

# Two New Combinatorial Problems involving Dominating Sets for Lottery Schemes

Werner R Gründlingh



Dissertation presented for the degree  
**Doctor of Philosophy**  
at the Department of Applied Mathematics of the  
University of Stellenbosch, South Africa

Promoter: Prof JH van Vuuren  
Co-promoter: Dr AP Burger

December 2004



# Declaration

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

# Abstract

Suppose a lottery scheme consists of randomly selecting an unordered winning  $n$ -subset from a universal set of  $m$  numbers, while a player participates in the scheme by purchasing a playing set of any number of unordered  $n$ -subsets from the same universal set prior to a winning draw, and is awarded a prize if  $k$  or more numbers in the winning  $n$ -set match those of at least one of the player's  $n$ -sets in his/her playing set ( $k \leq n \leq m$ ). Such a prize is called a  $k$ -prize. A player may wish to construct a smallest playing set for which he/she is at least  $100\psi\%$  sure of winning a  $k$ -prize ( $0 < \psi \leq 1$ ). From a different perspective, a player may wish to construct a playing set of specified cardinality in such a way that the probability of winning a  $k$ -prize is maximised, regardless of the winning  $n$ -set. These situations lead to the following two related combinatorial problems: (i) the *incomplete lottery problem* and (ii) the *resource utilisation problem*. The questions posed in these problems are: (i) What is the smallest possible cardinality of a playing set for which the probability of winning a  $k$ -prize is at least  $\psi$ ? and (ii) What is the largest possible probability of winning a  $k$ -prize via a playing set of specified cardinality  $\ell$ ? The answers to these two questions are given by the so-called incomplete lottery and resource utilisation numbers,  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$ , respectively. The well-known complete lottery problem, which asks for a 100% guarantee of winning a  $k$ -prize, is a special case of the incomplete lottery problem, namely where  $\psi = 1$ . However, both the incomplete lottery and resource utilisation problems constitute novel contributions of this dissertation to the combinatorial literature that may be of greater practical interest than their theoretical counterpart, the complete lottery problem, which appears for the first time in the literature in 1964.

The incomplete lottery and resource utilisation problems are presented as combinatorial optimisation problems and translated to within the realm of graph theory, introducing the notion of a so-called lottery graph. From this background, both analytic and algorithmic arguments are used to establish bounds on incomplete lottery and resource utilisation numbers. New, small (in)complete lottery and resource utilisation numbers are also found. A technique for characterising overlapping structures of optimal solutions to both problems is developed. This technique is then used to determine the number,  $\eta_\psi(m, n; k)$  [ $\eta_{\Psi_\ell}(m, n; k)$ , respectively], of structurally different optimal solutions to the incomplete lottery problem [resource utilisation problem, respectively]. Various characteristics of the sequences  $\eta_\bullet(m, n; k)$ , for variations in the arguments  $\bullet$ ,  $m$ ,  $n$  and  $k$ , are also uncovered.

Although both the above combinatorial optimisation problems are far from being resolved completely, the reader is provided with an impression of the underlying problem complexity within the scope of this dissertation. More specifically, the broad aims of this dissertation are threefold: first to develop a structured, efficient solution methodology for establishing solution bounds to the above two hard combinatorial problems, secondly to contribute toward optimal solutions of small instances of the problems and thirdly to contribute toward characterising optimal solutions to both problems, in terms of overlapping structure.

# Opsomming

Gestel 'n lotery bestaan uit die lukrake seleksie van 'n ongeordende wen  $n$ -tal uit 'n universele versameling van  $m$  getalle, terwyl spelers aan die lotery deelneem deur 'n spel-versameling van enige aantal ongeordende  $n$ -talle uit dieselfde universele versameling voor die wentrekking te kies. 'n Prys word aan 'n speler toegeken indien  $k$  of meer getalle in die wen  $n$ -tal ooreenstem met minstens een van die speler se  $n$ -talle in sy/haar spel-versameling ( $k \leq n \leq m$ ). Na só 'n prys word verwys as 'n  $k$ -prys. 'n Speler mag ten doel hê om 'n minimale grootte spel-versameling te konstrueer sodat hy/sy 'n  $k$ -prys met 'n waarskynlikheid van minstens  $100\psi\%$  sal wen, ongeag watter wen  $n$ -tal getrek word ( $0 < \psi \leq 1$ ). Alternatiewelik mag 'n speler poog om 'n spel-versameling van gespesifiseerde kardinaliteit só te konstrueer dat die kans om 'n  $k$ -prys te wen, gemaksimeer word, ongeag die waardes in die wen  $n$ -tal. Hierdie situasies het die volgende twee verwante kombinatoriese optimeringsprobleme tot gevolg: (i) die *onvolledige lotery-probleem* en (ii) die *hulpbron-benuttingsprobleem*. Die vrae wat in die onderskeie probleme gestel word, is: (i) Wat is die kleinste kardinaliteit van 'n spel-versameling wat, met 'n waarskynlikheid van minstens  $\psi$ , aan 'n speler 'n  $k$ -prys sal waarborg? en (ii) Wat is die grootste kans om 'n  $k$ -prys te wen, deur gebruikmaking van 'n spel-versameling van gespesifiseerde kardinaliteit  $\ell$ ? Die antwoorde op hierdie vrae word deur die sogenaamde onvolledige lotery- en hulpbron-benuttingsgetalle,  $L_\psi(m, n; k)$  en  $\Psi_\ell(m, n; k)$ , onderskeidelik aangedui. Die bekende volledige lotery-probleem, waarin om 'n 100% waarborg van 'n  $k$ -prys gevra word, is 'n spesiale geval van die onvolledige lotery-probleem, en wel waar  $\psi = 1$ . Beide die onvolledige lotery- en hulpbron-benuttingsprobleem is egter nuwe toevoegings tot die kombinatoriese literatuur oor loterye, wat moontlik groter praktiese belangstelling as die (teoretiese) volledige lotery-probleem, wat vir die eerste keer in 1964 in die literatuur verskyn, sou kon ontlok.

Die onvolledige lotery- en hulpbron-benuttingsprobleme word formeel as kombinatoriese optimeringsprobleme gedefinieer en spesifiek vanuit 'n grafiekteoretiese oogpunt ondersoek, deur middel van die invoering van 'n sogenaamde lotery-grafiek. Beide analitiese sowel as algoritmiese grafiekteoretiese argumente word gebruik om grense op die onvolledige lotery- en hulpbron-benuttingsgetalle daar te stel. Nuwe, klein (on)volledige lotery- en hulpbron-benuttingsgetalle word ook gevind. 'n Tegniek om oorvleuelingstrukture van optimale oplossings tot beide probleme te karakteriseer, word ontwikkel. Met hierdie tegniek kan die aantal,  $\eta_\psi(m, n; k)$  [ $\eta_{\Psi_\ell}(m, n; k)$ , respektiewelik], struktureel verskillende optimale oplossings tot die onvolledige lotery-probleem [hulpbron-benuttingsprobleem, respektiewelik] bepaal word. Verskeie eienskappe van die rye  $\eta_\bullet(m, n; k)$ , vir variasies in die argumente  $\bullet$ ,  $m$ ,  $n$  en  $k$ , word ook ondersoek.

Beide kombinatoriese optimeringsprobleme is vêr van volledig opgelos, maar daar word binne die omvang van hierdie proefskrif gepoog om die onderliggende kompleksiteite van die probleme bloot te lê. Meer spesifiek, die breë doelstelling in hierdie proefskrif is drieledig, naamlik om eerstens 'n gestruktureerde, doeltreffende metodologie vir die konstruksie van oplossingsgrense vir hierdie moeilike kombinatoriese probleme daar te stel, om tweedens bydraes in terme van eksakte oplossings vir klein gevalle van die probleme te maak en om derdens die oorvleuelingstrukture van optimale oplossings tot beide probleme te karakteriseer.

# Terms of Reference

This dissertation forms part of a larger project at the University of Stellenbosch devoted to the *lottery problem*, which appears for the first time in the combinatorial literature during the early 1960s. The novel contribution of the approach taken in this project is that the lottery problem is viewed within a graph theoretic domination context, whereas traditional approaches to this problem are usually launched from a combinatorial design theory platform. The project was initiated with an inquiry by Prof Dirk Laurie (Department of Mathematics, University of Stellenbosch) on the 9<sup>th</sup> of June 2000. Dr Alewyn Burger (School of Mathematics and Statistics, University of Victoria) and Prof Jan van Vuuren (Department of Applied Mathematics, University of Stellenbosch) started working on this project in September 2000. At the time of writing, Dr Burger and Prof Van Vuuren had been involved in the development of both analytic and algorithmic bounds for four-parameter complete lottery numbers, utilising a wide range of mathematical tools from graph domination theory to combinatorial designs. This dissertation builds on some of their ideas and makes a number of novel contributions, most notably the introduction and subsequent study of the *incomplete context for the lottery problem*, as well as that of the *resource utilisation problem*, a problem closely related to the lottery problem. New, small (in)complete lottery and resource utilisation numbers may be found in this dissertation. A method of characterising overlapping structures of optimal solutions to both problems is also included. Another important contribution of this dissertation toward the greater lottery project is the computerised implementation of a number of heuristics, used to obtain lower and upper bounds on respectively resource utilisation and incomplete lottery numbers.

Prof Van Vuuren was the promoter for this dissertation, while Dr Burger acted as co-promoter. The computing facilities of the Departments of Applied Mathematics, Electrical & Electronic Engineering and Computer Science at Stellenbosch University (South Africa), as well as the Departments of Mathematics & Statistics and Mechanical Engineering at the University of Victoria (Canada) were used in parallel during numerical investigations contained in this dissertation. Work on the current dissertation was commenced in February 2001 in the form of an MSc thesis, which was upgraded to a PhD dissertation in March 2003 and was completed in December 2004. The larger lottery problem project at the University of Stellenbosch supersedes both the scope and completion date of this dissertation.

# Acknowledgements

Nearly four years ago, a choice was made culminating in this dissertation. The author hereby wishes to personally express his gratitude towards

- the Departments of Applied Mathematics, Electrical & Electronic Engineering and Computer Science of the University of Stellenbosch (South Africa), for the use of their computing facilities;
- the Departments of Mathematics & Statistics and Mechanical Engineering of the University of Victoria (Canada), for the use of their computing facilities and for receiving the author as an international visiting student during the period 5 January – 6 November 2004;
- prof JH van Vuuren, for his insight, patience and diligence throughout. It was a truly great pleasure working on a project such as this, utilising all his available academic and intuitive resources;
- dr AP Burger, for his skill on the technical side of this dissertation, the verification of some computer results and his broad ability to “see things”;
- proff EJ Cockayne and CM Mynhardt (of the Department of Mathematics & Statistics, University of Victoria) and dr W Myrvold (of the Department of Computer Science, University of Victoria) for the time they spent with the author when adding the finishing touches to this dissertation; and
- friends and family, for their support, comfort and enthusiasm when late nights became early mornings (if at all).

This dissertation is based upon work supported by the South African National Research Foundation (NRF) under grant number GUN 2053755. Additionally, the financial assistance of the National Department of Labour (DoL) towards this research is hereby acknowledged. Any opinions, findings and conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the NRF or the DoL. Financial assistance contributing towards this research project was also granted by the Post-graduate Bursary Office and Research Sub-Committee B of the University of Stellenbosch, the Harry Crossley Foundation and the Wilhelm Frank Stipendium Trust.





# Table of Contents

List of Figures	v
List of Tables	vii
List of Algorithms	ix
Glossary	xi
Translation of Terminology (Afrikaans)	xix
List of Reserved Symbols	xxiii
<b>1 Introduction</b>	<b>1</b>
1.1 Historical background . . . . .	1
1.2 Problem descriptions . . . . .	2
1.3 Literature on the lottery, packing and covering problems . . . . .	7
1.3.1 Complete lottery numbers . . . . .	7
1.3.2 Packing and covering numbers . . . . .	8
1.4 Scope and objectives of this dissertation . . . . .	10
1.5 Preview of dissertation layout . . . . .	10
<b>2 Properties of <math>L_\psi(m, n; k)</math> and <math>\Psi_\ell(m, n; k)</math></b>	<b>13</b>
2.1 Boundedness of $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$ . . . . .	13
2.2 Growth properties of $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$ . . . . .	15
2.3 Explicit values for $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$ . . . . .	17
2.4 Binary programming solution approaches . . . . .	21
2.5 Chapter summary . . . . .	24
<b>3 The Lottery Graph</b>	<b>25</b>
3.1 Fundamentals from graph and complexity theory . . . . .	25
3.2 Definition and properties of the lottery graph . . . . .	30
3.3 Symmetric representation of the lottery graph . . . . .	39
3.4 Analysis of small lotteries . . . . .	40

---

3.5	Chapter summary . . . . .	40
<b>4</b>	<b>Analytic bounds</b>	<b>51</b>
4.1	Graph theoretic bounds on $L_\psi(m, n; k)$ . . . . .	51
4.1.1	Lower bounds on $L_1(m, n; k)$ . . . . .	51
4.1.2	Upper bounds on $L_\psi(m, n; k)$ . . . . .	52
4.2	Other bounds on $L_\psi(m, n; k)$ . . . . .	54
4.2.1	Lower bounds on $L_1(m, n; k)$ . . . . .	54
4.2.2	Upper bounds on $L_\psi(m, n; k)$ . . . . .	55
4.3	Comparison of analytic bounds on $L_1(m, n; k)$ . . . . .	56
4.4	Chapter summary . . . . .	56
<b>5</b>	<b>Algorithmic bounds</b>	<b>61</b>
5.1	Classical random algorithm . . . . .	62
5.2	Distributed random algorithm . . . . .	63
5.3	Minimal overlapping algorithm . . . . .	65
5.4	Neighbourhood removal algorithm . . . . .	66
5.5	Tabu search algorithm . . . . .	69
5.6	Genetic algorithm . . . . .	75
5.6.1	Classical genetic algorithm . . . . .	76
5.6.2	Intelligent genetic algorithm . . . . .	77
5.7	Comparison of algorithms for small lotteries . . . . .	79
5.8	Comparison of algorithms for larger lotteries . . . . .	83
5.8.1	The lottery class $\langle m, 5; 2 \rangle$ . . . . .	83
5.8.2	The lottery $\langle 49, 6; 3 \rangle$ . . . . .	84
5.9	Chapter summary . . . . .	84
<b>6</b>	<b>Optimal solution characterisations</b>	<b>87</b>
6.1	Characterisation of $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set structures . . . . .	87
6.1.1	Lottery tree method . . . . .	87
6.1.2	nauty tree method . . . . .	95
6.2	General properties of the parameter $\eta_\psi(m, n; k)$ . . . . .	98
6.3	Analysis of small lotteries . . . . .	107
6.4	New complete lottery numbers and improved bounds . . . . .	107
6.5	Chapter summary . . . . .	123
<b>7</b>	<b>Conclusions</b>	<b>127</b>
7.1	Dissertation summary . . . . .	127
7.2	Possible future work . . . . .	128

---

<b>References</b>	<b>137</b>
<b>Appendices</b>	<b>150</b>
<b>A Computer programs</b>	<b>151</b>
A.1 Classical random algorithm (Algorithm 2) . . . . .	151
A.2 Distributed random algorithm (Algorithm 3) . . . . .	153
A.3 Minimal overlapping algorithm (Algorithm 4) . . . . .	155
A.4 Neighbourhood removal algorithm (Algorithm 5) . . . . .	159
A.5 Tabu search algorithm (Algorithm 6) . . . . .	161
A.6 Classical genetic algorithm . . . . .	165
A.7 Intelligent genetic algorithm (Algorithm 7) . . . . .	167
A.8 $L_\psi(m, n; k)$ -set characterisation algorithm (Algorithm 8) . . . . .	172
<b>B Best Internet bounds on <math>L_1(m, n; k)</math></b>	<b>179</b>
<b>C Optimal <math>\vec{X}^{(\ell)}</math>-vector encodings</b>	<b>183</b>



# List of Figures

2.1	Graphical representation of counting argument in Lemma 2.2 . . . . .	17
2.2	Qualitative schematic representation of $\mathcal{F}(\ell)$ when determining $L_\psi(m, n; 1)$ . . . . .	20
3.1	Examples of basic graphs . . . . .	26
3.2	Two isomorphic graphs . . . . .	27
3.3	A vertex-transitive 3-regular graph and a non vertex-transitive 3-regular graph . . . . .	28
3.4	The lottery graph $G\langle 7, 3; 2 \rangle$ (Example 3.1) . . . . .	32
3.5	Vertex-induced subgraphs of $G\langle 7, 3; 2 \rangle$ dominated by $\{1, 5, 7\}$ and $\{3, 4, 6\}$ . . . . .	33
3.6	Argument for determining which $G\langle m, n; k \rangle$ is strongly regular . . . . .	34
3.7	Complete lottery set structures for $\langle 14, 6; 3 \rangle$ (Example 3.2) . . . . .	36
3.8	Graphical representation of counting argument in Lemma 3.1 . . . . .	38
3.9	Vertex generators and lottery graph for $\langle 5, 3; 2 \rangle$ (Example 3.3) . . . . .	41
3.10	Symmetric graph representation of $G\langle 7, 3; 2 \rangle$ . . . . .	47
3.11	Lottery graphs $G\langle m, n; k \rangle$ or $\overline{G\langle m, n; k \rangle}$ for $1 \leq k < n < m \leq 10$ . . . . .	48
5.1	Lower bound on the resource utilisation $\Psi_{100}(20, 4; 3)$ of Algorithm 2 . . . . .	63
5.2	Lower bound on the resource utilisation $\Psi_{100}(20, 4; 3)$ of Algorithm 3 . . . . .	65
5.3	Lower bound on the resource utilisation $\Psi_{100}(20, 4; 3)$ of Algorithm 4 . . . . .	67
5.4	Step-by-step analysis of Algorithm 5 as applied to $\langle 8, 3; 2 \rangle$ . . . . .	68
5.5	Three defined moves for the tabu search algorithm (Shift, Flip and Swap) . . . . .	72
5.6	Lower bound on the resource utilisation $\Psi_{100}(20, 4; 3)$ of Algorithm 6 . . . . .	74
5.7	Typical structure of an evolution program . . . . .	75
5.8	Crossover and mutation procedures of genetic algorithm . . . . .	77
5.9	Influence on resource utilisation due to genetic algorithm parameter variation . . . . .	83
5.10	Lower bounds on the resource utilisation $\Psi_\ell(49, 6; 3)$ using Algorithms 2–4 . . . . .	85
6.1	Complete lottery tree and domination test for the lottery $\langle 5, 3; 2 \rangle$ . . . . .	90
6.2	Fragmented lottery tree for $\langle 7, 3; 2 \rangle$ (Example 6.2) . . . . .	91
6.3	Partial lottery tree for $\langle 14, 6; 3 \rangle$ (Example 6.5) . . . . .	94
6.4	Algorithmic statistics for and characterisations of $L_1(m, 6; 3)$ -sets (Example 6.6) . . . . .	96
6.5	Graphical representation of $L_\psi(m, n; k)$ -sets for use with <b>nauty</b> . . . . .	97

---

6.6	Fragmented nauty tree for $\langle 7, 3; 2 \rangle$ . . . . .	99
6.7	The resource utilisation characterisation number $\eta_\psi(15, 6; 3)$ as a function of $0.4 \leq \psi \leq 1$ .	101
6.8	Values of $\eta_1(m, 6; k)$ and $L_1(m, 6; k)$ for $6 \leq m \leq 35$ and $k = 1, 2$ . . . . .	102
6.9	All $\eta_\psi(m, n; k)$ $L_\psi(m, n; k)$ -set structure characterisations for $1 \leq k < n < m \leq 10$ . . . . .	111
7.1	Ticket efficiency parameter $\mathcal{E}_\ell(m, n; k)$ . . . . .	131
7.2	Example of determining $G(5, 2; 1)^\bullet$ . . . . .	135

# List of Tables

1.1	Typical parameters for lotteries around the world . . . . .	3
1.2	The class of complete lottery numbers $L_1(m, 6; 2)$ . . . . .	8
1.3	Bounds on complete lottery numbers of the form $L_1(m, 6; 3)$ . . . . .	8
1.4	The class of covering numbers $C(m, 5; 2)$ where $m \leq 27$ . . . . .	10
2.1	Binary programming approach toward the complete and incomplete lottery problems . . .	22
2.2	The number of formulations to be solved to uniquely determine $L_\psi(m, n; k)$ with (2.11) .	23
3.1	Complete lottery numbers, $L_1(m, n; k)$ , for $\langle m, n; k \rangle$ , $1 \leq k \leq n \leq m \leq 10$ . . . . .	49
4.1	Known $L_1(m, n; k)$ and analytic bounds on $L_1(m, n; k)$ , $1 \leq k \leq n \leq 6$ and $3 \leq m \leq 50$ . .	57
4.2	Analytic bounds on the complete lottery number $L_1(m, 6; 3)$ for all $6 \leq m \leq 50$ . . . . .	59
5.1	Step-by-step analysis of Algorithm 5 on $\langle 20, 4; 3 \rangle$ . . . . .	70
5.2	Resource utilisation $\Psi_\ell(m, n; k)$ for $\langle m, n; k \rangle$ , $1 \leq k < n < m \leq 10$ and $2 \leq \ell \leq L_1(m, n; k)$	80
5.3	Comparitive bounds on $L_1(m, 5; 2)$ , determined by Algorithms 2–7, where $5 \leq m \leq 30$ . .	84
6.1	All structurally different $\eta_\psi(7, 3; 2)$ $L_\psi(7, 3; 2)$ -sets ( $0 < \psi \leq 1$ ) for $\langle 7, 3; 2 \rangle$ . . . . .	92
6.2	Alternative approach toward determining winning probabilities of $\vec{X}^{(2)}$ for $\langle 5, 3; 2 \rangle$ . . . .	93
6.3	Values for $\zeta_\ell(m, n)$ for $3 \leq \ell \leq 5$ , $4 \leq m \leq 20$ and $3 \leq n \leq 12$ . . . . .	105
6.4	Values of $(100\Psi_\ell(m, n; 2), \eta_{\Psi_\ell}(m, n; 2))$ . . . . .	108
6.5	Values of $(100\Psi_\ell(m, n; 3), \eta_{\Psi_\ell}(m, n; 3))$ . . . . .	109
6.6	Values of $(100\Psi_\ell(m, n; 4), \eta_{\Psi_\ell}(m, n; 4))$ . . . . .	110
6.7	Improved bounds on and exact values for $L_1(m, n; k)$ in lotteries $\langle m, n; k \rangle$ . . . . .	124
6.8	Lottery characterisation numbers $\eta_1(m, n; k)$ for $\langle m, n; k \rangle$ , $1 \leq k \leq n \leq m \leq 10$ . . . . .	125
7.1	Lottery graph core $G\langle m, n; k \rangle^\bullet$ . . . . .	134
B.1	Playing and lottery set for $\langle 49, 6; 3 \rangle$ . . . . .	179
B.2	Known $L_1(m, n; k)$ and best known bounds on $L_1(m, n; k)$ . . . . .	180
B.3	Best bounds on the lottery number $L_1(m, 6; 3)$ for all $6 \leq m \leq 50$ . . . . .	182





# List of Algorithms

1	Symmetric representation algorithm for $G\langle m, n; k \rangle$ . . . . .	39
2	Classical random algorithm . . . . .	63
3	Distributed random algorithm . . . . .	64
4	Minimal overlapping algorithm . . . . .	66
5	Neighbourhood removal algorithm . . . . .	69
6	Tabu search algorithm . . . . .	73
7	(Classical/intelligent) Genetic algorithm . . . . .	78
8	$L_\psi(m, n; k)$ -set characterisation algorithm . . . . .	89



# Glossary

**100(1 -  $\psi$ )%–incomplete lottery set** A *playing set* that *dominates* at least 100 $\psi$ % of the vertices of a *lottery graph*.

**100 $\psi$ %–partially dominating set** A *vertex subset* of minimum *cardinality* of a *graph* that *dominates* at least 100 $\psi$ % of the *graph vertices*.

**adjacency matrix** A  $p \times p$  matrix associated with a *graph*  $\mathcal{G}$  of *order*  $p$ , where the matrix element  $a_{ij}$  takes the value 1 if *vertex*  $i$  is *adjacent* to *vertex*  $j$  and the value 0 otherwise. All diagonal entries are ones, by (a non-standard) convention.

**acyclic graph** A *graph* without *cycles*.

**adjacent** Said of two *vertices* of a *graph* if they are *joined* by an *edge*.

**algorithm** An ordered sequence of procedural operations for solving a problem within a finite number of steps.

**aspiration criterion** A rule incorporated in a *tabu search* approach towards solving a combinatorial optimisation problem approximately, whereby the *tabu status* of a *neighbouring move* is overridden if the resulting *trial solution* is better than the best solution obtained thus far.

**asymptotic** Said of a bound on a function that is valid for all values of the function argument greater than some fixed value.

**automorphism** An *isomorphism* of *graph* onto itself. Such an *isomorphism* is a permutation of the *vertex set*, which preserves *adjacency*.

**automorphism group** The *group* of all *automorphisms* of a *graph*.

**basic operation** A single (binary) operation, performed by the *central processing unit* of a computer.

**binary programming problem** An *integer programming problem* with the additional constraint that the *decision variables* may only take binary values (0 or 1).

**block design** A 6-tuple  $(v, l, t, s, \lambda, b)$ , consisting of a carefully selected subset of  $b$   $l$ -subsets from  $v$  combinatorial elements (also called varieties) which ensures the existence of a minimum of at least  $\lambda$   $l$ -subsets containing a  $t$ -subset match with any  $s$ -subset from the  $v$  elements.

**branch-and-bound** An *exact* optimisation method consisting of exhaustive *tree* search enumeration of the solution space to an integer optimisation problem by means of relaxations of the problem. Branching occurs iteratively on non-integer solution components of relaxation subproblems and bounding occurs when it becomes apparent that a subtree cannot contain an optimal solution.

**candidate list strategy** A protocol whereby a proportion/subset of the possible *neighbouring moves* are considered when making a choice with respect to the next *trial solution* in a *tabu search* approach towards solving a combinatorial optimisation problem approximately.

**cardinality** The number of elements in a set.

- 
- central processing unit** The hardware component of a computer in which all *basic operations* are performed during computations (abbr. CPU).
- chromosome** The individuals in a *population* of candidate solutions to a combinatorial optimisation problem in a *genetic algorithmic* approach toward solving the problem approximately.
- class NP** Acronym for *Non-deterministic Polynomial*. The set of all *decision problems* which may be answered “yes” by a *polynomial time algorithm*, given additional information (called a certificate to the problem instance at hand).
- class NP-complete** The set of all *NP-complete decision problems*.
- class P** Acronym for *Polynomial*. The set of all *decision problems* that may be solved by a *polynomial time algorithm*.
- combinatorial matrix of type  $(m, n)$**  A matrix that has  $\binom{m}{n}$  rows and columns, indexed by the  $n$ -element subsets of an  $m$ -element *universal set*. The matrix entry in row  $u$  and column  $v$  should depend only on the cardinality of  $|u \cap v|$ .
- complement** A *graph* (denoted  $\bar{\mathcal{G}}$ ) associated with a given graph  $\mathcal{G}$  whose *vertex set* is  $V(\bar{\mathcal{G}}) = V(\mathcal{G})$  and which contains an *edge* if and only if the *edge* is not an *edge* of  $\mathcal{G}$ .
- complete graph** A *graph* of order  $p$  that is  $(p - 1)$ -regular.
- complete lottery number** The minimum *cardinality* of a *complete lottery set*.
- complete lottery set** A *playing set* for the *lottery*  $\langle m, n; k \rangle$  consisting of  $n$ -subsets from an  $m$ -element *universal set*, which contains a member with at least  $k$  elements in common with any chosen winning  $n$ -subset.
- component** A maximally *connected subgraph* of a given *graph*.
- connected** Said of a *graph*  $\mathcal{G}$  if there exists a  $u$ - $v$  *path* for every *vertex pair*  $u, v \in V(\mathcal{G})$ .
- core** A *graph*  $\mathcal{G}$  for which any *homomorphism* from  $\mathcal{G}$  onto itself is necessarily a bijection.
- covering number** The minimum *cardinality* of a *covering set* for a *lottery*.
- covering set** A set  $\mathcal{C}$  associated with the *lottery*  $\langle m, n; k \rangle$  consisting of  $n$ -subsets from an  $m$ -element *universal set* with the property that every  $k$ -subset of the *universal set* of elements is contained in some element of  $\mathcal{C}$ .
- crossover** The procedure (specific to a *genetic algorithm*) in which two (parent) *chromosomes* are paired and subsequently used to generate offspring.
- cycle** A *walk* of length  $n \geq 3$  with the property that the first and last *vertices* are the same and no other (internal) *vertices* are repeated.
- decision problem** A problem that may be interpreted as a binary question, which may be answered either “yes” or “no.”
- decision variable** An indeterminate variable quantity in an *integer programming problem*.
- degree** The number of *vertices adjacent* to a *vertex* in a *graph*.
- density** The ratio between the *size* of a *graph* and the *size* of a *complete graph* of the same *order*.
- diameter** The maximum *eccentricity* of a *vertex* in a *graph*.
- disconnected** Said of a *graph* that is not *connected*.
- disjoint** Said of two sets if their intersection is empty.
- distance** (between two *vertices*  $u$  and  $v$  in a *graph*) The minimum *length*, of all  $u$ - $v$  *paths* in a *graph*, if any such *paths* exist. If no  $u$ - $v$  *path* exists, then the distance between  $u$  and  $v$  is taken as infinite.

- diversification strategy** A search protocol (typically employed by *heuristic optimisation techniques*) whereby new areas of the solution space of a combinatorial optimisation problem are explored.
- (lower) domination number** The minimum *cardinality* of a *minimal dominating set* of a *graph*.
- dominated** A *vertex*  $u$  in a *graph*  $\mathcal{G}$  is dominated by a *vertex*  $v$  in  $\mathcal{G}$  if  $u$  and  $v$  are *adjacent* (or the same). A *vertex*  $u$  is said to be dominated by a *vertex subset*  $\mathcal{D} \subseteq V(\mathcal{G})$  if  $u$  is dominated by some *vertex*  $v \in \mathcal{D}$ . The *graph*  $\mathcal{G}$  is said to be dominated by a *vertex subset*  $\mathcal{D} \subseteq V(\mathcal{G})$  if every *vertex* in  $\mathcal{G}$  is dominated by  $\mathcal{D}$ .
- dominating set** A *vertex subset*  $\mathcal{D} \subseteq V(\mathcal{G})$  of a *graph*  $\mathcal{G}$  with the property that every *vertex* of  $\mathcal{G}$  is either an element of  $\mathcal{D}$ , or *adjacent* to an element of  $\mathcal{D}$  (or both). Also sometimes called a domination set.  $\mathcal{G}$  is sometimes said to be *dominated* by  $\mathcal{D}$ .
- domination test** The test performed on the last level of the *lottery tree* in order to determine which of the nodes on the penultimate level of the *lottery tree* constitute *incomplete lottery sets* of minimum *cardinality*.
- eccentricity** The *distance* from a *vertex*  $v$  in a *graph*  $\mathcal{G}$  to a *vertex* in  $\mathcal{G}$  furthest from  $v$ .
- efficient** Said of a *polynomial time algorithm*.
- edge** The elements of the *edge set* of a *graph*.
- edge set** A (possibly empty) finite set of two–element subsets of the *vertex set* of a *graph*.
- edge–transitive** Said of a *graph*  $\mathcal{G}$  if, for all *edges*  $e, f \in E(\mathcal{G})$ , there exists an *automorphism* that maps the endpoints of  $e$  to the endpoints of  $f$  (in either order).
- exact** Said of an optimisation technique that is guaranteed to find an optimal solution to a combinatorial optimisation problem.
- gene** An attribute of a feasible or candidate solution in a *genetic algorithmic* approach toward solving a combinatorial optimisation problem approximately.
- gene pool** The collection of all *genes* in a *genetic algorithmic* toward solving a combinatorial optimisation problem approximately.
- generation** The set or *population* of candidate solutions at any time step in a *genetic algorithmic* approach toward solving a combinatorial optimisation problem approximately.
- genetic algorithm** A *heuristic* optimisation technique designed with the intention of simulating the natural process of biological evolution when attempting to solve a combinatorial optimisation problem approximately.
- girth** The *length* of a shortest *cycle* in a *graph*. If no *cycles* exist in the *graph*, the girth is taken as infinite, by convention.
- graph** A combinatorial object  $\mathcal{G} = (V, E)$  consisting of a non–empty, finite set  $V$  of combinatorial objects called *vertices* as well as a (possibly empty) finite set  $E$  of two–element subsets of  $V$ , called *edges*.
- group** A set  $\mathcal{W}$ , along with a binary operator  $\bullet$  for which  $\bullet$  is associative over the elements of  $\mathcal{W}$ , for which  $\mathcal{W}$  is closed with respect to the application of  $\bullet$ , for which there is an identity element for  $\bullet$  in  $\mathcal{W}$ , and for which each element in  $\mathcal{W}$  has an inverse under  $\bullet$ .
- heuristic** Said of an optimisation technique that is not necessarily *exact*.
- homomorphism** A mapping  $f$  from a *vertex set* of a *graph*  $\mathcal{G} = (V, E)$  to a *vertex set* of a *graph*  $\mathcal{H} = (V', E')$  such that for each *edge*  $uv \in E$ ,  $f(u)f(v) \in E'$ .
- incomplete lottery characterisation number** The number of structurally different optimal solutions to the incomplete lottery problem.

- incomplete lottery number** The minimum *cardinality* of a  $100(1 - \psi)\%$ -incomplete lottery set.
- (vertex) independence number** The maximum *cardinality* of a *maximal independent set* of a graph.
- (vertex) independent set** A *vertex subset* of a graph containing no adjacent vertex pairs.
- integer programming problem** A *linear programming problem* with the additional constraint that the *decision variables* may only take integer values.
- intensification strategy** A search protocol (typically employed by *heuristic optimisation techniques*) that exploits those good solutions obtained during the search, by intensifying the local search around their respective neighbourhoods in the solution space of a combinatorial optimisation problem.
- intractable** Said of a *decision problem* for which no *polynomial time algorithm* is known.
- isolated** Said of a *vertex* with *degree* zero.
- isomorphic** Said of two *graphs* between which there exists an *isomorphism*.
- isomorphism** A bijection between the *vertex sets* of two *graphs* that preserves *adjacency*.
- jackpot** The maximum prize that is awarded in a *lottery*.
- joined** Said of two *vertices* that are *adjacent*.
- $k$ -prize** The prize awarded in the *lottery*  $\langle m, n; k \rangle$  if there is at least one  $n$ -subset from an  $m$ -element *universal set* in the participant's *playing set* coinciding in at least  $k$  elements with the *winning draw*.
- length** The number of *edges* contained in a *cycle*, *path* or *walk*.
- linear programming problem** A mathematical programming problem in which the *objective* is to optimise some linear function of indeterminate variables, subject to a set of linear constraints (defining a feasible domain for the problem).
- loop** An *edge* in a *pseudograph* that *joins* a *vertex* to itself.
- lottery** A procedure (denoted by  $\langle m, n; k \rangle$ ) of randomly selecting (without replacement) a winning  $n$ -subset  $w$  from a *universal set*  $\mathcal{U}_m$  and awarding a  $k$ -prize to a participant if he/she has a *playing set* that contains an  $n$ -subset coinciding with  $w$  in at least  $k$  elements.
- lottery graph** A *graph* associated with the *lottery*  $\langle m, n; k \rangle$  whose *vertices* represent  $n$ -subsets of an  $m$ -element *universal set*, in which two *vertices* are *adjacent* if their corresponding *vertex labels* share a common  $k$ -subset.
- lottery tree** A search *tree* associated with the *lottery*  $\langle m, n; k \rangle$  containing all structurally different  $n$ -subset *overlapping structures* of *cardinality*  $i$  from an  $m$ -element *universal set* on level  $i$  of the *tree*.
- $L_1(m, n; k)$ -set** A *complete lottery set* of minimum *cardinality* for the *lottery*  $\langle m, n; k \rangle$ .
- $L_\psi(m, n; k)$ -set** A  $100(1 - \psi)\%$ -incomplete lottery set of minimum *cardinality* for the *lottery*  $\langle m, n; k \rangle$ .
- maximal independent set** An *independent set* of a *graph* with the property that no proper superset is an *independent set* of the *graph*.
- minimal dominating set** A *dominating set* of a *graph* with the property that no proper subset is a *dominating set* of the *graph*.
- minimal  $100\psi\%$ -partially dominating set** A  $100\psi\%$ -*partially dominating set* of a *graph* with the property that no proper subset is a  $100\psi\%$ -*partially dominating set* of the *graph*.
- minimum [maximum] degree** The minimum [maximum] of all *vertex degrees* of a *graph*.

- minimum dominating set** A *minimal dominating set* of minimum cardinality.
- minimum 100 $\psi$ %-partially dominating set** A *minimal 100 $\psi$ %-partially dominating set* of minimum cardinality.
- multigraph** A *graph* with more than one *edge* between some pair of *vertices*.
- multiplicity** The number of *playing sets* in the *lottery*  $\langle m, n; k \rangle$  that conforms to the same *n*-set *overlapping structure*.
- mutate** The *diversification strategy* specific to a *genetic algorithm* whereby a solution candidate to a combinatorial optimisation problem is perturbed away from its current position in solution space.
- nauty graph** A *graph* associated with a *lottery* representing the *overlapping structure* of a *playing set* in the *lottery*.
- nauty tree** A search *tree* consisting of *i* levels that represent all *non-isomorphic nauty graphs* of order *i* for that *lottery*.
- (open) [(closed)] neighbourhood set** The set of all *vertices adjacent* to a given *vertex v* in a *graph* [including *v* itself].
- neighbouring move** The set of all *trial solutions* that may be reached from a given candidate solution by means of a single solution structure modification in a *tabu search* approach towards solving a combinatorial optimisation problem approximately. Also called the solution neighbourhood.
- NP-complete** Said of a *decision problem A* in the class **NP** satisfying  $\mathcal{A}' \preceq \mathcal{A}$  for all  $\mathcal{A}'$  in the class **NP**.
- objective function** A real function of indeterminate variables in an optimisation problem that is to be maximised/minimised, possibly subject to a collection of constraints on the indeterminate variables.
- operating system** The software that controls the execution of computer programs (abbr. OS).
- order** The number of *vertices* of a *graph*.
- overlapping structure** A complete specification (in terms of the inclusion-exclusion principle) of elements of an *m*-element *universal set* shared by members of a given *playing set* in the *lottery*  $\langle m, n; k \rangle$ .
- packing number** The maximum *cardinality* of a *packing set* for a *lottery*.
- packing set** A set  $\mathcal{P}$  associated with the *lottery*  $\langle m, n; k \rangle$  consisting of *n*-subsets from an *m*-element *universal set* with the property that no *k*-subset of the *universal set* is common to any two elements of  $\mathcal{P}$ .
- path** A *walk* in a *graph* with the property that no *vertex* is repeated.
- parallel edges** Two or more *edges* in a *multigraph* that *join* the same *vertex pair*.
- (lower) partial domination number** The minimum *cardinality* of a *minimal 100 $\psi$ %-partial dominating set*.
- playing set** A set associated with the *lottery*  $\langle m, n; k \rangle$ , consisting of a number of *n*-subsets from an *m*-element *universal set*.
- polynomial time** Said of an *algorithm* whose *worst case complexity* is a polynomial function of its input size.
- population** A collection of *chromosomes* in a *genetic algorithmic* approach toward solving a combinatorial optimisation problem approximately. Sometimes also referred to as a *generation* of candidates.
- pseudocode** An *algorithm* written independently of any implemented computer programming language, using common or natural language.

- 
- pseudograph** A graph containing *loops* and/or *parallel edges*.
- $\Psi_\ell(m, n; k)$ -**set** A *playing set* in the *lottery*  $\langle m, n; k \rangle$  of cardinality  $\ell$  that yields a maximum *resource utilisation*.
- radius** The minimum of all *eccentricities* of *vertices* in a *graph*.
- regular** Said of a *graph* if the *degrees* of all *vertices* are the same ( $r$ , say, in which case the *graph* is said to be  $r$ -regular).
- resource utilised** The ratio between the number of *vertices* of the *lottery graph dominated* by a *playing set* in the *lottery* and the *lottery graph order*.
- resource utilisation number** The maximum *resource utilised* by a *playing set* of given *cardinality*.
- size** The number of *edges* in a *graph*.
- space complexity** The amount of (computer) memory expended by an *algorithm*, as a function of input size of an *algorithm*.
- spanning subgraph** A *subgraph*  $\mathcal{H} = (V', E')$  of a *graph*  $\mathcal{G} = (V, E)$  with the properties that  $V' = V$  and  $E' \subseteq E$ .
- spanning tree** A *spanning subgraph* of a given *graph* that is also a *tree*.
- Steiner system** A *playing set* in the *lottery*  $\langle m, n; k \rangle$  with the property that any  $k$ -subset of an  $m$ -element *universal set* appears in exactly one of the members of the *playing set*.
- strongly regular** (with parameters  $(p, r, a, c)$ ) Said of an  $r$ -regular, order  $p$  *graph* for which every pair of *adjacent [non-adjacent] vertices* share  $a$  [ $c$ ] common neighbouring vertices.
- subgraph** A *graph*  $\mathcal{H} = (V', E')$  associated with a *graph*  $\mathcal{G} = (V, E)$  with the property that  $V' \subseteq V$  and  $E' \subseteq E$ .
- tabu list** One (or more) list(s) containing information about recent choices with respect to *neighbouring moves* in a *tabu search* approach towards solving a combinatorial optimisation problem approximately.
- tabu search** A *heuristic optimisation technique* which is based on designing and exploiting adaptive memory structures to determine iteratively locally optimal solutions to a combinatorial optimisation problem.
- tabu status** The status given to a specific *trial solution* contained in a *tabu list* when using a *tabu search* approach towards solving a combinatorial optimisation problem approximately.
- tabu tenure** The length of a *tabu list*.
- ticket efficiency parameter** The difference between the *resource utilisation number* of a *playing set* of given *cardinality*  $\ell$  less the *resource utilisation number* of a *playing set* of *cardinality*  $\ell - 1$ .
- time complexity** An estimate of the number of *basic operations* performed by an *algorithm* as a function of its input size.
- tractable** Said of a *decision problem* that may be solved by a *polynomial time algorithm*.
- tree** A *connected, acyclic graph*.
- trial solution** A (not necessarily optimal) solution to a combinatorial optimisation problem that usually forms part of an iterative step in a *heuristic optimisation technique*.
- universal set** The set of integers  $\{1, \dots, m\}$  in the *lottery*  $\langle m, n; k \rangle$ .
- vertex** A combinatorial object in terms of which the *vertex set* and *edge set* of a *graph* is defined.



---

**vertex-induced subgraph** A *subgraph*  $\mathcal{H} = (V', E')$  of a *graph*  $\mathcal{G} = (V, E)$  with the properties that  $V' \subseteq V$  and  $uv \in E'$  if  $uv \in E$  for all vertex pairs  $u, v \in V'$ .

**vertex set** A non-empty, finite set containing the *vertices* of a *graph*.

**vertex-transitive** Said of a *graph* if, for which all *vertex* pairs  $u, v$ , there exists an *automorphism* that maps  $u$  to  $v$ .

**walk** An alternating sequence of *vertices* and *edges*  $\mathcal{W}_{\mathcal{G}} : v_1, e_1, v_2, e_2, \dots, v_{n-1}, e_{n-1}, v_n$  in a *graph*  $\mathcal{G}$  where  $v_i \in V(\mathcal{G})$  and  $e_i = v_i v_{i+1} \in E(\mathcal{G})$  for all  $i = 1, \dots, n - 1$ , both beginning and ending in a *vertex*.

**winning draw** A (unordered) random selection of  $n$  elements (without replacement) from an  $m$ -element *universal set* in the lottery  $\langle m, n; k \rangle$ .

**worst case complexity** A (possibly *asymptotic*) upper bound on the space [time] complexity of an *algorithm*.



# Translation of Terminology

ENGLISH	AFRIKAANS
$100(1 - \psi)\%$ -incomplete lottery set	$100(1 - \psi)\%$ -onvolledige lotery-versameling
$100\psi\%$ -partially dominated	$100\psi\%$ -gedeeltelik gedomineer
$100\psi\%$ -partially dominating set	$100\psi\%$ -gedeeltelike dominasieversameling
adjacency matrix	naasliggendheidsmatriks
acyclic graph	asikliese grafiek
adjacent	naasliggend
algorithm	algoritme
aspiration criterion	doelwitreël
asymptotic	asimptoties
automorphism	outomorfisme
automorphism group	outomorfismegroep
basic operation	basiese berekening
binary programming problem	binêre programmeringsprobleem
block design	blokontwerp
branch-and-bound	vertak-en-begrens
candidate list strategies	kandidaatoplossing seleksiestrategie
cardinality	kardinaliteit
central processing unit (CPU)	sentrale verwerkingseenheid (SVE)
chromosome	chromosoom
class NP	klas NP
class NP-complete	klas NP-volledig
class P	klas P
combinatorial matrix of type $(m, n)$	kombinatoriese matriks van tipe $(m, n)$
complement	komplement
complete graph	volledige grafiek
complete lottery number	volledige lotery-getal
complete lottery set	volledige lotery-versameling
component	komponent
connected	samehangend
core	bron
covering number	oordekkingsgetal
covering set	oordekkingsversameling
crossover	oorgang, voortplanting
cycle	siklus
decision problem	beslissingsprobleem
decision variable	beslissingsveranderlike
degree	graad
density	digtheid
diameter	deursnee, deursnit
disconnected	onsamehangend
disjoint	disjunk
distance	afstand
diversification strategy	diversifikasie strategie

ENGLISH	AFRIKAANS
(lower) domination number	(laer) dominasiegetal
dominated	gedomineer
dominating/domination set	dominasieversameling
domination test	dominasietoets
eccentricity	eksentrisiteit
efficient	doeltreffent
edge	lyn
edge set	lyn-versameling
edge transitive	lyn-transitief
exact	eksak
gene	geen
gene pool	genepoel
generation	generasie, geslag
genetic algorithm	genetiese algoritme
girth	wydte
graph	grafiek
group	groep
heuristic	heuristiek
homomorphism	homomorfisme
incomplete lottery characterisation number	onvolledige lotery karakteriseringsgetal
incomplete lottery number	onvolledige lotery-getal
incomplete lottery set	onvolledige lotery-versameling
(vertex) independence number	(punt-) onafhanklikheidsgetal
(vertex) independent set	onafhanklike (punt-) versameling
integer programming problem	heeltallige programmeringsprobleem
intensification strategy	lokaliseringstrategie
intractable	moeilik-oplosbaar
isolated	geïsoleerd
isomorphic	isomorf
isomorphism	isomorfisme
jackpot	boerpot
joined	verbind
jump sequence	sprong-ry
$k$ -prize	$k$ -prys
length	lengte
linear programming problem	lineêre programmeringsprobleem
loop	lus
lottery	lotery
lottery graph	lotery-grafiek
lottery tree	lotery-soekboom
$L_1(m, n; k)$ -set	$L_1(m, n; k)$ -versameling
$L_\psi(m, n; k)$ -set	$L_\psi(m, n; k)$ -versameling
maximal (vertex) independent set	maksimaal onafhanklike (punt-) versameling
minimal dominating set	minimale dominasieversameling
minimal 100 $\psi$ %-partially dominating set	minimale 100 $\psi$ %-gedeeltelik dominasieversameling
minimum [maximum] degree	minimum [maksimum] graad
minimum dominating set	(minimale) dominasieversameling van minimum kardinaliteit
minimum 100 $\psi$ %-partially dominating set	(minimale) 100 $\psi$ %-gedeeltelike dominasieversameling van minimum kardinaliteit
multigraph	multigrafiek
multiplicity	multiplisiteit
mutate/mutation	muteer/mutasie
nauty graph	nauty-grafiek

ENGLISH	AFRIKAANS
nauty tree	nauty-soekboom
NP-complete	NP-volledig
(open) [(closed)] neighbourhood set	(oop) [(geslote)] buurpuntversameling
neighbouring move	buuroplossing
objective function	doelfunksie
operating system	bedryfstelsel
order	orde
overlapping structure	oorvleuelingstruktuur
packing number	inpakkingsgetal
packing set	inpakkingsversameling
path	pad
parallel edge	parallele lyn
(lower) partial domination number	(laer) gedeeltelike dominasiegetal
playing set	spel-versameling
polynomial time	polinoomtyd
population	populasie
pseudocode	pseudokode
pseudograph	pseudografiek
$\Psi_\ell(m, n; k)$ -set	$\Psi_\ell(m, n; k)$ -versameling
radius	radius
regular	regulier
resource utilised	hulpbron benut
resource utilisation number	hulpbron-benuttingsgetal
size	grootte
space complexity	ruimte-kompleksiteit
spanning subgraph	spangrafiek
spanning tree	spanboom
Steiner system	Steiner-sisteem
strongly regular	sterk regulier
subgraph	deelgrafiek
tabu list	tabu lys
tabu search	tabu soektog
tabu status	tabu status
tabu tenure	tabu tydspan
ticket efficiency parameter	kaartjie-effektieweitiesparameter
time complexity	tyd-kompleksiteit
tractable	maklik-oplosbaar
tree	boom
trial solution	tussentydse oplossing
universal set	universele versameling
vertex	punt
vertex-induced subgraph	punt-geïnduseerde deelgrafiek
vertex set	puntversameling
vertex-transitive	punt-transitief
walk	lynry
winning draw	wentrekking
worst case complexity	ergste-geval kompleksiteit



# List of Reserved Symbols

$A_{\mathcal{G}}$	$p \times p$ adjacency matrix for an order $p$ graph $\mathcal{G}$
$\approx$	the relation $R_1 \approx R_2$ (defined between real numbers) states that the value $R_1$ may be approximated to four decimal places by the value $R_2$
$\beta(\mathcal{G})$	(vertex) independence number of a graph $\mathcal{G}$
$\binom{a}{b}$	binomial coefficient given by $\frac{a!}{b!(a-b)!}$ when $a \geq b$ and by 0 when $a < b$
$C(m, n; k)$	the minimum cardinality of a covering set for the lottery $\langle m, n; k \rangle$ (also called the covering number)
$\mathcal{C}$	a covering set
$\mathbb{C}$	the set of complex numbers
$C_t$	cycle of length $t$
$\deg_{\mathcal{G}}(v)$	degree of a vertex $v$ in a graph $\mathcal{G}$
$\delta(\mathcal{G})$	minimum vertex degree in a graph $\mathcal{G}$
$\Delta(\mathcal{G})$	maximum vertex degree in a graph $\mathcal{G}$
$\text{diam}(\mathcal{G})$	diameter of a graph $\mathcal{G}$
$d_{\mathcal{G}}(u, v)$	distance between a vertex pair $(u, v)$ in a graph $\mathcal{G}$
$E(\mathcal{G})$	edge set of a graph $\mathcal{G}$
$e_{\mathcal{G}}(v)$	eccentricity of a vertex $v$ in a graph $\mathcal{G}$
$\mathcal{E}_{\ell}(m, n; k)$	ticket efficiency parameter for the lottery $\langle m, n; k \rangle$
$\mathcal{L}$	a lottery or playing set
$m$	cardinality of the universal set, $\mathcal{U}_m$ , for the lottery $\langle m, n; k \rangle$ , where $1 \leq k \leq n \leq m$
$\eta_{\psi}(m, n; k)$	number of different (or distinct) $L_{\psi}(m, n; k)$ -set structures for the lottery $\langle m, n; k \rangle$ (also called the incomplete lottery characterisation number)
$n$	cardinality of the subsets in a playing or lottery set for the lottery $\langle m, n; k \rangle$ , where $1 \leq k \leq n \leq m$
$\mathbb{N}$	the set of natural numbers $\{1, 2, 3, \dots\}$
$\mathbb{N}_0$	the set of counting numbers $\mathbb{N} \cup \{0\}$
<b>NP</b>	class of decision problems which may be answered “yes” by a polynomial time algorithm, given additional information (called a certificate to the problem instance at hand)
$\aleph_{\mathcal{L}}$	a <b>n</b> auty graph for the playing set $\mathcal{L}$
$\phi$	isomorphism between two graphs

$\Phi(\mathcal{A}, n)$	the set of all $n$ -subsets from an unordered set $\mathcal{A}$ , so that $ \Phi(\mathcal{A}, n)  = \binom{ \mathcal{A} }{n}$
$f_{\mathcal{F}}(i)$	the frequency of occurrence (number of occurrences) of the element $i$ in the list $\mathcal{F}$
$\gamma(\mathcal{G})$	(lower) domination number of a graph $\mathcal{G}$
$g(\mathcal{G})$	girth of a graph $\mathcal{G}$
$\langle m, n; k \rangle$	a lottery where players select $n$ -subsets from a universal $m$ -set, winning a prize if a player has chosen at least one $n$ -subset sharing a $k$ -subset from the universal set with an arbitrarily chosen winning $n$ -subset
$G\langle m, n; k \rangle$	the lottery graph for the lottery $\langle m, n; k \rangle$
$k$	minimum cardinality of a subset of an $n$ -set that is required to match with an arbitrarily chosen winning $n$ -set in order for the player to win a prize
$K_t$	complete graph on $t$ vertices
$\ell$	cardinality of a lottery or playing set
$L_{\psi}(m, n; k)$	the incomplete lottery number for the lottery $\langle m, n; k \rangle$ (in the special case where $\psi = 1$ , the incomplete lottery number is referred to as the complete lottery number)
$\min\{\mathcal{F}\}$ [ $\max\{\mathcal{F}\}$ ]	the minimum [maximum] value of all the elements in a set $\mathcal{F} = \{f_1, \dots, f_{ \mathcal{F} }\}$
$\mathcal{N}(\mathcal{L})$	a neighbourhood of candidate solutions of a specific lottery or playing set $\mathcal{L}$ in the Tabu search algorithm implementation for determining lottery or playing sets
$N_{\mathcal{G}}[v]$ [ $N_{\mathcal{G}}(v)$ ]	closed [open] neighbourhood set associated with a vertex $v$ in a graph $\mathcal{G}$
$\mathcal{O}(g(n))$	a function $f(n)$ grows no faster than $g(n)$ as $n \rightarrow \infty$ (denoted $f(n) = \mathcal{O}(g(n))$ ), if there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
$\Omega(g(n))$	a function $f(n)$ is said to grow asymptotically at least as fast as $g(n)$ as $n \rightarrow \infty$ (denoted $f(n) = \Omega(g(n))$ ), if there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$
$p$	order of a graph
$P(m, n; k)$	the maximum cardinality of a packing set for the lottery $\langle m, n; k \rangle$ (also called the packing number)
<b>P</b>	class of decision problems that may be solved by polynomial time algorithms
$\mathcal{P}$	a packing set
$\Psi_{\ell}(m, n; k)$	resource utilisation number of the lottery $\langle m, n; k \rangle$ , when playing $\ell$ $n$ -sets from $\mathcal{U}_m$
$q$	size of a graph
$\text{rad}(\mathcal{G})$	radius of a graph $\mathcal{G}$
$r$	degree of regularity of the lottery graph $G\langle m, n; k \rangle$
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of positive real numbers
$\text{RU}(\mathcal{L})$	the resource utilisation of a lottery or playing set $\mathcal{L}$
$\simeq$	the relation $\mathcal{G}_1 \simeq \mathcal{G}_2$ (defined between graphs) states that a graph $\mathcal{G}_1$ is isomorphic to a graph $\mathcal{G}_2$
$\preceq$	the relation $L_1 \preceq L_2$ (defined between decision problems) states that an algorithm exists that solves a decision problem $L_1$ as a subroutine of an algorithm that solves a decision problem $L_2$ . Simplistically this means that the problem $L_1$ is no harder to solve than the problem $L_2$



---

$\subseteq$	the relation $\mathcal{H}_1 \subseteq \mathcal{H}_2$ (defined between sets) states that the set $\mathcal{H}_1$ is a subset of (or equal to) the set $\mathcal{H}_2$ and the same relation (defined between graphs) states that the graph $\mathcal{H}_1$ is a subgraph of the graph $\mathcal{H}_2$
$\mathcal{U}_m$	universal set consisting of the integers $\{1, \dots, m\}$ in the lottery $\langle m, n; k \rangle$ , where $1 \leq k \leq n \leq m$
$V(\mathcal{G})$	vertex set of a graph $\mathcal{G}$

# Chapter 1

## Introduction

**lōt**, *n.* One of a set of objects used to secure a chance decision in dividing goods, selecting officials, *etc.*

**lōtt'erỹ**, *n.* Arrangement for distributing prizes by chance among purchasers of tickets; *~–wheel*, wheel with box used for shuffling numbers corresponding to those tickets.

**lōtt'ō**, *n.* Game of chance with drawing of numbers as in lottery.

*The Concise Oxford Dictionary* (1950) [157]

### 1.1 Historical background

Recorded use of lotteries dates back a number of centuries, even to biblical times [221]. Land division was mostly achieved by the casting of lots, as mentioned a number of times in the Bible [24]. Also, recall how the possession of Jesus' garment was gambled for during His crucifixion<sup>1</sup>. Certain Roman emperors (Augustus Caesar B.C. 12 – 14 A.D., Nero 54 – 68 A.D. and Heliogabalus 218 – 222 A.D.) instituted a form of lottery called door-prize drawings as feature entertainment for their imperial gatherings, where guests drew for prizes (which included slaves, fashionable villas, *etc.*).

Apart from Augustus Caesar, who sponsored the first known public lottery in order to raise funds to repair the city of Rome, records indicate a similar kind of commercial lottery, dating back to 1420 in the Burgundian town of L'Ecluse, which was used to raise money to strengthen the town's fortifications. The Low Countries (the Netherlands, Belgium and Luxembourg) had similar lotteries at this time and used them as a means of selling land, livestock, works of art and other commodities. Even council and government members were chosen by lot, using numbered balls drawn from an urn [29].

In contrast to the initial main focus of fund raising lotteries for various public projects, the first known public lottery paying money prizes was *La Lotto de Firenze*, which began during the Renaissance (in 1530) in Florence, Italy [29]. By 1643, the Genoese republic institutionalised lotteries (drawing a selection of numbered balls from an urn without replacement), with similar lotteries being established during 1665 in Milan, Naples and Venice. This custom soon also spread through the whole of Italy and grew into the Italian national lottery, which was launched in 1870. In 1533 France was also introduced to the use of lotteries by the Italians (references indicate that lotteries were held at the court of Louis XIV during the late 17th century). Soon after the institution of lotteries in Italy, Britain also initiated her first lotteries, starting with royal lotteries, but later also including the public. The first government-sponsored lottery in England (with a first prize of £5 000) was announced by Queen Elizabeth in 1566, raising funds to

---

<sup>1</sup>“And they crucified Him, and parted his garments, casting **lots**: that it might be fulfilled which was spoken by the prophet. They parted My garments among them, and upon My vesture did they cast lots.” (Matthew 27:35), King James Version [24].

be used “toward the reparation of the harbours, strength of the Realme and other public good works” [221]. Even the British colonisation of America was partly funded by royal lotteries in 1612, raising roughly £29 000 for settlers of the Virginia Company. With the explosion of lotteries thereafter, the British Parliament abolished public lotteries in 1699 and again in 1721, after lotteries were re-permitted in 1710.

American lotteries were founded as early as 1665 in New Amsterdam. Many prominent citizens sponsored lotteries to raise funds for the improvement of various public facilities. These included Benjamin Franklin, who bought arms for the defense of Philadelphia with the proceeds; John Hancock (1762), who rebuilt Boston’s Faneuil Hall and George Washington (1768), who built a road over the Cumberland Mountains. These first American lotteries were referred to as a form of voluntary taxation.

The first lotteries in Japan are said to date from the 1630s [120]. The Japanese went through a series of bannings and revivals of lottery schemes until 1842, when lotteries were banned completely. This lasted until shortly before the end of the Second World War, in 1945, when lotteries were revived to obtain funds for the war effort. In October 1945, immediately after the war, the Japanese government began selling lottery tickets under the name *Takara-kujū*, meaning “fortune” or “treasure” lottery. The government’s aims then were to soak up idle capital in order to contain rampant inflation and to procure funds for post-war reconstruction.

Lotteries have since been utilised for a variety of purposes other than the above mentioned reasons, most of which are closely monitored and run by government mandated public companies, although some private lotteries do exist. The manner in which draws for lotteries are performed today also varies (from drawing specific numbered tickets to drawing numbers from a container), although the notion of drawing a selection of numbered balls without replacement from an urn seems to have remained the favourite for more than three centuries. In comparison to centuries ago, lottery winning prize payouts have increased dramatically (see, for example, the winning prize pool of over £1 Billion for the Spanish *El Gordo* lottery in December 2002 [73]). The frequency of winning draws has also been increased by some authorities to include daily draws and facilities for buying lottery tickets over the Internet.

In summary, government and private lotteries are widespread and highly functional today, supporting numerous welfare institutions, charitable groups and various public demands. The vast majority of lotteries around the world have conformed to the practice of drawing numbers from a prespecified set (without replacement). In most lotteries around the world today, players of the lottery draw 6 numbers on a ticket, from a set of  $m$  numbers and are awarded a prize if they have drawn at least  $k \geq 3$  numbers in common with the set of 6 winning numbers, drawn at random by the governing body. A comprehensive survey of lottery parameters in use across the world today is given in Table 1.1.

## 1.2 Problem descriptions

Suppose a participant of a lottery scheme wishes to play in a manner so as to ensure that he/she wins a prize. Let the lottery  $\langle m, n; k \rangle$  consist of randomly selecting a winning  $n$ -set  $w$  from the universal set  $\mathcal{U}_m = \{1, 2, \dots, m\}$ , while a player participates in the scheme by purchasing a playing set  $\mathcal{L}$  of any number of  $n$ -sets (informally also called tickets) from  $\mathcal{U}_m$  prior to the winning draw and is awarded a prize if at least  $k$  elements of  $w$  match those of at least one of the participant’s  $n$ -sets (tickets) in the playing set  $\mathcal{L}$ . The participant may wish to select  $n$ -sets in order to construct his/her playing set  $\mathcal{L}$  in such a way that there will necessarily be at least one  $n$ -set in  $\mathcal{L}$  which contains at least  $k$  elements of  $\mathcal{U}_m$  matching those of  $w$ , no matter which winning  $n$ -set  $w$  is chosen from  $\mathcal{U}_m$ . Such a playing set is usually called a lottery set. We shall refer to such a lottery set as a complete lottery set. With this objective in mind, the participant may well wonder what the cardinality of such a *smallest* possible complete lottery set  $\mathcal{L}$  might be and how to go about constructing a complete lottery set of smallest cardinality.

In order to make the above description more precise, let  $\Phi(\mathcal{A}, n)$  denote the set of all (unordered)  $n$ -sets from a set  $\mathcal{A}$ , so that  $|\Phi(\mathcal{A}, n)| = \binom{|\mathcal{A}|}{n}$ . The following well-known problem is a special case of a more general problem to be considered in this dissertation.

State or Country	Lottery
Malaysia [2]	$\langle 10, n; k \rangle^a$
Yugoslavia [179]	$\langle 24, 12; k \rangle$
West Virginia [266]	$\langle 25, 6; k \rangle$
Chile [5], Venezuela [241]	$\langle 25, 15; k \rangle$
Illinois [111], Lithuania [9]	$\langle 30, 5; k \rangle$
Iowa [112]	$\langle 30, 6; k \rangle$
Chile [5]	$\langle 30, 7; k \rangle$
Wisconsin [269]	$\langle 31, 5; k \rangle$
Colorado [54], Kansas [124]	$\langle 32, 5; k \rangle$
District of Columbia [58], Michigan [163]	$\langle 33, 5; k \rangle$
New Mexico [186], Turkey [164], Virginia [262]	$\langle 34, 5; k \rangle$
Norway [191]	$\langle 34, 7; k \rangle$
Arizona [7], Connecticut [51], Latvia [227], Massachusetts [154], Slovak Republic [245], Slovenia [232], South Dakota [233]	$\langle 35, 5; k \rangle$
Kansas [124]	$\langle 35, 6; k \rangle$
Hungary [230], Sweden [1]	$\langle 35, 7; k \rangle$
Florida [77], Indiana [106], Kazakhstan [182], Yugoslavia [179]	$\langle 36, 5; k \rangle$
Maine [211], New Hampshire [184], Vermont [261], Wisconsin [269]	$\langle 36, 6; k \rangle$
China [48], Denmark [53]	$\langle 36, 7; k \rangle$
Montana [176], Ohio [196], Texas [84]	$\langle 37, 5; k \rangle$
Jamaica [118]	$\langle 37, 6; k \rangle$
Iceland [116], Nebraska [183]	$\langle 38, 5; k \rangle$
Australia [192], Delaware [215]	$\langle 38, 6; k \rangle$
California [121], Georgia [87], Iowa [112], Malta [205], Maryland [150], Montana [176], New York [187], Pennsylvania [69], South Dakota [233]	$\langle 39, 5; k \rangle$
District of Columbia [58]	$\langle 39, 6; k \rangle$
Croatia [108]	$\langle 39, 7; k \rangle$
Czech Republic [220], New Jersey [185]	$\langle 40, 5; k \rangle$
Ghana, Kazakhstan [182], Louisiana [146], New Zealand [195], Perú [243]	$\langle 40, 6; k \rangle$
Arizona [7]	$\langle 41, 6; k \rangle$
Minnesota [173]	$\langle 42, 5; k \rangle$
Belgium [18], Colorado [54], Ireland [6, 55], Maine [211], Malaysia [2], Massachusetts [154], New Hampshire [184], Philippines [201], Puerto Rico [140], Taiwan [242], Vermont [261]	$\langle 42, 6; k \rangle$
Japan [120]	$\langle 43, 6; k \rangle$
Missouri [175], Portugal [80], Texas [84], Uruguay [67]	$\langle 44, 5; k \rangle$
Australia [192], Connecticut [51], Missouri [175]	$\langle 44, 6; k \rangle$
Argentina [70], Australia [93, 143, 144], Austria [200], Croatia [108], Hungary [230], Israel [117], Netherlands [63], Perú [243], Philippines [201], Singapore [228], Switzerland [240], Ukraine [115], Yugoslavia [179]	$\langle 45, 6; k \rangle$
California [121]	$\langle 47, 5; k \rangle$
Hong Kong [86]	$\langle 47, 6; k \rangle$
British Columbia [32], Québec [142], Western Canada [267]	$\langle 47, 7; k \rangle$
Denmark [53], Finland [260], Indiana [106], Oregon [198]	$\langle 48, 6; k \rangle$
Malaysia [2]	$\langle 49, 4; k \rangle$
Alberta [4], British Columbia [32], Colorado [54], Connecticut [51], Delaware [215], France [47], Georgia [87], Germany [222], Greece [109], Idaho [110], Iowa [112], Kansas [124], Kentucky [126], Louisiana [146], Malaysia [2], Manitoba [149], Maryland [150], Massachusetts [154], Minnesota [173], Missouri [175], Montana [176], New Hampshire [184], New Jersey [185], New Mexico [186], Ohio [196], Philippines [201], Poland [255], Québec [142], Rhode Island [209], Slovak Republic [245], South Africa [181], South Dakota [233], Spain [141], Turkey [164], United Kingdom [40], Virginia [262], Washington [271], Western Canada [267], Wisconsin [269]	$\langle 49, 6; k \rangle^b$
Georgia [87], Illinois [111], Maryland [150], Massachusetts [154], Michigan [163], New Jersey [185], Virginia [262]	$\langle 50, 6; k \rangle^c$

Table 1.1: Typical parameters for lotteries around the world, gathered from an extensive Internet survey (see [113] for a collection of lottery websites). Some variations to the complete lottery problem (as defined in Definition 1.1) exist and are: <sup>a</sup>This lottery allows players to select  $n$ -sets ( $n = 4, 5, 6$ ) containing duplicate numbers from  $\mathcal{U}_{10}$ ; <sup>b</sup>In various of these US states the lottery players select 5 numbers from  $\mathcal{U}_{49}$  and a single Powerball number from  $\mathcal{U}_{42}$ . These lotteries are referred to as the Powerball Lotteries; <sup>c</sup>In these US states the player of the lottery selects 5 numbers from  $\mathcal{U}_{50}$  and a single Big Game number from  $\mathcal{U}_{36}$ . These lotteries are referred to as the Big Game Lottery.

State or Country	Lottery
Michigan [163]	$\langle 51, 6; k \rangle$
Maryland [150], Massachusetts [154], Ohio [196], Texas [84], Virginia [262]	$\langle 52, 5; k \rangle$
Georgia [87], Illinois [111]	$\langle 52, 6; k \rangle$
Iowa [112], Kansas [124], Missouri [175], Montana [176], New Mexico [186], South Dakota [233], Wisconsin [269]	$\langle 53, 5; k \rangle$
Florida [77]	$\langle 53, 6; k \rangle$
India [188], Texas [84]	$\langle 54, 6; k \rangle$
Delaware [215], Minnesota [173], Montana [176], Nebraska [183], New Hampshire [184], South Dakota [233], West Virginia [266]	$\langle 55, 5; k \rangle$
New York [187]	$\langle 59, 6; k \rangle$
Pennsylvania [69]	$\langle 69, 6; k \rangle$
Canada [11]	$\langle 70, 10; k \rangle^d$
Maryland [150]	$\langle 80, 10; k \rangle^d$
Hungary [230], Italy [145], Nigeria [50]	$\langle 90, 5; k \rangle$
Argentina [70], Brazil [64]	$\langle 100, 20; k \rangle$

Table 1.1 (continued): Typical parameters for lotteries around the world, gathered from an extensive Internet survey (see [113] for a collection of lottery websites). Some variations to the complete lottery problem (as defined in Definition 1.1) exist and are: <sup>d</sup>These lotteries conform to the quadruple  $\langle m, n, 20; k \rangle$ , as described in footnote 3.

**Definition 1.1 (The complete lottery problem)** Define a playing set  $\mathcal{L}$  for the lottery  $\langle m, n; k \rangle$  as a set  $\mathcal{L} \subseteq \Phi(\mathcal{U}_m, n)$ . Then a complete lottery set for the lottery  $\langle m, n; k \rangle$  is a playing set with the property that, for any element  $\phi_n \in \Phi(\mathcal{U}_m, n)$ , there exists an element  $l \in \mathcal{L}$  such that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$ . The complete lottery problem is: What is the smallest possible cardinality of a complete lottery set  $\mathcal{L}$  for  $\langle m, n; k \rangle$ ? Denote the answer to this question by the complete lottery number  $L_1(m, n; k)$ <sup>3</sup>. ■

A complete lottery set  $\mathcal{L}$  of minimum cardinality  $L_1(m, n; k)$  will be referred to as an  $L_1(m, n; k)$ -set for the lottery  $\langle m, n; k \rangle$ .

The complete lottery problem is certainly of interest from a *combinatorial* point of view and has been studied extensively since 1964, as will be apparent later in the survey of literature. But perhaps the following two generalisations of the complete lottery problem are of more interest from a *practical* point of view. Suppose a participant of the lottery scheme  $\langle m, n; k \rangle$  wishes to construct a playing set  $\mathcal{L}_\psi$  in such a manner that at least a certain proportion,  $100\psi\%$  say, of the possible  $\binom{m}{n}$   $n$ -sets from  $\mathcal{U}_m$  share a common  $k$ -subset with (*some*) elements in  $\mathcal{L}_\psi$ . In this case, he/she would be at least  $100\psi\%$  sure of winning a  $k$ -prize. Let us call such a playing set  $\mathcal{L}_\psi$  a  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m, n; k \rangle$ . With this objective in mind, the participant may well wonder what the cardinality of a smallest  $100(1 - \psi)\%$ -incomplete lottery set might be and how to go about constructing such a set of smallest cardinality. This leads to the following novel problem definition.

**Definition 1.2 (The incomplete lottery problem)** Define a  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m, n; k \rangle$  as a subset  $\mathcal{L}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  with the property that there exists some subset  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\lceil \psi \binom{m}{n} \rceil$  such that, for any element  $\phi_n \in \mathcal{V}_\psi$ , it holds that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$  for some  $l \in \mathcal{L}_\psi$ . The incomplete lottery problem is: What is the smallest possible cardinality of a  $100(1 - \psi)\%$ -incomplete lottery set  $\mathcal{L}_\psi$ ? Denote the answer to this question by the incomplete lottery number  $L_\psi(m, n; k)$ . ■

<sup>2</sup>In the combinatorial literature this is often referred to as a **block design** [45, 265], defined by the 6-tuple  $(v, l, t, s, \lambda, b)$  (other parameter definitions also exist [17, 33, 134]). Such a design consists of a carefully selected combinatorial subset of  $b$   $l$ -sets from  $v$  elements (also called varieties). It ensures the existence of a minimum of at least  $\lambda$   $l$ -sets containing a  $t$ -subset match with any  $s$ -set drawn from  $v$  elements. In terms of the lottery parameters, this translates to the 6-tuple  $(m, n, k, n, 1, L_1(m, n; k))$ .

<sup>3</sup>In the construction of a so-called *wheel* or *reduced system*, some authors allow the governing body of the lottery scheme to draw a different number of elements,  $t$ , from  $\mathcal{U}_m$  to form a winning  $t$ -set,  $w$ , while players of the lottery scheme construct playing sets consisting of  $n$ -sets from  $\mathcal{U}_m$ , with  $n \neq t$  in general, thereby instead defining the lottery by the quadruple  $\langle m, n, t; k \rangle$ . Although this more general definition of a lottery scheme sometimes allows convenient upper bound constructions for minimum cardinality complete lottery sets, it will be ignored in this dissertation.

A  $100(1 - \psi)\%$ -incomplete lottery set  $\mathcal{L}_\psi$  of minimum cardinality  $L_\psi(m, n; k)$  will be referred to as an  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$ . In the case where  $\psi = 1$  in the above definition, the incomplete lottery problem reduces to the well-known complete lottery problem in Definition 1.1.

Conversely, suppose a participant of a lottery scheme wishes to play in a manner so as to maximise his/her chance of winning a  $k$ -prize in the lottery  $\langle m, n; k \rangle$  by constructing a playing set of fixed cardinality less than or equal to  $L_1(m, n; k)$ . Using the same notation as above, let  $N[v]$  denote the set of all  $n$ -sets from  $\mathcal{U}_m$  having at least one  $k$ -subset in common with  $v \in \Phi(\mathcal{U}_m, n)$  (including  $v$  itself), that is,

$$N[v] = \bigcup_{\substack{\varphi \in \Phi(\mathcal{U}_m, n) \\ \Phi(\varphi, k) \cap \Phi(v, k) \neq \emptyset}} \{\varphi\}$$

where  $1 \leq k \leq n \leq m$ . Then this alternative problem may be defined as follows.

**Definition 1.3 (The resource utilisation problem)** *The resource utilisation of a playing set  $\mathcal{L} = \{v_1, \dots, v_\ell\} \subseteq \Phi(\mathcal{U}_m, n)$  of  $\langle m, n; k \rangle$  is defined as the proportion  $|\bigcup_{i=1}^\ell N[v_i]| / \binom{m}{n}$ , and the resource utilisation problem is: Given a fixed playing set cardinality  $\ell$  ( $1 \leq \ell \leq L_1(m, n; k)$ ), what is the maximum resource utilisation that may be achieved? Denote the answer to this problem by the resource utilisation number*

$$\Psi_\ell(m, n; k) = \max_{v_1, \dots, v_\ell \in \Phi(\mathcal{U}_m, n)} \frac{\left| \bigcup_{i=1}^\ell N[v_i] \right|}{\binom{m}{n}}. \quad \blacksquare$$

A playing set of cardinality  $\ell$  that realises the maximum resource utilisation of  $\Psi_\ell(m, n; k)$  will be referred to as a  $\Psi_\ell(m, n; k)$ -set for  $\langle m, n; k \rangle$ . The following example is presented to clarify the newly defined incomplete lottery and resource utilisation problems, which are the two problems to be studied in this dissertation.

**Example 1.1** *Consider the small lottery  $\langle 7, 3; 2 \rangle$ . A player constructs a playing set  $\mathcal{L}$  by selecting 3-sets from  $\mathcal{U}_7 = \{1, \dots, 7\}$  and wins a prize if at least one 3-set has a 2-subset match with the winning 3-set. In order to guarantee that every possible 2-set from  $\mathcal{U}_7$  is in the player's playing set, the player may play the complete lottery set  $\bar{\mathcal{L}} = \{\{1, 2, 3\}, \{1, 4, 5\}, \{1, 6, 7\}, \{2, 4, 6\}, \{2, 5, 7\}, \{3, 4, 7\}, \{3, 5, 6\}\}$  (the reader may verify that, in fact, every 2-subset of  $\mathcal{U}_7$  is contained exactly once in (some element of)  $\bar{\mathcal{L}}$ ) and suppose the winning 3-set is  $w = \{2, 4, 7\}$ . There exists an element  $\bar{\varphi} = \{2, 4, 6\} \in \bar{\mathcal{L}}$  such that  $\Phi(w, 2) \cap \Phi(\bar{\varphi}, 2) = \{2, 4\} \neq \emptyset$ . This is true for any winning 3-set  $w$ , implying that  $L_1(7, 3; 2) \leq 7$ . A different, smaller complete lottery set  $\tilde{\mathcal{L}} = \{\{1, 2, 3\}, \{1, 5, 7\}, \{2, 5, 7\}, \{3, 4, 6\}\}$  would yield a similar result, regardless of the winning 3-set (for  $\tilde{\varphi} = \{2, 5, 7\} \in \tilde{\mathcal{L}}$ ,  $\Phi(w, 2) \cap \Phi(\tilde{\varphi}, 2) = \{2, 7\} \neq \emptyset$ , for example), establishing an improved upper bound of  $L_1(7, 3; 2) \leq 4$ . The reader may verify exhaustively that none of the possible  $\binom{35}{3}$  3-set combinations from  $\Phi(\mathcal{U}_7, 3)$  yield a complete lottery set. For example,  $\check{\mathcal{L}} = \{\{1, 3, 4\}, \{2, 5, 6\}, \{3, 5, 7\}\}$  is not a complete lottery set for  $\langle 7, 3; 2 \rangle$ , since  $\Phi(w, 2) \cap \Phi(\check{\varphi}, 2) = \emptyset$  for all  $\check{\varphi} \in \check{\mathcal{L}}$ . This yields the lower bound  $L_1(7, 3; 2) > 3$ . Hence the complete lottery set  $\tilde{\mathcal{L}}$  is, in fact, an  $L_1(7, 3; 2)$ -set for  $\langle 7, 3; 2 \rangle$ , with cardinality  $L_1(7, 3; 2) = 4$ .*

Now suppose that a participant of the lottery is not interested in a guarantee (100% assurance) of winning a 2-prize, but will instead settle for a lesser assurance of at least (say) 90%. One incomplete lottery set that would achieve this goal is given by  $\check{\mathcal{L}} = \{\{1, 4, 5\}, \{2, 4, 5\}, \{3, 6, 7\}\}$ . This is true because  $|N[\{1, 4, 5\}] \cup N[\{2, 4, 5\}] \cup N[\{3, 6, 7\}]| / \binom{7}{3} = \frac{32}{35} \geq 0.9$  (where  $N[\{1, 4, 5\}] = \{\{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{1, 4, 5\}, \{1, 4, 6\}, \{1, 4, 7\}, \{1, 5, 6\}, \{1, 5, 7\}, \{2, 4, 5\}, \{3, 4, 5\}, \{4, 5, 6\}, \{4, 5, 7\}\}$ ,  $N[\{2, 4, 5\}] = \{\{1, 2, 4\}, \{1, 2, 5\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{2, 4, 5\}, \{2, 4, 6\}, \{2, 4, 7\}, \{2, 5, 6\}, \{2, 5, 7\}, \{3, 4, 5\}, \{4, 5, 6\}, \{4, 5, 7\}\}$  and  $N[\{3, 6, 7\}] = \{\{1, 3, 6\}, \{1, 3, 7\}, \{1, 6, 7\}, \{2, 3, 6\}, \{2, 3, 7\}, \{2, 6, 7\}, \{3, 4, 6\}, \{3, 4, 7\}, \{3, 5, 6\}, \{3, 5, 7\}, \{3, 6, 7\}, \{4, 6, 7\}, \{5, 6, 7\}\}$ ), implying that  $L_{0.9}(7, 3; 2) \leq 3$ . Although other 3-sets from  $\Phi(\mathcal{U}_7, 3)$  that yield a similar result exist, no set of 2 elements from  $\Phi(\mathcal{U}_7, 3)$  provides a probability of winning (a 2-prize) in excess of 90%. We therefore conclude that  $L_{0.9}(7, 3; 2) > 2$  and have established that  $L_{0.9}(7, 3; 2) = 3$ . In fact,  $\check{\mathcal{L}}$  is also an  $L_{\frac{32}{35}}(7, 3; 2)$ -set for the lottery  $\langle 7, 3; 2 \rangle$ .

Now consider the playing set  $\hat{\mathcal{L}} = \{\{1, 5, 7\}, \{3, 4, 6\}\}$  of cardinality  $\ell = 2 < L_1(7, 3; 2)$ . Since we know that  $N[\{1, 5, 7\}] = \{\{1, 2, 5\}, \{1, 2, 7\}, \{1, 3, 5\}, \{1, 3, 7\}, \{1, 4, 5\}, \{1, 4, 7\}, \{1, 5, 6\}, \{1, 5, 7\}, \{1, 6, 7\},$

$\{2, 5, 7\}, \{3, 5, 7\}, \{4, 5, 7\}, \{5, 6, 7\}\}$  and  $N[\{3, 4, 6\}] = \{\{1, 3, 4\}, \{1, 3, 6\}, \{1, 4, 6\}, \{2, 3, 4\}, \{2, 3, 6\}, \{2, 4, 6\}, \{3, 4, 5\}, \{3, 4, 6\}, \{3, 4, 7\}, \{3, 5, 6\}, \{3, 6, 7\}, \{4, 5, 6\}, \{4, 6, 7\}\}$ , the resource utilisation of  $\widehat{\mathcal{L}}$  is  $\frac{26}{35}$ . This yields the lower bound  $\Psi_2(7, 3; 2) \geq \frac{26}{35}$ . The reader may again verify exhaustively, by considering all  $\binom{35}{2}$  possible 3-set combinations of cardinality 2 from  $\Phi(\mathcal{U}_7, 3)$ , that  $\Psi_2(7, 3; 2) \leq \frac{26}{35}$ . Hence, we have established that  $\widehat{\mathcal{L}}$  is indeed a  $\Psi_2(7, 3; 2)$ -set for the lottery  $\langle 7, 3; 2 \rangle$ , implying that  $\Psi_2(7, 3; 2) = \frac{26}{35} \approx 74.2857\%$ . Note that this of course implies that  $L_{\frac{26}{35}}(7, 3; 2) \leq 2$ . It is possible to show that  $\Psi_1(7, 3; 2) = \frac{13}{35}$ , leading to a confirmation that  $\widehat{\mathcal{L}}$  is also an  $L_{\frac{26}{35}}(7, 3; 2)$ -set for  $\langle 7, 3; 2 \rangle$  and consequently that  $L_{\frac{26}{35}}(7, 3; 2) = 2$ . ■

From Definitions 1.2 and 1.3 it follows that the incomplete lottery problem and resource utilisation problem are inverses of each other, in the sense of the following proposition.

**Proposition 1.1** *Let  $0 < \psi \leq 1$  be a real number and let  $\ell$  be any natural number. Then  $L_\psi(m, n; k) \leq \ell$  if and only if  $\Psi_\ell(m, n; k) \geq \psi$ , for all  $1 \leq k \leq n \leq m$ .*

**Proof**

Suppose  $L_\psi(m, n; k) \leq \ell$  for some real number  $0 < \psi \leq 1$  and some  $\ell \in \mathbb{N}$ . This implies that there exists a playing set  $\mathcal{L}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  with the property that some set  $\mathcal{U}_\psi^* \in \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\psi \binom{m}{n}$  exists such that, for any element  $\phi_n \in \mathcal{U}_\psi^*$ , it holds that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$  for at least one  $l \in \mathcal{L}_\psi$  (according to Definition 1.2). Hence, there exists a set of cardinality  $\ell$  yielding a resource utilisation of at least  $|\mathcal{U}_\psi^*| = (\psi \binom{m}{n}) / \binom{m}{n} = \psi$ , implying that  $\Psi_\ell(m, n; k) \geq \psi$ .

Conversely, suppose  $\Psi_\ell(m, n; k) \geq \psi$  for some real number  $0 < \psi \leq 1$  and some  $\ell \in \mathbb{N}$ . This implies that there exists a playing set  $\mathcal{L}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality  $\ell$  yielding a resource utilisation of  $\psi$ . Hence there exists a set  $\mathcal{U}_\psi^*$  of cardinality at least  $\psi \binom{m}{n}$  such that, for each  $\phi_n \in \mathcal{U}_\psi^*$ , it holds that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$  for some  $l \in \mathcal{L}_\psi$ . Hence,  $L_\psi(m, n; k) \leq \ell$ . ■

Two different, well-studied combinatorial problems closely related to the complete lottery problem, are the so-called packing and covering problems. Although further investigations into the determination of covering and packing numbers will not be considered in this dissertation, these problem definitions are presented to clarify their distinction from the complete lottery problem in Definition 1.1, and to obtain upper bounds for complete lottery numbers, as will be demonstrated later.

**Definition 1.4 (The packing problem)** *Define a packing set for  $\langle m, n; k \rangle$  as a subset  $\mathcal{P} \subseteq \Phi(\mathcal{U}_m, n)$  with the property that, for any elements  $p_1, p_2 \in \mathcal{P}$ , it holds that  $\Phi(p_1, k) \cap \Phi(p_2, k) = \emptyset$ . Then the packing problem is: what is the largest possible cardinality of a packing set  $\mathcal{P}$  for  $\langle m, n; k \rangle$ ? Denote the answer to this question by the packing number  $P(m, n; k)$ .* ■

**Definition 1.5 (The covering problem)** *Define a covering set for  $\langle m, n; k \rangle$  as a subset  $\mathcal{C} \subseteq \Phi(\mathcal{U}_m, n)$  with the property that, for any element  $\phi_k \in \Phi(\mathcal{U}_m, k)$ , there exists an element  $c \in \mathcal{C}$  such that  $\{\phi_k\} \cap \Phi(c, k) \neq \emptyset$ . Then the covering problem is: what is the smallest possible cardinality of a covering set  $\mathcal{C}$  for  $\langle m, n; k \rangle$ ? Denote the answer to this question by the covering number  $C(m, n; k)$ .* ■

The subtle difference between the complete lottery and covering problems given in Definitions 1.1 and 1.5 may easily be overlooked (see, for instance, one incorrect solution to the complete lottery problem posed by Jans [119], where the proposed solution determined the *maximal* number of different combinations of  $n$ -sets with less than  $n-k$  correspondences for a *fixed*  $n$ -set, yielding a poor upper bound for the complete lottery number  $L_1(m, n; k)$ , which is the solution to a *minimisation problem*). Note that, although in both the complete lottery and covering problems minimum cardinality sets consisting of  $n$ -sets from  $\mathcal{U}_m$  are sought, the difference is that in the covering problem we wish to have “covered” *all*  $k$ -subsets from  $\mathcal{U}_m$ , whilst in the complete lottery problem the weaker requirement is made that we need only “cover” *at least one*  $k$ -subset in the winning  $n$ -set  $w \in \Phi(\mathcal{U}_m, n)$ , no matter what  $w$  might be. For clarity, Example 1.1 is continued with a discussion of both packing and coverings sets.

**Example 1.2 (continuation of Example 1.1)** Consider again the lottery  $\langle 7, 3; 2 \rangle$  of Example 1.1. The first complete lottery set given in Example 1.1 as  $\bar{\mathcal{L}} = \{\{1, 2, 3\}, \{1, 4, 5\}, \{1, 6, 7\}, \{2, 4, 6\}, \{2, 5, 7\}, \{3, 4, 7\}, \{3, 5, 6\}\}$  is, in fact, a covering set for  $\langle 7, 3; 2 \rangle$  due to the fact that every pair from  $\mathcal{U}_7$  is contained in (exactly) one element of the set  $\bar{\mathcal{L}}$ , implying that  $C(7, 3; 2) \leq 7$ . Minimality of this set may be verified by noting that every number  $i \in \{1, 2, \dots, 7\}$  must occur in at least three different 3-sets (in order to pair with every one of the six other numbers), giving a minimum of 21 occurrences of the numbers  $\{1, 2, \dots, 7\}$  in the elements of the covering set. This implies the existence of at least seven elements in any covering set, or  $C(7, 3; 2) \geq 7$  and hence  $C(7, 3; 2) = 7$ . This covering set  $\bar{\mathcal{L}}$  is indeed also a packing set for  $\langle 7, 3; 2 \rangle$  (using a similar argument as above, 24 occurrences of the numbers  $\{1, 2, \dots, 7\}$  will necessarily force two elements in  $\bar{\mathcal{L}}$  to share a common pair, implying that  $P(7, 3; 2) < 8$ ). Hence  $P(7, 3; 2) = 7$ . It rarely happens, for a 3-tuple  $\langle m, n; k \rangle$ , that  $C(m, n; k) = P(m, n; k)$ , as is the case here. If this happens,  $\langle m, n; k \rangle$  is called a **Steiner system**<sup>4</sup>. Hence  $\langle 7, 3; 2 \rangle$  is a Steiner system. Note that  $4 = L_1(7, 3; 2) < C(7, 3; 2) = P(7, 3; 2) = 7$ . ■

## 1.3 Literature on the lottery, packing and covering problems

The author could only find references in the combinatorial literature to studies performed on the complete lottery problem (yielding bounds and exact values on  $L_1(m, n; k)$ ) dating back as early as 1964. No literature has been encountered on the incomplete lottery and resource utilisation problems. The author believes that these problem definitions constitute wholly novel contributions of this dissertation.

The first subsection, §1.3.1 of this section, contains a survey of the only literature found on the complete lottery problem. This is followed, in §1.3.2, by a similar survey of literature on the two related combinatorial problems, the packing and covering problems. The latter survey is by no means exhaustive due to the age of and widespread interest in the covering and packing problems, relative to that of the complete lottery problem.

### 1.3.1 Complete lottery numbers

The earliest work encountered by the author in the combinatorial literature on the complete lottery problem is due to Hanani, *et al.* [99] in 1964, followed by the PhD dissertation in 1978 of Bate [13], who proved a formula for the class of complete lottery numbers  $L_1(m, n; 1)$ , presented later in this dissertation. Sterboul [236] established general lower bounds on  $L_1(m, n; k)$  in 1978. Further complete lottery number formulas for lottery classes of the form  $\langle m, 3; 2 \rangle$ , in terms of the covering number, were established independently by Brouwer & Voorhoeve [37] in 1979, by Brouwer [36] in 1981 and by Bate & Van Rees [17] in 1998, stating that

$$\left. \begin{aligned} L_1(2m+1, 3; 2) &= \underline{C}(m, 3; 2) + \underline{C}(m+1, 3; 2) \\ L_1(4m, 3; 2) &= \underline{C}(2m-1, 3; 2) + \underline{C}(2m+1, 3; 2) \\ L_1(4m+2, 3; 2) &= 2\underline{C}(2m+1, 3; 2), \end{aligned} \right\} \quad (1.1)$$

where  $\underline{C}(m, n; k)$  denotes the well-known (recursive) Schönheim covering bound [223, 224]. Bate & Van Rees [17] determined the values of the complete lottery numbers  $L_1(m, 6; 2)$  for  $m \leq 54$ , given in Table 1.2, in 1998.

Both Hanani, *et al.* [99] (in 1964) and Füredi, *et al.* [83] (in 1996) focused their attention on the lottery class  $\langle m, n; 2 \rangle$ , while Bate & Stanton [15] (in 1981) derived explicit formulas<sup>5</sup> for the complete lottery

<sup>4</sup>A Steiner system  $S(m, n; k)$  is a set  $X$  of  $m$  elements and a collection of subsets of  $X$  of cardinality  $n$  (called blocks), with the property that any  $k$  elements of  $X$  are contained in exactly one of the blocks [265].

<sup>5</sup>The author believes that the result derived by Bate & Stanton [15] is incorrect for the specific case  $m \equiv 11 \pmod{12}$  (due to conflicting results found on Internet repository tables [19, 44, 133, 237]), which instead conform to the formula  $\left\lceil \frac{m^2-m}{12} \right\rceil$  for  $m \equiv 11 \pmod{12}$  and  $m \leq 50$ .



$m$	6–10	11–15	16–20	21–25	26–30	31–33	34	35–36	37	38–39		
$L_1(m, 6; 2)$	1	2	3	4	5	7	8	9	10	11		
$m$	40	41–42	43	44–45	46	47	48	49–50	51	52	53	54
$L_1(m, 6; 2)$	12	13	14	15	16	17	18	19	20	21	22	23

Table 1.2: The class of complete lottery numbers  $L_1(m, 6; 2)$ , for  $6 \leq m \leq 54$ , due to Bate & Van Rees [17] in 1998.

$m$	36–37	38	39	40	41	42	43	44–45	46	47	48	49	50	51
$L_1(m, 6; 3) \leq$	96	101	102	105	112	123	138	154	160	161	165	174	187	203

Table 1.3: Bounds obtained on a specific class of complete lottery numbers  $L_1(m, 6; 3)$  by Colbourn [50] in 1996.

class  $L_1(m, 3; 2)$ , given by

$$L_1(m, 3; 2) = \begin{cases} \left\lceil \frac{m^2 - 2m}{12} \right\rceil & \text{if } m \equiv 2, 4, 6 \pmod{12} \\ \left\lceil \frac{m^2 - 2m}{12} \right\rceil + 1 & \text{if } m \equiv 0, 8, 10 \pmod{12} \\ \left\lceil \frac{m^2 - m}{12} \right\rceil & \text{if } m \equiv 1, 3, 5, 7 \pmod{12} \\ \left\lceil \frac{m^2 - m}{12} \right\rceil + 1 & \text{if } m \equiv 9, 11 \pmod{12}. \end{cases} \quad (1.2)$$

They also determined bounds on  $L_1(m, 4; 2)$  using a design theoretic approach [14]. Probabilistic methods to obtain upper bounds on  $L_1(m, n; k)$  were implemented by Harant, *et al.* [100] (in 1993). Colbourn [50] (in 1996) used combinatorial construction techniques to determine upper bounds on the complete lottery class  $L_1(m, 6; 3)$  for  $36 \leq m \leq 51$  (these bounds are contained in Table 1.3), as well as the bound  $L_1(34, 5; 3) \leq 136$ .

Li & Van Rees [137] derived general lower bounds on  $L_1(m, n; k)$  in 1999, using combinatorial theory. Li & Van Rees also determined some explicit complete lottery numbers and bounds on complete lottery numbers  $L_1(m, n; k)$  for the values  $m \leq 20$ ,  $n \leq 12$  and  $k \leq 5$  [133, 136, 138] in 2002. Finally, Belic [19], Li [133] and Stojiljkovic [237] all maintain and frequently update Internet repositories listing complete lottery numbers.

### 1.3.2 Packing and covering numbers

Fort & Hedlund [79] (in 1958) investigated the covering of pairs by triples, hence establishing some of the first covering numbers of the form  $C(m, 3; 2)$ . A more general case of covering numbers were considered by Kalbfleisch & Stanton [123] in 1968. Research on the packing and covering problems, obtaining the well-known (recursive) Schönheim packing and covering bounds

$$P(m, n; k) \leq \overline{P}(m, n; k) := \left\lceil \frac{m}{n} P(m-1, n-1; k-1) \right\rceil = \left\lceil \frac{m}{n} \left\lceil \frac{m-1}{n-1} \cdots \left\lceil \frac{m-k+1}{n-k+1} \right\rceil \cdots \right\rceil \right\rceil \quad \text{and}$$

$$C(m, n; k) \geq \underline{C}(m, n; k) := \left\lfloor \frac{m}{n} C(m-1, n-1; k-1) \right\rfloor = \left\lfloor \frac{m}{n} \left\lfloor \frac{m-1}{n-1} \cdots \left\lfloor \frac{m-k+1}{n-k+1} \right\rfloor \cdots \right\rfloor \right\rfloor,$$

was conducted by Schönheim [223, 224] (respectively in 1964 and 1969). Using the notation of the bounds obtained by Schönheim, the mentioned result by Fort & Hedlund [79] states that  $C(m, 3; 2) = \underline{C}(m, 3; 2)$ . Bounds on packing numbers were also found by Di Paola [65] (in 1966), who combined the fields of design and graph theory in his approach. In two successive papers by Mills [168, 169] in 1972 and 1973 respectively, the value of the covering class  $C(m, 4; 2)$  was determined to be

$$C(m, 4; 2) = \begin{cases} \underline{C}(m, 4; 2) + 1 & \text{if } m = 7, 9 \text{ or } 10, \\ \underline{C}(m, 4; 2) + 2 & \text{if } m = 19, \\ \underline{C}(m, 4; 2) & \text{otherwise.} \end{cases}$$

Much of this work was independently performed by Horton, *et al.* [107] in 1971. In 1973, Mills [167] also published a survey article of covering related research up to 1973. In 1978, Brouwer [34] determined  $P(m, 4; 3)$  for  $m$  a multiple of 6, while Brouwer [35] (in 1979) investigated general bounds on both  $P(m, n; k)$  and  $C(m, n; k)$ . For a comprehensive collection of work performed on covering numbers until 1979, the reader is referred to Mills [166]. This report also included some new results (actual constructed covering sets of specified cardinality) which include the determination of all covering numbers satisfying the inequality  $C(m, n; k) \leq 3(k+1)/2$ , as well as establishing all covering numbers of the form  $C(m, n; 2)$  for which  $m \leq 3n$ . Todorov [248] stated in 1980 that for  $n \geq kp^{k+1}$  (where  $p$  is a power of a prime) the covering number

$$C(pn + j, n; k) = \frac{p^{k+1} - 1}{p - 1} \quad (0 < j \leq p - 1)$$

and

$$\frac{p^{k+1} - p}{p - 1} \leq C(pn - i, n; k) \leq \frac{p^{k+1} - 1}{p - 1} \quad (0 \leq i < p - 1).$$

In 1980, Stanton & Bate [234] employed computer searches, using a backtracking algorithm, to determine some covering numbers  $C(m, n; k)$ , where  $m \leq 16$ . Stanton & Mullin [235] (in 1980) presented a survey of what had been done on covering numbers up to 1979 and also determined the covering numbers  $C(m, 4; 3) = \underline{C}(m, 4; 3) + 1$ , where  $m \equiv 7 \pmod{12}$  (thereby completing the single case not investigated by Mills [170] in 1974, stating that  $C(m, 4; 3) = \underline{C}(m, 4; 3)$ , where  $m \not\equiv 7 \pmod{12}$ ). In 1981, Todorov [253] determined a class of covering numbers stating that  $C(a_0p^2 + a_1p + a_2, a_0p + a_1; 2) = p^2 + p + 1$  if  $p$  is a prime power and  $a_0 \geq a_1 \geq a_2 > 0$  are non-negative integers. The values for the covering numbers  $C(7, 4; 3) = 12$ ,  $C(9, 5; 3) = 12$ ,  $C(11, 6; 3) = 11$  and  $C(13, 7; 3) = 13$  were determined by Todorov & Tonchev [254] in 1982. They also showed that  $13 \leq C(2m - 1, m; 3) \leq 14$  if  $8 \leq m \leq 14$ , while  $C(2m - 1, m; 3) = 14$  if  $m \geq 15$ . Nurmela & Östergård [193, 194] (in 1983) and Godbole, *et al.* [91] (in 1996) independently used probabilistic techniques to obtain general upper bounds on  $C(m, n; k)$ . The following result, for  $p$  a prime power and non-negative integers  $a_0 \geq a_1 \geq \dots \geq a_l$ , was proved by Todorov [246] in 1984:

$$C\left(\sum_{j=0}^l a_j p^{l-j}, \sum_{j=0}^{l-1} a_j p^{l-j-1}; l\right) = \frac{p^{l+1} - 1}{p - 1} \quad \text{if } a_l > 0$$

and

$$\frac{p^{l+1} - p}{p - 1} \leq C\left(\sum_{j=0}^l a_j p^{l-j}, \sum_{j=0}^{l-1} a_j p^{l-j-1}; l\right) \leq \frac{p^{l+1} - 1}{p - 1} \quad \text{if } a_l = 0.$$

Later that same year, Todorov [249] determined all  $m$ ,  $n$  and  $k$  for which  $C(m, n; k) \leq 3(k+2)/2$ , thus extending his result published in 1985 [250]. By 1984, the value of  $C(m, 5; 2)$  was known for all  $m \leq 23$  except  $m = 16$ . Mills [165] determined this single exception to be  $C(16, 5; 2) = 15$ . Both Rödl [213] and Spencer [231] (in 1985) independently proved the asymptotic covering and packing result

$$\lim_{m \rightarrow \infty} C(m, n; t) \binom{n}{k} \binom{m}{k}^{-1} = \lim_{m \rightarrow \infty} P(m, n; k) \binom{n}{k} \binom{m}{k}^{-1} = 1,$$

which was originally conjectured by Erdős & Hanani [71] in 1963. In two papers by Todorov [247, 252] in 1986, he respectively investigates the covering numbers  $C(m, n; 2)$  and  $C(m, n; 3)$ . In [247] upper and lower bounds on  $C(m, n; 2)$  (when  $5 \leq n \leq 9$  and  $3k < m \leq 30$ ) together with some explicit values for  $C(m, n; 2)$  are given, while in [252] the author finds all  $m$  and  $n$  for which  $C(m, n; 3) = 8$ . With certain exceptions, these are the values of  $m$  and  $n$  satisfying  $\frac{17}{11} < \frac{m}{n} \leq \frac{8}{5}$ . The exceptions are when  $m = 17k + 3$ ,  $n = 11k + 2$  with  $k \geq 1$ , in which case  $C(m, n; 3) = 8$ , even though  $\frac{m}{n}$  is not in the given range, and  $m = 8k + 3$ ,  $n = 5k + 2$  with  $k \geq 2$ , in which case  $C(m, n; 3) > 8$ . The class of covering numbers  $C(m, 5; 2)$  for  $m \equiv 2 \pmod{4}$  was determined to be  $C(m, 5; 2) = \underline{C}(m, 5; 2)$  (with the exception of  $m = 270, 274$ ) by Lamken, *et al.* [131] in 1987. Mills & Mullin [172], in 1988, also focused their attention on the class of covering numbers  $C(m, 5; 2)$ . They proved that  $C(15, 5; 2) = \underline{C}(15, 5; 2) + 1$ , while  $C(m, 5; 2) = \underline{C}(m, 5; 2)$  if  $m \equiv 3 \pmod{4}$  where  $m \geq 7$  and  $m \neq 15$ . In 1990, Morley & Van Rees [178] determined explicit covering number formulas as well as bounds on the infinite class of covering

$m$	6-7	8	9	10	11	12	13	14	15	16	17	18	19	20-21	22	23	24-25	26	27
$C(m, 5; 2)$	3	4	5	6	7	9	10	12	13	15	16	18	19	21	27	28	30	37	38

Table 1.4: The class of covering numbers  $C(m, 5; 2)$ , for  $6 \leq m \leq 27$ , known by 1992 as reported in [66]. It was also known that  $C(m, 5; 2) = \underline{C}(m, 5; 2)$  whenever  $m \equiv 2 \pmod{4}$ ,  $m \equiv 3 \pmod{4}$  and  $m \neq 15$ ,  $m \equiv 9 \pmod{20}$  and  $m > 3149$ , or  $m \equiv 17 \pmod{20}$  and  $m > 757$ . Furthermore  $C(m, 5; 2) = \underline{C}(m, 5; 2) + 1$  whenever  $m \equiv 13 \pmod{20}$  and  $m > 753$ .

numbers  $C(4m - 3, 2m - 1; m - 1)$  (for  $m \geq 3$ ) and  $C(4m - 4, 2m - 2; m)$  (for  $m \geq 2$ ). In a paper on the covering of  $n$ -sets by  $(n + 1)$ -sets in 1991, De Caen, *et al.* [60] determined that  $C(9, 5; 4) = 30$  and  $C(10, 6; 5) = 50$ . Mills [171] proved that  $C(11, 5; 3) = 20$  in 1992, thereby contradicting a conjecture by Morley & Van Rees [178] that  $C(11, 5; 3) = 21$ . As reported in [66] (with the exception of some equivalence classes), up to 1992, the value of  $C(m, 5; 2)$  was only known where  $m < 28$  (see Table 1.4). In 1995, Gordon, *et al.* [94] determined upper bounds on  $C(m, n; k)$  for  $m \leq 32$ ,  $n \leq 16$  and  $k \leq 8$ , using greedy algorithms and modification techniques that synthesize new coverings from existing ones. Some general constructions for packing designs were developed by Yin & Assaf [270] (in 1998). Bluskov & Hämäläinen [27] (in 1998) and Bluskov & Heinrich [28] (in 1999) determined upper bounds on several families and infinite classes of covering numbers  $C(m, n; k)$  where  $3 \leq k \leq n \leq 7$ . Explicit values for and bounds on an infinite class of covering numbers  $C(m, n; 2)$  were determined by Bluskov, *et al.* [26] (in 2000). Li & Van Rees [135] (in 2002) determined the individual covering number  $C(17, 10; 3)$ . Bate, *et al.* [16] determined  $C(19, 6; 2)$  in 2002. Current research on covering numbers is still performed by Greig, *et al.* [95], who determined all  $m$  and  $n$  for which  $C(m, n; 2) = 13$ , thus extending a result originally investigated by Todorov [251] in 1985. They also determined that  $C(28, 9; 2) = C(41, 13; 2) = 14$ . Gordon [44] maintains and frequently updates an Internet repository table containing bounds on and exact values for covering numbers  $C(m, n; k)$  where  $m \leq 32$ ,  $n \leq 16$  and  $k \leq 8$ .

## 1.4 Scope and objectives of this dissertation

Four objectives are pursued in this dissertation.

**Objective I:** To *introduce* a transparent and elementary framework in which the incomplete lottery and resource utilisation problems may be investigated. The foundations of such an environment should include establishing the boundedness (and hence existence) as well as growth properties (with respect to variations in their arguments) of incomplete lottery and resource utilisation numbers;

**Objective II:** To *establish* both analytical and algorithmic approaches toward solving (or at least approximating solutions to) the incomplete lottery and resource utilisation problems and to be able to compare the qualities and efficiencies of such approaches;

**Objective III:** To *develop* and *implement* a technique for distinguishing between structurally different (optimal) solutions to the incomplete lottery and resource utilisation problems;

**Objective IV:** To *pose* a number of questions and open problems that may spawn future research on the incomplete lottery and resource utilisation problems.

## 1.5 Preview of dissertation layout

The existence and certain basic properties of the parameters  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  are derived in Chapter 2 of this dissertation. More specifically, growth properties (§2.2) and specific values in some simple special cases (§2.3) are established. The chapter is concluded with a discussion on a possible solution approach to both the incomplete lottery and resource utilisation problems, using integer programming (§2.4).

Chapter 3 opens with a brief background on basic concepts from graph and complexity theory, in §3.1. Most of the basic graph theoretic results and terminology used throughout this dissertation are described in this section. The chapter also contains a detailed description (in §3.2) of a graph theoretic solution approach that will be used to attack the incomplete lottery and resource utilisation problems. A characterisation is provided of when  $L_1(m, n; k) = 1, 2$  or  $3$ . As a case study, lotteries of the form  $\langle m, n; k \rangle$  where  $1 \leq k \leq n \leq m \leq 10$  are considered in §3.4 with a section devoted to the symmetric graphical representation of lottery graphs in §3.3.

A further study of larger (typical) lottery parameter values is conducted in Chapter 4, with several theoretical bounds from the literature presented in a comparative manner. In particular, lower and upper bounds on complete lottery numbers from graph theory (§4.1) and other related disciplines (§4.2) are discussed.

The main focus of Chapter 5 is on the improvement of analytic upper bounds on incomplete lottery numbers, using a variety of algorithms. Lower bounds on resource utilisation numbers are found in the same way. Seven heuristic and greedy algorithms that were implemented are discussed, and analyses of their relative complexities, performances, advantages and disadvantages are given.

In Chapter 6 two algorithmic characterisation techniques for  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set overlapping structures is given. The first characterisation algorithm (§6.1.1) constructs evolving  $n$ -set overlapping structures in a so-called lottery tree, while the second characterisation algorithm (§6.1.2) uses a graph theoretic approach of representing playing sets in conjunction with a graph automorphism package, called **nauty**, to construct a so-called **nauty** tree. All structurally different  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets for  $1 \leq k \leq n \leq m \leq 10$ , satisfying  $m + k > 2n$  and  $n \leq \lfloor \frac{m}{2} \rfloor$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  are determined, using these techniques. This constitutes another original contribution of this dissertation with respect to lottery sets. An inquiry into the characteristics of the number of structurally different solutions with respect to variations of the parameters  $m, n, k$  and  $\psi$  is launched in §6.2. A number of new complete lottery numbers  $L_1(m, n; k)$  are also established in §6.4.

The dissertation is concluded (in Chapter 7) with a summary in §7.1 of the achievements of this dissertation and some proposals with respect to possible future work related to the lottery and resource utilisation problems, in §7.2.

The source code to algorithms described and implemented in the dissertation is included in Appendix A. Best available bounds (obtained from the Internet) on the complete lottery number  $L_1(m, n; k)$ , are given in Appendix B. Finally, Appendix C contains tables of optimal overlapping structure encodings as a reference.



## Chapter 2

# Properties of $L_\psi$ and $\Psi_\ell$

“Everything actual must first have been possible,  
before having actual *existence*.”

*Albert Pike* (1809–1891)

“If you have built castles in the air, your work need not be lost;  
that is where they should be. Now put the foundations under them.”

*Henry D Thoreau* (1817–1861) [244]

In this chapter the focus will be on establishing the existence of solutions to the incomplete lottery and resource utilisation problems in general, as well as deriving some of their basic properties and, where possible, providing explicit values for incomplete lottery and resource utilisation numbers.

### 2.1 Boundedness of $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$

The existence of solutions to the incomplete lottery and resource utilisation problems is established in Theorem 2.1, by giving both lower and upper bounds on the incomplete lottery number  $L_\psi(m, n; k)$  and resource utilisation number  $\Psi_\ell(m, n; k)$  in closed form for all feasible values of the parameters  $m$ ,  $n$ ,  $k$ ,  $\psi$  and  $\ell$ . However, the following preliminary result is necessary in order to prove the theorem.

**Lemma 2.1** *The number of  $n$ -sets from  $\mathcal{U}_m$  that have at least one  $k$ -subset in common with some  $n$ -set  $v \in \Phi(\mathcal{U}_m, n)$  (excluding  $v$  itself) is given by*

$$r = |N[v]| - 1 = \sum_{i=k}^{n-1} \binom{n}{i} \binom{m-n}{n-i}, \quad (2.1)$$

for all  $1 \leq k \leq n \leq m$ .

#### **Proof**

Consider any  $n$ -set,  $v$ , from  $\mathcal{U}_m$  in the lottery  $\langle m, n; k \rangle$ . The number of  $n$ -sets from  $\mathcal{U}_m$  that have exactly  $i$  elements in common with  $v$  is  $\binom{n}{i} \binom{m-n}{n-i}$  (the first factor indicating the number of ways in which the coinciding  $i$ -set may be fixed and the second factor counting the number of combinations in which the remainder of the  $n$ -set may be completed). The number of  $n$ -sets that have at least one  $k$ -subset in common with  $v$  is obtained by letting  $i$  vary between  $k$  and  $n-1$  inclusively, yielding the result presented in (2.1). ■

It is now possible to derive the boundedness (and hence existence) of incomplete lottery and resource utilisation numbers in the following theorem.

**Theorem 2.1 (Existence of  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$ )** *The incomplete lottery number  $L_\psi(m, n; k)$  and the resource utilisation number  $\Psi_\ell(m, n; k)$  exist and, in fact,*

$$\left\lceil \frac{\psi \binom{m}{n}}{r+1} \right\rceil \leq L_\psi(m, n; k) \leq P(m, n; k) \leq C(m, n; k) \leq \binom{m}{k} \quad (2.2)$$

and

$$\frac{r+\ell}{\binom{m}{n}} \leq \Psi_\ell(m, n; k) \leq \min \left\{ 1, \frac{(r+1)\ell}{\binom{m}{n}} \right\}, \quad (2.3)$$

for all  $1 \leq k \leq n \leq m$ ,  $0 < \psi \leq 1$  and all  $0 < \ell \leq L_1(m, n; k)$ , where  $r$  is given in (2.1).

**Proof**

Suppose  $\mathcal{L}_\psi = \{l_1, l_2, \dots, l_{L_\psi(m, n; k)}\}$  is an  $L_\psi(m, n; k)$ -set for the lottery  $\langle m, n; k \rangle$  and that the set  $\mathcal{V}_i$  contains all  $n$ -sets,  $\phi_n$ , from  $\Phi(\mathcal{U}_m, n)$  such that  $\Phi(\phi_n, k) \cap \Phi(l_i, k) \neq \emptyset$ , excluding  $l_i$  itself, for  $i = 1, \dots, L_\psi(m, n; k)$ . Then  $|\mathcal{V}_i| = r$  according to Lemma 2.1 for all  $i$ . At best, the sets  $\{\mathcal{V}_i \cup l_i\}$  and  $\{\mathcal{V}_j \cup l_j\}$  each contain  $(r+1)$  mutually disjoint elements of  $\Phi(\mathcal{U}_m, n)$  for all  $i \neq j$ , in which case  $(r+1)L_\psi(m, n; k) \geq \psi \binom{m}{n}$ , from which the first inequality in (2.2) follows.

For the second inequality in (2.2) it suffices to show that every packing set is also an incomplete lottery set. Suppose  $\mathcal{P}$  is a packing set of cardinality  $P(m, n; k)$  for  $\langle m, n; k \rangle$ . Then, for every  $\varphi \in \Phi(\mathcal{U}_m, n) \setminus \mathcal{P}$ , it holds that  $\Phi(\varphi, k) \cap \Phi(\varphi', k) \neq \emptyset$  for some  $\varphi' \in \mathcal{P}$ , implying that  $\mathcal{P}$  is indeed also a (possibly non-minimal) incomplete lottery set for any  $0 < \psi \leq 1$ . Therefore the second inequality in (2.2) holds.

Let  $\mathcal{C}$  be a covering set of cardinality  $C(m, n; k)$  for  $\langle m, n; k \rangle$ , but suppose, to the contrary, that  $P(m, n; k) > C(m, n; k)$ . Then there exists a  $\phi_n^* \in \Phi(\mathcal{U}_m, n) \setminus \mathcal{C}$  such that  $\Phi(\phi_n^*, k) \cap \Phi(c, k) = \emptyset$  for all  $c \in \mathcal{C}$ , contradicting the fact that  $\mathcal{C}$  is a covering set for  $\langle m, n; k \rangle$  and therefore yielding the third inequality  $P(m, n; k) \leq C(m, n; k)$  in (2.2).

The final upper bound in the inequality chain (2.2) is achieved by the existence of a covering set  $\mathcal{C}$  of cardinality  $\binom{m}{k}$  for  $\langle m, n; k \rangle$  where the first  $k$  entries of set elements consist of all  $\binom{m}{k}$  different, unordered  $k$ -sets from  $\mathcal{U}_m$  and by randomly adding  $(n-k)$  elements to every  $n$ -set from the remaining  $(m-k)$  elements in  $\mathcal{U}_m$ .  $\mathcal{C}$  is a (possibly non-minimal) covering set by construction and therefore  $C(m, n; k) \leq \binom{m}{k}$ .

The lower bound on  $\Psi_\ell(m, n; k)$  in (2.3) for  $\langle m, n; k \rangle$  may be obtained by constructing a playing set  $\mathcal{L} = \{l_1, l_2, \dots, l_\ell\} \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality  $0 < \ell \leq L_1(m, n; k)$  as follows: Select any  $n$ -set  $l_1 \in \Phi(\mathcal{U}_m, n)$ , and suppose  $l_1$  shares  $k$ -subsets with a subset  $\mathcal{V}_1$  of elements from  $\Phi(\mathcal{U}_m, n)$ , excluding  $l_1$  itself. Now  $\mathcal{L}' = \Phi(\mathcal{U}_m, n) \setminus \mathcal{V}_1$  is clearly a complete lottery set for  $\langle m, n; k \rangle$ . This is true because if the winning  $n$ -set,  $w$ , is an element of  $\mathcal{L}'$ , then certainly there exists an element  $l' \in \mathcal{L}'$  such that  $\Phi(l', k) \cap \Phi(w, k) \neq \emptyset$  (namely the element  $l' = w$ ). Otherwise, if  $w \in \mathcal{V}_1$ , then  $\Phi(l_1, k) \cap \Phi(w, k) \neq \emptyset$ . Therefore  $\ell \leq L_1(m, n; k) \leq |\mathcal{L}'| = \binom{m}{n} - r$ . Hence there exist at least  $(\ell - 1)$  elements in the set  $\Phi(\mathcal{U}_m, n) \setminus \{\mathcal{V}_1 \cup \{l_1\}\}$ . Choose any  $(\ell - 1)$  elements  $l_2, l_3, \dots, l_\ell$  from this last set to complete the construction of  $\mathcal{L}$ . Then the resource utilised by  $\mathcal{L}$  is at least  $\frac{r+\ell}{\binom{m}{n}}$ .

The trivial upper bound  $\Psi_\ell(m, n; k) \leq 1$  in (2.3) is obtained by any  $L_1(m, n; k)$ -set (i.e., by letting  $\ell = L_1(m, n; k)$ ) for the lottery  $\langle m, n; k \rangle$ . The non-trivial upper bound  $\Psi_\ell(m, n; k) \leq (r+1)\ell / \binom{m}{n}$  in (2.3) is attained (in a best possible case) if it is possible to construct a playing set  $\mathcal{L} = \{l_1, \dots, l_\ell\}$  for the lottery  $\langle m, n; k \rangle$  with the property that  $\{\mathcal{V}_i \cup l_i\} \cap \{\mathcal{V}_j \cup l_j\} = \emptyset$  for all  $i \neq j = 1, \dots, \ell$  where  $\mathcal{V}_i$  denotes the set of all  $n$ -sets from  $\mathcal{U}_m$  that share a  $k$ -subset with  $l_i$  (excluding  $l_i$  itself),  $i = 1, \dots, \ell$ . In this case, the resource utilised by  $\mathcal{L}$  is

$$\left[ \sum_{i=1}^{\ell} |\mathcal{V}_i \cup \{l_i\}| \right] / \binom{m}{n} = \left[ \sum_{i=1}^{\ell} (r+1) \right] / \binom{m}{n} = \frac{(r+1)\ell}{\binom{m}{n}},$$

by Lemma 2.1. In cases where it is not possible to construct the playing set  $\mathcal{L}$  in a manner that all sets  $\mathcal{V}_1, \dots, \mathcal{V}_\ell$  are pairwise disjoint, it follows by the inclusion-exclusion principle that the resource utilised by  $\mathcal{L}$  is strictly less than  $\frac{(r+1)\ell}{\binom{m}{n}}$ . ■

These bounds on  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  serve the purpose of establishing existence of solutions to the incomplete lottery problem and the recourse utilisation problem, but are typically very weak (wide apart) for large values of  $m$ , prompting an investigation into methods for improving both lower and upper bounds. Analytic and algorithmic improvements of bounds on  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  will be considered in Chapters 4 and 5 of this dissertation respectively.

## 2.2 Growth properties of $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$

A number of growth properties of incomplete lottery and resource utilisation numbers with respect to their arguments, may also be established.

### Theorem 2.2 (Growth properties of $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$ )

- (a)  $L_\psi(m', n; k) \leq L_\psi(m, n; k)$ , for all  $1 \leq k \leq n \leq m' < m$  and  $0 < \psi \leq 1$ .
- (b)  $L_\psi(m, n'; k) \geq L_\psi(m, n; k)$ , for all  $1 \leq k \leq n' < n \leq m$  and  $0 < \psi \leq 1$ .
- (c)  $L_\psi(m, n; k') \leq L_\psi(m, n; k)$ , for all  $1 \leq k' < k \leq n \leq m$  and  $0 < \psi \leq 1$ .
- (d)  $L_{\psi'}(m, n; k) \leq L_\psi(m, n; k)$ , for all  $1 \leq k \leq n \leq m$  and  $0 < \psi' < \psi \leq 1$ .
- (e)  $\Psi_\ell(m', n; k) \geq \Psi_\ell(m, n; k)$ , for all  $1 \leq \ell < L_1(m', n; k)$  and  $1 \leq k \leq n \leq m' < m$ .
- (f)  $\Psi_\ell(m, n'; k) \leq \Psi_\ell(m, n; k)$ , for all  $1 \leq \ell < L_1(m, n; k)$  and  $1 \leq k \leq n' < n \leq m$ .
- (g)  $\Psi_\ell(m, n; k') \geq \Psi_\ell(m, n; k)$ , for all  $1 \leq \ell \leq L_1(m, n; k')$  and  $1 \leq k' < k \leq n \leq m$ .
- (h)  $\Psi_{\ell'}(m, n; k) \leq \Psi_\ell(m, n; k)$ , for all  $1 \leq \ell' < \ell \leq L_1(m, n; k)$  and  $1 \leq k \leq n \leq m$ .

### Proof

(a) Suppose  $\mathcal{L}_\psi$  is an  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$ , and that  $m' < m$ , for some  $0 < \psi \leq 1$ . Consider the following reduction method to obtain a (possibly non-minimal)  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m - 1, n; k \rangle$  from  $\mathcal{L}_\psi$ .

**Reduction method:** Consider a two-dimensional tabular representation of  $\mathcal{L}_\psi$ , consisting of  $L_\psi(m, n; k)$  rows (denoting the  $n$ -sets in  $\mathcal{L}_\psi$ ) and  $m$  columns (denoting the elements of  $\mathcal{U}_m$ ) in which the  $(i, j)$ -th cell contains a cross if  $j \in \mathcal{U}_m$  is an element of the  $i$ -th  $n$ -set of  $\mathcal{L}_\psi$ , and is otherwise empty.

The remaining part of  $\mathcal{L}_\psi$ , obtained by temporarily disregarding some  $j$ -th column in the above tabular representation (some resulting rows will represent  $n$ -subsets of  $\mathcal{U}_m \setminus \{j\}$  and some might represent  $(n - 1)$ -subsets of  $\mathcal{U}_m \setminus \{j\}$ ), will collectively share  $k$ -subsets of  $\mathcal{U}_m \setminus \{j\}$  with a proportion,  $\psi_j$ , of elements from  $\Phi(\mathcal{U}_m \setminus \{j\}, n)$ . Note that an  $n$ -set that has a  $k$ -match with  $\mathcal{L}_\psi$ , will have a  $k$ -match with exactly  $m - n$  of the  $m$  possible reductions (when not considering  $(n - 1)$ -sets for  $k$ -matches with  $\mathcal{L}_\psi$ , of course). Hence, counting the total number of  $n$ -sets with a  $k$ -match over all possible reductions, we have  $\sum_{j=1}^m \binom{m-1}{n} \psi_j = \binom{m}{n} \psi \times (m - n)$ . The average of the proportions  $\psi_1, \dots, \psi_m$  (i.e., when removing columns  $1, \dots, m$ ) is given by

$$\frac{\sum_{j=1}^m \psi_j}{m} = \frac{\sum_{j=1}^m \binom{m-1}{n} \psi_j}{\binom{m-1}{n} m} = \frac{\binom{m}{n} \psi (m - n)}{\binom{m-1}{n} m} = \psi.$$

Thus there exists a  $j^* \in \mathcal{U}_m$  such that  $\psi_{j^*} \geq \psi$ . Now permanently remove column  $j^*$  from the tabular representation of  $\mathcal{L}_\psi$  and place a cross in *any* empty cell of each row that now contains only  $n - 1$  crosses as a result of the permanent column deletion. The result is a tabular representation of a  $100(1 - \psi_{j^*})\%$ -incomplete lottery set for  $\langle m - 1, n; k \rangle$ , which is also a  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m - 1, n; k \rangle$ .

By (possibly repeated) application of the above reduction method to  $\mathcal{L}_\psi$ , a (possibly non-minimal)  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m', n; k \rangle$  may be obtained. Hence we conclude that  $L_\psi(m', n; k) \leq L_\psi(m, n; k) = |\mathcal{L}_\psi|$ .

(b) Suppose  $\mathcal{L}'_\psi$  is an  $L_\psi(m, n'; k)$ -set for  $\langle m, n'; k \rangle$ , and that  $n' < n$  for some  $0 < \psi \leq 1$ . It will be shown, by considering the construction method outlined below, that a (possibly non-minimal)  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m, n' + 1; k \rangle$  may be constructed from  $\mathcal{L}'_\psi = \{T'_1, \dots, T'_{L_\psi(m, n'; k)}\}$ .



**Construction method:** Append, to each  $n'$ -set,  $T'_i \in \mathcal{L}'_\psi$ , an arbitrary element of  $\mathcal{U}_m \setminus T'_i$  to form an  $(n' + 1)$ -set,  $T_i$ , for each  $i = 1, \dots, L_\psi(m, n'; k)$ . Define the set

$$\mathcal{L}_\psi = \bigcup_{i=1}^{L_\psi(m, n'; k)} T_i.$$

It is easy to see that if any  $w \in \Phi(\mathcal{U}_m, n')$  has a  $k$ -match with  $\mathcal{L}'_\psi$ , then  $w$  also has a  $k$ -match with  $\mathcal{L}_\psi$ . By definition of  $\mathcal{L}'_\psi$ , it follows that at least  $100\psi\%$  of the elements of  $\Phi(\mathcal{U}_m, n')$  have a  $k$ -match with  $\mathcal{L}'_\psi$ . It remains to be shown that at least  $100\psi\%$  of the elements of  $\Phi(\mathcal{U}_m, n' + 1)$  also have a  $k$ -match with  $\mathcal{L}'_\psi$  and hence with  $\mathcal{L}_\psi$ .

Append to each  $\phi_{n'} \in \Phi(\mathcal{U}_m, n')$  all possible single elements from  $\mathcal{U}_m \setminus \phi_{n'}$  to form  $m - n'$  new  $(n' + 1)$ -sets. Now, if a particular  $n'$ -set,  $\phi_{n'}$ , has a  $k$ -match with  $\mathcal{L}'_\psi$ , then each of its  $m - n'$  associated  $(n' + 1)$ -sets will also have a  $k$ -match with  $\mathcal{L}'_\psi$ . Therefore at least  $100\psi\%$  of all the appended  $(n' + 1)$ -sets have a  $k$ -match with  $\mathcal{L}'_\psi$ . Note that every element of  $\Phi(\mathcal{U}_m, n' + 1)$  occurs exactly  $n' + 1$  times amongst the set of appended  $(n' + 1)$ -sets, because each of the elements of  $\Phi(\mathcal{U}_m, n' + 1)$  is constructed from a different  $n'$ -set  $\phi_{n'} \in \Phi(\mathcal{U}_m, n')$ . Therefore, if we remove all duplicates amongst the appended  $(n' + 1)$ -sets, the proportion of  $(n' + 1)$ -sets that have a  $k$ -match with  $\mathcal{L}'_\psi$  remains unchanged. Hence, at least  $100\psi\%$  of the elements of  $\Phi(\mathcal{U}_m, n' + 1)$  have a  $k$ -match with  $\mathcal{L}_\psi$ . Therefore,  $\mathcal{L}_\psi$  is a  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m, n' + 1; k \rangle$ .

By (possibly repeated) application of the above construction method to the set  $\mathcal{L}'_\psi$ , a (possibly non-minimal)  $100(1 - \psi)\%$ -incomplete lottery set for  $\langle m, n; k \rangle$  may be obtained. Hence we conclude that  $L_\psi(m, n; k) \leq L_\psi(m, n'; k) = |\mathcal{L}'_\psi|$ .

(c) Suppose  $\mathcal{L}_\psi$  is an  $L_\psi(m, n; k)$ -set for the lottery  $\langle m, n; k \rangle$  and that  $k' < k$  for some  $0 < \psi \leq 1$ . Therefore, there exists a subset  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\lceil \psi \binom{m}{n} \rceil$  with the property that, for any  $\phi_n \in \mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$ , there exists an  $l \in \mathcal{L}_\psi$  such that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$ . But for any  $\phi_k \in \Phi(\phi_n, k) \cap \Phi(l, k)$  it holds that  $\Phi(\phi_k, k') \cap \Phi(l, k') \neq \emptyset$  (or equivalently that  $\Phi(\phi_n, k') \cap \Phi(l, k') \neq \emptyset$ ), implying that  $\mathcal{L}_\psi$  is also a  $100(1 - \psi)\%$ -incomplete lottery set for the lottery  $\langle m, n; k' \rangle$ . Hence  $L_\psi(m, n; k') \leq L_\psi(m, n; k)$ .

(d) Suppose  $\mathcal{L}_\psi$  is an  $L_\psi(m, n; k)$ -set for the lottery  $\langle m, n; k \rangle$  and that  $1 \leq k \leq n \leq m$  and  $0 < \psi' < \psi \leq 1$ . Therefore, there exists a subset  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\lceil \psi \binom{m}{n} \rceil$  with the property that, for any  $\phi_n \in \mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$ , there exists an  $l \in \mathcal{L}_\psi$  such that  $\Phi(\phi_n, k) \cap \Phi(l, k) \neq \emptyset$ . This condition also holds for any  $\phi'_n \in \mathcal{V}'_{\psi'} \subseteq \mathcal{V}_\psi$  (where  $\mathcal{V}'_{\psi'}$  has a cardinality of at least  $\psi' \binom{m}{n}$ ), implying that  $\mathcal{L}_\psi$  is a (possibly non-minimal)  $100(1 - \psi')\%$ -incomplete lottery set for the lottery  $\langle m, n; k \rangle$ . Hence  $L_{\psi'}(m, n; k) \leq L_\psi(m, n; k)$ .

(e) Suppose, to the contrary, that  $\Psi_\ell(m', n; k) < \Psi_\ell(m, n; k)$  for some  $m' < m$ . Then, according to Proposition 1.1,  $L_{\Psi_\ell(m', n; k)}(m', n; k) > \ell$ . However, due to the (trivial) inequality  $\Psi_\ell(m, n; k) \geq \Psi_\ell(m', n; k)$  it also follows that  $L_{\Psi_\ell(m', n; k)}(m, n; k) \leq \ell$  by Proposition 1.1. Therefore  $L_{\Psi_\ell(m', n; k)}(m, n; k) \leq \ell < L_{\Psi_\ell(m', n; k)}(m', n; k)$ , which contradicts the result of Theorem 2.2(a). This implies that  $\Psi_\ell(m', n; k) \geq \Psi_\ell(m, n; k)$ .

(f) Suppose, to the contrary, that  $\Psi_\ell(m, n'; k) > \Psi_\ell(m, n; k)$  for some  $n' < n$ . Then, according to Proposition 1.1,  $L_{\Psi_\ell(m, n'; k)}(m, n'; k) > \ell$ . However, due to the (trivial) inequality  $\Psi_\ell(m, n'; k) \geq \Psi_\ell(m, n; k)$  it also follows that  $L_{\Psi_\ell(m, n'; k)}(m, n; k) \leq \ell$  by Proposition 1.1. Therefore  $L_{\Psi_\ell(m, n'; k)}(m, n'; k) \leq \ell < L_{\Psi_\ell(m, n'; k)}(m, n; k)$ , which contradicts the result of Theorem 2.2(b). This implies that  $\Psi_\ell(m, n'; k) \leq \Psi_\ell(m, n; k)$ .

(g) Consider a  $\Psi_\ell(m, n; k)$ -set  $\mathcal{L}$  for the lottery  $\langle m, n; k \rangle$  and suppose  $\ell \leq L_1(m, n; k')$  with  $k' < k$ . From (2.1) it follows that  $r$  is a decreasing function of  $k$ , implying that every  $l \in \mathcal{L}$  shares a common  $k'$ -subset with more  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  than it shares  $k$ -subsets with  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ . Consequently  $\Psi_\ell(m, n; k') \geq \Psi_\ell(m, n; k)$ .

(h) Let  $\mathcal{L}'$  be a  $\Psi_{\ell'}(m, n; k)$ -set for the lottery  $\langle m, n; k \rangle$  and suppose that  $\ell' < \ell \leq L_1(m, n; k)$ . By adding any  $\ell - \ell'$   $n$ -sets  $l \notin \cup_{l \in \mathcal{L}'} N[l]$  to  $\mathcal{L}'$ , a new playing set  $\mathcal{L}$  (of cardinality  $\ell$ ) is constructed with at least one more element in  $\cup_{l \in \mathcal{L}} N[l]$  than in  $\cup_{l \in \mathcal{L}'} N[l']$  (i.e.,  $\Psi_\ell(m, n; k) \geq \Psi_{\ell'}(m, n; k) + 1/\binom{m}{n}$ ). Hence  $\Psi_{\ell'}(m, n; k) \leq \Psi_\ell(m, n; k)$ . ■

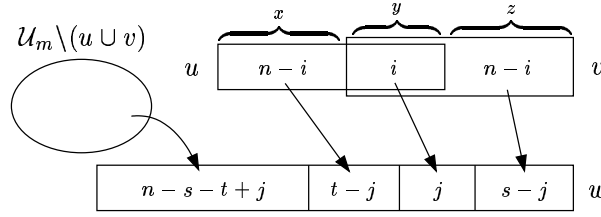


Figure 2.1: Counting the number of  $n$ -sets  $w$  sharing a common  $t$ -subset of  $\mathcal{U}_m$  with  $u$  and an  $s$ -subset of  $\mathcal{U}_m$  with  $v$ , which in turn share at most an  $i$ -subset of  $\mathcal{U}_m$ .

The following results are direct consequences of Theorem 2.2(a)–(c) and correspond to known growth property results for the complete lottery problem by Li [136]<sup>1</sup>.

**Corollary 2.1 (Growth properties of  $L_1(m, n; k)$ )**

- (a)  $L_1(m', n; k) \leq L_1(m, n; k)$ , for all  $1 \leq k \leq n \leq m' \leq m$ .
- (b)  $L_1(m, n'; k) \geq L_1(m, n; k)$ , for all  $1 \leq k \leq n' \leq n \leq m$ .
- (c)  $L_1(m, n; k') \leq L_1(m, n; k)$ , for all  $1 \leq k' \leq k \leq n \leq m$ .

Growth results regarding a change in a combination of the parameters of the complete lottery number (such as  $L_1(m, n; k) \geq L_1(m+1, n+1; k)$ ,  $L_1(m, n; k) \leq L_1(m+1, n; k+1)$  and  $L_1(m, n; k) \leq L_1(m+1, n+1; k+1)$ ) are also due to Li [134] and their proofs will not be repeated here. Similar results may be established for the incomplete lottery number. It is interesting to note that no direct comparison between the values of  $L_1(m, n; k)$  and  $L_1(m, n+1; k+1)$  has yet been established. In some instances the inequality  $L_1(m, n; k) \leq L_1(m, n+1; k+1)$  is observed, while for other combinations of values of  $m$ ,  $n$  and  $k$  the opposite is observed [136].

## 2.3 Explicit values for $L_\psi(m, n; k)$ and $\Psi_\ell(m, n; k)$

In order to determine explicit values for the resource utilisation number in certain special cases, the following intermediate result is necessary.

**Lemma 2.2** *The number of  $n$ -sets in  $\Phi(\mathcal{U}_m, n)$  having at least a  $k$ -subset in common with any two specified  $n$ -sets in  $\Phi(\mathcal{U}_m, n)$ , which in turn share a common  $i$ -subset of  $\mathcal{U}_m$  but no common  $(i+1)$ -subset of  $\mathcal{U}_m$  ( $m > 2n - i$ ), is given by*

$$\xi_2^i(m, n; k) = \sum_{t=k}^{n-1} \sum_{s=k}^{n-1} \sum_{j=0}^i \binom{n-i}{t-j} \binom{i}{j} \binom{n-i}{s-j} \binom{m-2n+i}{n-s-t+j}. \quad (2.4)$$

**Proof**

Let  $u, v \in \Phi(\mathcal{U}_m, n)$  share an  $i$ -subset of  $\mathcal{U}_m$ , with  $0 \leq i \leq n$ , but not an  $(i+1)$ -subset of  $\mathcal{U}_m$ . Define the sets  $x = u \setminus v$ ,  $y = u \cap v$  and  $z = v \setminus u$  (see Figure 2.1). Then  $|x| = n-i$ ,  $|y| = i$  and  $|z| = n-i$ . The number of sets  $w \in \Phi(\mathcal{U}_m, n)$  that share a  $t$ -subset of  $\mathcal{U}_m$  with  $u$  and an  $s$ -subset of  $\mathcal{U}_m$  with  $v$ . If  $w$  shares a  $j$ -subset of  $\mathcal{U}_m$  with  $y$ , then it shares a  $(t-j)$ -subset of  $\mathcal{U}_m$  with  $x$  and an  $(s-j)$ -subset of  $\mathcal{U}_m$  with  $z$ . Hence the  $(s+t-j)$  elements of  $\mathcal{U}_m$  that  $w$  shares with  $u \cup v$  may be chosen in  $\binom{n-i}{t-j} \binom{i}{j} \binom{n-i}{s-j}$  different ways. But then the remaining  $n - (s+t-j)$  elements of  $w \setminus (u \cup v)$  may be chosen in  $\binom{m-2n+i}{n-s-t+j}$  different ways (note that in the case where  $s+t-j > n$ , this factor evaluates to 0). Finally, both  $s$  and  $t$  should be at least  $k$  (and of course at most  $n-1$ ), and  $j$  must be at least zero and not exceed  $i$ , in which case the formula in (2.4) is obtained for  $\xi_2^i(m, n; k)$ . ■

<sup>1</sup>The growth property results for the incomplete lottery numbers presented in Theorem 2.2(a)–(c) were determined by the author independently from Li [136], and before the author became aware of the work of Li.

In certain special cases, incomplete lottery and resource utilisation numbers may be determined explicitly. These special cases are summarised in the following theorem.

**Theorem 2.3 (Special cases of  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$ )**

- (a)  $L_\psi(m, m; k) = 1$ , for all  $1 \leq k \leq m$  and all  $0 < \psi \leq 1$ .
- (b)  $L_\psi(m, n; n) = \lceil \psi \binom{m}{n} \rceil$ , for all  $1 \leq n \leq m$  and all  $0 < \psi \leq 1$ .
- (c)  $L_\psi(m, n; k) = 1$ , for all  $1 \leq k \leq n \leq m$  such that  $2n \geq m + k$  and all  $0 < \psi \leq 1$ .
- (d)  $\Psi_\ell(m, n; n) = \ell / \binom{m}{n}$ , for all  $1 \leq n \leq m$  and all  $1 \leq \ell \leq \binom{m}{n}$ .
- (e)  $L_\psi(m, n; k) = 1$ , for all  $1 \leq k \leq n \leq m$  and all  $0 < \psi \leq (r+1) / \binom{m}{n}$ .
- (f) If  $1 < \ell \leq \lfloor \frac{m}{n} \rfloor$ ,  $n < 3k$  and all  $1 \leq k \leq n \leq m$ , then

$$\Psi_\ell(m, n; k) = \left[ (r+1)\ell - \binom{\ell}{2} \sum_{t=k}^{n-1} \sum_{s=k}^{n-1} \binom{n}{s} \binom{n}{t} \binom{m-2n}{n-s-t} \right] / \binom{m}{n},$$

where  $r$  is given in (2.1).

- (g) For all  $1 \leq n \leq m$  and  $0 < \psi \leq 1$ ,  $L_\psi(m, n; 1) = \ell$  is the smallest integer solution to the inequality

$$\prod_{i=0}^{n-1} (m - \ell n - i) \leq \frac{m!(1-\psi)}{(m-n)!}. \quad (2.5)$$

**Proof**

(a) Consider the set  $\mathcal{L}_\psi = \{l\}$ , with  $l = \{1, \dots, m\}$ .  $\mathcal{L}_\psi$  is an  $L_\psi(m, m; k)$ -set for the lottery  $\langle m, m; k \rangle$ , since  $\Phi(\mathcal{U}_m, m) = \{\mathcal{U}_m\}$ , and clearly  $\Phi(\mathcal{U}_m, k) \cap \Phi(l, k) \neq \emptyset$ , so that the desired result follows for all  $1 \leq k \leq m$  and  $0 < \psi \leq 1$ .

(b) In order to guarantee that, for any element  $\phi_n$  in some set  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\psi \binom{m}{n}$  (as given in Definition 1.2), there exists an element  $l \in \mathcal{L}_\psi$  such that  $\Phi(\phi_n, n) \cap \Phi(l, n) \neq \emptyset$ ,  $\mathcal{L}$  has to consist of at least  $100\psi\%$  of all possible  $n$ -sets from  $\mathcal{U}_m$  in order to be a  $100(1-\psi)\%$ -incomplete lottery set for  $\langle m, n; n \rangle$ , implying that  $L_\psi(m, n; n) \geq \lceil \psi \binom{m}{n} \rceil$ . Any additional elements considered for inclusion in  $\mathcal{L}_\psi$ , render an unnecessary increase in the cardinality of  $\mathcal{L}_\psi$ . Therefore  $L_\psi(m, n; n) \leq \lceil \psi \binom{m}{n} \rceil$  and the desired result holds.

(c) If  $2n - m \geq k$ , then any two  $n$ -sets of  $\mathcal{U}_m$  have at least  $k$  elements in common. Therefore any one  $n$ -set  $l$  from  $\mathcal{U}_m$  forms a complete lottery set and the desired result holds by (2.2).

(d) Because  $r = 0$  in the lottery  $\langle m, n; n \rangle$  (according to (2.1)), no  $n$ -set shares a common  $k$ -subset with any other  $n$ -set. Hence, the resource utilised by any  $n$ -set, is exactly  $1 / \binom{m}{n}$  and the result follows.

(e)  $\Psi_1(m, n; k) = (r+1) / \binom{m}{n}$ , where  $r$  is given in (2.1). For any  $0 < \psi' \leq (r+1) / \binom{m}{n}$ ,  $1 \leq L_{\psi'}(m, n; k) \leq L_\psi(m, n; k) = 1$ , according to (2.2) and Theorem 2.2(d), from which the result follows.

(f) Let  $\mathcal{L}$  be a playing set of  $\langle m, n; k \rangle$  consisting of  $\ell$  disjoint  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ . Such a set exists, because  $\ell \leq \lfloor \frac{m}{n} \rfloor$ . Every  $v \in \mathcal{L}$  shares a common  $k$ -subset with itself and  $r$  (as defined in (2.1)) other  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ , although some  $n$ -sets are counted twice when considering all possible elements  $v \in \mathcal{L}$ . No  $n$ -set is counted three (or more) times, since no  $n$ -set may consist of 3 (disjoint)  $k$ -subsets ( $n < 3k$ ). Since  $\mathcal{L}$  consists of disjoint  $n$ -sets (sharing no common  $i$ -subset from  $\mathcal{U}_m$ ), the expression in (2.4) in Lemma 2.2 reduces to  $\xi_2^0(m, n; k) = \sum_{t=k}^{n-1} \sum_{s=k}^{n-1} \binom{n}{s} \binom{n}{t} \binom{m-2n}{n-s-t}$ , summed over all  $\binom{\ell}{2}$  pairs of  $n$ -sets.

(g) Let  $\mathcal{L}$  be an  $L_\psi(m, n; 1)$ -set for  $\langle m, n; 1 \rangle$  of cardinality  $\ell$ . Every additional (distinct) element of  $\mathcal{U}_m$  incorporated into an  $n$ -set of  $\mathcal{L}$  (utilised by  $\mathcal{L}$ ) necessarily increases the probability of winning a 1-prize via  $\mathcal{L}$ . Therefore, choosing disjoint  $n$ -sets from  $\mathcal{U}_m$  is the optimal way of constructing  $\mathcal{L}$  for lotteries of the form  $\langle m, n; 1 \rangle$ . Furthermore, the number of elements of  $\mathcal{U}_m$  not utilised by  $\mathcal{L}$  (there are  $m - \ell n$  such elements) should not be so large that the probability of winning a 1-prize via  $\mathcal{L}$  drops below  $\psi$ . Equivalently stated,

$$\binom{m - \ell n}{n} \leq \binom{m}{n} (1 - \psi), \quad (2.6)$$

which simplifies to (2.5), from which  $\ell$  may, in principle, be solved. ■

The following special cases are a direct consequence of Theorem 2.3.

**Corollary 2.2 (Special cases of  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$ )**

- (a)  $\Psi_1(m, m; k) = 1$ , for all  $1 \leq k \leq m$ .
- (b)  $\Psi_1(m, n; k) = (r+1)/\binom{m}{n}$ , for all  $1 \leq k \leq n \leq m$ , where  $r$  is given in (2.1).
- (c)  $\Psi_1(m, n; k) = 1$ , for all  $1 \leq k \leq n \leq m$  such that  $2n \geq m+k$ .
- (d)  $L_1(m, n; 1) = \lfloor \frac{m}{n} \rfloor$ , for all  $1 \leq n \leq m$ .
- (e) For all  $0 < \psi \leq 1$ ,

$$L_\psi(m, n; 1) = \begin{cases} \lceil \psi m \rceil, & \text{if } n = 1 \leq m \\ \left\lfloor \frac{2m-1-\sqrt{1+4m(m-1)(1-\psi)}}{4} \right\rfloor, & \text{if } n = 2 \leq m \\ \left\lfloor \frac{2m-3-\sqrt{5+4\sqrt{1+m(m-1)(m-2)(m-3)(1-\psi)}}}{8} \right\rfloor, & \text{if } n = 4 \leq m. \end{cases} \quad (2.7)$$

**Proof**

- (a) This result follows as a direct consequence of Theorem 2.3(a) when  $\psi = 1$ .
- (b) This result follows as a direct consequence of Theorem 2.3(c) when  $\ell = 1$ .
- (c) This result follows as a direct consequence of Theorem 2.3(c) when  $\psi = 1$ .
- (d) In this special case ( $\psi = 1$ ) of Theorem 2.3(g), we seek the smallest  $\ell \in \mathbb{N}$ ,  $\ell = \ell^*$  say, for which

$$(m - \ell n)(m - \ell n - 1)(m - \ell n - 2) \cdots (m - \ell n - n + 1) \leq 0. \quad (2.8)$$

The lefthand side of (2.8) vanishes for any  $\ell \in \{ \frac{m-n+1}{n}, \frac{m-n+2}{n}, \dots, \frac{m-2}{n}, \frac{m-1}{n}, \frac{m}{n} \} = \mathcal{I}$ . Hence, (2.8) is satisfied in the intervals

$$\frac{m+2s-1}{n} - 1 \leq \ell \leq \frac{m+2s}{n} - 1, \text{ for all } s = 1, 2, \dots, \frac{n}{2},$$

when  $n$  is even, or in the intervals

$$\frac{m+2s-1}{n} - 1 \leq \ell \leq \frac{m+2s}{n} - 1, \text{ for all } s = 1, 2, \dots, \frac{n-1}{2}, \quad \text{or} \quad \ell \geq \frac{m}{n},$$

when  $n$  is odd. The set  $\mathcal{I}$  contains at most one integer, since  $\frac{m}{n} - \frac{m-n+1}{n} = \frac{n-1}{n} < 1$ . To see that  $\mathcal{I}$  contains at least one integer, observe that the elements of  $\mathcal{I}$  may be written as  $\frac{m-i}{n}$ ,  $i = 0, \dots, n-1$ . Now, if  $m \equiv i \pmod{n}$ , then  $\frac{m-i}{n}$  is integer. And since  $m \equiv i \pmod{n}$  for some  $i \in \{0, \dots, n-1\}$  it follows that  $\ell^* = L_1(m, n; 1) \in \mathcal{I}$ . Moreover, this unique integer element of  $\mathcal{I}$  is exactly  $\frac{m-i}{n} = \lfloor \frac{m}{n} \rfloor$  (in accordance with a result in [13]).

- (e) For  $n \in \{1, 2, 4\}$  the roots of the  $n$ -th degree polynomial

$$\mathcal{F}^{(n)}(\ell) = \prod_{i=0}^{n-1} (m - \ell n - i) - \frac{m!(1-\psi)}{(m-n)!}, \quad (2.9)$$

derived from (2.5), are given by

$$\begin{aligned} \ell^* &= \psi m && \text{if } n = 1 \leq m, \\ \ell_{\pm}^* &= \frac{2m-1 \pm \sqrt{1+4m(m-1)(1-\psi)}}{4} && \text{if } n = 2 \leq m, \\ \left. \begin{aligned} \ell_{-, \pm}^* &= \frac{2m-3 - \sqrt{5 \pm 4\sqrt{1+m(m-1)(m-2)(m-3)(1-\psi)}}}{8} \\ \ell_{+, \pm}^* &= \frac{2m-3 + \sqrt{5 \pm 4\sqrt{1+m(m-1)(m-2)(m-3)(1-\psi)}}}{8} \end{aligned} \right\} && \text{if } n = 4 \leq m. \end{aligned}$$

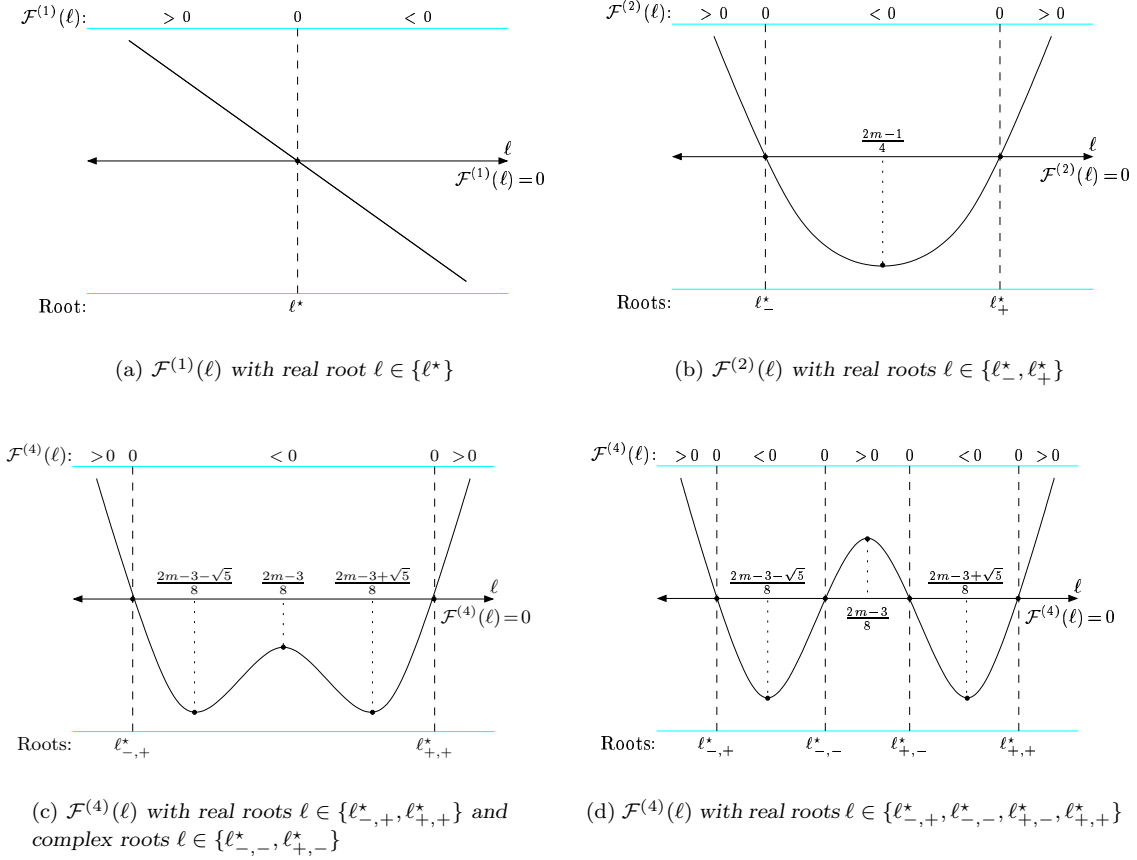


Figure 2.2: Qualitative schematic representation of the polynomial  $\mathcal{F}^{(n)}(\ell)$  given in (2.9) for determining (a)  $L_\psi(m, 1; 1)$ , (b)  $L_\psi(m, 2; 1)$  and (c)–(d)  $L_\psi(m, 4; 1)$ . The formula for  $L_\psi(m, n; 1)$  for  $n \in \{1, 2, 4\}$  is given in (2.7).

For  $n = 1$  the polynomial satisfies

$$\mathcal{F}^{(1)}(\ell) \begin{cases} > 0 & \text{if } \ell < \ell^* \\ < 0 & \text{if } \ell > \ell^* \end{cases}$$

in a neighbourhood of  $\ell^*$  (see Figure 2.2(a)). Hence, we immediately have that  $L_\psi(m, 1; 1) = \lceil \psi m \rceil$ .

Now consider the case  $n = 2$  (see Figure 2.2(b)). For any fixed  $m \geq 2$  the interval  $[\ell_{-}^*, \ell_{+}^*]$  is smallest when  $\psi = 1$ , in which case  $[\ell_{-}^*, \ell_{+}^*]_{\min} = [\frac{m}{2}, \frac{m+1}{2}]$ , which contains the integer  $\lceil \ell_{-}^* \rceil$  for both equivalence classes of  $m \pmod{2}$ . Hence, for  $0 < \psi \leq 1$ , the interval  $[\ell_{-}^*, \ell_{+}^*]$  contains the integer  $\lceil \ell_{-}^* \rceil$ , from which the formula in (2.7) follows for  $L_\psi(m, 2; 1)$ .

Finally, consider the case  $n = 4$ . First observe that  $\ell_{-,+}^*, \ell_{+,+}^* \in \mathbb{R}$  for all  $m \geq 4$  and  $0 < \psi \leq 1$ . Now we distinguish between two different subcases, depending on whether  $\ell_{-,-}^*, \ell_{+,-}^*$  are real or complex.

Suppose  $\ell_{-,-}^*, \ell_{+,-}^* \in \mathbb{C} \setminus \mathbb{R}$  (see Figure 2.2(c)). This occurs when  $0 < \psi < 1 - \frac{9}{16m(m-1)(m-2)(m-3)}$  and implies that

$$\mathcal{F}^{(4)}(\ell) \begin{cases} > 0 & \text{if } \ell < \ell_{-,+}^* \\ < 0 & \text{if } \ell_{-,+}^* < \ell < \ell_{+,+}^* \\ > 0 & \text{if } \ell > \ell_{+,+}^*. \end{cases}$$

For any fixed  $m \geq 4$  the width of the interval  $[\ell_{-,+}^*, \ell_{+,+}^*]$  is

$$\ell_{+,+}^* - \ell_{-,+}^* = \frac{\sqrt{5 + 4\sqrt{1 + m(m-1)(m-2)(m-3)(1-\psi)}}}{4}.$$

This width is smallest when  $\psi = 1$ , in which case  $[\ell_{-,+}^*, \ell_{+,+}^*]_{\min} = [\frac{m-3}{4}, \frac{m}{4}]$ , which clearly contains an integer for all equivalence classes of  $m \pmod{4}$ , since  $\frac{m-1}{4}, \frac{m-2}{4} \in [\frac{m-3}{4}, \frac{m}{4}]$ . Moreover, this integer is given by  $\lceil \ell_{-,+}^* \rceil$ , from which the formula in (2.7) follows for  $L_\psi(m, 4; 1)$ .

Now suppose  $\ell_{-,-}^*, \ell_{+,-}^* \in \mathbb{R}$  (see Figure 2.2(d)). It has been established (above) that the interval  $[\ell_{-,+}^*, \ell_{+,+}^*]$  contains an integer. Furthermore, the width of the interval  $(\ell_{-,-}^*, \ell_{+,-}^*)$  is

$$\ell_{+,-}^* - \ell_{-,-}^* = \frac{\sqrt{5 - 4\sqrt{1 + m(m-1)(m-2)(m-3)(1-\psi)}}}{4}.$$

This width is largest when  $\psi = 1$ , in which case  $(\ell_{-,-}^*, \ell_{+,-}^*)_{\max} = (\frac{m-2}{4}, \frac{m-1}{4})$  clearly does not contain an integer for any equivalence class of  $m \pmod{4}$ . Hence, for all  $1 - \frac{9}{16m(m-1)(m-2)(m-3)} \leq \psi \leq 1$  the interval  $[\ell_{-,+}^*, \ell_{+,+}^*] \setminus (\ell_{-,-}^*, \ell_{+,-}^*)$  contains the integer  $\lceil \ell_{-,+}^* \rceil$ , from which the formula in (2.7) follows for  $L_\psi(m, 4; 1)$ . ■

The reader might question the absence of the special cases for  $n = 3$  and  $n > 4$  in Theorem 2.2(e). This is due to the fact that the theoretical manipulation of the roots determined from (2.9) are far more intricate than those for the cases  $n = 1, 2, 4$ , even in the small case of  $n = 3$ .

If it were possible to factor the polynomial in (2.9) analytically for general values of  $n \leq m$  and  $0 < \psi \leq 1$ , we would have an explicit, closed-form formula for the incomplete lottery number  $L_\psi(m, n; 1)$ . However, such a factorisation seems to be a very hard problem. In contrast, the polynomial in (2.9) is immediately factored for any  $n \leq m$  if  $\psi = 1$ . This situation seems to indicate that, for the class of incomplete lottery numbers  $L_\psi(m, n; 1)$  at the very least, the incomplete lottery problem is harder to resolve than the complete lottery problem. This case will be argued further in the next section, for more general classes of lottery numbers.

Finally, note that the bounds in (2.2) on  $L_\psi(m, n; k)$  are best possible for general values of the parameters  $m, n, k, \psi$  and  $\ell$ : the lower bound is achieved when  $2n \geq m+k$  by Theorem 2.3(c) (since then  $r+1 = \binom{m}{n}$ ), while the upper bound is achieved when  $k = n$  by Theorem 2.3(b). Similarly, the lower and upper bounds in (2.3) are equal when  $\ell = 1$ , while the trivial upper bound in (2.3) is achieved when  $\ell = L_1(m, n; k)$ , by definition.

## 2.4 Binary programming solution approaches

The construction of  $L_\psi(m, n; k)$ -sets and  $\Psi_\ell(m, n; k)$ -sets for  $\langle m, n; k \rangle$ , such as those presented in Examples 1.1 and 1.2, is not a trivial task. One intuitive solution approach towards the incomplete lottery and resource utilisation problems is to formulate integer programming problems containing binary set construction decision variables, with the objective of minimising the number of  $n$ -sets in an incomplete lottery set or maximising the resource utilisation proportion of playing sets of a fixed cardinality. We first consider the special case of the incomplete lottery problem, where  $\psi = 1$ , in an attempt to formulate an integer program for solving the complete lottery problem. Formally this may be achieved by numbering the  $\binom{m}{n}$   $n$ -sets from  $\mathcal{U}_m$  in increasing lexicographic order and then

$$\left. \begin{array}{l} \text{minimising} \\ \text{subject to} \end{array} \right\} \begin{array}{l} y_1 = \sum_{i=1}^{\binom{m}{n}} x_i \\ \sum_{j=1}^{\binom{m}{n}} a_{ij} x_j \geq 1, \quad i = 1, \dots, \binom{m}{n} \\ x_i = 0 \text{ or } 1, \quad i = 1, \dots, \binom{m}{n}. \end{array} \quad (2.10)$$

Here  $x_i$  is a binary decision variable taking the value 1 if the  $i$ -th  $n$ -set should be included in the complete lottery set and  $x_i = 0$  otherwise,  $a_{ij}$  is an adjacency parameter taking the value 1 if the  $i$ -th and  $j$ -th  $n$ -set share a common  $k$ -subset or if  $i = j$ , and taking the value 0 otherwise. The non-trivial constraint requires that, given any  $n$ -set  $w \in \Phi(\mathcal{U}_m, n)$ , there should be at least one  $n$ -set from

$\langle m, n; k \rangle$	$\binom{m}{n}$	Branches traversed	Execution time (dd:hh:mm:ss)	$L_1(m, n; k)$
$\langle 7, 3; 2 \rangle$	35	19	00:00:00:01	4
$\langle 8, 3; 2 \rangle$	56	6	00:00:00:03	5
$\langle 9, 4; 3 \rangle$	126	39 972	00:00:13:52	9
$\langle 11, 3; 2 \rangle$	165	274 781	00:03:32:28	10
$\langle 10, 5; 4 \rangle$	252	18 504 408	51:13:20:41	<b>14</b>

(a) Solutions to the complete lottery problem binary formulation in (2.10)

$\langle m, n; k \rangle$	$\psi$	$\binom{m}{n}$	Branches traversed	Execution time (dd:hh:mm:ss)	$L_\psi(m, n; k)$	$L_1(m, n; k)$
$\langle 7, 4; 3 \rangle$	0.8	35	4	00:00:00:01	$\leq \mathbf{3}$	4
$\langle 8, 3; 2 \rangle$	0.6	56	13	00:00:00:01	$\leq \mathbf{4}$	5
$\langle 9, 5; 4 \rangle$	0.7	126	3 695	00:00:01:45	$\leq \mathbf{7}$	9
$\langle 11, 3; 2 \rangle$	0.9	165	819 385	00:08:14:48	$\leq \mathbf{10}$	10
$\langle 10, 5; 4 \rangle$	0.7	252	1 047 142	04:16:19:36	$\leq \mathbf{12}$	<b>14</b>

(b) Solutions to the incomplete lottery problem binary formulation in (2.11)

Table 2.1: Execution times of and branches traversed during the binary programming approach toward solving (a) the complete lottery problem formulation in (2.10) and (b) the incomplete lottery problem formulation in (2.11), using LINGO [139] (all executions were performed on an AMD Athlon 1.8 GHz personal computer with 256 MB memory). Boldface entries represent previously undetermined results.

$\Phi(\mathcal{U}_m, n)$ , corresponding to a non-zero decision variable, sharing a common  $k$ -subset with  $w$ . If an optimal solution to (2.10) is found, then  $L_1(m, n; k) = y_1$ . Although it is possible to achieve global optima for small values of the parameters  $m$  and  $n$  via the binary programming problem (2.10), realistic parameter values ( $m = 49$  and  $n = 6$ , for instance) render this approach practically infeasible. Table 2.1(a) gives an indication of the execution time of this approach, using the linear/integer programming solver LINGO [139] (all executions were performed on an AMD Athlon 1.8 GHz personal computer with 256 MB memory). LINGO employs the well-known branch-and-bound optimisation method for searching through the solution space. The number of branches traversed during the solution of (2.10) for certain values of  $m$ ,  $n$  and  $k$  are also shown in Table 2.1(a).

One attempt at generalising the integer program in (2.10) to incorporate the incomplete version of the lottery problem (*i.e.*, when  $0 < \psi < 1$ ), would rather be to consider

$$\left. \begin{array}{l}
 \text{minimising} \quad y'_\psi = \sum_{i=1}^{\binom{m}{n}} x_i \\
 \text{subject to} \quad \sum_{j=1}^{\binom{m}{n}} a_{ij} x_j \geq 1, \quad i = 1, \dots, \lceil \psi \binom{m}{n} \rceil \\
 \quad \quad \quad x_i = 0 \text{ or } 1, \quad i = 1, \dots, \binom{m}{n},
 \end{array} \right\} \quad (2.11)$$

using a similar notation as above. The difference between the formulation in (2.11) as opposed to that in (2.10) is that only the lexicographic *first*  $\lceil \psi \binom{m}{n} \rceil$   $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  are required to share a common  $k$ -subset with  $w$ . An optimal solution to (2.11) will, however, only yield an upper bound  $L_\psi(m, n; k) \leq y'_\psi$  to the incomplete lottery number, because a *different* selection of *fewer* than  $y'_\psi$  elements from  $\Phi(\mathcal{U}_m, n)$  may, in fact, share a common  $k$ -subset with a (possibly completely) different set of  $\lceil \psi \binom{m}{n} \rceil$  elements of  $\Phi(\mathcal{U}_m, n)$  than merely the lexicographic *first*  $\lceil \psi \binom{m}{n} \rceil$  elements of  $\Phi(\mathcal{U}_m, n)$ . In fact, the formulation in (2.11) should be performed  $\binom{a}{b}$  times (where  $a = \binom{m}{n}$  and  $b = \lceil \psi \binom{m}{n} \rceil$ ) in order to pinpoint the value of  $L_\psi(m, n; k)$ , each time requiring that a *different* subset of  $\lceil \psi \binom{m}{n} \rceil$   $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  share a common  $k$ -subset with  $w$ . However, this approach is not practically feasible in view of the prohibitively large values of  $\binom{a}{b}$  (where  $a = \binom{m}{n}$  and  $b = \lceil \psi \binom{m}{n} \rceil$ ), as shown in Table 2.2. Table 2.1(b) gives a similar indication of the execution times of above approach to establishing upper bounds on  $L_\psi(m, n; k)$  (for

$n$	$\psi$	$m$		
		5	10	15
$\lceil \frac{m}{6} \rceil$	0.1 or 0.9	$5.000 \times 10^0$	$1.221 \times 10^6$	$3.201 \times 10^{63}$
	0.2 or 0.8	$5.000 \times 10^1$	$8.861 \times 10^8$	$3.555 \times 10^{97}$
	0.3 or 0.7	$1.000 \times 10^1$	$1.668 \times 10^{11}$	$3.181 \times 10^{119}$
	0.4 or 0.6	$1.000 \times 10^1$	$1.715 \times 10^{12}$	$3.726 \times 10^{131}$
	0.5	$1.000 \times 10^1$	$4.116 \times 10^{12}$	$3.474 \times 10^{135}$
$\lceil \frac{m}{3} \rceil$	0.1 or 0.9	$1.000 \times 10^1$	$4.065 \times 10^{28}$	$1.049 \times 10^{423}$
	0.2 or 0.8	$4.500 \times 10^1$	$2.981 \times 10^{44}$	$1.314 \times 10^{651}$
	0.3 or 0.7	$1.200 \times 10^2$	$3.090 \times 10^{54}$	$8.291 \times 10^{794}$
	0.4 or 0.6	$2.100 \times 10^2$	$1.364 \times 10^{60}$	$1.109 \times 10^{876}$
	0.5	$2.520 \times 10^2$	$9.049 \times 10^{61}$	$1.432 \times 10^{902}$
$\lceil \frac{m}{2} \rceil$	0.1 or 0.9	$1.000 \times 10^1$	$1.781 \times 10^{35}$	$1.587 \times 10^{907}$
	0.2 or 0.8	$4.500 \times 10^1$	$8.316 \times 10^{53}$	$3.644 \times 10^{1396}$
	0.3 or 0.7	$1.200 \times 10^2$	$5.480 \times 10^{65}$	$2.470 \times 10^{1705}$
	0.4 or 0.6	$2.100 \times 10^2$	$2.514 \times 10^{72}$	$7.279 \times 10^{1878}$
	0.5	$2.520 \times 10^2$	$3.633 \times 10^{74}$	$1.335 \times 10^{1935}$

Table 2.2: An estimate of the number of formulations  $\binom{a}{b}$  (where  $a = \binom{m}{n}$  and  $b = \lceil \psi \binom{m}{n} \rceil$ ) to be solved when using the formulation in (2.11) to determine the exact value of  $L_\psi(m, n; k)$ .

different values of the parameters  $m$ ,  $n$ ,  $k$  and  $\psi$ ) as well as the number of branches traversed during the solution of (2.11), when using LINGO (all executions were performed on an AMD Athlon 1.8 GHz personal computer with 256 MB memory). To determine an exact integer programming formulation of the incomplete lottery problem, a different (certainly non-trivial and non-linear) constraint is required. One solution approach might be to consider

$$\left. \begin{array}{l}
 \text{minimising} \quad y_\psi = \sum_{i=1}^{\binom{m}{n}} x_i \\
 \text{subject to} \quad \sum_{p=1}^{\binom{m}{n}} (-1)^{p+1} \left( \sum_{1 \leq i_1 < \dots < i_p}^{\binom{m}{n}} \left( \sum_{j=1}^{\binom{m}{n}} \prod_{q=1}^p a_{i_q j} x_{i_q} \right) \right) \geq \lceil \psi \binom{m}{n} \rceil \\
 x_i = 0 \text{ or } 1, \quad i = 1, \dots, \binom{m}{n}.
 \end{array} \right\} \quad (2.12)$$

The non-trivial constraint dictates that there should exist a subset  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\lceil \psi \binom{m}{n} \rceil$  with the property that, given *any*  $n$ -set  $w \in \mathcal{V}_\psi$ , there should be at least one  $n$ -set corresponding to a non-zero decision variable (which is therefore included in the incomplete lottery set  $\mathcal{L}_\psi$ ) sharing a common  $k$ -subset with  $w$ . In particular, the innermost sum of the non-trivial constraint in (2.12) counts the number of  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  that share at least one  $k$ -subset with *all* members of the specific combination of the  $i_1$ -th,  $i_2$ -th,  $\dots$ ,  $i_p$ -th lexicographic  $n$ -sets from  $\mathcal{V}_\psi$  if these members were to be included in a playing set for  $\langle m, n; k \rangle$ . The middle summation merely allows for the inclusion of all possible combinations of  $p$   $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  into the playing set. Finally, the outermost summation utilises the inclusion-exclusion principle to evaluate the total number of  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  that share at least one  $k$ -subset with all possible combinations of up to  $\binom{m}{n}$   $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ . If an optimal solution to (2.12) is found, then  $L_\psi(m, n; k) = y_\psi$ . The non-linearity of the constraint in (2.12), however, renders this solution approach practically infeasible.

An additional shortcoming of the above mentioned binary programming formulation of the incomplete lottery problem is that any optimal solution found to (2.12) (or to (2.10), for that matter) sheds no light on the possible set-overlapping structures of *other* optimal solutions. Typically more than one solution exists for any given incomplete lottery problem. A detailed discussion on characterising the different possible  $L_\psi(m, n; k)$ -set structures will follow in Chapter 6 of this dissertation.



Using a similar notation as in (2.12), the resource utilisation problem may be solved by

$$\left. \begin{array}{l} \text{maximising} \\ \text{subject to} \end{array} \right\} \begin{array}{l} y_\ell = \sum_{p=1}^{\ell} (-1)^{p+1} \left( \sum_{1 \leq i_1 < \dots < i_p}^{\binom{m}{n}} \left( \sum_{j=1}^{\binom{m}{n}} \prod_{q=1}^p a_{i_q j} x_{i_q} \right) \right) \\ \sum_{j=1}^{\binom{m}{n}} x_j \leq \ell \\ x_j = 0 \text{ or } 1, j = 1, \dots, \binom{m}{n}. \end{array} \quad (2.13)$$

The summation explanations for the objective function in (2.13) are similar to those described for (2.12), although the outermost summation is restricted to evaluating the total number of  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$  that share at least one  $k$ -set with all possible combinations of only  $\ell$   $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ , as limited by the non-trivial constraint. Note that the non-trivial constraint in (2.13) will be binding for all  $1 \leq \ell \leq L_1(m, n; k)$ . If an optimal solution to (2.13) is found, then  $\Psi_\ell(m, n; k) = y_\ell / \binom{m}{n}$ . However, the highly non-linear structure of the objective function in (2.13) altogether prohibits the use of binary programming as a practical solution technique for the resource utilisation problem.

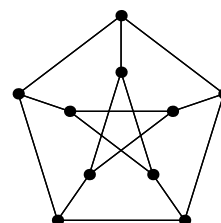
The binary programming problem (2.10) is nonlinear only by virtue of its (trivial) binary variable constraints. In contrast, the constraint in (2.12) and objective function in (2.13) exhibit a far more serious nonlinear structure, strengthening the argument that the incomplete lottery and resource utilisation problems may be more difficult to solve than the well-established complete lottery problem.

## 2.5 Chapter summary

In this chapter, the existence and certain growth properties of both the incomplete lottery number  $L_\psi(m, n; k)$  and resource utilisation number  $\Psi_\ell(m, n; k)$  were established (in §2.1 and §2.2 respectively). Explicit values for these variables were found for special cases of the parameters  $m$ ,  $n$ ,  $k$ ,  $\psi$  and  $\ell$  (in §2.3). Finally, binary programming formulations for both the incomplete lottery and resource utilisation problems were derived in §2.4. From the results presented in §2.4, it is clear that a more efficient solution procedure to the incomplete lottery and resource utilisation problems than mere binary programming is desirable.

## Chapter 3

# The Lottery Graph



*The Petersen Graph*

“Some men see things as they are and say, ‘Why?’  
I dream things that never were and say, ‘Why not?’”  
*George B Shaw (1856–1950) [132]*

This chapter extends the discussion of the incomplete lottery and resource utilisation problems to within the realm of graph theory, with the main focus on defining the so-called lottery graph in §3.2. For the sake of completeness, certain fundamentals from graph and complexity theory are discussed in §3.1, followed by a description of a symmetric representation for the visual representation of the inherent symmetry of the lottery graph in §3.3. The chapter closes with a complete analysis of small lotteries in §3.4 (that is, for all  $1 \leq k \leq n \leq m \leq 10$ ) in terms of the newly defined graph theoretic approach.

### 3.1 Fundamentals from graph and complexity theory

This section is devoted to providing the reader with the minimal necessary graph theoretic background in order to understand the concepts and ideas employed throughout the rest of the dissertation.

As described earlier, lottery schemes will be modelled by means of graphs. A **graph**  $\mathcal{G}$  (sometimes indicated by  $\mathcal{G} = (V, E)$ ) consists of a non-empty, finite set  $V = V(\mathcal{G})$ , called the **vertex set**, as well as a (possibly empty) set  $E = E(\mathcal{G})$  of 2-element subsets of  $V(\mathcal{G})$ , called the **edge set** [45]. The elements of the set  $V(\mathcal{G})$  are called the **vertices** of  $\mathcal{G}$ , while the elements of the set  $E(\mathcal{G})$  are referred to as **edges** of  $\mathcal{G}$ . The number of vertices of a graph, indicated by  $p = |V(\mathcal{G})|$ , is called the **order** of the graph  $\mathcal{G}$ , while the **size** of a graph  $\mathcal{G}$  is given by the number of edges, or  $q = |E(\mathcal{G})|$ . A graph of order  $p$  and size  $q \leq \binom{p}{2}$  is called a  $(p, q)$  graph. Define the **density** of a graph  $\mathcal{G}$  as the ratio between the size of the graph  $\mathcal{G}$  of specific order and the size of a same-order graph containing every possible edge. This means that the density of a graph is  $q/\binom{p}{2}$ . A graphical representation of a graph  $\mathcal{G}_1$  with vertex set  $V(\mathcal{G}_1) = \{v_1, \dots, v_7\}$ , edge set  $E(\mathcal{G}_1) = \{v_1v_3, v_1v_7, v_2v_4, v_3v_7, v_5v_7\}$  and a density of  $\frac{5}{21} \approx 0.2381$  is shown in Figure 3.1(a) as an example of a  $(7, 5)$  graph.

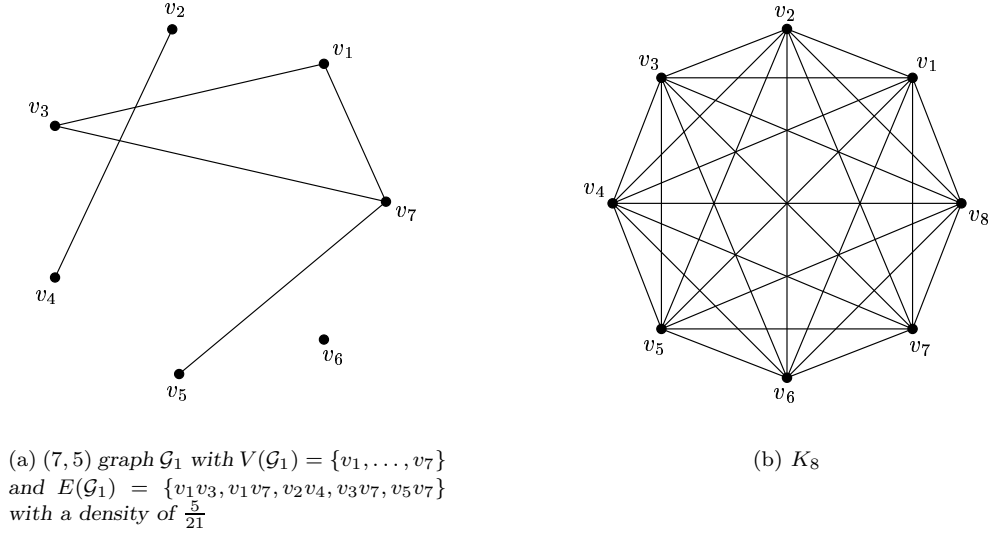


Figure 3.1: Examples of basic graphs.

An **open** [closed] **neighbourhood set**,  $N_{\mathcal{G}}(v) \subseteq V(\mathcal{G})$  [ $N_{\mathcal{G}}[v] \subseteq V(\mathcal{G})$ ], is associated with every vertex  $v$  of  $\mathcal{G}$ . This open [closed] neighbourhood set contains all the vertices of  $\mathcal{G}$  that have an edge in common with  $v$  [additionally including  $v$  itself]. Formally,  $N_{\mathcal{G}}(v) = \{u \in V(\mathcal{G}) \mid uv \in E(\mathcal{G})\}$  [ $N_{\mathcal{G}}[v] = \{v\} \cup N_{\mathcal{G}}(v)$ ]. The vertices  $u \in N_{\mathcal{G}}(v)$  are said to be **adjacent** to the vertex  $v$  in  $\mathcal{G}$ . In addition, an edge  $uv$  is said to **join** vertices  $u$  and  $v$ . The number of vertices adjacent to a vertex  $v$ , is referred to as the **degree**<sup>1</sup> of  $v$  (denoted by  $\deg_{\mathcal{G}}(v) = |N_{\mathcal{G}}(v)|$ ). A vertex  $v$  with  $\deg_{\mathcal{G}}(v) = 0$  is called an **isolated vertex** of  $\mathcal{G}$ . The **maximum** [minimum] **degree** of an order  $p$  graph  $\mathcal{G}$  is defined as  $\Delta(\mathcal{G}) = \max_{1 \leq i \leq p} \{\deg_{\mathcal{G}}(v_i)\}$  [ $\delta(\mathcal{G}) = \min_{1 \leq i \leq p} \{\deg_{\mathcal{G}}(v_i)\}$ ] where  $v_i \in V(\mathcal{G})$  for  $i = 1, \dots, p$ . The following result, relating the sum of vertex degrees and the number of edges in any graph, is sometimes referred to as the fundamental theorem of graph theory [45].

**Theorem 3.1 (Fundamental theorem of graph theory)** *Let  $\mathcal{G}$  be a graph of order  $p$  and size  $q$ , with  $V(\mathcal{G}) = \{v_1, v_2, \dots, v_p\}$ . Then*

$$\sum_{i=1}^p \deg_{\mathcal{G}}(v_i) = 2q. \quad (3.1)$$

### Proof

When the degrees of the vertices of  $\mathcal{G}$  are summed, each edge is counted twice, once for each of the two adjacent vertices. ■

A  $v_1$ - $v_n$  **walk**  $\mathcal{W}_{\mathcal{G}}$  of **length**  $n - 1$  in a graph  $\mathcal{G}$  is defined as an alternating sequence

$$\mathcal{W}_{\mathcal{G}} : v_1, e_1, v_2, e_2, \dots, v_{n-1}, e_{n-1}, v_n \quad (n \geq 0)$$

of vertices and edges, both beginning and ending in a vertex, such that  $e_i = v_i v_{i+1}$  for all  $i = 1, 2, \dots, n-1$  (note that vertices and edges may appear more than once in a walk). The walk  $\mathcal{W}_{\mathcal{G}}$  may also be unambiguously written as  $\mathcal{W}_{\mathcal{G}} : v_1, v_2, \dots, v_n$ . A  $v_1$ - $v_n$  **path** is a  $v_1$ - $v_n$  walk in which no vertex is repeated (*i.e.*,  $v_i \neq v_j$  if  $i \neq j$  for all  $i, j = 1, 2, \dots, n - 1$  in  $V(\mathcal{G})$ ) and is denoted  $P_n$ . In the graph  $\mathcal{G}_1$  of Figure 3.1(a),  $\mathcal{W}_{\mathcal{G}_1} : v_5, v_7, v_1, v_3, v_7, v_5$  is an example of a walk of length 5, while  $P_{\mathcal{G}_1} : v_1, v_3, v_7, v_5$  represents a path of length 3 in  $\mathcal{G}_1$ . If there exists a  $u$ - $v$  path for every vertex pair  $(u, v)$  of a graph, the graph is said to be **connected** [102]. Conversely, a graph that contains at least one vertex pair  $(u, v)$  for

<sup>1</sup>This is also sometimes referred to in literature as the **valency** of a vertex [92].

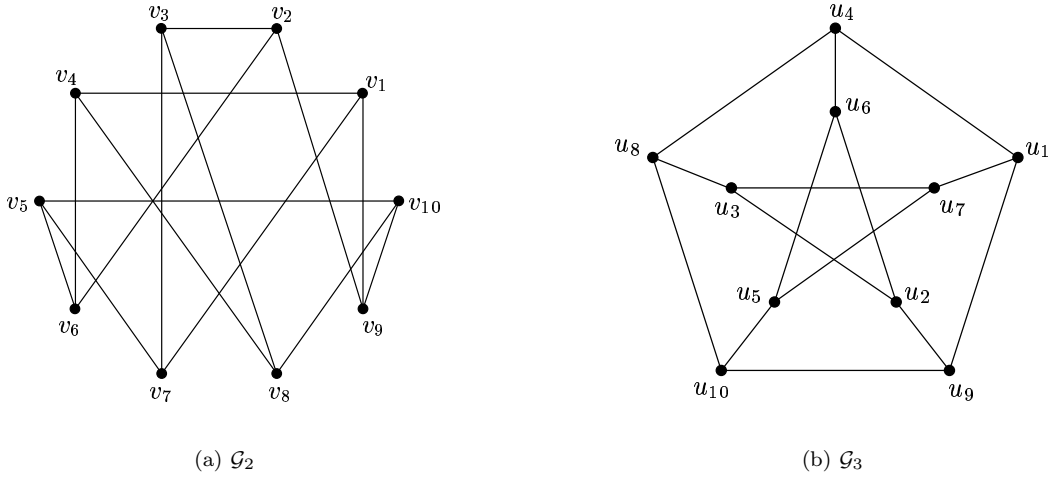


Figure 3.2: Two isomorphic graphs, both known as Petersen's graph.

which there exists no  $u-v$  path, is called **disconnected**. The graph  $\mathcal{G}_1$  in Figure 3.1(a) is disconnected, because there exists no path between the vertices  $v_i$  and  $v_6$  for all  $i \in \{1, \dots, 5, 7\}$  ( $v_6$  is an isolated vertex of  $\mathcal{G}_1$ ). If  $v_1 = v_n$  in the walk  $\mathcal{W}_{\mathcal{G}}$  and no other (internal) vertices in  $\mathcal{W}_{\mathcal{G}}$  are repeated, the walk  $\mathcal{W}_{\mathcal{G}}$  is called a **cycle of length**  $n - 1$  or an  $(n - 1)$ -**cycle** (denoted by  $C_{n-1}$ ). The graph  $K_8$  in Figure 3.1(b) contains a cycle  $C_i : v_1, \dots, v_i, v_1$ , for all  $i = 3, \dots, 8$ . The length of a shortest cycle (if any) in a graph  $\mathcal{G}$  is referred to as the **girth** of  $\mathcal{G}$ , denoted by  $g(\mathcal{G})$ . Both the graphs  $\mathcal{G}_1$  and  $K_8$  in Figure 3.1 have  $g(\mathcal{G}_1) = g(K_8) = 3$  while both the graphs  $\mathcal{G}_2$  and  $\mathcal{G}_3$  in Figure 3.2 have  $g(\mathcal{G}_2) = g(\mathcal{G}_3) = 5$ . The **distance**  $d_{\mathcal{G}}(u, v)$  between a vertex pair  $(u, v)$  of a graph  $\mathcal{G}$  is the minimum length of all  $u-v$  paths in  $\mathcal{G}$ , if any such paths exist, otherwise  $d_{\mathcal{G}}(u, v) = \infty$ . The **eccentricity**  $e_{\mathcal{G}}(v)$  of a vertex  $v$  in a graph  $\mathcal{G}$  is the distance from  $v$  to a vertex furthest from  $v$ , *i.e.*,

$$e_{\mathcal{G}}(v) = \max\{d_{\mathcal{G}}(v, u) \mid u \in V(\mathcal{G})\}.$$

For example, let  $\mathcal{G}'_1$  be the  $(7, 7)$  graph with  $V(\mathcal{G}'_1) = V(\mathcal{G}_1)$  and  $E(\mathcal{G}'_1) = E(\mathcal{G}_1) \cup \{v_4v_6, v_5v_6\}$ . Then  $e_{\mathcal{G}'_1}(v_1) = e_{\mathcal{G}'_1}(v_2) = e_{\mathcal{G}'_1}(v_3) = 5$ ,  $e_{\mathcal{G}'_1}(v_4) = e_{\mathcal{G}'_1}(v_7) = 4$  and  $e_{\mathcal{G}'_1}(v_5) = e_{\mathcal{G}'_1}(v_6) = 3$ . The **radius** of a graph  $\mathcal{G}$  (denoted by  $\text{rad}(\mathcal{G})$ ) is defined by

$$\text{rad}(\mathcal{G}) = \min\{e_{\mathcal{G}}(v) \mid v \in V(\mathcal{G})\},$$

while the **diameter** of a graph  $\mathcal{G}$  (denoted by  $\text{diam}(\mathcal{G})$ ) is defined by

$$\text{diam}(\mathcal{G}) = \max\{e_{\mathcal{G}}(v) \mid v \in V(\mathcal{G})\}.$$

Both the graphs  $\mathcal{G}_2$  and  $\mathcal{G}_3$  in Figure 3.2, for example, have radius and diameter 2. A **subgraph**  $\mathcal{H} = (V', E')$  of a graph  $\mathcal{G} = (V, E)$  is a graph with the properties that  $V' \subseteq V$  and  $E' \subseteq E$  [102].  $\mathcal{H} = (V', E')$  is said to be a **vertex-induced subgraph** of  $\mathcal{G} = (V, E)$  if  $\mathcal{H}$  is a subgraph of  $\mathcal{G}$  with the properties that  $V' \subseteq V$  and  $uv \in E'$  if  $uv \in E$  for all vertex pairs  $(u, v) \in V'$ . A subgraph  $\mathcal{H}$  of a graph  $\mathcal{G}$  is called a **component** of  $\mathcal{G}$  if  $\mathcal{H}$  is a maximally connected subgraph of  $\mathcal{G}$ . The subgraph  $\mathcal{H}_1 = (V, E)$ , defined by  $V(\mathcal{H}_1) = \{v_1, v_3, v_5, v_7\}$  and  $E(\mathcal{H}_1) = \{v_1v_3, v_1v_7, v_3v_7, v_5v_7\}$ , of the graph  $\mathcal{G}_1$  in Figure 3.1(a), is both a vertex-induced subgraph and a component of  $\mathcal{G}_1$ , while the graph  $\mathcal{H}_2 = (V, E)$ , defined by  $V(\mathcal{H}_2) = V(\mathcal{H}_1)$  and  $E(\mathcal{H}_2) = \{v_1v_3, v_3v_7, v_5v_7\}$ , is a subgraph of  $\mathcal{G}_1$ , but not a vertex-induced subgraph of  $\mathcal{G}_1$ . A **spanning subgraph**  $\mathcal{H}$  of a graph  $\mathcal{G}$  is a subgraph of  $\mathcal{G}$  with the property that  $V(\mathcal{H}) = V(\mathcal{G})$ . A **tree** is defined as a connected graph without cycles. Also, a **spanning tree** of a graph  $\mathcal{G}$  is a spanning subgraph of  $\mathcal{G}$  that is a tree.

Two graphical representations of the same graph may look completely different. Their similarities may only become noticeable by re-ordering and/or re-labelling the vertices in one of the representations. This

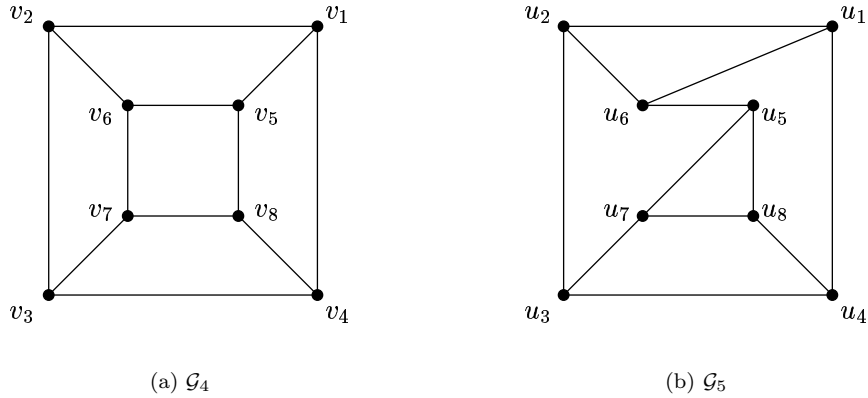


Figure 3.3: A vertex-transitive 3-regular graph  $\mathcal{G}_4$  and a non vertex-transitive 3-regular graph  $\mathcal{G}_5$ .

leads to the concept of an isomorphism. Two graphs  $\mathcal{G}$  and  $\mathcal{H}$  are **isomorphic** if there exists a bijective function  $\phi$ , mapping  $V(\mathcal{G})$  onto  $V(\mathcal{H})$  such that  $uv \in E(\mathcal{G})$  if and only if  $\phi(u)\phi(v) \in E(\mathcal{H})$ . In such a case the function  $\phi$  is called an **isomorphism** between  $\mathcal{G}$  and  $\mathcal{H}$  [122]. An **automorphism**  $\alpha$  of a graph  $\mathcal{G}$  is an isomorphism of  $\mathcal{G}$  onto itself, meaning that  $\alpha$  is a permutation of the set  $V(\mathcal{G})$  which preserves adjacency (more on this subject may be obtained from [42, 101]). The set of all automorphisms of a graph  $\mathcal{G}$  form a group, called the **automorphism group** of  $\mathcal{G}$ . The two graphs  $\mathcal{G}_2$  and  $\mathcal{G}_3$  in Figure 3.2 are isomorphic, the function  $\phi(v_i) = u_i$  ( $i = 1, 2, \dots, 10$ ) being an isomorphism. We denote this isomorphism by writing  $\mathcal{G}_2 \simeq \mathcal{G}_3$ . The **complement** of a graph  $\mathcal{G}$ , denoted by  $\overline{\mathcal{G}}$ , is a graph with  $V(\overline{\mathcal{G}}) = V(\mathcal{G})$ , that contains the edge  $uv$  if and only if the edge  $uv$  is not an edge of  $\mathcal{G}$ . Therefore, the complement of the graph  $\mathcal{G}_1$  in Figure 3.1(a) has vertex set  $V(\overline{\mathcal{G}_1}) = V(\mathcal{G}_1)$  and edge set  $E(\overline{\mathcal{G}_1}) = \{v_1v_2, v_1v_4, v_1v_5, v_1v_6, v_2v_3, v_2v_5, v_2v_6, v_2v_7, v_3v_4, v_3v_5, v_3v_6, v_4v_5, v_4v_6, v_4v_7, v_5v_6, v_6v_7\}$ .

If  $\deg_{\mathcal{G}}(v) = r$  for all vertices  $v \in V(\mathcal{G})$  and some  $r \in \mathbb{N}_0$ , then the graph  $\mathcal{G}$  is said to be  **$r$ -regular**, or **regular of degree  $r$** . In general, a graph is said to be **regular** if it is  $r$ -regular. If an order  $n$  graph,  $\mathcal{G}$ , is  $r$ -regular such that every pair of adjacent vertices has  $n_a$  common neighbours, and every pair of distinct nonadjacent vertices has  $n_c$  common neighbours, then  $\mathcal{G}$  is called **strongly regular** with parameters  $(n, r, n_a, n_c)$  [92]. A graph  $\mathcal{G}$  is said to be **vertex-transitive** if, for all vertex pairs  $(u, v) \in V(\mathcal{G})$ , there is an automorphism of  $\mathcal{G}$  that maps  $u$  to  $v$  [97]. Simplistically this means that “every vertex is like every other vertex” in a vertex-transitive graph (at the very least, vertex-transitive graphs have to be regular). Equivalently, a graph  $\mathcal{G}$  is said to be **edge-transitive** if, for all vertex pairs  $(u, v) \in V(\mathcal{G})$  with  $uv \in E(\mathcal{G})$ , there is an automorphism  $\alpha$  of  $\mathcal{G}$  that maps  $uv$  to  $\alpha(u)\alpha(v)$ . Both the graphs  $\mathcal{G}_2$  and  $\mathcal{G}_3$  in Figure 3.2 are 3-regular and vertex-transitive. Moreover, they are both strongly regular with parameters  $(10, 3, 0, 1)$ . The graph  $\mathcal{G}_4$  in Figure 3.3(a) is vertex-transitive, while the graph  $\mathcal{G}_5$  in Figure 3.3(b) is neither vertex-transitive (vertex  $u_3$  forms part of no 3-cycle, while 3-cycles exist in  $\mathcal{G}_5$ ) nor strongly regular (the adjacent vertex pair  $u_1$  and  $u_2$  share a single common neighbouring vertex, while the adjacent vertex pair  $u_3$  and  $u_4$  share no common neighbouring vertex; the non-adjacent vertex pair  $u_2$  and  $u_4$  share two common neighbouring vertices, while the non-adjacent vertex pair  $u_6$  and  $u_8$  only share a single common neighbouring vertex). Vertex-transitivity [edge-transitivity] of a graph may be advantageous in certain applications. Given a certain (possibly greedy) algorithm to perform some task involving vertices [edges] of the graph, any vertex [edge] may be used for initialisation of the algorithm without loss of generality or efficiency of the algorithm in the case of a vertex-transitive [edge-transitive] graph. A graph of order  $p$  that is  $(p-1)$ -regular is referred to as a **complete graph** and is denoted by  $K_p$ <sup>2</sup> (see Figure 3.1(b)). It trivially follows that the density of  $K_p$  is 1, for all  $p \in \mathbb{N}$ .

A graph  $\mathcal{G}$  is said to be **dominated** by a vertex subset  $\mathcal{D} \subseteq V(\mathcal{G})$  if every vertex of  $\mathcal{G}$  is either an element of  $\mathcal{D}$ , or adjacent to an element of  $\mathcal{D}$ . In such a case  $\mathcal{D}$  is called a **dominating set** of  $\mathcal{G}$  (originally

<sup>2</sup>By definition, the graphs  $K_p$  and  $\overline{K_p}$  are not considered strongly regular, because  $K_p$  [ $\overline{K_p}$ ] only has [non]adjacent vertices.

defined by Berge [22] and Ore [197] as **externally stable sets**). By definition, a graph  $\mathcal{G} = (V, E)$  is dominated by its own vertex set,  $V$ . A dominating set  $\mathcal{D}$  is **minimal dominating** if no proper subset of  $\mathcal{D}$  is a dominating set of  $\mathcal{G}$ . The minimum cardinality of a minimal dominating set of a graph  $\mathcal{G}$  is denoted by  $\gamma(\mathcal{G})$  (called the **(lower) domination number** of  $\mathcal{G}$ ). Similarly, a set  $\mathcal{B} \subseteq V(\mathcal{G})$  is called an **independent set** of  $\mathcal{G}$  if no two vertices in  $\mathcal{B}$  are adjacent in  $\mathcal{G}$ , and an independent set  $\mathcal{B}$  is **maximal independent** if no proper superset of  $\mathcal{B}$  is an independent set of  $\mathcal{G}$ . The maximum cardinality of a maximal independent set of  $\mathcal{G}$  is denoted by  $\beta(\mathcal{G})$  (called the **vertex independence number** of  $\mathcal{G}$ ). For example, the set  $\mathcal{D}_1 = \{u_1, u_2, u_6, u_8\}$  is a dominating set for the graph  $\mathcal{G}_3$  in Figure 3.2(b), although it is not minimal dominating. However, the set  $\mathcal{D}_2 = \mathcal{D}_1 \setminus \{u_2\}$  is a minimal dominating set (of minimum cardinality) for  $\mathcal{G}_3$ , because no proper subset of  $\mathcal{D}_2$  yields a dominating set for  $\mathcal{G}_3$ , implying that  $\gamma(\mathcal{G}_3) \leq 3$ . Because no dominating set of cardinality 2 exists for  $\mathcal{G}_3$ , it follows that, in fact,  $\gamma(\mathcal{G}_3) = 3$ . Similarly, the set  $\mathcal{B}_1 = \{u_1, u_7\}$  is an independent set for the graph  $\mathcal{G}_5$  in Figure 3.3(b), although it is not maximal independent. The set  $\mathcal{B}_2 = \{u_1, u_3, u_8\}$ , however, is a maximal independent set (of maximum cardinality) for  $\mathcal{G}_5$ , because no proper superset of  $\mathcal{B}_2$  yields an independent set for  $\mathcal{G}_5$  and no independent set of cardinality 4 exists for  $\mathcal{G}_5$ , implying that  $\beta(\mathcal{G}_5) = 3$ . In general, it may be shown that  $\gamma(\mathcal{G}) \leq \beta(\mathcal{G})$  for any graph  $\mathcal{G}$  [103]. For further information on domination, independence and related notions, the reader may consult [103]. Similarly, a graph  $\mathcal{G}$  is said to be  $100\psi\%$ -**partially dominated** (where  $0 < \psi \leq 1$ ) by a vertex subset  $\partial_\psi \subseteq V(\mathcal{G})$  if there exists a vertex-induced subgraph  $\mathcal{H}$  of  $\mathcal{G}$  for which  $\partial_\psi$  is a dominating set with  $|V(\mathcal{H})| \geq \lceil \psi |V(\mathcal{G})| \rceil$  (i.e., every vertex in  $V(\mathcal{H})$  is either an element of  $\partial_\psi$ , or adjacent to an element in  $\partial_\psi$ ). In such a case,  $\partial_\psi$  is called a  $100\psi\%$ -**partial dominating set** of  $\mathcal{G}$ . A  $100\psi\%$ -partial dominating set  $\partial_\psi$  is **minimal  $100\psi\%$ -partial dominating** if no proper subset of  $\partial_\psi$  is a minimal  $100\psi\%$ -partial dominating set of  $\mathcal{G}$ . The minimum cardinality of a minimal  $100\psi\%$ -partial dominating set of a graph  $\mathcal{G}$  is denoted by  $\gamma_\psi(\mathcal{G})$  (called the **(lower) partial domination number**<sup>3</sup> of  $\mathcal{G}$ ). For example, the set  $\bar{\partial}_{0.7} = \mathcal{D}_2 \setminus \{u_6\}$  is a minimal  $70\%$ -partial dominating set (of minimum cardinality) for the graph  $\mathcal{G}_3$  in Figure 3.2(b), implying that  $\gamma_{0.7}(\mathcal{G}_3) = 2$ . In general, it may be shown that  $\gamma_\psi(\mathcal{G}) \leq \gamma_1(\mathcal{G}) = \gamma(\mathcal{G})$  for all  $0 < \psi \leq 1$ .

Algorithms involving graph manipulations may easily be performed with the aid of computers. Graphs are usually represented in computer memory using the notion of an **adjacency matrix** (although other more efficient memory representations also exist). The adjacency matrix for an order  $p$  graph  $\mathcal{G}$  is a  $p \times p$  matrix  $A_{\mathcal{G}} = [a_{ij}]$ , where  $a_{ij} = 1$  if and only if the vertex pair  $(i, j)$  is adjacent or  $i = j$ , while  $a_{ij}$  takes the value 0 otherwise<sup>4</sup>. For example, the adjacency matrix for the graph  $\mathcal{G}_4$  in Figure 3.3(a) is given by

$$A_{\mathcal{G}_4} = \begin{matrix} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{matrix} & \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{matrix}.$$

The purpose of algorithmic complexity analysis is usually to predict algorithmic behaviour independently from implementation details, such as the programming language or hardware used [148]. Although it is typically impossible to predict the *exact* behaviour of any algorithm, an *approximation* is usually made during algorithmic complexity analyses, extracting the main characteristics of the algorithm's behaviour in some sense (such as an average case or worst case scenario). Grimaldi [96] stipulates five conditions that have to hold in order that the complexity of an algorithm may be studied.

**Algorithmic complexity** is usually measured by two functional variables: the **time complexity**  $T$  and **space complexity**  $S$  of the algorithm. The functions  $T$  and  $S$  are usually written as  $T(\theta)$  and

<sup>3</sup>Hence,  $\gamma_1(\mathcal{G}) = \gamma(\mathcal{G})$  for a graph  $\mathcal{G}$ . In the special case where  $\psi = 1$ , the parameter  $\gamma(\mathcal{G})$  will be used (instead of the general notation,  $\gamma_1(\mathcal{G})$ ).

<sup>4</sup>The additional requirement that the main diagonal elements of an adjacency matrix take the value 1, yields a non-standard definition. However, this extra condition ties in very nicely with the notion of domination (since every vertex of a graph dominates itself) as well as with the adjacency parameters  $a_{ij}$  in the integer programming formulation (2.10)–(2.13), and will hence be adopted here. The reader should not infer from this that the graph contains a loop at every vertex. All graphs in this dissertation are classical or simple graphs (as opposed to pseudographs or multigraphs [45]).

$S(\theta)$ , where  $\theta$  refers to the size of the algorithm input and these functions measure respectively the number of basic operations performed by the algorithm and the amount of memory required by the algorithm in terms of  $\theta$ . The order of magnitude of the algorithmic complexity, denoted by  $\mathcal{O}$  (“big  $\mathcal{O}$ ”), is given by the term (disregarding the coefficient) which grows fastest as the input size  $\theta$  increases [45, 88]. For example, if  $T(\theta) = 7\theta^3 + 19\theta^2 + 2\theta + 31$ ,  $T$  is said to grow asymptotically as  $\theta^3$  grows, denoted by  $T(\theta) = \mathcal{O}(\theta^3)$  [229]. More precisely speaking, let  $f$  and  $g$  be two functions. Then we write  $f(\theta) = \mathcal{O}(g(\theta))$  if there exists a  $c \in \mathbb{R}^+$  and  $\theta_0 \in \mathbb{N}$  such that  $0 \leq f(\theta) \leq c \cdot g(\theta)$  for all  $\theta \geq \theta_0$ . The function  $g$  is said to be an asymptotic upper bound for  $f$ . An algorithm is called a **polynomial time algorithm** if its time complexity may be bounded asymptotically by a polynomial with respect to input size. If a problem may be solved by a polynomial time algorithm, the problem is called **tractable** and the relevant algorithm is called **time efficient**. Conversely, if no such polynomial time algorithm is known, the problem investigated is called an **intractable** or hard problem. Trade-offs usually exist between the time and space complexities of an algorithm, in the sense that attempts at reducing the time complexity cause an increase in the memory required to execute the algorithm (and hence the space complexity), and *vice-versa*. All references to algorithmic complexity in this dissertation will be in terms of the time complexity (the reader should, however, keep in mind that space complexity estimates are also important – the space complexity of an extremely time efficient algorithm often renders the algorithm impractical for large instances of the problem at hand).

Decision theory is that branch of complexity theory in which the problem to be solved is interpreted as a binary question that may be answered either “yes” or “no” (it is possible to show, without loss of generality, that all computational problems may be reduced to decision problems). The class of decision problems **P** (acronym for *Polynomial*) is defined as the set of all decision problems that may be solved by polynomial time algorithms. For example, the problem of deciding whether or not there exists a path of length  $n$  in some graph may be solved by a polynomial time algorithm and is therefore a problem in the class **P**. The class of decision problems **NP** (acronym for *Non-deterministic Polynomial*) comprises the set of all decision problems which may be answered “yes” by a polynomial time algorithm, given additional information (called a certificate to the problem instance at hand). Suppose  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are two decision problems. We write  $\mathcal{A}_1 \preceq \mathcal{A}_2$  if an algorithm exists that solves  $\mathcal{A}_1$  as a subroutine of an algorithm that solves  $\mathcal{A}_2$  (*i.e.*, if  $\mathcal{A}_1 \preceq \mathcal{A}_2$ , then  $\mathcal{A}_2$  is computationally at least as hard as  $\mathcal{A}_1$ ). A decision problem  $\mathcal{A}$  is called **NP**-complete if  $\mathcal{A} \in \mathbf{NP}$  and  $\mathcal{A}_1 \preceq \mathcal{A}$  for all  $\mathcal{A}_1 \in \mathbf{NP}$  (*i.e.*, **NP**-complete decision problems are at least as hard to solve, from a computational point of view, as *any* other problem in **NP**). For example, no polynomial time algorithm is known to solve the problem of deciding whether or not there exists a packing set of cardinality at least  $\lambda$  for  $\langle m, n; k \rangle$ . The packing problem may therefore not be classified as a member of the class **P**. However, given the trivial certificate consisting of a packing set  $\mathcal{P}$  of cardinality  $\lambda$  for  $\langle m, n; k \rangle$ , the question “Does there exist a packing set of cardinality at least  $\lambda$ ?” may be answered in the affirmative within polynomial time, by merely affirming the validity of  $\mathcal{P}$  as a packing set for  $\langle m, n; k \rangle$ . Hence the packing problem is a member of the class **NP**. In fact, it is also a member of the class **NP**-complete (the reader is referred to a landmark paper by Karp [125] in 1972 presenting 21 intractable combinatorial computational problems that are all **NP**-complete). Other **NP** classes also exist, although they do not fall within the scope of this dissertation.

Using the above fundamentals from graph and complexity theory, the concept of a lottery graph is introduced and formalised in the following section.

## 3.2 Definition and properties of the lottery graph

Following a suggestion by Berge [20], Di Paola [65] introduced the notion of a *graph on the binomial coefficient*  $\binom{m}{n}$  in 1966.

**Definition 3.1 (Graph  $\mathcal{G}_B(m, n; \lambda)$  on the binomial coefficient [65])** A graph  $\mathcal{G}_B(m, n; \lambda)$  on the binomial coefficient  $\binom{m}{n}$  with edge parameter  $\lambda$  is a graph whose vertices are the  $\binom{m}{n}$  possible  $n$ -sets which may be formed from  $m$  elements and having as adjacent vertices those pairs of vertices which have more than  $\lambda$ , but less than  $n$  elements in common. ■

This definition also relates to the family of graphs  $J(m, n; k)$ <sup>5</sup>, defined below, that has been used to translate many combinatorial problems to within the realm of graph theory [92].

**Definition 3.2 (The graph  $J(m, n; k)$  [92])** Let  $m, n$  and  $k$  be positive integers, with  $k \leq n \leq m$  and let  $\Theta$  be a fixed set of elements of cardinality  $m$ . The vertices of  $J(m, n; k)$  represent the subsets from  $\Theta$  of cardinality  $n$ , and two vertices in  $J(m, n; k)$  are adjacent if the intersection of their corresponding subsets of  $\Theta$  has cardinality  $k$ . ■

Using the above definitions, the notion of a lottery graph may now be introduced.

**Definition 3.3 (The lottery graph  $G\langle m, n; k \rangle$ )** The lottery graph, denoted by  $G\langle m, n; k \rangle$ , is defined as the graph  $\mathcal{G}_B(m, n; k - 1)$ , which has as vertex set,  $V(G\langle m, n; k \rangle)$ , all possible  $\binom{m}{n}$   $n$ -sets from  $\mathcal{U}_m = \{1, \dots, m\}$  in the lottery  $\langle m, n; k \rangle$  (hence  $V(G\langle m, n; k \rangle) = \Phi(\mathcal{U}_m, n)$ ). Two vertices in  $G\langle m, n; k \rangle$  are adjacent if the corresponding two  $n$ -sets share a common  $k$ -subset<sup>6</sup>. ■

The complete [incomplete] lottery problem then translates to determining the lower [partial] domination number,  $\gamma(G\langle m, n; k \rangle)$  [ $\gamma_\psi(G\langle m, n; k \rangle)$ ], of the lottery graph, because (according to the definition of a [100 $\psi$ %-partial] dominating set) any winning  $n$ -set [a proportion,  $\psi$ , of all possible winning  $n$ -sets]  $w$  in  $\langle m, n; k \rangle$  will either be (i) in a [100 $\psi$ %-partial] dominating set of  $G\langle m, n; k \rangle$  or (ii) adjacent to an  $n$ -set in the [100 $\psi$ %-partial] dominating set: So, if a participant of the lottery  $\langle m, n; k \rangle$  plays a [100 $\psi$ %-partial] dominating set of the lottery graph, he/she will win the jackpot in the former case, or else at least a  $k$ -matching prize [with probability  $\psi$ ] in the latter case. The following simple example is presented to provide the reader with an impression of the newly defined lottery graph and graph theoretic translations of the complete and incomplete lottery problems.

**Example 3.1 (continuation of Example 1.2)** Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 1.2, where a participant forms a playing set  $\mathcal{L}$  by selecting 3-sets from  $\mathcal{U}_7 = \{1, \dots, 7\}$  and wins a prize if some element in  $\mathcal{L}$  has a 2-set in common with the winning 3-set. The lottery graph  $G\langle 7, 3; 2 \rangle$  has the vertex set  $V(G\langle 7, 3; 2 \rangle)$  consisting of all  $\binom{7}{3} = 35$  possible 3-sets from  $\mathcal{U}_7$  and two vertices are adjacent if their corresponding 3-sets share a pair from  $\mathcal{U}_7$ , thereby defining the edge set  $E(G\langle 7, 3; 2 \rangle)$ . This graph is shown schematically in Figure 3.4. The complete lottery set  $\tilde{\mathcal{L}} = \{\{1, 2, 3\}, \{1, 5, 7\}, \{2, 5, 7\}, \{3, 4, 6\}\}$  presented in Example 1.1 is an  $L_1(7, 3; 2)$ -set and hence a minimal dominating set for  $G\langle 7, 3; 2 \rangle$  of minimum cardinality, and is denoted by embossed vertices in Figure 3.4. Hence  $L_1(7, 3; 2) = \gamma(G\langle 7, 3; 2 \rangle) = 4$ . The reader may also verify that the playing set  $\hat{\mathcal{L}} = \{\{1, 5, 7\}, \{3, 4, 6\}\}$  collectively dominates 26 vertices (the two vertex-induced subgraphs  $\mathcal{H}_{\{1,5,7\}}$  and  $\mathcal{H}_{\{3,4,6\}}$  of  $G\langle 7, 3; 2 \rangle$  dominated by the vertices  $\{1, 5, 7\}$  and  $\{3, 4, 6\}$  of  $G\langle 7, 3; 2 \rangle$  respectively, are presented in Figure 3.5). Since the vertex sets  $V(\mathcal{H}_{\{1,5,3\}})$  and  $V(\mathcal{H}_{\{3,4,6\}})$  are disjoint, no other combination of two 3-sets yielding an improvement on the number of vertices collectively dominated by  $\hat{\mathcal{L}}$  may be found. Hence it follows that  $\Psi_2(7, 3; 2) = \frac{26}{35}$  and  $L_{\frac{26}{35}}(7, 3; 2) = \gamma_{\frac{26}{35}}(G\langle 7, 3; 2 \rangle) = 2$ . ■

The construction of a minimal dominating set of minimum cardinality of a graph may also be achieved via an integer programming approach (as in (2.10)). Here the binary decision variables  $x_i$  refer to whether or not a vertex should be included in the complete dominating set, with the objective of minimising the number of vertices,  $y_1$ , in the dominating set. The adjacency parameters  $a_{ij}$  in (2.10) may be seen as the entries of the adjacency matrix of  $G\langle m, n; k \rangle$ . The problem of efficiently finding a dominating set of minimum cardinality for a general graph is still an unresolved problem to this day, with no known polynomial time algorithm for determining such a set. In particular, the problem of finding the lower partial domination number of a lottery graph may be reduced to the following decision problem.

<sup>5</sup>One specific subset of the family of graphs  $J(m, n; k)$ , is the so-called *Johnson graph* denoted by  $\mathcal{G}_B(m, n; n - 2) \simeq J(m, n; n - 1)$ .

<sup>6</sup>Computer representation of the lottery graph  $G\langle m, n; k \rangle$  relates to so-called combinatorial matrices. In general, a **combinatorial matrix of type**  $(m, n)$  has  $\binom{m}{n}$  rows and columns indexed by the  $n$ -element subsets of an  $m$ -element set  $\mathcal{U}_m$ . The matrix entries  $a_{uv}$  (in row  $u$  and column  $v$ ) should depend only on the cardinality  $|u \cap v|$  [127]. Hence the adjacency matrix for  $G\langle m, n; k \rangle$  may be defined as  $A_{G\langle m, n; k \rangle} = a_{ij}$  where  $a_{ij} = 1$  if  $a_{uv} \geq k$  for  $i = u$  and  $j = v$  and  $a_{ij} = 0$  otherwise.



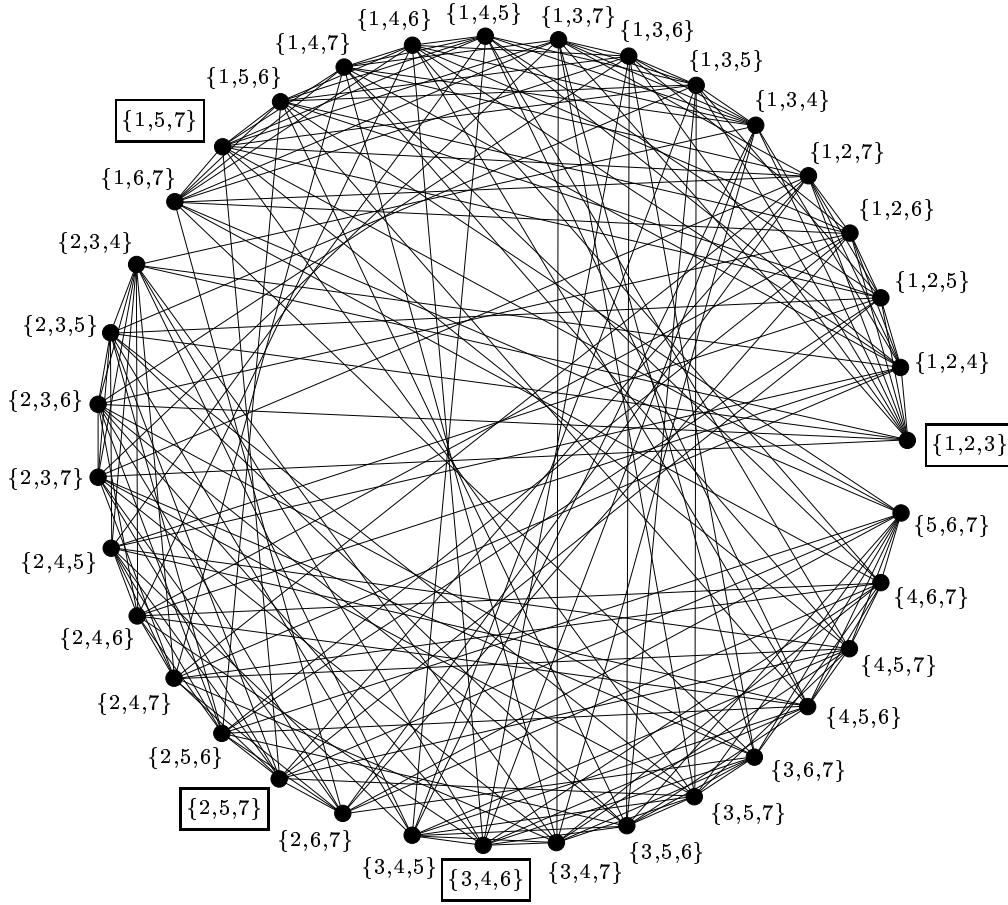


Figure 3.4: The lottery graph  $G(7, 3; 2)$ . Emboxed vertices indicate elements of the (dominating or) complete lottery set  $\hat{\mathcal{L}} = \{\{1, 2, 3\}, \{1, 5, 7\}, \{2, 5, 7\}, \{3, 4, 6\}\}$ . The playing set  $\hat{\mathcal{L}} = \{\{1, 5, 7\}, \{3, 4, 6\}\}$  collectively dominates 26 vertices (13 unique vertices by each element of  $\hat{\mathcal{L}}$ ) and hence is a  $\frac{26}{35}$ -partial dominating set for  $G(7, 3; 2)$  (i.e.,  $\Psi_2(7, 3; 2) = \frac{26}{35}$  and  $L_{\frac{26}{35}}(7, 3; 2) = \gamma_{\frac{26}{35}}(G(7, 3; 2)) = 2$ ).

Decision problem: **INCOMPLETE LOTTERY SET**

**INSTANCE:** A lottery graph  $G(m, n; k)$ , a positive integer  $\ell$  and a real number  $0 < \psi \leq 1$ .

**QUESTION:** Does  $G(m, n; k)$  have a  $100\psi\%$ -partial dominating set of cardinality not exceeding  $\ell$ ?

The following result is derived from the well-known **NP**-completeness of finding the lower domination parameter,  $\gamma$ , for a graph.

### Theorem 3.2 (Incomplete lottery set)

The decision problem **INCOMPLETE LOTTERY SET** is **NP**-complete.

#### Proof

See [103, 148] for a proof that the special case of the complete lottery problem ( $\psi = 1$ ), is **NP**-complete. Hence the general case of the incomplete lottery problem is also **NP**-complete. ■

It is possible to determine certain properties of the lottery graph. In order to prove the following theorem, the notion of the complement of a vertex is required. The **vertex complement**  $v'$  of a vertex  $v$  in the lottery graph  $G(m, n; k)$  is defined by the  $(m - n)$ -set consisting of all the elements from  $\mathcal{U}_m$  that are not contained in the label of  $v$  (e.g., the vertex  $\{1, 3, 7\}$  in the lottery graph  $G(7, 3; 2)$  in Figure 3.4 has vertex complement  $\{2, 4, 5, 6\}$ ).

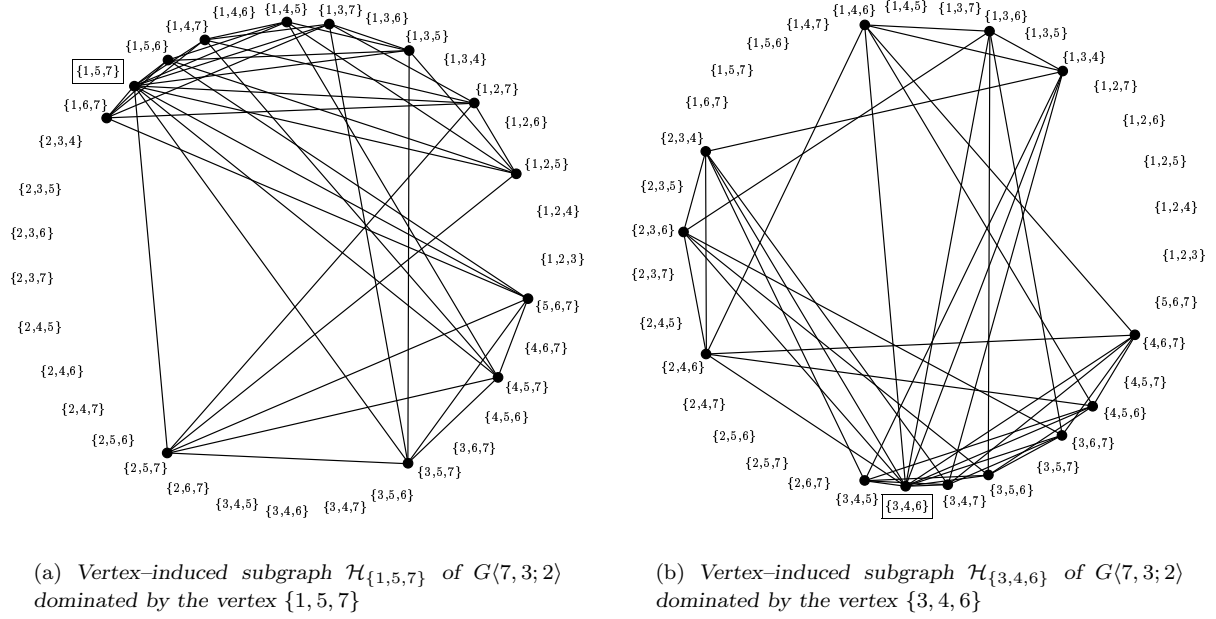


Figure 3.5: Vertex-induced subgraphs of  $G(7, 3; 2)$  dominated by the vertices (a)  $\{1, 5, 7\}$  and (b)  $\{3, 4, 6\}$ , respectively.

**Theorem 3.3 (Properties of the lottery graph  $G(m, n; k)$ )**

- (a) The lottery graph  $G(m, n; k)$  is  $r$ -regular, with  $r$  given as in (2.1) for any  $1 \leq k \leq n \leq m$ .  
 (b) The lottery graph  $G(m, n; k)$  is vertex-transitive for any  $1 \leq k \leq n \leq m$ .  
 (c) The density of the lottery graph  $G(m, n; k)$  is  $\frac{r}{\binom{m}{n}-1}$ , with  $r$  given in (2.1) for any  $1 \leq k \leq n \leq m$ .  
 (d)

$$\begin{aligned} \text{Rad}(G(m, n; k)) &= \text{diam}(G(m, n; k)) \\ &= \begin{cases} \left\lceil \frac{n}{n-k} \right\rceil & \text{if } m \geq 2n \text{ and } k < n, \\ \left\lceil \frac{m-n}{n-k} \right\rceil & \text{if } m < 2n \text{ and } k < n, \\ \infty & \text{if } n = k. \end{cases} \end{aligned} \quad (3.2)$$

- (e)  $G(m, n; k) \simeq G(m, m-n; m+k-2n)$ , for all  $1 \leq k < n < m$  satisfying  $m+k > 2n$ .  
 (f)  $g(G(m, n; k)) = 3$ , for all  $1 \leq k < n < m$ .  
 (g) The only class strongly regular lottery graphs are  $G(m, 2; 1)$  with parameters  $(\binom{m}{2}, r, m-2, 4)$  for all  $m \geq 2n$ , where  $r$  is given in (2.1).

**Proof**

(a) Two vertices in  $G(m, n; k)$  are adjacent if and only if they share a common  $i$ -subset for all  $k \leq i \leq n-1$ , yielding the degree of regularity  $r$  as in (2.1), according to the argument used to prove Lemma 2.1.

(b) Define  $\Pi$  as the set of all possible permutations of the elements of  $\mathcal{U}_m$ . The function  $\pi(v)$  (with  $\pi \in \Pi$  and  $v \in \Phi(\mathcal{U}_m, n) = V(G(m, n; k))$ ) permutes the elements of  $v = \{v_1, \dots, v_n\}$  according to the elements of  $\pi$  (i.e.,  $\pi(v) = \{\pi(v_1), \dots, \pi(v_n)\}$ ). The set of all possible functions  $\pi(v)$  with  $\pi \in \Pi(\mathcal{U}_m)$  and  $v \in V(G(m, n; k))$  form an automorphism group on  $V(\mathcal{G})$  (i.e., adjacency between any two vertices in  $G(m, n; k)$  is preserved, given any permutation of the elements of  $\mathcal{U}_m$ ). This follows by the ubiquity of the roles of the numbers in  $\mathcal{U}_m$ , which may be seen as arbitrary symbols. Therefore  $G(m, n; k)$  is vertex-transitive.

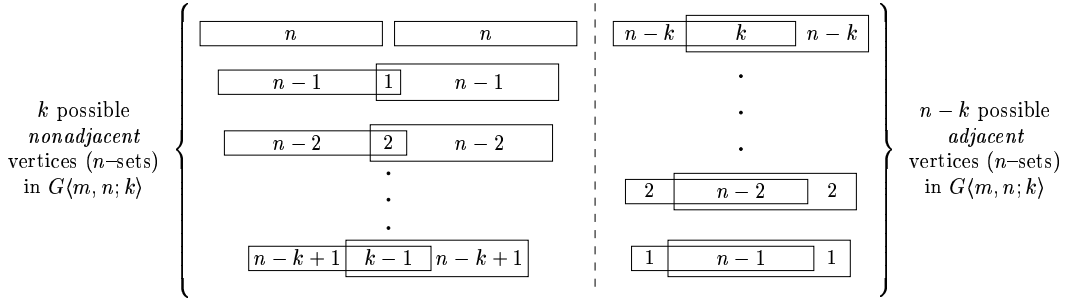


Figure 3.6: All possible configurations of [non]adjacent vertices in  $G\langle m, n; k \rangle$ .

(c) From (3.1) we have that the density of the lottery graph  $G\langle m, n; k \rangle$  of order  $p$  and size  $q$  is given by

$$\frac{q}{\binom{p}{2}} = \frac{\frac{1}{2} \sum_{i=1}^p \deg_{G\langle m, n; k \rangle}(v_i)}{\frac{p(p-1)}{2}} = \frac{\binom{m}{n} r}{\binom{m}{n} (\binom{m}{n} - 1)} = \frac{r}{\binom{m}{n} - 1}. \quad (3.3)$$

(d) The first equality in (3.2) follows from the fact that the lottery graph  $G\langle m, n; k \rangle$  is vertex-transitive (yielding equal eccentricities for all vertices). Furthermore, suppose the two vertices  $v_0$  and  $v$  are furthest apart in  $G\langle m, n; k \rangle$  and that their labels have  $x$  elements of  $\mathcal{U}_m$  in common. Then  $x = 0$  if  $m > 2n$  and  $x = 2n - m$  if  $m < 2n$ . Therefore, construct a shortest path  $\mathcal{W} : v_0, v_1, v_2, \dots, v_{l-1}, v$  in the following way: For any  $v_i$  select a neighbouring vertex  $v_{i+1}$  in  $G\langle m, n; k \rangle$  whose label has the most elements of  $\mathcal{U}_m$  in common with  $v_i$ . Because the labels of any two neighbouring vertices in  $G\langle m, n; k \rangle$  differ in at most  $n - k$  elements of  $\mathcal{U}_m$ , it follows that the labels  $v_i$  and  $v$  have exactly  $x + i(n - k)$  elements of  $\mathcal{U}_m$  in common, for all  $0 \leq i \leq l - 1$ . Because  $v_{l-1} \in N(v)$  (and therefore shares a common  $k$ -subset with  $v$ ), it follows that  $x + (l - 1)(n - k) \geq k$ . This yields  $l \geq \lceil n/(n - k) \rceil$  if  $m \geq 2n$  and  $l \geq \lfloor (m - n)/(n - k) \rfloor$  if  $m < 2n$ . Since  $l$  is minimal, the desired result follows. Note that  $G\langle m, n; n \rangle$  contains  $\binom{m}{n}$  isolated vertices and therefore no edge. It therefore follows that  $\text{rad}(G\langle m, n; n \rangle) = \infty$ .

(e) Let  $u'$  and  $v'$  be vertex complements of  $u$  and  $v$  respectively in  $G\langle m, n; k \rangle$ . It remains to be shown that  $u$  and  $v$  share a  $k$ -subset if and only if  $u'$  and  $v'$  share a  $(m + k - 2n)$ -subset. Suppose the labels of  $u$  and  $v$  have  $k$  elements in common. The number of elements common to the labels of  $u'$  and  $v'$  is  $m - |\{\text{elements common to } u \text{ and } v\} + \{\text{elements unique to } u\} + \{\text{elements unique to } v\}| = m - [k + (n - k) + (n - k)] = m + k - 2n$ . A similar argument may be used to prove the converse. Hence two vertices in  $G\langle m, n; k \rangle$  are adjacent if and only if the complements of these vertices are adjacent in  $G\langle m, m - n; m + k - 2n \rangle$ .

(f) It is shown that the labels of at least 3 vertices share a common  $k$ -subset in  $G\langle m, n; k \rangle$ , if  $1 \leq k < n < m$ . For this we require that  $\binom{m-k}{3} > 0$  when considering (say) the lexicographic ordered vertex labels  $\{1, \dots, n-1, n\}$ ,  $\{1, \dots, n-1, n+1\}$  and  $\{1, \dots, n-1, n+2\}$  (sharing at least a  $k$ -subset). This condition holds if  $k \leq m - 3$ . The special (worst) case where  $k = m - 2$  results in  $G\langle m, m - 1; m - 2 \rangle \simeq K_m$ , having girth 3.

(g) Consider all the possible [non]adjacent configurations of vertices in  $G\langle m, n; k \rangle$ , as shown in Figure 3.6. A unique configuration is required for both adjacent and nonadjacent vertices, because the common neighbourhood shared by *any* two [non]adjacent vertices (say  $u$  and  $v$ ) in  $G\langle m, n; k \rangle$  is dependent on (an increasing function of) the common elements between  $u$  and  $v$ , as will be argued later in this section. Requiring a unique [non]adjacent vertex configuration (*i.e.*,  $[k = 1] \ n - k = 1$ ) applies only to lotteries of the form  $\langle m, 2; 1 \rangle$ . In the case of the vertices  $u$  and  $v$  being adjacent (sharing  $k = 1$  element), the common neighbourhood cardinality is given by  $\xi_2^1(m, 2; 1) = m - 2$  in (2.4). In the case were  $u$  and  $v$  are nonadjacent, the equation in (2.4) reduces to  $\xi_2^0(m, 2; 1) = 4$ . ■

The graph  $G\langle 7, 3; 2 \rangle$  in Figure 3.4 of Example 3.1 is  $r = \binom{3}{2} \binom{4}{1} = 12$ -regular according to (2.1), a radius and diameter of 3 according to (3.2) (attained by the path  $\{1, 2, 3\}$ ,  $\{1, 2, 5\}$ ,  $\{2, 5, 6\}$ ,  $\{5, 6, 7\}$ ). The girth is given by  $g(G\langle 7, 3; 2 \rangle) = 3$  with a 3-cycle being  $\{1, 2, 3\}$ ,  $\{1, 2, 4\}$ ,  $\{1, 2, 5\}$ ,  $\{1, 2, 3\}$ . According

to Theorem 3.3(e),  $G\langle 7, 3; 2 \rangle \simeq G\langle 7, 4; 3 \rangle$ , implying that  $L_1(7, 3; 2) = L_1(7, 4; 3) = 4$  by Example 1.1, so that no search for minimal (partial) dominating sets in the lottery graph  $G\langle 7, 4; 3 \rangle$  is necessary, if such sets are known for  $G\langle 7, 3; 2 \rangle$ .

The following corollary is a direct consequence of Theorem 2.3 and Theorem 3.3.

**Corollary 3.1 (Properties of the lottery graph  $G\langle m, n; k \rangle$ )**

- (a) The lottery graph  $G\langle m, n; k \rangle$  is edge-transitive for all  $1 \leq k \leq n \leq m$ .
- (b) The lottery graph  $G\langle m, n; k \rangle$  is connected for all  $1 \leq k < n \leq m$ .
- (c) The lottery graph  $G\langle m, n; k \rangle \simeq K_{\binom{m}{n}}$  for all  $1 \leq k < n \leq m$  satisfying  $2n \geq m + k$ .

**Proof**

- (a) For any permutation  $\rho \in \Pi(\mathcal{U}_m)$  the automorphism defined by  $\pi_\rho(vu) = \pi_\rho(v)\pi_\rho(u)$  acts transitively on  $E(G\langle m, n; k \rangle)$  for all  $1 \leq k \leq n \leq m$ .
- (b)  $\text{Rad}(G\langle m, n; k \rangle) = \text{diam}(G\langle m, n; k \rangle)$  is finite for all  $1 \leq k < n \leq m$ .
- (c) The proof of this result is similar to the proof of Theorem 2.3(e). ■

Finally, it is possible to characterise completely when the complete lottery number  $L_1(m, n; k)$  takes any one of the values 1, 2 or 3, as is established<sup>7</sup> in the following theorem. However, before this characterisation may be achieved, it is necessary to introduce a notation able to capture the overlapping structure of a playing set for the lottery  $\langle m, n; k \rangle$ .

Given a vertex subset  $\mathcal{L}_\psi \subseteq \Phi(\mathcal{U}_m, n) = V(G\langle m, n; k \rangle)$  of the lottery graph  $G\langle m, n; k \rangle$ , it is possible to interchange the roles of elements in  $\mathcal{U}_m$ , thereby yielding different vertex subsets (of the same cardinality) for  $G\langle m, n; k \rangle$ . This merely constitutes a relabelling of the vertices of  $G\langle m, n; k \rangle$ , with the relabelled graph being isomorphic to  $G\langle m, n; k \rangle$ . Although these subsets may be considered different, they still inherit the same structure regarding the *overlappings* of their vertex label ( $n$ -set) elements. The ( $n$ -set) overlapping structure of a vertex subset  $\mathcal{L}_\ell = \{T_1, T_2, \dots, T_\ell\}$  may be captured by defining the function

$$x_{(t_\ell t_{\ell-1} \dots t_2 t_1)_2}^{(\ell)} = \left| \bigcap_{i=1}^{\ell} \begin{cases} T_i & \text{if } t_i = 1 \\ T'_i & \text{if } t_i = 0 \end{cases} \right|,$$

where  $(t_\ell t_{\ell-1} \dots t_2 t_1)_2$  denotes the binary representation of an integer in the range  $\{0, 1, \dots, 2^\ell - 1\}$  and where  $T'_i$  denotes the complement  $\mathcal{U}_m \setminus T_i$ . This function induces the  $2^\ell$ -vector

$$\vec{X}^{(\ell)} = \left( x_{(000 \dots 00)_2}^{(\ell)}, x_{(000 \dots 01)_2}^{(\ell)}, \dots, x_{(111 \dots 11)_2}^{(\ell)} \right),$$

which represents all the information required to describe the  $n$ -set overlapping structure of any vertex subset of cardinality  $\ell$  for  $G\langle m, n; k \rangle$  (and therefore also any 100%–partial dominating set  $[\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set] for  $G\langle m, n; k \rangle$   $[\langle m, n; k \rangle]$ ). The entries of the vector  $\vec{X}^{(\ell)}$  add up to  $m$  and may be interpreted as follows:

- there are  $x_{(000 \dots 00)_2}^{(\ell)}$  elements of  $\mathcal{U}_m$  contained in no label of  $\mathcal{L}_\ell$ ,
- there are  $x_{(000 \dots 01)_2}^{(\ell)}$  elements of  $\mathcal{U}_m$  contained in the single label  $T_1$  of  $\mathcal{L}_\ell$ ,
- there are  $x_{(000 \dots 10)_2}^{(\ell)}$  elements of  $\mathcal{U}_m$  contained in the single label  $T_2$  of  $\mathcal{L}_\ell$ ,
- there are  $x_{(000 \dots 11)_2}^{(\ell)}$  elements of  $\mathcal{U}_m$  contained in the overlapping labels of  $T_1$  and  $T_2$  of  $\mathcal{L}_\ell$ , etc.

It may sometimes be more convenient to write the subscripts of entries in the vector  $\vec{X}^{(\ell)}$  in decimal form. The multiplicity of an overlapping  $n$ -set structure  $\vec{X}^{(\ell)}$  (i.e., the number of different vertex subsets of  $G\langle m, n; k \rangle$  conforming to the structure  $\vec{X}^{(\ell)}$ ), given by

$$\mathcal{M}(\vec{X}^{(\ell)}) = \frac{m!}{\prod_{j=0}^{2^\ell-1} x_j^{(\ell)}!}, \quad (3.4)$$

<sup>7</sup>To the best knowledge of the author, no such characterisation of when the complete lottery numbers  $L_1(m, n; k) = 2, 3$  has ever been attempted. However, the characterisation of when  $L_1(m, n; k) = 1$  is known [234].

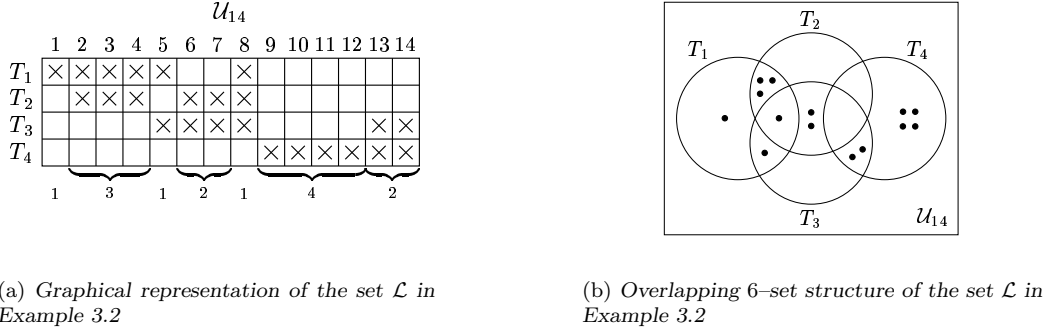


Figure 3.7: Graphical representations of complete lottery set structures for the lottery  $\langle 14, 6; 3 \rangle$ .

represents the actual number of structurally similar overlappings that are possible when interchanging the roles of the elements of  $\mathcal{U}_m$ <sup>8</sup>. The following example is presented to clarify the above notation.

**Example 3.2** Consider the lottery graph  $G\langle 14, 6; 3 \rangle$ , for which the lower domination number [complete lottery number] is known as  $\gamma(G\langle 14, 6; 3 \rangle) = L_1(14, 6; 3) = 4$ . A minimum dominating set [ $L_1(14, 6; 3)$ -set] for  $G\langle 14, 6; 3 \rangle$  [ $\langle 14, 6; 3 \rangle$ ] is given by  $\mathcal{L} = \{T_1, T_2, T_3, T_4\}$  where the vertex labels are given by  $T_1 = \{1, 2, 3, 4, 5, 8\}$ ,  $T_2 = \{2, 3, 4, 6, 7, 8\}$ ,  $T_3 = \{5, 6, 7, 8, 13, 14\}$  and  $T_4 = \{9, 10, 11, 12, 13, 14\}$ . Figure 3.7(a) represents the set  $\mathcal{L}$  in tabular format, while Figure 3.7(b) captures the overlapping 6-set structure of  $\mathcal{L}$  in Venn-diagram format. From Figure 3.7(b) it is clear that

$$x_{(0001)_2}^{(4)} = x_1^{(4)} = 1, \quad x_{(0011)_2}^{(4)} = x_3^{(3)} = 3, \quad x_{(0101)_2}^{(4)} = x_5^{(4)} = 1,$$

$$x_{(0110)_2}^{(4)} = x_6^{(4)} = 2, \quad x_{(0111)_2}^{(4)} = x_7^{(4)} = 1, \quad x_{(1000)_2}^{(4)} = x_8^{(4)} = 4 \quad \text{and} \quad x_{(1100)_2}^{(4)} = x_{12}^{(4)} = 2$$

and that  $x_{(t_4 t_3 t_2 t_1)_2}^{(4)} = 0$  for all other combinations of the bits  $t_1, t_2, t_3$  and  $t_4$ . This implies that  $\vec{X}^{(4)} = (0, 1, 0, 3, 0, 1, 2, 1, 4, 0, 0, 2, 0, 0, 0)$ . Note that the notation  $\vec{X}^{(4)}$  and the Venn-diagram representation in Figure 3.7(b) are unique for any given overlapping structure of the vertex labels  $T_1, T_2, T_3$  and  $T_4$  as opposed to the tabular form presented in Figure 3.7(a). If we choose the elements of  $\mathcal{U}_{14}$  in a different order, a different table results. In fact, there are  $\mathcal{M}(\vec{X}^{(4)}) = 14!/(3!2!4!2!) = 151\,351\,200$  different ways to form a table such as the one shown in Figure 3.7(a) from the overlapping 6-set structure shown in Figure 3.7(b), if the order of the 6-set listing is not altered. To pinpoint a unique tabular representation for the given lottery set structure, we may consider all permutations of  $\mathcal{U}_{14}$  and choose, for example, the lexicographic first one. ■

We are now in a position to characterise exactly when the complete lottery number assumes the values 1, 2 or 3.

**Theorem 3.4 (Characterisation of small values of  $L_1(m, n; k)$ )** For all  $1 \leq k \leq n \leq m$ ,

- (a)  $L_1(m, n; k) = 1$  if and only if  $2n \geq m + k$ .
- (b)  $L_1(m, n; k) = 2$  if and only if  $2k - 1 + \max\{m - 2n, 0\} \leq n \leq m - n + k - 1$ .
- (c)  $L_1(m, n; k) = 3$  if and only if

$$n \leq \min\{2k - 2 + \max\{m - 2n, 0\}, m - n + k - 1\} \tag{3.5}$$

and

$$n \geq \begin{cases} 3k - 2 + \max\{m - 3n, 0\} & \text{if } m \geq 2n \\ \frac{3}{2}k - 1 + \max\{m - \frac{3}{2}n, 0\} & \text{if } m < 2n. \end{cases} \tag{3.6}$$

<sup>8</sup>A different way of determining the multiplicity of the structure  $\vec{X}^{(\ell)}$  is to consider the number of ways (combinations) in which the structure  $\vec{X}^{(\ell)}$  may be fixed when choosing elements from  $\mathcal{U}_m$ . Consider, for example,  $\vec{X}^{(4)} = (0, 1, 0, 3, 0, 1, 2, 1, 4, 0, 0, 2, 0, 0, 0)$ . There are  $\binom{14}{1}\binom{13}{3}\binom{10}{1}\binom{9}{2}\binom{7}{1}\binom{6}{4}\binom{2}{2} = 151\,351\,200$  ways in which to choose such a vector from the elements of  $\mathcal{U}_{14}$ .

**Proof**

(a) If  $2n \geq m + k$ , then  $G\langle m, n; k \rangle \simeq K_{\binom{m}{n}}$  (by Corollary 3.1(c)) for which the lower domination number is known to be 1. Hence  $L_1(m, n; k) = 1$ . The converse is proved by means of a contra-positive argument. Consider an arbitrary vertex subset  $\mathcal{L}^* = \{T^*\}$  of  $G\langle m, n; k \rangle$  of cardinality one, but suppose  $2n < m + k$ . Then  $2n - m \leq k - 1 < k$ , implying that there exist two vertex labels sharing at most a  $(k - 1)$ -subset of  $\mathcal{U}_m$ . Hence  $\gamma(G\langle m, n; k \rangle) = L_1(m, n; k) \neq 1$  and the desired result follows.

(b) Suppose  $2k - 1 + \max\{m - 2n, 0\} \leq n$ . Let  $\mathcal{L}^* = \{T_1^*, T_2^*\}$  be a vertex subset of cardinality two of the lottery graph  $G\langle m, n; k \rangle$ , for which  $x_{(00)_2}^{(2)}$  is a minimum. Then  $x_{(00)_2}^{(2)} = \max\{m - 2n, 0\}$  and hence  $\Phi(T_i^*, k) \cap \Phi(w, k) \neq \emptyset$  for at least one  $i \in \{1, 2\}$ , where  $w$  is an arbitrary winning vertex in  $G\langle m, n; k \rangle$ , since  $n \geq \max\{m - 2n, 0\} + (2k - 1)$  (i.e.,  $w$  is adjacent to either  $T_1^*$  or  $T_2^*$ ). Hence,  $L_1(m, n; k) = 2$ .

Conversely, suppose  $L_1(m, n; k) = 2$ ; an  $L_1(m, n; k)$ -set being  $\mathcal{L}^* = \{T_1^*, T_2^*\}$ . Then  $n \leq m + k - n - 1$  from part (a) of this theorem. It remains to be shown that  $n > 2(k - 1) + \max\{m - 2n, 0\}$ . Suppose, to the contrary, that  $n \leq 2(k - 1) + \max\{m - 2n, 0\}$ . Hence, by choosing at most  $(k - 1)$  elements from each of the vertex labels  $T_1^*$  and  $T_2^*$  and the remainder from  $\mathcal{U}_m \setminus (T_1^* \cup T_2^*)$ , a winning vertex label  $w$  may be constructed such that  $\Phi(T_i^*, k) \cap \Phi(w, k) = \emptyset$  (for  $i \in \{1, 2\}$ ), implying that  $G\langle m, n; k \rangle$  is not dominated. This contradiction, however, implies that  $n > 2(k - 1) + \max\{m - 2n, 0\}$ , and the desired result follows.

(c) It follows, by parts (a) and (b) of this theorem, that  $L_1(m, n; k) \neq 1, 2$  if and only if  $n \leq m - n + k - 1$  and  $n \leq 2(k - 1) + \max\{m - 2n, 0\}$ . Therefore

$$L_1(m, n; k) > 2 \text{ if and only if } n \leq \min\{2(k - 1) + \max\{m - 2n, 0\}, m - n + k - 1\}. \quad (3.7)$$

We first prove the theorem for the case  $m \geq 2n$ . Suppose that (3.5) and the first inequality in (3.6) hold. Then it follows, by (3.7), that  $L_1(m, n; k) > 2$ . We now show that  $L_1(m, n; k) \leq 3$ . Choose a vertex subset  $\mathcal{L}^\circ = \{T_1^\circ, T_2^\circ, T_3^\circ\}$  of cardinality three of  $G\langle m, n; k \rangle$ , for which  $x_{(000)_2}^{(3)}$  is a minimum. Then  $x_{(000)_2}^{(3)} = \max\{m - 3n, 0\}$  and hence  $w$  is adjacent to  $T_i^\circ$  for at least one  $i \in \{1, 2, 3\}$ , where  $w$  is an arbitrary winning vertex in  $G\langle m, n; k \rangle$ , since  $n > \max\{m - 3n, 0\} + 3(k - 1)$ . Therefore,  $\mathcal{L}^\circ$  is a dominating set for  $G\langle m, n; k \rangle$ , and we conclude that  $L_1(m, n; k) = 3$ .

Conversely, suppose  $L_1(m, n; k) = 3$ . Then (3.5) follows from (3.7). We show that  $n > 3(k - 1) + \max\{m - 3n, 0\}$  by proving that, for any dominating set,

$$\max\{m - 3n, 0\} \leq x_{(000)_2}^{(3)} \leq n - 3k + 2. \quad (3.8)$$

The first inequality in (3.8) is obvious. To prove the second inequality in (3.8), suppose, to the contrary, that  $x_{(000)_2}^{(3)} > n - 3k + 2$  for some minimum dominating set  $\mathcal{L}^\circ = \{T_1^\circ, T_2^\circ, T_3^\circ\}$  for  $G\langle m, n; k \rangle$ . In this case, if the winning vertex,  $w$ , consists of (at least)  $n - 3(k - 1)$  elements of  $\mathcal{U}_m \setminus (T_1^\circ \cup T_2^\circ \cup T_3^\circ)$  and (at most)  $k - 1$  elements of each of  $T_1^\circ \setminus (T_2^\circ \cup T_3^\circ)$ ,  $T_2^\circ \setminus (T_1^\circ \cup T_3^\circ)$  and  $T_3^\circ \setminus (T_1^\circ \cup T_2^\circ)$ , it follows that  $w$  is not adjacent to any element in  $\mathcal{L}^\circ$ , contradicting the fact that  $\mathcal{L}^\circ$  is a dominating set for  $G\langle m, n; k \rangle$ . Note that in the proof we assume  $x_{(001)_2}^{(3)}, x_{(010)_2}^{(3)}, x_{(100)_2}^{(3)} \geq k - 1$ . However, if this is not the case, the following possible ‘‘worst case’’ is considered: we may have (say)  $x_{(100)_2}^{(3)} = n$  and  $x_{(010)_2}^{(3)} = x_{(001)_2}^{(3)} = k - 1 - y$ , where  $x_{(000)_2}^{(3)} = n - 3(k - 1) + y > n - 3(k - 1)$  for some  $y > 0$ . When constructing  $w$  to contain  $k - 1 - y$ ,  $k - 1 - y$ ,  $y$ ,  $k - 1$  and  $n - 3(k - 1) + y$  elements from respectively  $T_1^\circ \setminus T_2^\circ$ ,  $T_2^\circ \setminus T_1^\circ$ ,  $T_1^\circ \cap T_2^\circ$ ,  $T_3^\circ$  and  $\mathcal{U}_m \setminus (T_1^\circ \cup T_2^\circ \cup T_3^\circ)$ , it follows that  $\Phi(T_i^\circ, k) \cap \Phi(w, k) = \emptyset$  for all  $i \in \{1, 2, 3\}$ . Again this contradicts the fact that  $\mathcal{L}^\circ$  is a dominating set for  $G\langle m, n; k \rangle$ . This completes the proof for the case  $m \geq 2n$ .

For the case  $m < 2n$ , we consider the complementary complete lottery problem  $\langle m', n'; k' \rangle \equiv \langle m, m - n; m + k - 2n \rangle$  by virtue of the isomorphism result in Theorem 3.3(e). We then have  $m' > 2n'$ , which is the first case, proved above. Therefore  $L_1(m', n'; k') = 3$  if and only if

$$n' \leq \min\{2k' - 2 + \max\{m' - 2n', 0\}, m' - n' + k' - 1\} \quad (3.9)$$

and

$$n' \geq 3k' - 2 + \max\{m' - 3n', 0\}. \quad (3.10)$$

We only have to show that (3.10) is equivalent to the second inequality in (3.6). From (3.10) we have

$$m - n \geq 3(m + k - 2n) - 2 + \max\{m - 3(m - n), 0\}.$$

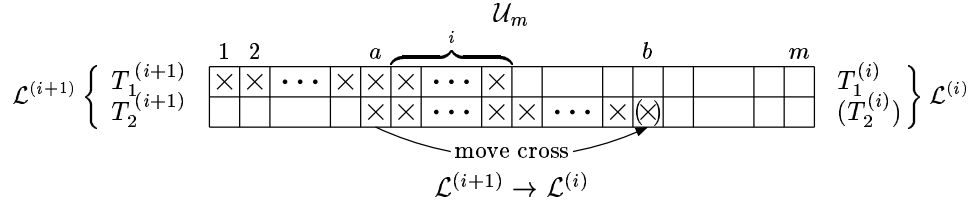


Figure 3.8: Graphical representation of the vertex subsets  $\mathcal{L}^{(i)}$  and  $\mathcal{L}^{(i+1)}$  in Lemma 3.1.

This inequality simplifies to

$$2n \geq 3k - 2 + \max\{2m - 3n, 0\},$$

which is equivalent to the second inequality in (3.6).  $\blacksquare$

It is also possible to determine explicitly the impact (difference in the number of common neighbours in  $G\langle m, n; k \rangle$ ) of an increment in the number of common elements between any two  $n$ -set vertex labels (i.e.,  $x_{(11)_2}^{(2)}$ ) in a vertex subset of  $G\langle m, n; k \rangle$ . From the next lemma, it follows that the resource utilised by a vertex subset of cardinality two in  $G\langle m, n; k \rangle$  decreases as  $x_{(11)_2}^{(2)}$  increases, as long as  $L_1(m, n; k) \geq 3$ .

**Lemma 3.1** *Suppose  $L_1(m, n; k) \geq 3$  and let  $\mathcal{L}^{(i+1)} = \{T_1^{(i+1)}, T_2^{(i+1)}\}$  be any cardinality 2 vertex subset of  $G\langle m, n; k \rangle$ , where the labels of  $\mathcal{L}^{(i+1)}$  share a common  $(i+1)$ -subset of  $\mathcal{U}_m$  (i.e.,  $0 < x_{(11)_2}^{(2)} = i+1 < n$ ). Additionally, let  $\mathcal{L}^{(i)} = \{T_1^{(i)}, T_2^{(i)}\}$  be any cardinality 2 vertex subset of  $G\langle m, n; k \rangle$ , where the labels of  $\mathcal{L}^{(i)}$  share a common  $i$ -subset of  $\mathcal{U}_m$  (i.e.,  $x_{(11)_2}^{(2)} = i$ ). Then the number of vertices in  $G\langle m, n; k \rangle$  adjacent to both  $T_1^{(i)}$  and  $T_2^{(i)}$  is strictly less than the number of vertices adjacent to both  $T_1^{(i+1)}$  and  $T_2^{(i+1)}$ . This difference is given by*

$$\sum_{j=0}^i \binom{i}{j} \binom{n-1-i}{k-1-j}^2 \binom{m-2n+i}{n+1-2k+j}. \quad (3.11)$$

**Proof**

Consider a tabular representation of  $\mathcal{L}^{(i+1)}$  comprising two rows (denoting the elements of  $\mathcal{L}^{(i+1)}$ ) and  $m$  columns (denoting the elements of  $\mathcal{U}_m$ ), in which the  $(x, y)$ -th cell contains a cross if  $y \in \mathcal{U}_m$  is an element of  $T_x^{(i+1)}$ , and is empty otherwise. We only have to move one cross, from column  $a$  to column  $b$  (say), in this representation of  $\mathcal{L}^{(i+1)}$  to obtain a similar representation of the vertex subset  $\mathcal{L}^{(i)}$ , as shown in Figure 3.8. We wish to count the difference between the number of vertices adjacent to both  $T_1^{(\bullet)}$  and  $T_2^{(\bullet)}$  in these two representations. When the cross is moved as depicted in Figure 3.8, then only the number of vertices adjacent to  $T_2^{(\bullet)}$  can change. We therefore count:

- (1) the number of vertices in  $G\langle m, n; k \rangle$  adjacent to both  $T_1^{(i+1)}$  and  $T_2^{(i+1)}$ , but not adjacent to  $T_2^{(i)}$ ;
- (2) the number of vertices in  $G\langle m, n; k \rangle$  adjacent to both  $T_1^{(i)}$  and  $T_2^{(i)}$ , but not adjacent to  $T_2^{(i+1)}$ .

The number of adjacent vertices obtained in (1) above, less that obtained in (2) above, gives the desired quantity.

**Number obtained from (1):** Vertices of  $G\langle m, n; k \rangle$  in (1) should contain the element  $a \in \mathcal{U}_m$ , but not  $b \in \mathcal{U}_m$ . Furthermore, they should also contain at least  $k-1$  elements from  $T_1^{(i+1)}$  (not counting  $a$ ) and exactly  $k-1$  elements from  $T_2^{(i+1)}$ ;

**Number obtained from (2):** Vertices of  $G\langle m, n; k \rangle$  in (2) should contain the element  $b \in \mathcal{U}_m$ , but not  $a \in \mathcal{U}_m$ . Furthermore, they should also contain at least  $k$  elements from  $T_1^{(i+1)}$  and exactly  $k-1$  elements from  $T_2^{(i+1)}$  (not counting  $b$ ).

It is easy to see that the enumeration in (1) less that of (2) is achieved by the number of vertices in  $G\langle m, n; k \rangle$  with labels that share exactly  $k-1$  elements of  $T_1^{(i+1)}$  and exactly  $k-1$  elements of  $T_2^{(i+1)}$

(recall that  $a$  and  $b$  are not counted). This number of  $(n-1)$ -sets is given by

$$\binom{i}{j} \binom{n-1-i}{k-1-j}^2 \binom{m-2n+i}{n+1-2k+j},$$

(by virtue of Lemma 2.2) where  $j$  denotes the number of common elements between the vertex labels in  $\mathcal{L}^{(i+1)}$  (i.e.,  $x_{(11)_2}^{(2)}$ ). By allowing  $j$  to vary within the range  $\{0, \dots, i\}$  in the above expression, the desired result is obtained.

It remains to be shown that the expression in (3.11) cannot be zero. This may be achieved by showing that there exists a  $j$  such that  $n-1-i \geq k-1-j$  and that  $m-2n+i \geq n+1-2k+j$ . That is,  $k-n+i \leq j \leq m-3n+i+2k-1$ . Thus, for such a  $j$  to exist, we require that  $k-n+i \leq m-3n+i+2k-1$ , which simplifies to  $2n < m+k$ , which is exactly the requirement that the vertex subset should contain at least two vertices ( $n$ -sets), according to the characterisation in Theorem 3.4(a). ■

It follows from Lemma 3.1 that the resource utilised by a vertex subset of cardinality two of  $G\langle m, n; k \rangle$  decreases as  $x_{(11)_2}^{(2)}$  increases, as long as  $L_1(m, n; k) \geq 3$ . Therefore, all  $L_{\Psi_2(m, n; k)}(m, n; k)$ -sets have the structure  $\vec{X}^{(2)} = (\max\{m-2n, 0\}, n - \max\{2n-m, 0\}, n - \max\{2n-m, 0\}, \max\{2n-m, 0\})$  in cases where  $L_1(m, n; k) \geq 3$ . In such cases the corresponding resource utilisations,  $\Psi_2(m, n; k)$ , are given by  $(2(r+1) - \xi_2^{\max\{2n-m, 0\}}) / \binom{m}{n}$ , with  $r$  given in (2.1).

### 3.3 Symmetric representation of the lottery graph

The lottery graph contains an inherent symmetry which becomes apparent in its graphical representation by ordering the vertices according to certain rules. These rules are described somewhat imprecisely in Algorithm 1. Algorithm 1 is illustrated with a step-by-step inspection of drawing the lottery graph  $G\langle 5, 3; 2 \rangle$  in the following example.

---

**Algorithm 1** Symmetric representation algorithm for  $G\langle m, n; k \rangle$

---

**Input:** The lottery parameters  $m$ ,  $n$  and  $k$ .

**Output:** A symmetrical representation of the lottery graph  $G\langle m, n; k \rangle$  in the plane.

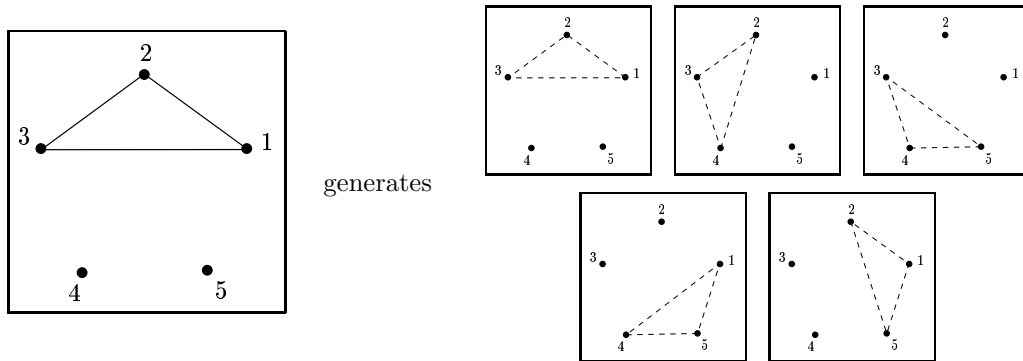
- 1: Place  $m$  vertices of an auxiliary graph on the circumference of a circle and label these vertices lexicographically, using the numbers  $1, \dots, m$ .
  - 2: Draw all possible cycles  $C_n$  on a subset of  $n$  of the  $m$  vertices, excluding equivalent rotations of cycles of the resulting  $(m, n)$ -graph. The vertex labels of every non-equivalent  $C_n$  correspond to some  $n$ -subset, say  $T_j = \{t_1, t_2, \dots, t_n\}$  (called a vertex generator), of the possible  $\binom{m}{n}$  vertices of  $G\langle m, n; k \rangle$ .
  - 3: From every vertex generator  $T_j$ , a specified selection/partition of  $n$ -subsets (say  $\mathcal{T}_i$ ) may be generated by determining  $t_i + j \pmod{m}$  for all  $t_i \in T_j$ ,  $i = 1, \dots, n$  and  $j = 1, \dots, m-1$ . All these vertices from a single generator are placed on the circumference of a circle of a certain radius in the lottery graph.
  - 4: The lottery graph  $G\langle m, n; k \rangle$  is drawn with vertices arranged (lexicographically) according to the different vertex generators in concentric circles of different radii.
- 

**Example 3.3** Consider the lottery graph  $G\langle 5, 3; 2 \rangle$  on  $\binom{5}{3} = 10$  vertices. The only two possible (lexicographically first) vertex generators (up to rotation isomorphism) are shown in the large boxes in Figure 3.9(a)–(b). Here

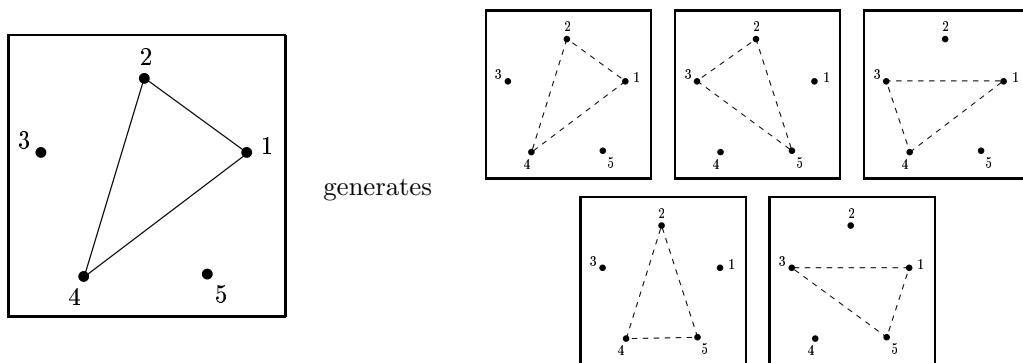
$$\begin{aligned} \{1, 2, 3\} \text{ generates } \mathcal{T}_1 &= \{\{1, 2, 3\}, \{2, 3, 4\}, \{3, 4, 5\}, \{1, 4, 5\}, \{1, 2, 5\}\}, \\ \{1, 2, 4\} \text{ generates } \mathcal{T}_2 &= \{\{1, 2, 4\}, \{2, 3, 5\}, \{1, 3, 4\}, \{2, 4, 5\}, \{1, 3, 5\}\}. \end{aligned}$$

Arranging the elements of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  lexicographically on the circumference of two concentric circles and adding the relevant edges, a (well-known) symmetric representation of the lottery graph [complement]  $G\langle 5, 3; 2 \rangle$  [ $G\langle 5, 3; 2 \rangle$ ] is obtained, as shown in Figure 3.9(c) [(d)]. The labels of the vertices are shown in Figure 3.9(e). ■

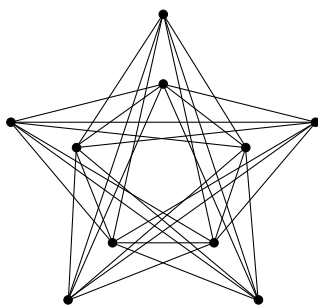




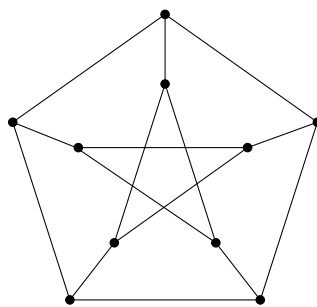
(a) Vertex generator  $\{1, 2, 3\}$  that generates the vertex subset  $T_1$  of  $G\langle 5, 3; 2 \rangle$



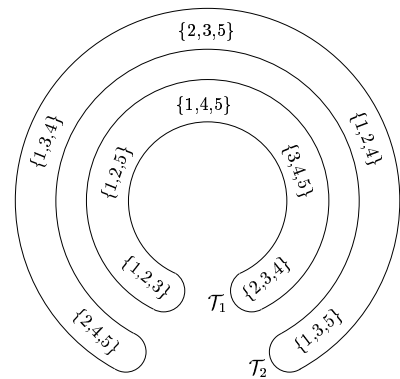
(b) Vertex generator  $\{1, 2, 4\}$  that generates the vertex subset  $T_2$  of  $G\langle 5, 3; 2 \rangle$



(c)  $G\langle 5, 3; 2 \rangle$



(d)  $\overline{G\langle 5, 3; 2 \rangle}$



(e) (Partition of) 3-sets for  $\langle 5, 3; 2 \rangle$

Figure 3.9: Subfigures (a)–(b) represent the only two possible vertex generators for  $\langle 5, 3; 2 \rangle$ , while the corresponding lottery graph [complement]  $G\langle 5, 3; 2 \rangle$  [ $\overline{G\langle 5, 3; 2 \rangle}$ ] is displayed in subfigure (c) [(d)], with the corresponding vertex labels in subfigure (e). Interestingly enough, the well-known Petersen graph emerges as the symmetric representation of the lottery graph complement  $\overline{G\langle 5, 3; 2 \rangle}$ , according to Algorithm 1.

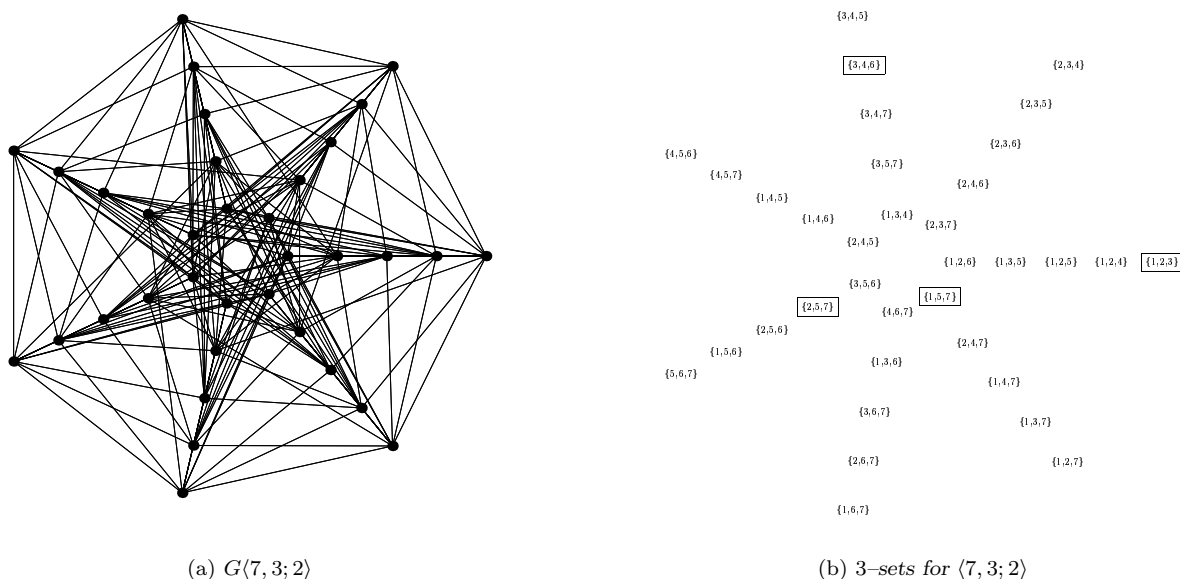


Figure 3.10: Symmetric graph representation of  $G\langle 7, 3; 2 \rangle$  obtained by Algorithm 1.

Another (abbreviated) application of Algorithm 1 is presented in the following example for the lottery  $\langle 7, 3; 2 \rangle$ .

**Example 3.4 (continuation of Example 3.1)** *Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 3.1. Using Algorithm 1, the symmetric representation of the lottery graph  $G\langle 7, 3; 2 \rangle$ , presented in Figure 3.10, was obtained. The construction is rather tedious, and hence only the final result is shown. ■*

The reader should note that different vertex generators (as described in Algorithm 1) would yield different (yet isomorphic) representations of  $G\langle m, n; k \rangle$ . Hence, the choice of vertex generators (and therefore the placement and relative rotation of the concentric circles) might be a tedious process, which may be abbreviated somewhat through experience, although a lexicographic approach to these uncertainties was found to be best suited.

Algorithm 1 was used to represent graphically some of the lottery graphs presented in the following section.

### 3.4 Analysis of small lotteries

Graphical representations of the lottery graphs  $G\langle m, n; k \rangle$  or their complements  $\overline{G\langle m, n; k \rangle}$  for all lotteries  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$ , are shown in Figure 3.11. Only cases where the lesser dense of the two graphs has density not exceeding 0.5 and/or order not exceeding 35 (due to clarity of visual appearance) are shown. Embossed vertices of different line styles were used to represent elements of one possible  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$ , yielding a sharp upper bound on  $L_\psi(m, n; k)$  in each case, for the special separating values  $\psi = \Psi_i(m, n; k)$ , where  $i = 1, \dots, L_1(m, n; k)$ .

Table 3.1 contains the values of the complete lottery numbers  $L_1(m, n; k)$  for the small feasible values of  $1 \leq k \leq n \leq m \leq 10$  (i.e., including cases not shown in Figure 3.11), as a summary of the results presented in this section.

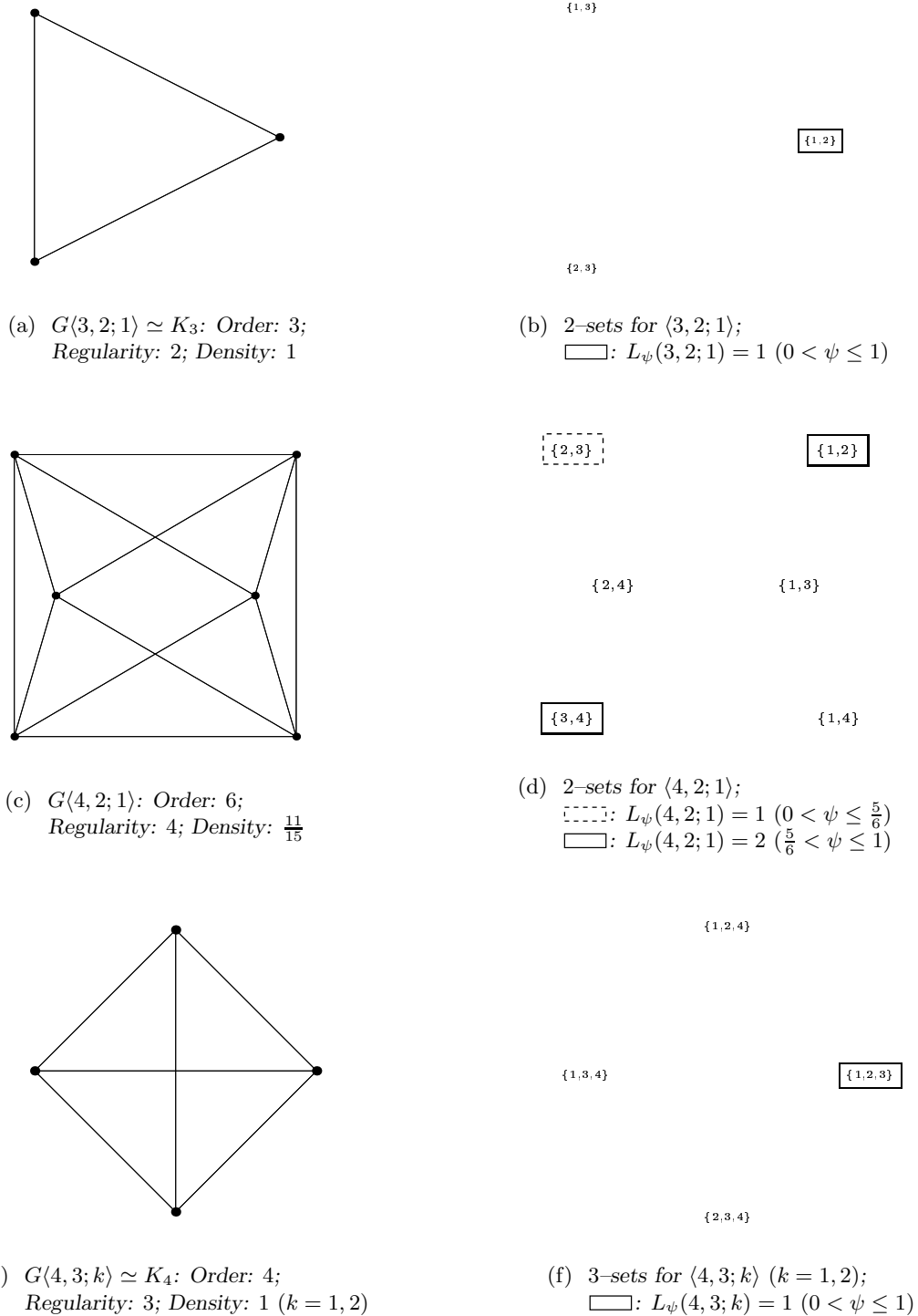
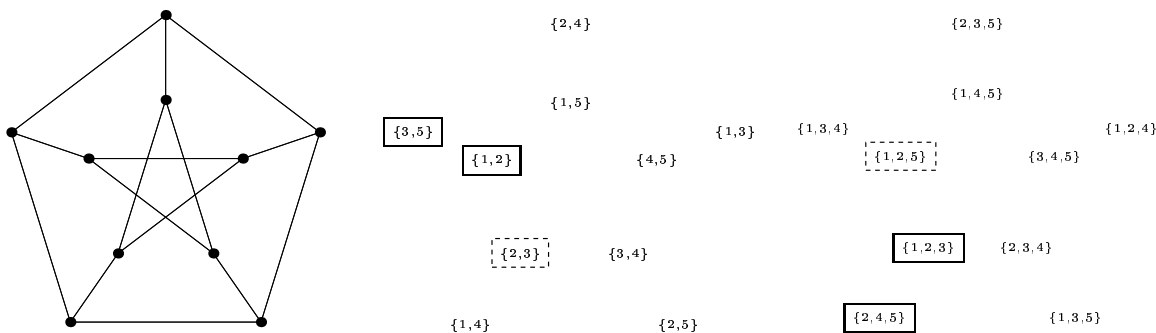
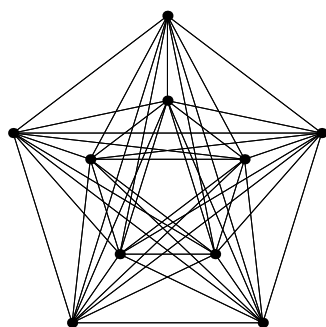


Figure 3.11: Lottery graphs  $G(m, n; k)$  or  $\overline{G(m, n; k)}$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Emboxed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal 100 $\psi$ %-partial dominating sets of minimum cardinality for  $G(m, n; k)$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .

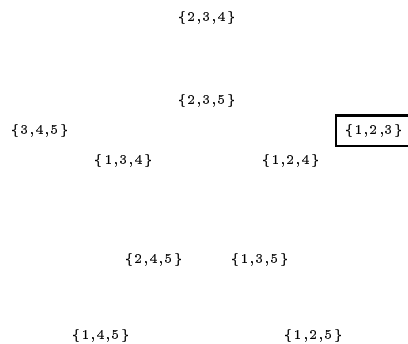


(g)  $\overline{G\langle 5, 2; 1 \rangle} \simeq \overline{G\langle 5, 3; 2 \rangle}$ :  
 Order: 10; Regularity: 3;  
 Density:  $\frac{15}{45}$

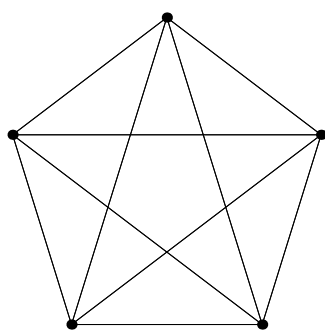
(h) 2-sets for  $\langle 5, 2; 1 \rangle$ ; (i) 3-sets for  $\langle 5, 3; 2 \rangle$ ;  
 $\text{---} \text{---} \text{---} \text{---}$ :  $L_\psi(5, 2; 1) = L_\psi(5, 3; 2) = 1$  ( $0 < \psi \leq \frac{2}{5}$ )  
 $\square$ :  $L_\psi(5, 2; 1) = L_\psi(5, 3; 2) = 2$  ( $\frac{2}{5} < \psi \leq 1$ )



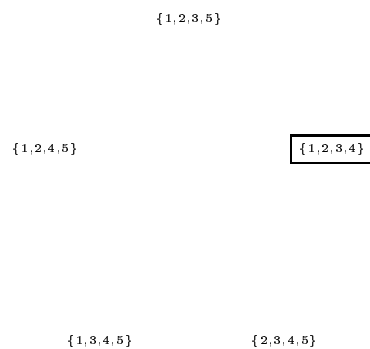
(j)  $G\langle 5, 3; 1 \rangle \simeq K_{10}$ : Order: 10;  
 Regularity: 9; Density: 1



(k) 3-sets for  $\langle 5, 3; 1 \rangle$ ;  
 $\square$ :  $L_\psi(5, 3; 1) = 1$  ( $0 < \psi \leq 1$ )

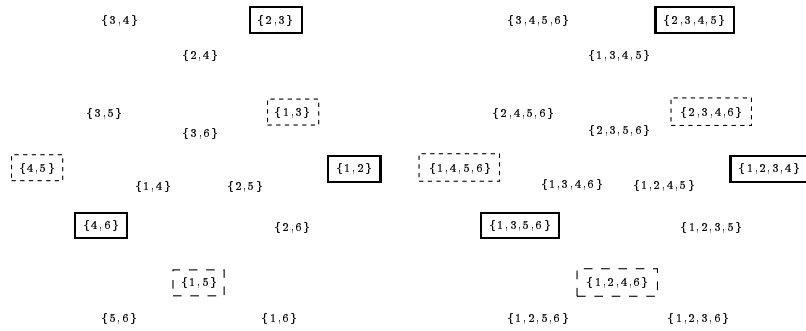
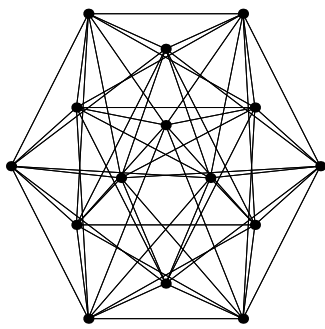


(l)  $G\langle 5, 4; k \rangle \simeq K_5$ : Order: 5; Regularity: 4;  
 Density: 1 ( $k = 1, 2, 3$ )



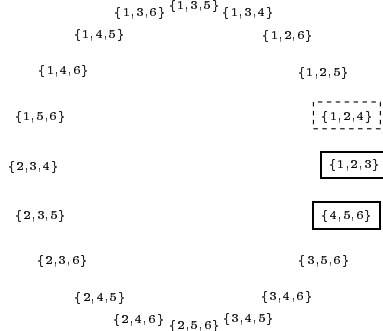
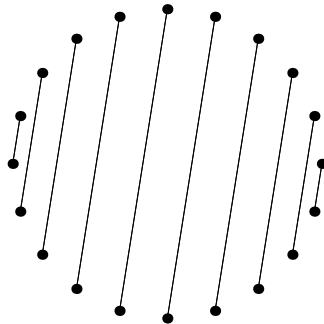
(m) 4-sets for  $\langle 5, 4; k \rangle$  ( $k = 1, 2, 3$ );  
 $\square$ :  $L_\psi(5, 4; k) = 1$  ( $0 < \psi \leq 1$ )

Figure 3.11 (continued): Lottery graphs  $G\langle m, n; k \rangle$  or  $\overline{G\langle m, n; k \rangle}$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Emboxed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal  $100\psi\%$ -partial dominating sets of minimum cardinality for  $G\langle m, n; k \rangle$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .



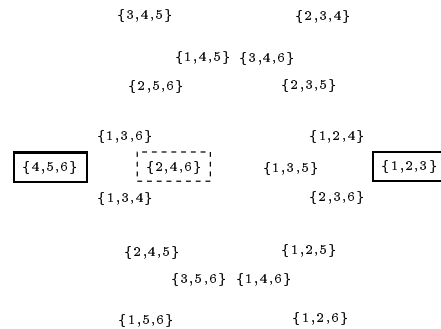
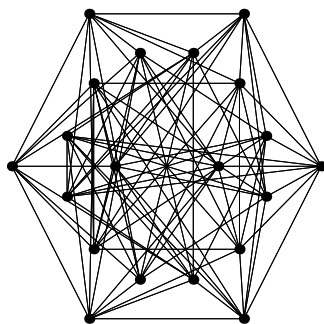
(n)  $G(6, 2; 1) \simeq G(6, 4; 3)$ :  
 Order: 15; Regularity: 8;  
 Density:  $\frac{60}{105}$

(o) 2-sets for  $\langle 6, 2; 1 \rangle$ ; (p) 4-sets for  $\langle 6, 4; 3 \rangle$ ;  
 $\square \square \square$ :  $L_\psi(6, 2; 1) = L_\psi(6, 4; 3) = 1$  ( $0 < \psi \leq \frac{9}{15}$ )  
 $\square \square \square \square$ :  $L_\psi(6, 2; 1) = L_\psi(6, 4; 3) = 2$  ( $\frac{9}{15} < \psi \leq \frac{14}{15}$ )  
 $\square$ :  $L_\psi(6, 2; 1) = L_\psi(6, 4; 3) = 3$  ( $\frac{14}{15} < \psi \leq 1$ )



(q)  $\overline{G(6, 3; 1)}$ : Order: 20;  
 Regularity: 2; Density:  $\frac{10}{190}$

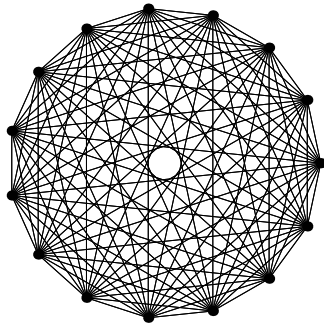
(r) 3-sets for  $\langle 6, 3; 1 \rangle$ ;  
 $\square \square \square$ :  $L_\psi(6, 3; 1) = 1$  ( $0 < \psi \leq \frac{19}{20}$ )  
 $\square$ :  $L_\psi(6, 3; 1) = 2$  ( $\frac{19}{20} < \psi \leq 1$ )



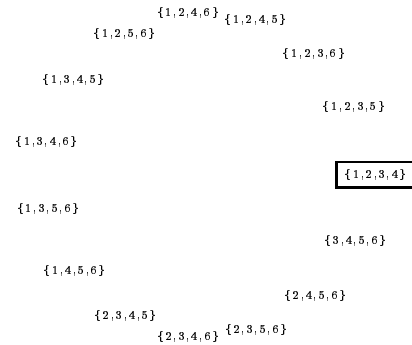
(s)  $G(6, 3; 2)$ : Order: 20;  
 Regularity: 9; Density:  $\frac{90}{190}$

(t) 3-sets for  $\langle 6, 3; 2 \rangle$ ;  
 $\square \square \square$ :  $L_\psi(6, 3; 2) = 1$  ( $0 < \psi \leq \frac{1}{2}$ )  
 $\square$ :  $L_\psi(6, 3; 2) = 2$  ( $\frac{1}{2} < \psi \leq 1$ )

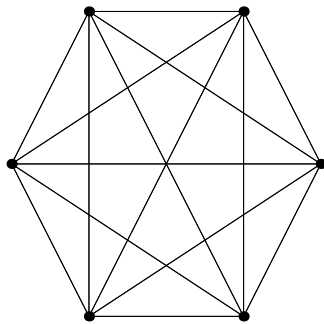
Figure 3.11 (continued): Lottery graphs  $G\langle m, n; k \rangle$  or  $\overline{G\langle m, n; k \rangle}$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Emboxed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal  $100\psi\%$ -partial dominating sets of minimum cardinality for  $G\langle m, n; k \rangle$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .



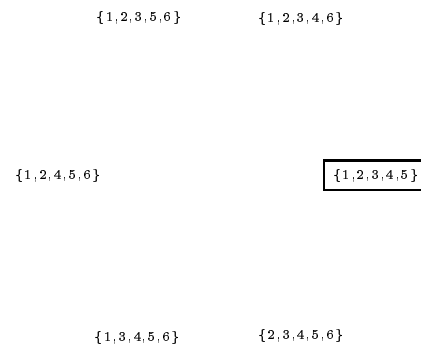
(u)  $G(6, 4; k) \simeq K_{15}$ : Order: 15;  
Regularity: 14; Density: 1 ( $k = 1, 2$ )



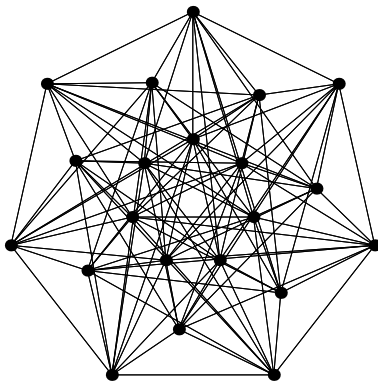
(v) 4-sets for  $(6, 4; k)$  ( $k = 1, 2$ );  
□:  $L_\psi(6, 4; k) = 1$  ( $0 < \psi \leq 1$ )



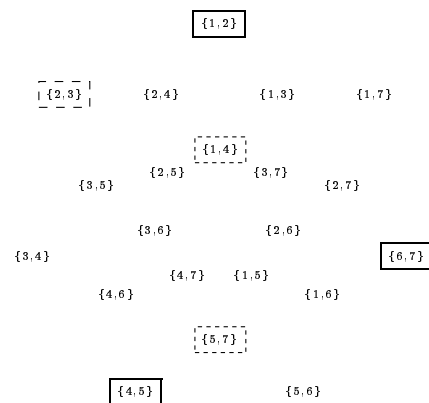
(w)  $G(6, 5; k) \simeq K_6$ : Order: 6;  
Regularity: 5; Density: 1 ( $k = 1, \dots, 4$ )



(x) 5-sets for  $(6, 5; k)$  ( $k = 1, \dots, 4$ );  
□:  $L_\psi(6, 5; k) = 1$  ( $0 < \psi \leq 1$ )

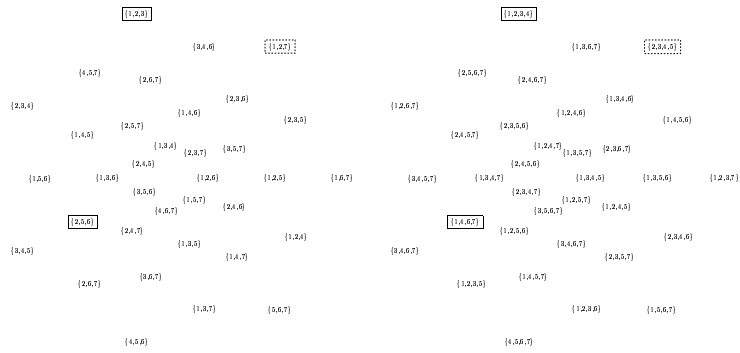
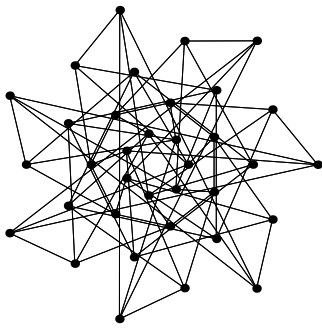


(y)  $G(7, 2; 1)$ : Order: 21;  
Regularity: 10; Density:  $\frac{105}{210}$



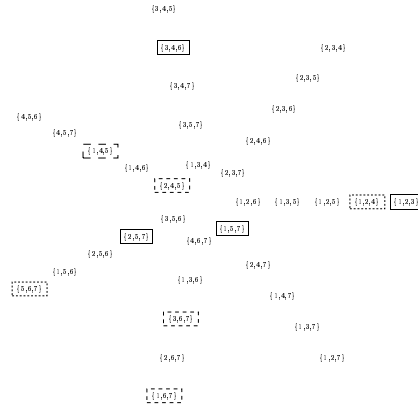
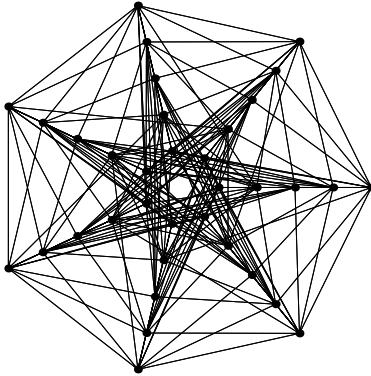
(z) 2-sets for  $(7, 2; 1)$ ;  
□:  $L_\psi(7, 2; 1) = 1$  ( $0 < \psi \leq \frac{11}{21}$ )  
□:  $L_\psi(7, 2; 1) = 2$  ( $\frac{11}{21} < \psi \leq \frac{18}{21}$ )  
□:  $L_\psi(7, 2; 1) = 3$  ( $\frac{18}{21} < \psi \leq 1$ )

Figure 3.11 (continued): Lottery graphs  $G\langle m, n; k \rangle$  or  $\overline{G\langle m, n; k \rangle}$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Emboxed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal  $100\psi\%$ -partial dominating sets of minimum cardinality for  $G\langle m, n; k \rangle$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .



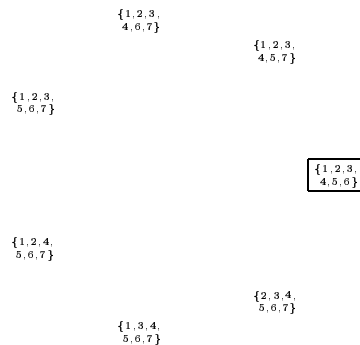
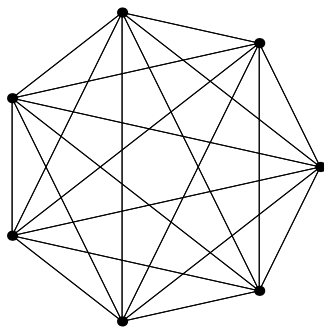
(aa)  $\overline{G(7, 3; 1)} \simeq \overline{G(7, 4; 2)}$ :  
 Order: 35; Regularity: 4;  
 Density:  $\frac{70}{595}$

(bb) 3-sets for  $\langle 7, 3; 1 \rangle$ ;      (cc) 4-sets for  $\langle 7, 4; 2 \rangle$ ;  
 $\square \square \square$ :  $L_\psi(7, 3; 1) = L_\psi(7, 4; 2) = 1$  ( $0 < \psi \leq \frac{32}{35}$ )  
 $\square$ :  $L_\psi(7, 3; 1) = L_\psi(7, 4; 2) = 2$  ( $\frac{32}{35} < \psi \leq 1$ )



(dd)  $G(7, 3; 2)$ : Order: 35;  
 Regularity: 12; Density:  $\frac{210}{595}$

(ee) 3-sets for  $\langle 7, 3; 2 \rangle$ ;  
 $\square \square \square$ :  $L_\psi(7, 3; 2) = 1$  ( $0 < \psi \leq \frac{13}{35}$ )  
 $\square \square \square$ :  $L_\psi(7, 3; 2) = 2$  ( $\frac{13}{35} < \psi \leq \frac{26}{35}$ )  
 $\square \square \square$ :  $L_\psi(7, 3; 2) = 3$  ( $\frac{26}{35} < \psi \leq \frac{32}{35}$ )  
 $\square$ :  $L_\psi(7, 3; 2) = 4$  ( $\frac{32}{35} < \psi \leq 1$ )



(ff)  $G(7, 6; k) \simeq K_7$ : Order: 7;  
 Regularity: 6; Density: 1 ( $k = 1, \dots, 5$ )

(gg) 6-sets for  $\langle 7, 6; k \rangle$  ( $k = 1, \dots, 5$ );  
 $\square$ :  $L_\psi(7, 6; k) = 1$  ( $0 < \psi \leq 1$ )

Figure 3.11 (continued): Lottery graphs  $G\langle m, n; k \rangle$  or  $\overline{G\langle m, n; k \rangle}$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Emboxed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal  $100\psi\%$ -partial dominating sets of minimum cardinality for  $G\langle m, n; k \rangle$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .





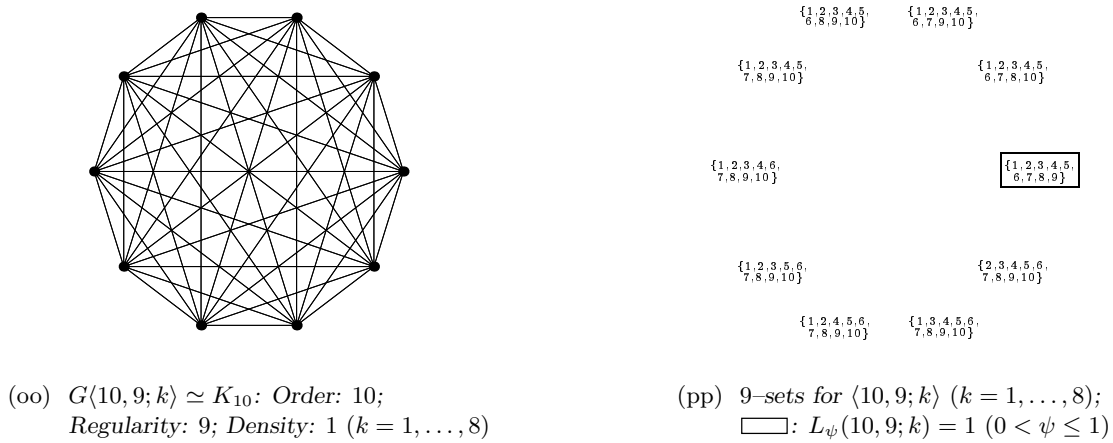


Figure 3.11 (continued): Lottery graphs  $G\langle m, n; k \rangle$  or  $\overline{G}\langle m, n; k \rangle$  for  $1 \leq k < n < m \leq 10$  (having density and/or order at most 0.5 and 35 respectively) and their relevant  $n$ -set ordering. Embossed vertex labels represent  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (i.e., minimal  $100\psi\%$ -partial dominating sets of minimum cardinality for  $G\langle m, n; k \rangle$ ), for the separating values of  $\psi = \Psi_i(m, n; k)$ , where  $1 \leq i \leq L_1(m, n; k)$ .

### 3.5 Chapter summary

The chapter opened with a discussion of some basic notions from graph and complexity theory (in §3.1), laying the basic foundation for the methodology that will be adopted in the rest of the dissertation. The lottery graph  $G\langle m, n; k \rangle$  was introduced, yielding a rich structural representation of the lottery  $\langle m, n; k \rangle$ .

The lottery graph was utilised to determine upper bounds on  $L_1(m, n; k)$  using graph domination theory in analyses of small lotteries of the form  $\langle m, n; k \rangle$  where  $1 \leq k \leq n \leq m \leq 10$  (in §3.4). An algorithm for exposing the symmetric structure of the lottery graph  $G\langle m, n; k \rangle$  was discussed and used to draw some of the graphs in Figure 3.11. For the specific lottery graphs included in Figure 3.11, all (incomplete) lottery numbers  $L_\psi(m, n; k)$ , where  $\psi = \Psi_i(m, n; k)$  for  $i = 1, \dots, L_1(m, n; k)$ , are presented. The (complete) lottery numbers  $L_1(m, n; k)$  for the small lotteries investigated are summarised in Table 3.1.

<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="2" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td></tr> <tr><td rowspan="2" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(a) <math>L_1(2, n; k)</math></p>			$n$				1	2	$k$	1	$2^b$	$1^a$	2	-	$1^a$	<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="3" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td></tr> <tr><td rowspan="3" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>3^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(b) <math>L_1(3, n; k)</math></p>			$n$					1	2	3	$k$	1	$3^b$	$1^b$	$1^a$	2	-	$1^c$	$1^a$	3	-	-	$1^a$	<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="4" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td></tr> <tr><td rowspan="4" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>4^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>6^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(c) <math>L_1(4, n; k)</math></p>			$n$						1	2	3	4	$k$	1	$4^b$	$2^b$	$1^b$	$1^a$	2	-	$6^c$	$1^d$	$1^a$	3	-	-	$1^c$	$1^a$	4	-	-	-	$1^a$																																																																																																																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2																																																																																																																																																																																																																																																						
$k$	1	$2^b$	$1^a$																																																																																																																																																																																																																																																						
	2	-	$1^a$																																																																																																																																																																																																																																																						
		$n$																																																																																																																																																																																																																																																							
		1	2	3																																																																																																																																																																																																																																																					
$k$	1	$3^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																					
	2	-	$1^c$	$1^a$																																																																																																																																																																																																																																																					
	3	-	-	$1^a$																																																																																																																																																																																																																																																					
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4																																																																																																																																																																																																																																																				
$k$	1	$4^b$	$2^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																				
	2	-	$6^c$	$1^d$	$1^a$																																																																																																																																																																																																																																																				
	3	-	-	$1^c$	$1^a$																																																																																																																																																																																																																																																				
	4	-	-	-	$1^a$																																																																																																																																																																																																																																																				
<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="5" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td></tr> <tr><td rowspan="5" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>5^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>10^c</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>10^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>5^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(d) <math>L_1(5, n; k)</math></p>			$n$							1	2	3	4	5	$k$	1	$5^b$	$2^b$	$1^b$	$1^b$	$1^a$	2	-	$10^c$	$2^e$	$1^d$	$1^a$	3	-	-	$10^c$	$1^d$	$1^a$	4	-	-	-	$5^c$	$1^a$	5	-	-	-	-	$1^a$	<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="6" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">6</td></tr> <tr><td rowspan="6" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>6^b</math></td><td style="text-align: center;"><math>3^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>15^c</math></td><td style="text-align: center;"><math>2^{Jg}</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>20^c</math></td><td style="text-align: center;"><math>3^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>15^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(e) <math>L_1(6, n; k)</math></p>			$n$								1	2	3	4	5	6	$k$	1	$6^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^a$	2	-	$15^c$	$2^{Jg}$	$1^d$	$1^d$	$1^a$	3	-	-	$20^c$	$3^e$	$1^d$	$1^a$	4	-	-	-	$15^c$	$1^d$	$1^a$	5	-	-	-	-	$1^c$	$1^a$	6	-	-	-	-	-	$1^a$																																																																																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5																																																																																																																																																																																																																																																			
$k$	1	$5^b$	$2^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																			
	2	-	$10^c$	$2^e$	$1^d$	$1^a$																																																																																																																																																																																																																																																			
	3	-	-	$10^c$	$1^d$	$1^a$																																																																																																																																																																																																																																																			
	4	-	-	-	$5^c$	$1^a$																																																																																																																																																																																																																																																			
	5	-	-	-	-	$1^a$																																																																																																																																																																																																																																																			
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6																																																																																																																																																																																																																																																		
$k$	1	$6^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																		
	2	-	$15^c$	$2^{Jg}$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																		
	3	-	-	$20^c$	$3^e$	$1^d$	$1^a$																																																																																																																																																																																																																																																		
	4	-	-	-	$15^c$	$1^d$	$1^a$																																																																																																																																																																																																																																																		
	5	-	-	-	-	$1^c$	$1^a$																																																																																																																																																																																																																																																		
	6	-	-	-	-	-	$1^a$																																																																																																																																																																																																																																																		
<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="7" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">6</td><td style="text-align: center;">7</td></tr> <tr><td rowspan="7" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>7^b</math></td><td style="text-align: center;"><math>3^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>21^c</math></td><td style="text-align: center;"><math>4^h</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>35^c</math></td><td style="text-align: center;"><math>4^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>35^c</math></td><td style="text-align: center;"><math>3^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>21^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>7^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(f) <math>L_1(7, n; k)</math></p>			$n$									1	2	3	4	5	6	7	$k$	1	$7^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^a$	2	-	$21^c$	$4^h$	$2^e$	$1^d$	$1^d$	$1^a$	3	-	-	$35^c$	$4^e$	$1^d$	$1^d$	$1^a$	4	-	-	-	$35^c$	$3^e$	$1^d$	$1^a$	5	-	-	-	-	$21^c$	$1^d$	$1^a$	6	-	-	-	-	-	$7^c$	$1^a$	7	-	-	-	-	-	-	$1^a$	<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="8" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">6</td><td style="text-align: center;">7</td><td style="text-align: center;">8</td></tr> <tr><td rowspan="8" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>8^b</math></td><td style="text-align: center;"><math>4^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>28^c</math></td><td style="text-align: center;"><math>5^i</math></td><td style="text-align: center;"><math>2^{Jj}</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>56^c</math></td><td style="text-align: center;"><math>6^k</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>70^c</math></td><td style="text-align: center;"><math>5^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>56^c</math></td><td style="text-align: center;"><math>4^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>28^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>8^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(g) <math>L_1(8, n; k)</math></p>			$n$										1	2	3	4	5	6	7	8	$k$	1	$8^b$	$4^b$	$2^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^a$	2	-	$28^c$	$5^i$	$2^{Jj}$	$1^d$	$1^d$	$1^d$	$1^a$	3	-	-	$56^c$	$6^k$	$2^e$	$1^d$	$1^d$	$1^a$	4	-	-	-	$70^c$	$5^e$	$1^d$	$1^d$	$1^a$	5	-	-	-	-	$56^c$	$4^e$	$1^d$	$1^a$	6	-	-	-	-	-	$28^c$	$1^d$	$1^a$	7	-	-	-	-	-	-	$8^c$	$1^a$	8	-	-	-	-	-	-	-	$1^a$																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7																																																																																																																																																																																																																																																	
$k$	1	$7^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																	
	2	-	$21^c$	$4^h$	$2^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																	
	3	-	-	$35^c$	$4^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																	
	4	-	-	-	$35^c$	$3^e$	$1^d$	$1^a$																																																																																																																																																																																																																																																	
	5	-	-	-	-	$21^c$	$1^d$	$1^a$																																																																																																																																																																																																																																																	
	6	-	-	-	-	-	$7^c$	$1^a$																																																																																																																																																																																																																																																	
	7	-	-	-	-	-	-	$1^a$																																																																																																																																																																																																																																																	
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8																																																																																																																																																																																																																																																
$k$	1	$8^b$	$4^b$	$2^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																																
	2	-	$28^c$	$5^i$	$2^{Jj}$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																
	3	-	-	$56^c$	$6^k$	$2^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																
	4	-	-	-	$70^c$	$5^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																																
	5	-	-	-	-	$56^c$	$4^e$	$1^d$	$1^a$																																																																																																																																																																																																																																																
	6	-	-	-	-	-	$28^c$	$1^d$	$1^a$																																																																																																																																																																																																																																																
	7	-	-	-	-	-	-	$8^c$	$1^a$																																																																																																																																																																																																																																																
	8	-	-	-	-	-	-	-	$1^a$																																																																																																																																																																																																																																																
<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="9" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">6</td><td style="text-align: center;">7</td><td style="text-align: center;">8</td><td style="text-align: center;">9</td></tr> <tr><td rowspan="9" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>9^b</math></td><td style="text-align: center;"><math>4^b</math></td><td style="text-align: center;"><math>3^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>36^c</math></td><td style="text-align: center;"><math>7^l</math></td><td style="text-align: center;"><math>2^{Jm}</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>84^c</math></td><td style="text-align: center;"><math>9^n</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>126^c</math></td><td style="text-align: center;"><math>9^e</math></td><td style="text-align: center;"><math>3^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>126^c</math></td><td style="text-align: center;"><math>7^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>84^c</math></td><td style="text-align: center;"><math>4^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>36^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>9^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">9</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(h) <math>L_1(9, n; k)</math></p>			$n$											1	2	3	4	5	6	7	8	9	$k$	1	$9^b$	$4^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^b$	$1^a$	2	-	$36^c$	$7^l$	$2^{Jm}$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$	3	-	-	$84^c$	$9^n$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$	4	-	-	-	$126^c$	$9^e$	$3^e$	$1^d$	$1^d$	$1^a$	5	-	-	-	-	$126^c$	$7^e$	$1^d$	$1^d$	$1^a$	6	-	-	-	-	-	$84^c$	$4^e$	$1^d$	$1^a$	7	-	-	-	-	-	-	$36^c$	$1^d$	$1^a$	8	-	-	-	-	-	-	-	$9^c$	$1^a$	9	-	-	-	-	-	-	-	-	$1^a$	<table style="margin: auto;"> <tr><td colspan="2"></td><td colspan="10" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">6</td><td style="text-align: center;">7</td><td style="text-align: center;">8</td><td style="text-align: center;">9</td><td style="text-align: center;">10</td></tr> <tr><td rowspan="10" style="vertical-align: middle;"><math>k</math></td><td style="text-align: center;">1</td><td style="text-align: center;"><math>10^b</math></td><td style="text-align: center;"><math>5^b</math></td><td style="text-align: center;"><math>3^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>2^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^b</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>45^c</math></td><td style="text-align: center;"><math>8^n</math></td><td style="text-align: center;"><math>3^o</math></td><td style="text-align: center;"><math>2^{Jp}</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>120^c</math></td><td style="text-align: center;"><math>14^q</math></td><td style="text-align: center;"><math>2^{Jp}</math></td><td style="text-align: center;"><math>2^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>210^c</math></td><td style="text-align: center;"><math>14^n</math></td><td style="text-align: center;"><math>3^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>252^c</math></td><td style="text-align: center;"><math>14^e</math></td><td style="text-align: center;"><math>3^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>210^c</math></td><td style="text-align: center;"><math>8^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>120^c</math></td><td style="text-align: center;"><math>5^e</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>45^c</math></td><td style="text-align: center;"><math>1^d</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">9</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>10^c</math></td><td style="text-align: center;"><math>1^a</math></td></tr> <tr><td style="text-align: center;">10</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;">-</td><td style="text-align: center;"><math>1^a</math></td></tr> </table> <p>(i) <math>L_1(10, n; k)</math></p>			$n$												1	2	3	4	5	6	7	8	9	10	$k$	1	$10^b$	$5^b$	$3^b$	$2^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^b$	$1^a$	2	-	$45^c$	$8^n$	$3^o$	$2^{Jp}$	$1^d$	$1^d$	$1^d$	$1^d$	$1^a$	3	-	-	$120^c$	$14^q$	$2^{Jp}$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$	4	-	-	-	$210^c$	$14^n$	$3^e$	$1^d$	$1^d$	$1^d$	$1^a$	5	-	-	-	-	$252^c$	$14^e$	$3^e$	$1^d$	$1^d$	$1^a$	6	-	-	-	-	-	$210^c$	$8^e$	$1^d$	$1^d$	$1^a$	7	-	-	-	-	-	-	$120^c$	$5^e$	$1^d$	$1^a$	8	-	-	-	-	-	-	-	$45^c$	$1^d$	$1^a$	9	-	-	-	-	-	-	-	-	$10^c$	$1^a$	10	-	-	-	-	-	-	-	-	-	$1^a$
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8	9																																																																																																																																																																																																																																															
$k$	1	$9^b$	$4^b$	$3^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																															
	2	-	$36^c$	$7^l$	$2^{Jm}$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	3	-	-	$84^c$	$9^n$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	4	-	-	-	$126^c$	$9^e$	$3^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	5	-	-	-	-	$126^c$	$7^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	6	-	-	-	-	-	$84^c$	$4^e$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	7	-	-	-	-	-	-	$36^c$	$1^d$	$1^a$																																																																																																																																																																																																																																															
	8	-	-	-	-	-	-	-	$9^c$	$1^a$																																																																																																																																																																																																																																															
	9	-	-	-	-	-	-	-	-	$1^a$																																																																																																																																																																																																																																															
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8	9	10																																																																																																																																																																																																																																														
$k$	1	$10^b$	$5^b$	$3^b$	$2^b$	$2^b$	$1^b$	$1^b$	$1^b$	$1^b$	$1^a$																																																																																																																																																																																																																																														
	2	-	$45^c$	$8^n$	$3^o$	$2^{Jp}$	$1^d$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	3	-	-	$120^c$	$14^q$	$2^{Jp}$	$2^e$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	4	-	-	-	$210^c$	$14^n$	$3^e$	$1^d$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	5	-	-	-	-	$252^c$	$14^e$	$3^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	6	-	-	-	-	-	$210^c$	$8^e$	$1^d$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	7	-	-	-	-	-	-	$120^c$	$5^e$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	8	-	-	-	-	-	-	-	$45^c$	$1^d$	$1^a$																																																																																																																																																																																																																																														
	9	-	-	-	-	-	-	-	-	$10^c$	$1^a$																																																																																																																																																																																																																																														
	10	-	-	-	-	-	-	-	-	-	$1^a$																																																																																																																																																																																																																																														

Table 3.1: Complete lottery numbers,  $L_1(m, n; k)$ , for the lottery  $\langle m, n; k \rangle$ ,  $1 \leq k \leq n \leq m \leq 10$ . Motivation of table entries is as follows: <sup>a</sup>Theorem 2.3(a). <sup>b</sup>Theorem 2.3(b). <sup>c</sup>Theorem 2.3(c). <sup>d</sup>Theorem 2.3(d). <sup>e</sup>Theorem 3.3(e). <sup>f</sup>Theorem 2.2(c). <sup>g</sup> $L_1(6, 3; 2) \leq 2$  (see Figure 3.11(t)). <sup>h</sup> $L_1(7, 3; 2) > 3$  (brute force) and  $L_1(7, 3; 2) \leq 4$  (see Figure 3.11(ee)). <sup>i</sup> $L_1(8, 3; 2) > 4$  (brute force) and  $L_1(8, 3; 2) \leq 5$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3\}, \{1, 2, 6\}, \{1, 4, 5\}, \{2, 5, 8\}, \{3, 4, 7\}\}$ . <sup>j</sup> $L_1(8, 4; 2) \leq 2$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$ . <sup>k</sup> $L_1(8, 4; 3) > 5$  (brute force) and  $L_1(8, 4; 3) \leq 6$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3, 6\}, \{1, 3, 5, 7\}, \{1, 4, 5, 8\}, \{2, 3, 4, 6\}, \{2, 6, 7, 8\}, \{3, 4, 6, 7\}\}$ . <sup>l</sup> $L_1(9, 3; 2) > 6$  (brute force) and  $L_1(9, 3; 2) \leq 7$  by the complete lottery set  $\mathcal{L} = \{\{1, 3, 7\}, \{1, 5, 6\}, \{1, 6, 8\}, \{2, 4, 7\}, \{2, 4, 8\}, \{3, 5, 9\}, \{4, 6, 9\}\}$ . <sup>m</sup> $L_1(9, 4; 2) \leq 2$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$ . <sup>n</sup>Using LINGO to solve (2.10). <sup>o</sup> $L_1(10, 4; 2) > 2$  (brute force) and  $L_1(10, 4; 2) \leq 3$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3, 4\}, \{3, 7, 9, 10\}, \{4, 5, 6, 8\}\}$ . <sup>p</sup> $L_1(10, 5; 2) \leq L_1(10, 5; 3) \leq 2$  by the complete lottery set  $\mathcal{L} = \{\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10\}\}$ . <sup>q</sup>Using a parent-child search tree technique described in §6.1.2. Boldface entries indicate previously unknown complete lottery numbers.



# Chapter 4

## Analytic bounds

“In science one tries to tell people, in such a way as to be understood by everyone, something that no one ever knew before. But in poetry, it’s the exact opposite.”

*Paul Dirac (1902–1984) [206]*

In this chapter known bounds from the literature on the lower domination number  $\gamma$  (and hence the complete lottery number  $L_1(m, n; k)$ ) are presented<sup>1</sup>. In some cases, bounds on the incomplete lottery number  $L_\psi(m, n; k)$  (and hence also bounds on the resource utilisation number  $\Psi_\ell(m, n; k)$ ) are also presented. These bounds are compared (in §4.3) to give the reader an indication of the best known analytic bounds on  $L_1(m, n; k)$ .

### 4.1 Graph theoretic bounds on $L_\psi(m, n; k)$

The concept of domination in graphs has been studied extensively [103] and was formally introduced by Ore [197] and Berge [22]. In this section a wide range of graph domination results are presented that are related to the complete lottery problem, as stated in §1.2. Throughout this section the parameters  $m$ ,  $n$  and  $k$  are in the relation  $1 \leq k \leq n \leq m$ , whilst  $0 < \psi \leq 1$ .

This section is divided into two parts, presenting respectively lower (§4.1.1) and upper (§4.1.2) bounds on the lower domination number  $\gamma$ , for general graph classes, with all results chronologically ordered. These bounds are then interpreted in the context of the lottery graph  $G(m, n; k)$ .

#### 4.1.1 Lower bounds on $L_1(m, n; k)$

A closed form general lower bound for the complete lottery number  $L_1(m, n; k)$  may be found by using a well-known result from graph domination theory due to Walikar, *et al.* [264] in 1979. In any graph  $\mathcal{G}$  of order  $p$  with maximal degree  $\Delta(\mathcal{G})$ , any vertex can dominate at most  $\Delta(\mathcal{G}) + 1$  vertices and hence at least  $p/(\Delta(\mathcal{G}) + 1)$  vertices are required to dominate the entire vertex set  $V(\mathcal{G})$ , thereby establishing the lower bound

$$L_1(m, n; k) \geq \frac{\binom{m}{n}}{r + 1}, \quad (4.1)$$

---

<sup>1</sup>The reader may wonder why mainly bounds on  $L_1(m, n; k)$  are presented. This is because domination in the field of graph theory has not been studied from a partial point of view — this is a novel contribution of this dissertation, to the best knowledge of the author. Moreover, the incomplete lottery and resource utilisation problems are completely new contributions to the combinatorial literature and hence no bounds currently exist for the parameters  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$ .

where  $r$  is the degree of regularity of the lottery graph, presented in (2.1). Note that this lower bound coincides with that given in Theorem 2.1 in the special case where  $\psi = 1$ . It is also possible to obtain the lower bound in (4.1) by considering the number of vertices dominated by a given vertex subset. Suppose  $\mathcal{D}$  is any subset of the vertex set  $V(\mathcal{G})$  of a graph  $\mathcal{G}$  and let  $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_{|\mathcal{D}|}$  denote the sets of vertices dominated by the vertices  $v_1, v_2, \dots, v_{|\mathcal{D}|}$  in  $\mathcal{D}$  respectively. Additionally define

$$\begin{aligned} \mathcal{D}_1 &= |\mathcal{A}_1| + |\mathcal{A}_2| + \dots + |\mathcal{A}_{|\mathcal{D}|}| \\ \mathcal{D}_2 &= |\mathcal{A}_1 \cap \mathcal{A}_2| + |\mathcal{A}_1 \cap \mathcal{A}_3| + \dots + |\mathcal{A}_{|\mathcal{D}|-1} \cap \mathcal{A}_{|\mathcal{D}|}| \\ \mathcal{D}_3 &= |\mathcal{A}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3| + |\mathcal{A}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_4| + \dots + |\mathcal{A}_{|\mathcal{D}|-2} \cap \mathcal{A}_{|\mathcal{D}|-1} \cap \mathcal{A}_{|\mathcal{D}|}| \\ &\vdots \\ \mathcal{D}_{|\mathcal{D}|} &= |\mathcal{A}_1 \cap \mathcal{A}_2 \cap \dots \cap \mathcal{A}_{|\mathcal{D}|}|, \end{aligned}$$

where  $\mathcal{D}_i$  denotes the number of vertices in  $\mathcal{G}$  dominated by  $i$  elements of  $\mathcal{D}$ . The total number of vertices in  $\mathcal{G}$  dominated by  $\mathcal{D}$  is given by

$$D = \sum_{i=1}^{|\mathcal{D}|} (-1)^{i+1} |\mathcal{D}_i|, \quad (4.2)$$

by utilisation of the inclusion–exclusion principle [153]. Note that, after each addition in (4.2), an upper bound on  $D$  is found, while a lower bound on  $D$  is established after each subtraction. Note also that the well-known domination lower bound (4.1) may be obtained from (4.2) by truncating the series (4.2) after the first term. To see this, observe that for the lottery graph  $G\langle m, n; k \rangle$

$$\mathcal{D}_1 = |\mathcal{A}_1| + |\mathcal{A}_2| + \dots + |\mathcal{A}_{|\mathcal{D}|}| = |\mathcal{D}|(r+1), \quad (4.3)$$

where  $r$  is defined in (2.1). Hence, to dominate the entire vertex set  $V(G\langle m, n; k \rangle)$ , we require that

$$\binom{m}{n} = D \leq \mathcal{D}_1 = |\mathcal{D}|(r+1),$$

rendering the lower bound

$$L_1(m, n; k) \geq \frac{\binom{m}{n}}{r+1}, \quad (4.4)$$

which is exactly the lower bound (4.1). In fact, a lower bound on  $L_1(m, n; k)$  may be obtained by truncating (4.2) after any odd-numbered term, and the lower bound of course improves as the truncation is postponed. However, the calculation of  $|\mathcal{D}_i|$  for  $i > 2$  becomes combinatorially rather complex and we therefore refrain from further investigation into this approach.

#### 4.1.2 Upper bounds on $L_\psi(m, n; k)$

Ore [197] first stated in 1962 that, for any connected order  $p$  graph  $\mathcal{G}$  with no isolated vertices,  $\gamma(\mathcal{G}) \leq \frac{p}{2}$ . This bound is obtained by including every vertex at either an even or odd distance from a specified vertex  $u$  in a spanning tree of  $\mathcal{G}$  in a dominating set for  $\mathcal{G}$ . In terms of the lottery numbers, this bound translates to

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{1}{2} \binom{m}{n}. \quad (4.5)$$

In a paper by Vizing [263] in 1965, it was also shown that, for any graph  $\mathcal{G}$  of order  $p$  and size  $q$ ,  $\gamma(\mathcal{G}) \leq \frac{1+2p-\sqrt{8q+1}}{2}$ , which implies that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{1 + 2\binom{m}{n} - \sqrt{4r\binom{m}{n} + 1}}{2} \quad (4.6)$$

where  $r$  is the degree of regularity of the lottery graph  $G\langle m, n; k \rangle$ , as defined in (2.1) and by utilisation of Theorem 3.1. Chen & Zhou [46] obtained similar results in terms of neighbourhood conditions on the dominating vertices in 1999. A contribution by Berge [21] in 1973 stated that, for any order  $p$

graph  $\mathcal{G}$  with maximum degree  $\Delta(\mathcal{G})$ , it holds that  $\gamma(\mathcal{G}) \leq p - \Delta(\mathcal{G})$ . Similar results were achieved by Sampathkumar & Latha [218]. In terms of the lottery parameters, this result translates to

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \binom{m}{n} - r. \quad (4.7)$$

A bound obtained by Arnautov [8] in 1974 (and independently by Lovász [147] in 1975), stating that  $\gamma(\mathcal{G}) \leq \frac{p}{\delta(\mathcal{G})+1} \left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{\delta(\mathcal{G})+1}\right)$ , holds for any graph  $\mathcal{G}$  on  $p$  vertices with minimum degree  $\delta(\mathcal{G})$ , from which it may be deduced that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{\binom{m}{n}}{r+1} \sum_{i=1}^{r+1} \frac{1}{i}. \quad (4.8)$$

Arnaudov [8] (1974) and Payan [203] (1975) both derived the asymptotic result of a well-known theorem stating that, for any order  $p$  graph  $\mathcal{G}$  without isolated vertices,  $\gamma(\mathcal{G}) \leq p[1 + \ln(\delta(\mathcal{G}) + 1)]/[\delta(\mathcal{G}) + 1]$ . In terms of the lottery parameters it therefore follows that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \binom{m}{n} \frac{1 + \ln(r+1)}{r+1}. \quad (4.9)$$

Independently, Payan [203] in 1975 and Marcu [152] in 1986 proved that  $\gamma(\mathcal{G}) \leq (p - \Delta(\mathcal{G}) - 1)(p - \delta(\mathcal{G}) - 2)/(p - 1) + 2$ , which translates to

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{\left(\binom{m}{n} - r - 1\right) \left(\binom{m}{n} - r - 2\right)}{\binom{m}{n} - 1} + 2. \quad (4.10)$$

In 1985, Caro & Roditty [43] proved that  $\gamma(\mathcal{G}) \leq p[1 - \delta(\mathcal{G})((\delta(\mathcal{G}) + 1)^{-1-1/\delta(\mathcal{G})})]$  for any order  $p$  graph  $\mathcal{G}$  with  $\delta(\mathcal{G}) \geq 7$ . It therefore follows that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \binom{m}{n} \left[1 - r \left(\frac{1}{r+1}\right)^{1+\frac{1}{r}}\right] \text{ if } r \geq 7. \quad (4.11)$$

A conjecture by Vizing [263] in 1965, stating that for any order  $p$  graph  $\mathcal{G}$ ,  $\gamma(\mathcal{G}) \leq \frac{1}{2}(p + 1 - \delta(\mathcal{G}))$  was proved by Flach & Volkmann [76] in 1990, implying that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{1}{2} \left( \binom{m}{n} - r + 1 \right). \quad (4.12)$$

In 1965, Vizing [263] proved a theorem for any  $(p, q)$  graph  $\mathcal{G}$  for which  $\Delta(\mathcal{G}) = p - \gamma(\mathcal{G})$ . This result was generalised in 1994 by Fulman [82] for graphs that satisfy  $2 \leq \gamma(\mathcal{G}) \leq p - \Delta(\mathcal{G})$  and stated that  $q \leq \lfloor \frac{1}{2}[(p - \gamma(\mathcal{G}))(p - \gamma(\mathcal{G}) + 2) - \Delta(\mathcal{G})(p - \gamma(\mathcal{G}) - \Delta(\mathcal{G}))] \rfloor$ . Recall that, by Theorem 3.4(a),  $L_1(m, n; k) \geq 2$  if and only if  $2n \geq m + k$ . In terms of the lottery parameters, this is equivalent to

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{\binom{m}{n}}{2} + 1 - \frac{r}{2} - \sqrt{\left(\frac{\binom{m}{n}}{2}\right)^2 - r \left(\frac{3r}{4} - \frac{3}{2} \binom{m}{n} + 1\right)} + 1 \text{ if } 2n \geq m + k \quad (4.13)$$

by (4.7). McCuaig & Shepherd [155] conjectured in 1989 that if an order  $p$  graph  $\mathcal{G}$  is connected and  $\delta(\mathcal{G}) \geq 3$ , then  $\gamma(\mathcal{G}) \leq \frac{3}{8}p$ . This conjecture was proved by Reed [208] in 1996, which translates to the lottery bound

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{3}{8} \binom{m}{n} \text{ if } r \geq 3. \quad (4.14)$$

The bound in (4.8) was improved by Clark, *et al.* [49] in 1998, who proved, for any graph  $\mathcal{G}$  on  $p$  vertices, that  $\gamma(\mathcal{G}) \leq (1 - S_{\delta(\mathcal{G})})p$  where  $S_{\delta(\mathcal{G})} = \prod_{i=1}^{\delta(\mathcal{G})+1} \frac{i}{i+1/\delta(\mathcal{G})}$ . With the additional constraint that  $\mathcal{G}$  is regular, they improved this bound even further, yielding the upper bound

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \binom{m}{n} \left(1 - \frac{r^3 + r}{r^3 + 1} \prod_{j=1}^r \left(1 + \frac{1}{jr}\right)^{-1}\right). \quad (4.15)$$

In the same paper numerous other bounds on  $\gamma(\mathcal{G})$  are given in terms of the minimum degree  $\delta(\mathcal{G})$  and degree sequence  $\mathbf{d} = (d_1, d_2, \dots, d_p)$  (where  $d_i = \deg_{\mathcal{G}} v_i$ ) of an order  $p$  graph  $\mathcal{G}$ . Harant, *et al.* [100] also proved bounds on  $\gamma(\mathcal{G})$  using degree sequence probability measures. Furthermore, Haynes, *et al.* [103] stated in 1998 that  $\gamma(\mathcal{G}) \leq \delta(\mathcal{G})$  for any order  $p$  graph  $\mathcal{G}$  with  $\text{diam}(\mathcal{G}) = 2$ . It therefore follows that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq r \text{ if } \text{diam}(G\langle m, n; k \rangle) = 2. \quad (4.16)$$

Finally, using the same notation as in §4.1.1 we may derive an upper bound on the complete lottery number by truncating the series in (4.2) after the second term. To dominate the entire vertex set  $V(G\langle m, n; k \rangle)$  and utilising (4.3), we require that

$$\binom{m}{n} = D \geq \mathcal{D}_1 - \mathcal{D}_2 \geq |\mathcal{D}|(r+1) - \binom{|\mathcal{D}|}{2}x,$$

where  $\mathcal{D}_2 \leq \binom{|\mathcal{D}|}{2}x$  and  $|\mathcal{A}_i \cap \mathcal{A}_j| \geq x$  for all  $i \neq j$ . Here  $x$  represents a lower bound on the minimum number of vertices collectively dominated by any pair of vertices in the dominating set  $\mathcal{D}$ . By Lemma 3.1 it follows that the number of common elements between any two  $n$ -sets from  $\mathcal{U}_m$ , is a strictly decreasing function of the number of uniquely dominated vertices of the respective vertices. We therefore deduce that  $x = \sum_{t=k}^{n-1} \sum_{s=k}^{n-1} \binom{n}{s} \binom{n}{t} \binom{m-2n}{n-s-t}$  (by utilisation of Lemma 2.2) from which  $|\mathcal{D}|$  may be solved in

$$2 \binom{m}{n} \geq |\mathcal{D}|(2(r+1) + x) - |\mathcal{D}|^2x.$$

This yields the upper bound

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{2(r+1) + x + \sqrt{(2(r+1) + x)^2 - 8x \binom{m}{n}}}{2x}. \quad (4.17)$$

## 4.2 Other bounds on $L_\psi(m, n; k)$

Bounds on  $L_1(m, n; k)$  may also be obtained from a different field in combinatorics, namely design theory, and also from linear algebra. This section is devoted to presenting these results and is again divided into two subsections; one presenting lower bounds on  $L_1(m, n; k)$  (§4.2.1) and one presenting upper bounds on  $L_\psi(m, n; k)$  (§4.2.2). Throughout the following chronologically ordered results the parameters  $m$ ,  $n$  and  $k$  are in the relation  $1 \leq k \leq n \leq m$ , whilst  $0 < \psi \leq 1$ .

### 4.2.1 Lower bounds on $L_1(m, n; k)$

It is clear that at least  $C(m-1, n-1; k-1)$  elements of a covering set  $\mathcal{C}$  for  $\langle m, n; k \rangle$  contain the number  $j \in \mathcal{U}_m$ . Such a number  $j \in \mathcal{U}_m$  may be chosen in  $m$  possible, different ways for any of  $x$   $n$ -sets in  $\mathcal{C}$ , where  $C(m, n; k) \geq x \geq \lceil mC(m-1, n-1; k-1)/n \rceil$ . Therefore

$$C(m, n; k) \geq \left\lceil \frac{m}{n} C(m-1, n-1; k-1) \right\rceil, \quad (4.18)$$

which is known in design theoretic literature as Schönheim's covering bound [223], and dates from 1964. A generalised recursive lower bound, from which (4.18) may be derived, was established by Li & Van Rees [138] in 1999 and states that

$$L_1(m, n; k) \geq \frac{\binom{m}{n-k+1} L_1(m-n+k-1, n; k)}{\binom{m}{n-k+1} - \binom{n}{n-k+1}} \text{ if } m \geq 2n-k+1. \quad (4.19)$$

In 1983 De Caen [59] proved a graph theoretic result giving a lower bound on the well-known Turán number<sup>2</sup>,

$$T(m, n; k) \geq \left\lceil \binom{m}{k} / \binom{n-1}{k-1} \right\rceil \left\lceil \frac{m-n+1}{m-k+1} \right\rceil.$$

<sup>2</sup>The Turán number  $T(m, n; k)$  [256] is defined as the minimal number of  $n$ -subsets from  $\mathcal{U}_m$  such that each  $k$ -subset of  $\mathcal{U}_m$  contains one of them as subset. It is known that  $T(m, n; k) = C(m, m-n; m-k)$ .

This, together with a result by Bate [13, 234] in 1978 and Brouwer & Voorhoeve [37] in 1979, stating that  $L_1(m, n; k) \geq T(m, n; k) / \binom{n}{k}$ , establishes the lower bound on the lottery number

$$L_1(m, n; k) \geq \frac{\binom{m}{k}}{\binom{n-1}{k-1} \binom{n}{k}} \frac{m-n+1}{m-k+1}. \quad (4.20)$$

Schrijver [225] (in 1979) reported two bounds on  $L_1(m, n; k)$  due to Sterboul [236]<sup>3</sup>, in 1978, stating that

$$L_1(m, n; k) \geq \max_{n \leq a \leq m} \left\{ \frac{(a-n+1) \binom{m}{a}}{\sum_{i=k}^n \binom{n}{i} \binom{m-n}{a-i} (i-k+1)} \right\} \quad (4.21)$$

and

$$L_1(m, n; k) \geq \max_{n \leq a \leq m} \left\{ \left\lceil \frac{\left\lfloor \frac{a-n+1}{n-k+1} \right\rfloor \binom{m}{a}}{\sum_{i=k}^n \binom{n}{i} \binom{m-n}{a-i}} \right\rceil \right\}. \quad (4.22)$$

A lower bound on  $L(m, n; k)$ , established by Nurmela & Östergård [194] in 1993, states that

$$L_1(m, n; k) \geq \frac{\binom{m}{n}}{\sum_{i=k}^n \binom{n}{i} \binom{m-n}{n-i}}. \quad (4.23)$$

This bound is, in fact, the same as the graph theoretic lower bound given in (4.1), by utilisation of (2.1). All of the above mentioned bounds are general. A theorem of Hanani, *et al.* [99] in 1964, proved for the specific class of lottery numbers  $L_1(m, n; 2)$ , states that

$$L_1(m, n; 2) \geq \frac{m(m-n+1)}{n(n-1)^2}. \quad (4.24)$$

The same class of lottery numbers was also considered in a paper by Füredi, *et al.* [83] in 1996, who additionally proved that

$$L_1(m, n; 2) \geq \min_{a_1 + \dots + a_{n-1} = m} \left\lceil \sum_{i=1}^{n-1} \frac{a_i}{n} \left\lfloor \frac{a_i - 1}{n-1} \right\rfloor \right\rceil. \quad (4.25)$$

### 4.2.2 Upper bounds on $L_\psi(m, n; k)$

A well-known upper bound for an  $\langle m, n; k \rangle$  packing design (with a similar proof to that of (4.18)) was formulated by Schönheim [223] in 1964. This result states that

$$P(m, n; k) \leq \left\lfloor \frac{m}{n} P(m-1, n-1; k-1) \right\rfloor, \quad (4.26)$$

which, when applied recursively (together with (2.2)), yields the closed form upper bound

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{m}{n} \left\lfloor \frac{m-1}{n-1} \left\lfloor \dots \left\lfloor \frac{m-k+1}{n-k+1} \right\rfloor \dots \right\rfloor \right\rfloor. \quad (4.27)$$

<sup>3</sup>The author could not obtain a copy of [236] to verify this claim by Schrijver [225]. Consequently the author cannot provide the reader with an idea of the nature of the arguments used to obtain the bounds (4.21) and (4.22).



A formula for a certain class of packing numbers (giving an upper bound on the same class of complete lottery numbers), established in 1978 by Brouwer [34], is given by  $P(m, 4; 3) = m(m^2 - 3m - 6)/24$ , where  $m$  is a multiple of 6, implying that

$$L_\psi(m, 4; 3) \leq L_1(m, 4; 3) \leq m(m^2 - 3m - 6)/24 \text{ if } m \equiv 0 \pmod{6}. \quad (4.28)$$

Let  $\lambda$  denote the largest multiplicity of an eigenvalue of the adjacency matrix  $A_{\mathcal{G}}$  of a graph  $\mathcal{G}$ . In 1994, Rowlinson [52, 214] improved a result by Van Nuffelen [259] (in 1982) relating eigenvalues of  $\mathcal{A}_{\mathcal{G}}$  and  $\gamma(\mathcal{G})$ , which translates to

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \binom{m}{n} - \lambda. \quad (4.29)$$

In 1996 Godbole, *et al.* [91] used probabilistic techniques (extending a result of Erdős & Spencer [72]) to obtain a general upper bound on  $C(m, n; k)$ , which translates to the bound

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \frac{\binom{m}{n} \log \binom{n}{k}}{\binom{n}{k}} \quad (4.30)$$

via utilisation of (2.2). Finally, two recursive bounds on  $L_1(m, n; k)$ , established by Li & Van Rees [138] in 1999, state that

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq L_1(m-1, n-1; k-1) + L_1(m-1, n; k) \quad (4.31)$$

and

$$L_\psi(m, n; k) \leq L_1(m, n; k) \leq \left\lfloor 2 - \frac{n-1}{m-1} L_1(m-1, n-1; k) \right\rfloor + L_1(m-1, n; k). \quad (4.32)$$

### 4.3 Comparison of analytic bounds on $L_1(m, n; k)$

Known complete lottery numbers and, in cases of yet undetermined complete lottery numbers, a selection of the best known bounds on  $L_1(m, n; k)$  for  $2 \leq k \leq n \leq 6$  and  $m \leq 50$  (obtained from literature and presented in §4.1 and §4.2), are given in Table 4.1. Table 4.2 contains a comparison of the best upper and lower bounds on the (realistic cases of the) complete lottery number  $L_1(m, 6; 3)$  for all  $6 \leq m \leq 50$ . As may be seen from Table 4.2, the upper and lower bounds for realistic values of  $m$  are typically far apart. In fact, the construction of specific playing sets (using algorithmic techniques) yield far better upper bounds than any of the analytic approaches. With this in mind, the focus in Chapter 5 will be on the evaluation and comparison of playing set construction algorithms.

Although the graph theoretic lower bound (given in (4.1)) does not perform as well in comparison to the best lower bounds (in §4.2.1), it presents a closed-form lower bound which has the potential of significant improvement via the method outlined in §4.1. For realistic cases of the complete lottery problem, the design and graph theoretic Turán bound in (4.20) yields the best lower bound on  $L_1(m, n; k)$  (*e.g.*, for the realistic case of the South African National Lottery *Lotto*, (4.20) yields the lower bound  $L_1(49, 6; 3) \geq 87$ , as shown in Table 4.2).

A small improvement of the Arnautov upper bound (4.8), with the additional regularity condition, represents the best analytic upper bound on  $L_1(m, n; k)$ , established by Clark, *et al.* (4.15) (*e.g.*, for the realistic case of the South African National Lottery *Lotto*, (4.15) yields the upper bound  $L_1(49, 6; 3) \leq 700$ , as shown in Table 4.2).

### 4.4 Chapter summary

This chapter contains a survey of known bounds on the complete lottery number  $L_1(m, n; k)$  from the graph theoretic literature in §4.1 and other areas of combinatorics in §4.2. The best known analytic bounds on  $L_1(m, n; k)$  for  $1 \leq k \leq n \leq 6$  and  $3 \leq m \leq 50$  are given in Table 4.1, while bounds on the specific class of complete lottery numbers  $L_1(m, 6; 3)$  are compared in Table 4.2 for all  $6 \leq m \leq 50$ .

$m$	$L_1(m, 3; 2)$	$L_1(m, 4; 2)$	$L_1(m, 4; 3)$	$L_1(m, 5; 2)$	$L_1(m, 5; 3)$	$L_1(m, 5; 4)$
3	$1^a$	—	—	—	—	—
4	$1^a$	$1^a$	$1^{o,r}$	—	—	—
5	$2^a$	$1^a$	$1^{o,r}$	$1^{o,r}$	$1^{o,r}$	$1^{o,r}$
6	$2^a$	$1^a$	$2^{o,g} : 3^s$	$1^{o,r}$	$1^{o,r}$	$1^{o,r}$
7	$4^a$	$2^a$	$3^{o,g} : 7^n$	$1^{o,r}$	$1^{o,r}$	$2^g, j, k, l, o : 5^{m, n, q, t}$
8	$5^a$	$2^a$	$5^{o,l} : 12^n$	$1^{o,r}$	$2^{o,j,k,l} : 3^{p,s}$	$4^{k,l,o} : 10^n$
9	$7^a$	$2^a$	$9^{o,k,l} : 20^n$	$2^{o,p}$	$2^{o,j,k,l} : 7^{m,n}$	$6^g, k, l, o : 20^n$
10	$8^a$	$3^a$	$9^{o,g,k,l} : 30^{2n}$	$2^o : 4^p$	$2^g, o, j, k, l : 10^{m,n}$	$10^{k,l,o} : 35^n$
11	$10^c$	$3^a$	$13^g : 42^n$	$2^o : 7^{m,n}$	$3^g, o, j, k, l : 14^{m,n}$	$15^g, k, l, o : 56^n$
12	$11^a$	$3^a$	$17^g : 58^n$	$2^o : 9^{m,q}$	$4^{o,k,l} : 19^{m,n}$	$22^g, k, l, o : 87^n$
13	$13^a$	$5^a$	$22^g : 77^n$	$3^{j,k} : 10^{m,n}$	$5^{o,l} : 25^{m,n}$	$33^g : 129^n$
14	$14^a$	$5^a$	$28^g : 100^n$	$3^{j,k} : 12^{m,n}$	$6^g, j : 32^{m,n}$	$46^g : 184^n$
15	$18^a$	$7^a$	$35^g : 127^n$	$3^{j,g} : 14^{m,n}$	$7^g, j : 40^{m,n}$	$63^g : 256^n$
16	$19^a$	$7^a$	$44^g : 160^n$	$3^{j,g} : 16^{m,n}$	$9^j : 50^{m,n}$	$84^g : 347^n$
17	$23^a$	$9^a$	$53^g : 197^n$	$4^{j,k} : 18^{m,q}$	$10^g : 61^{m,n}$	$111^g : 461^n$
18	$24^a$	$9^a$	$64^g : 240^n$	$4^{j,g} : 20^{m,n}$	$12^g : 73^n$	$143^g : 601^n$
19	$29^a$	$11^a$	$76^g : 289^n$	$4^{j,g} : 22^{m,n}$	$15^g : 88^{m,n}$	$182^g : 771^n$
20	$31^a$	$12^a$	$90^g : 344^n$	$4^{j,g} : 25^{m,n}$	$17^g : 104^{m,n}$	$228^g : 976^n$
21	$36^a$	$12^a : 14^a$	$105^g : 406^n$	$6^f : 27^{m,n}$	$20^g : 122^{m,n}$	$283^g : 1\ 219^n$
22	$38^a$	$13^a, f : 14^a$	$122^g : 475^n$	$6^f : 30^{m,n}$	$24^g : 142^n$	$347^g : 1\ 505^n$
23	$43^{c,f}$	$14^a, f : 17^a$	$141^g : 551^n$	$7^f : 33^{m,n}$	$27^g : 165^{m,n}$	$421^g : 1\ 840^n$
24	$45^a$	$16^a : 18^a$	$161^g : 636^n$	$7^f : 36^{m,n}$	$31^g : 189^n$	$506^g : 2\ 228^n$
25	$50^a$	$17^a, f : 18^a$	$184^g : 729^n$	$8^f : 39^{m,n}$	$35^g : 216^n$	$604^g : 2\ 676^n$
26	$52^a$	$18^a, f : 21^a$	$208^g : 832^n$	$9^f : 43^{m,n}$	$40^g : 246^{m,n}$	$715^g : 3\ 189^n$
27	$59^a$	$20^a : 22^a$	$234^g : 943^n$	$9^f : 46^{m,n}$	$45^g : 278^n$	$841^g : 3\ 773^n$
28	$61^a$	$21^a, f : 22^a$	$263^g : 1\ 065^n$	$10^f : 50^{m,n}$	$51^g : 313^n$	$983^g : 4\ 435^n$
29	$68^a$	$23^a : 25^a$	$294^g : 1\ 196^n$	$11^f : 54^{m,n}$	$57^g : 351^n$	$1\ 142^g : 5\ 181^n$
30	$70^a$	$24^a : 27^a$	$327^g : 1\ 338^n$	$11^f : 58^{m,n}$	$63^g : 392^n$	$1\ 320^g : 6\ 019^n$
31	$78^a$	$26^a, f : 27^a$	$362^g : 1\ 492^n$	$12^f : 62^{m,n}$	$70^g : 436^n$	$1\ 518^g : 6\ 956^n$
32	$81^a$	$28^a : 29^a$	$400^g : 1\ 657^n$	$12^{f,g,h} : 67^{m,n}$	$78^g : 484^{m,n}$	$1\ 736^g : 8\ 000^n$
33	$89^a$	$30^a : 31^a$	$440^g : 1\ 834^n$	$14^f : 71^{m,n}$	$86^g : 535^{m,n}$	$1\ 978^g : 9\ 158^n$
34	$92^a$	$32^a, f$	$484^g : 2\ 023^n$	$14^f : 76^{m,n}$	$94^g : 589^n$	$2\ 244^g : 10\ 439^n$
35	$100^{c,f}$	$33^a, f : 34^a$	$529^g : 2\ 225^n$	$14^{f,g,h} : 81^{m,n}$	$103^g : 647^n$	$2\ 537^g : 11\ 852^n$
36	$103^a$	$35^a$	$578^g : 2\ 441^n$	$15^{f,g,h} : 86^{m,n}$	$112^g : 709^n$	$2\ 856^g : 13\ 405^n$
37	$111^a$	$37^a, f$	$629^g : 2\ 671^n$	$17^f : 91^{m,n}$	$123^g : 775^n$	$3\ 206^g : 15\ 108^n$
38	$114^a$	$38^f$	$684^g : 2\ 915^n$	$18^f : 96^{m,n}$	$133^g : 845^n$	$3\ 586^g : 16\ 970^n$
39	$124^a$	$39^{f,g,h}$	$741^g : 3\ 174^n$	$18^{f,g,h} : 102^{m,n}$	$145^g : 919^n$	$3\ 999^g : 19\ 002^n$
40	$127^a$	$44^a, f$	$802^g : 3\ 447^n$	$19^f : 108^{m,n}$	$156^g : 998^{m,n}$	$4\ 446^g : 21\ 212^n$
41	$137^a$	$45^f$	$866^g : 3\ 737^n$	$21^f : 113^{m,n}$	$169^g : 1\ 080^n$	$4\ 931^g : 23\ 612^n$
42	$140^a$	$46^{f,g,h}$	$933^g : 4\ 043^n$	$22^f : 119^{m,n}$	$182^g : 1\ 168^n$	$5\ 453^g : 26\ 213^n$
43	$151^a$	$51^a, f$	$1\ 004^g : 4\ 365^n$	$23^f : 126^{m,n}$	$196^g : 1\ 260^n$	$6\ 017^g : 29\ 025^n$
44	$155^a$	$52^f$	$1\ 078^g : 4\ 705^n$	$23^f : 132^{m,n}$	$211^g : 1\ 357^n$	$6\ 622^g : 32\ 061^n$
45	$166^a$	$53^{f,g,h}$	$1\ 155^g : 5\ 062^n$	$26^f : 139^{m,n}$	$226^g : 1\ 459^n$	$7\ 273^g : 35\ 331^n$
46	$170^a$	$58^a, f$	$1\ 237^g : 5\ 437^n$	$26^f : 145^{m,n}$	$242^g : 1\ 566^n$	$7\ 970^g : 38\ 848^n$
47	$181^{c,f}$	$59^f$	$1\ 322^g : 5\ 831^n$	$27^f : 152^{m,n}$	$259^g : 1\ 679^n$	$8\ 716^g : 42\ 624^n$
48	$185^a$	$60^{f,g,h}$	$1\ 410^g : 6\ 243^n$	$27^{f,g,h} : 159^{m,n}$	$276^g : 1\ 797^n$	$9\ 513^g : 46\ 673^n$
49	$196^a$	$66^a, f$	$1\ 503^g : 6\ 675^n$	$30^f : 166^{m,n}$	$294^g : 1\ 920^n$	$10\ 364^g : 51\ 006^n$
50	$200^a$	$67^f$	$1\ 600^g : 7\ 128^n$	$30^f : 174^{m,n}$	$314^g : 2\ 049^n$	$11\ 270^g : 55\ 638^n$

Table 4.1: Known complete lottery numbers and analytic bounds (using the notation lower bound : upper bound) on  $L_1(m, n; k)$ ,  $1 \leq k \leq n$  and  $3 \leq m \leq 50$  for  $n = 3, 4$  and  $5$ . Motivation for table entries are as follows: <sup>a</sup>Class of complete lottery numbers  $L_1(m, 3; 2)$ , (1.2), and  $L_1(m, 4; 2)$  due to Bate & Stanton [15]. <sup>b</sup>Class of complete lottery numbers  $L_1(m, 6; 2)$  due to Bate & Van Rees [17]. <sup>c</sup>Actual bound on  $L_1(m, 3; 2)$  for the case  $m \equiv 11 \pmod{12}$  that differs from that derived by Bate & Stanton [15]. <sup>d</sup>Due to Colbourn [50]. <sup>e</sup>The Payan asymptotic upper bound, (4.9) [203]. <sup>f</sup>The Füredi lower bound, (4.25) [83]. <sup>g</sup>The Turán lower bound, (4.20) [59]. <sup>h</sup>The Hanani lower bound, (4.24) [99]. <sup>i</sup>The recursive lower bound, given in Theorem 2.2(a). <sup>j</sup>The generalised Schönheim lower bound, (4.19) [138]. <sup>k</sup>The Sterboul lower bound, (4.21) [236]. <sup>l</sup>The Sterboul lower bound, (4.22) [236]. <sup>m</sup>The Arnautov upper bound, (4.8) [8]. <sup>n</sup>The Clark, et al. [49] upper bound, (4.15). <sup>o</sup>The graph theoretic lower bound, (4.1) [264]. <sup>p</sup>The Payan and Marcu upper bound, (4.10) [152]. <sup>q</sup>The Caro & Roditty upper bound, (4.11) [43]. <sup>r</sup>The Vizing upper bound, (4.6) [263]. <sup>s</sup>The Fulman upper bound, (4.13) [82]. <sup>t</sup>The Reed upper bound, (4.14) [208].

$m$	$L_1(m, 6; 2)$	$L_1(m, 6; 4)$	$L_1(m, 6; 5)$
6	$1^{o,r}$	$1^{o,r}$	$1^{o,r}$
7	$1^{o,r}$	$1^{o,r}$	$1^{o,r}$
8	$1^{o,r}$	$1^{o,r}$	$3^{l,o} : 6^{m,n}$
9	$1^{o,r}$	$2^{j,k,l,o} : 6^{e,m,n,p,q,s}$	$5^{k,l,o} : 14^n$
10	$1^{o,r}$	$2^{j,k,l,o} : 9^{m,n}$	$9^{k,l,o} : 30^n$
11	$2^{o,p,s}$	$3^{o,l,k} : 14^{m,n}$	$15^{k,l,o} : 56^n$
12	$2^{f,j,k,l,o} : 3^{p,s}$	$4^{o,l,k} : 21^{m,n}$	$25^{k,l,o} : 100^n$
13	$2^{f,j,k,l,o} : 8^{m,n}$	$5^{o,l,k} : 30^{m,n}$	$40^{l,o} : 166^n$
14	$2^{f,j,k,l,o} : 9^{m,n}$	$7^{o,l} : 42^n$	$62^{l,o} : 264^n$
15	$2^{f,j,k,l,o} : 10^{m,n}$	$9^{o,l} : 58^{m,n}$	$91^{g,l,o} : 403^n$
16	$3^{f,j,k,l} : 11^{m,n}$	$11^{g,k,l,o} : 77^n$	$134^g : 597^n$
17	$3^{f,k,l} : 13^{e,m,n,q}$	$14^{g,l,o} : 101^n$	$191^g : 858^n$
18	$3^{f,j,k,l} : 14^{m,n}$	$18^{g,l,o} : 131^{m,n}$	$266^g : 1\ 205^n$
19	$3^{f,j,k,l} : 15^{m,n}$	$23^g : 167^{m,n}$	$362^g : 1\ 657^n$
20	$3^{f,j,k,l} : 17^{m,n}$	$29^g : 209^n$	$485^g : 2\ 236^n$
21	$4^{f,j,k,l} : 18^{m,n}$	$36^g : 259^n$	$639^g : 2\ 968^n$
22	$4^{f,j,k,l} : 20^{m,n}$	$44^g : 319^{m,n}$	$830^g : 3\ 880^n$
23	$4^{f,j,k,l} : 22^{m,n}$	$54^g : 387^n$	$1\ 063^g : 5\ 007^n$
24	$4^{f,g,j,k,l} : 24^{m,n}$	$65^g : 467^n$	$1\ 346^g : 6\ 383^n$
25	$4^{f,g,j,k,l} : 26^{m,n}$	$77^g : 559^{m,n}$	$1\ 687^g : 8\ 048^n$
26	$5^{f,g,k,l} : 28^{m,n}$	$91^g : 663^n$	$2\ 093^g : 10\ 045^n$
27	$5^{f,g,k,l} : 30^{m,n}$	$108^g : 782^n$	$2\ 574^g : 12\ 425^n$
28	$5^{f,g,h,j,k,l} : 32^{m,n}$	$126^g : 916^n$	$3\ 140^g : 15\ 238^n$
29	$5^{f,g,h,j,k,l} : 34^{m,n}$	$147^g : 1\ 068^n$	$3\ 801^g : 18\ 543^n$
30	$5^{f,g,h,j,k,l} : 37^{m,n}$	$170^g : 1\ 237^n$	$4\ 568^g : 22\ 401^n$
31	$7^f : 39^{m,n}$	$195^g : 1\ 427^n$	$5\ 454^g : 26\ 882^n$
32	$7^f : 42^{m,n}$	$224^g : 1\ 637^n$	$6\ 473^g : 32\ 057^n$
33	$7^{f,g,h} : 44^{m,n}$	$255^g : 1\ 871^n$	$7\ 639^g : 38\ 006^n$
34	$8^f : 47^{m,n}$	$290^g : 2\ 129^n$	$8\ 967^g : 44\ 813^n$
35	$8^f : 50^{m,n}$	$328^g : 2\ 413^n$	$10\ 472^g : 52\ 570^n$
36	$9^f : 53^{m,n}$	$369^g : 2\ 725^n$	$12\ 174^g : 61\ 372^n$
37	$10^f : 56^{m,n}$	$415^g : 3\ 066^n$	$14\ 090^g : 71\ 325^n$
38	$10^f : 59^{m,n}$	$464^g : 3\ 440^n$	$16\ 240^g : 82\ 537^n$
39	$11^f : 62^{m,n}$	$518^g : 3\ 847^n$	$18\ 644^g : 95\ 127^n$
40	$11^f : 66^{m,n}$	$577^g : 4\ 290^n$	$21\ 325^g : 109\ 219^n$
41	$12^f : 69^{m,n}$	$640^g : 4\ 770^n$	$24\ 305^g : 124\ 945^n$
42	$13^f : 73^{m,n}$	$708^g : 5\ 290^n$	$27\ 610^g : 142\ 445^n$
43	$13^f : 76^{m,n}$	$782^g : 5\ 853^n$	$31\ 264^g : 161\ 866^n$
44	$13^f : 80^{m,n}$	$861^g : 6\ 459^n$	$35\ 296^g : 183\ 365^n$
45	$13^f : 84^{m,n}$	$946^g : 7\ 112^n$	$39\ 732^g : 207\ 106^n$
46	$15^f : 88^{m,n}$	$1\ 038^g : 7\ 813^n$	$44\ 604^g : 233\ 261^n$
47	$15^f : 91^{m,n}$	$1\ 136^g : 8\ 567^n$	$49\ 943^g : 262\ 012^n$
48	$15^f : 96^{m,n}$	$1\ 240^g : 9\ 374^n$	$55\ 780^g : 293\ 549^n$
49	$16^f : 100^{m,n}$	$1\ 352^g : 10\ 238^n$	$62\ 151^g : 328\ 074^n$
50	$16^f : 104^{m,n}$	$1\ 470^g : 11\ 161^n$	$69\ 090^g : 365\ 795^n$

Table 4.1 (continued): *Known complete lottery numbers and analytic bounds (using the notation lower bound : upper bound) on  $L_1(m, n; k)$ ,  $k = 2, 4$  and  $6$ ,  $n = 6$  and  $3 \leq m \leq 50$ . Motivation for table entries are as follows: <sup>a</sup>Class of complete lottery numbers  $L_1(m, 3; 2)$ , (1.2), and  $L_1(m, 4; 2)$  due to Bate & Stanton [15]. <sup>b</sup>Class of complete lottery numbers  $L_1(m, 6; 2)$  due to Bate & Van Rees [17]. <sup>c</sup>Actual bound on  $L_1(m, 3; 2)$  for the case  $m \equiv 11 \pmod{12}$  that differs from that derived by Bate & Stanton [15]. <sup>d</sup>Due to Colbourn [50]. <sup>e</sup>The Payan asymptotic upper bound, (4.9) [203]. <sup>f</sup>The Füredi lower bound, (4.25) [83]. <sup>g</sup>The Turán lower bound, (4.20) [59]. <sup>h</sup>The Hanani lower bound, (4.24) [99]. <sup>i</sup>The recursive lower bound, given in Theorem 2.2(a). <sup>j</sup>The generalised Schönheim lower bound, (4.19) [138]. <sup>k</sup>The Sterboul lower bound, (4.21) [236]. <sup>l</sup>The Sterboul lower bound, (4.22) [236]. <sup>m</sup>The Arnautov upper bound, (4.8) [8]. <sup>n</sup>The Clark, et al. [49] upper bound, (4.15). <sup>o</sup>The graph theoretic lower bound, (4.1) [264]. <sup>p</sup>The Payan and Marcu upper bound, (4.10) [152]. <sup>q</sup>The Caro & Roditty upper bound, (4.11) [43]. <sup>r</sup>The Vizing upper bound, (4.6) [263]. <sup>s</sup>The Fulman upper bound, (4.13) [82]. <sup>t</sup>The Reed upper bound, (4.14) [208].*

$m$	$\binom{m}{6}$	$r$	$\underline{L}_1$	$\underline{L}_2$	$\underline{L}_3$	$\underline{L}_4$	$\underline{L}_5$	$\underline{L}_6$	$L$	$\overline{L}_1$	$\overline{L}_2$	$\overline{L}_3$	$\overline{L}_4$	$\overline{L}_5$
6	1	0	1	1	1	1	–	1	1	1	1	–	1	1
7	7	6	1	1	1	1	–	1	1	2	2	–	2	1
8	28	27	1	1	1	1	–	1	1	3	3	4	4	1
9	84	83	1	1	1	1	–	1	1	4	5	5	5	1
10	210	194	2	2	2	2	2	1	2	6	6	6	6	–
11	462	380	2	2	2	2	2	1	2	7	7	8	8	–
12	924	661	2	2	2	2	2	1	2	9	9	10	10	2
13	1716	1057	2	2	2	2	2	2	2	12	12	12	12	4
14	3003	1588	2	2	3	3	3	2	4	14	15	15	15	6
15	5005	2274	3	3	3	3	3	2	4	18	18	19	19	9
16	8008	3135	3	3	3	3	3	3	5	22	22	23	23	13
17	12376	4191	3	3	3	4	3	3	6	26	26	27	27	18
18	18564	5462	4	4	4	4	5	4	7	31	31	32	32	24
19	27132	6968	4	4	4	5	5	4	?	36	36	38	38	31
20	38760	8729	5	5	5	5	6	5	?	42	42	44	44	39
21	54264	10765	6	6	6	6	7	6	?	49	49	51	51	49
22	74613	13096	6	6	6	7	6	7	?	57	57	59	59	60
23	100947	15742	7	7	7	8	6	8	?	65	65	68	68	72
24	134596	18723	8	8	8	8	7	9	?	74	74	77	77	86
25	177100	22059	9	9	9	9	8	10	?	84	84	88	88	102
26	230230	25770	9	9	9	10	8	12	?	95	95	99	99	120
27	296010	29876	10	10	10	11	9	13	?	107	107	111	112	139
28	376740	34397	11	11	11	13	10	15	?	120	120	125	125	161
29	475020	39353	13	13	13	14	11	17	?	134	134	139	139	184
30	593775	44764	14	14	14	15	13	19	?	149	149	155	155	210
31	736281	50650	15	15	15	17	14	21	?	165	165	171	172	238
32	906192	57031	16	16	16	18	16	23	?	183	183	189	189	269
33	1107568	63927	18	18	18	20	18	25	?	201	201	209	209	302
34	1344904	71358	19	19	19	21	20	28	?	221	221	229	229	337
35	1623160	79344	21	21	21	23	22	30	?	242	242	251	251	376
36	1947792	87905	23	23	23	25	24	33	?	265	265	274	274	417
37	2324784	97061	24	24	24	27	26	36	?	288	288	298	298	461
38	2760681	106832	26	26	26	29	29	39	?	314	314	325	325	507
39	3262623	117238	28	28	28	32	31	42	?	340	340	352	352	557
40	3838380	128299	30	30	30	34	34	46	?	369	369	381	381	611
41	4496388	140035	33	33	33	36	37	50	?	398	399	412	412	667
42	5245786	152466	35	35	35	39	40	54	?	430	430	445	445	727
43	6096454	165612	37	37	37	42	43	58	?	463	463	479	479	790
44	7059052	179493	40	40	40	44	47	62	?	498	498	515	515	857
45	8145060	194129	42	42	42	47	51	66	?	535	535	552	552	927
46	9366819	209540	45	45	45	51	55	71	?	573	573	592	592	1001
47	10737573	225746	48	48	48	54	59	76	?	613	613	633	633	1080
48	12271512	242767	51	51	51	57	63	81	?	655	655	677	677	1162
49	13983816	260623	54	54	54	61	67	87	?	700	700	722	722	1248
50	15890700	279334	57	57	57	64	72	92	?	746	746	770	770	1338

Table 4.2: Bounds on the complete lottery number  $L_1(m, 6; 3)$  for all  $6 \leq m \leq 50$  in a comparative fashion. The table shows the order of the lottery graph,  $\binom{m}{6}$ ; the degree of regularity of the lottery graph,  $r$  given in (2.1); the lower bound,  $\underline{L}_1$ , from classical graph domination theory according to (4.1) (also given in (4.4) and (4.23)); the combinatorial lower bound,  $\underline{L}_2$ , (4.23); the Sterboul lower bound,  $\underline{L}_3$ , (4.22); the Sterboul lower bound,  $\underline{L}_4$ , (4.21); the generalised Schönheim lower bound,  $\underline{L}_5$ , (4.19); the Turán lower bound,  $\underline{L}_6$  (4.20); the known complete lottery number  $L_1(m, 6; 3)$ ,  $L$ ; the Clark, et al. upper bound,  $\overline{L}_1$ , (4.15); the Arnautov upper bound,  $\overline{L}_2$ , (4.8); the Caro & Roditty upper bound,  $\overline{L}_3$ , (4.11); the asymptotic upper bound,  $\overline{L}_4$ , (4.9); and the upper bound,  $\overline{L}_5$ , according to the generalised domination result (4.2), truncated after the second term, (4.17). The complete lottery number  $L_1(18, 6; 3) = 7$  was previously unknown and is determined in [39]. A question mark (?) denotes that the complete lottery number  $L_1(m, 6; 3)$  is not known.

Appendix B is given as reference and contains similar tables to that of Tables 4.1 & 4.2, but also contains the best known bounds obtained from Internet repository tables [19, 44, 133, 237].

It is evident that the analytic bounds in Table 4.2 are typically weak or far apart. This prompts an inquiry into using an algorithmic approach for better (or even optimal) lottery sets for  $\langle m, n; k \rangle$ . This will be the point of departure for Chapter 5.

## Chapter 5

# Algorithmic bounds

“The difference between art and science is that science is what we understand well enough to explain to a computer.”

*Donald E Knuth* (1938–) [206]

“I think the problem is not to find the best or most efficient method to proceed to a discovery, but to find any method at all.”

*Richard P Feynman* (1918–1988) [189]

In this chapter, a number of algorithms<sup>1</sup> are presented to obtain bounds on both  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  for small values of  $1 \leq k \leq n \leq m$ ,  $0 < \psi \leq 1$  and  $2 \leq \ell \leq L_1(m, n; k)$ . Upper bounds and lower bounds were obtained on  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  respectively. A total of seven algorithms were implemented, and this chapter has been organised into sections accordingly, presenting the algorithms in order of increasing quality of performance. In each section a description of the relevant algorithm is given, together with a discussion of the possible advantages and/or disadvantages of the algorithm with respect to the others, the performance of the algorithm and complexity considerations of the specific algorithmic approach. Each section is concluded with an illustrative example of the algorithm implementation. The lottery  $\langle 20, 4; 3 \rangle$ , for which the lottery number is known to satisfy  $111 \leq L_1(20, 4; 3) \leq 148$  [133], was chosen as example to demonstrate the working and efficiency of the algorithms throughout this chapter. The chapter is concluded with a section containing a more complete comparison of algorithm results for small lotteries, followed by a section considering specifically the class of lotteries  $\langle m, 5; 2 \rangle$  for  $5 \leq m \leq 30$ , and finally the realistic and popular lottery  $\langle 49, 6; 3 \rangle$ .

The main approaches behind the different algorithms presented in this chapter may be classified as follows:

- Repetitive *generation* of elements in a playing set  $\mathcal{L}^{(i)}$  of cardinality  $\ell$  performed independently from preceding playing sets  $\mathcal{L}^{(i-1)}$  (Classical random algorithm in §5.1 and Distributed random algorithm in §5.2).
- Iterative *construction* of a playing set  $\mathcal{L}^{(i)}$  of cardinality  $i$  from a previous playing set  $\mathcal{L}^{(i-1)}$  of cardinality  $i - 1$  (Minimal overlapping algorithm in §5.3 and Neighbourhood removal algorithm in §5.4).
- Successive *modification* of a playing set  $\mathcal{L}$  of cardinality  $\ell$  (Tabu search algorithm in §5.5 and (classical/intelligent) Genetic algorithms in §5.6).

The numerical work presented in this chapter was in certain instances computationally rather taxing, and hence all implemented algorithms were run on a Linux-based stand-alone personal computer or MOSIX cluster. MOSIX is a set of enhancements of the Linux kernel for supporting cluster computing.

---

<sup>1</sup>Each of the implemented algorithms is included for further reference (either in C++ or Microsoft Visual Basic for Applications, a macro/application extension to Microsoft Excel, code) in Appendix A.

The core of MOSIX consists of adaptive (on-line) resource sharing (load-balancing, memory ushering and I/O optimisation) algorithms and a pre-emptive process migration mechanism that allows a cluster of workstations and servers (nodes) to work cooperatively as if part of a single system. The cluster may include a large number (up to 65 535) of nodes, which can be any combination of workstations, servers, single-processors, *etc.* of any speed. MOSIX is implemented within the Linux kernel and it maintains 100% compatibility with standard Linux, hence there is no need to modify any user-level files, programs or binaries. When activated, normal Linux processes may be allocated and migrate automatically and transparently to other nodes within the cluster in order to achieve a better resource usage of the cluster. As the demands for resources change across the cluster, processes may migrate again, as many times as necessary, to continue optimising the overall resource usage. User manual-control of process migration is also available [12]. More specifically, the MOSIX cluster used for the numerical work in this chapter consisted of more than 50 CPUs that included Celeron 900 MHz and Intel Celeron 1 GHz processors (each with 256 MB memory) as well as a single Intel Pentium IV 1.5 GHz master node processor (with 512 MB memory), while the collection of stand-alone computers consisted of a Pentium III 800 MHz dual processor and a Pentium III 600 MHz dual processor (each with 512 MB memory), a Celeron 700 MHz (with 256 MB memory) and 59 Pentium 133 MHz processors (each with 64 MB memory).

In order to derive bounds on the (worst case) complexity measure of the algorithms in this chapter, an upper bound on the complexity of determining the resource utilisation  $\Psi_\ell(m, n; k)$  of a given playing set of cardinality  $\ell$  needs to be investigated. To determine whether two  $n$ -sets share a common  $k$ -subset,  $n$  comparisons are necessary. In a possible worst case scenario, it might be necessary to examine all  $\ell$  elements of a playing set in search of a  $k$ -subset intersection. We therefore deduce that the number of comparisons performed in order to determine the resource utilisation  $\Psi_\ell(m, n; k)$  of a playing set of cardinality  $\ell$  in a possible worst case is given by

$$\tau_{\Psi_\ell(m, n; k)} = \mathcal{O}\left(n\ell \binom{m}{n}\right). \quad (5.1)$$

## 5.1 Classical random algorithm

### Description

The Classical random algorithm (presented in pseudocode as Algorithm 2 and in C++ code in Appendix A.1) consists of repeatedly generating a random playing set of (user) prespecified cardinality  $\ell$ . In the event that a cardinality  $\ell$  playing set yields a resource utilisation of 1, an upper bound on  $L_1(m, n; k)$  is obtained (the upper bound being  $L_1(m, n; k) \leq \ell$ ), otherwise a lower bound on  $\Psi_\ell(m, n; k)$  (or equivalently, the upper bound on  $L_\psi(m, n; k) \leq \ell$ ) is established.

### Performance and complexity

The implementation of Algorithm 2 is intuitive and elementary. With the reasonable assumption that the generation of a random playing set of cardinality  $\ell$  has a worst case complexity  $\mathcal{O}(\ell)$ , the worst case complexity of Algorithm 2 is  $\mathcal{O}(\ell \tau_{\Psi_\ell(m, n; k)} \text{Iterations})$ , with  $\tau_{\Psi_\ell(m, n; k)}$  given in (5.1).

**Example 5.1** Consider the lottery  $\langle 20, 4; 3 \rangle$ . Algorithm 2 was initialised with `Iterations = 1000`, generating playing sets of cardinality  $\ell = 100$ . The best resource utilisation was achieved at iteration 173, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{3752}{4845} \approx 77.4247\%$ . The graph in Figure 5.1 presents the (percentage) resource utilisation of the random playing sets (of cardinality 100) generated by Algorithm 2 at every iteration. No convergence is observed, as expected. ■

The main advantage of this implementation is the fast unconditional playing set generation, although from Figure 5.1 the independence of successive iterations is evident, yielding no visible improvement trend during application of the algorithm. Furthermore, this method does not exploit the lottery graph structure, and is therefore not expected to yield very good results in general.

**Algorithm 2** Classical random algorithm

**Input:** The lottery parameters  $\langle m, n; k \rangle$ , a playing set cardinality  $\ell$ , the number of iterations Iterations.

**Output:**  $L_{\text{ruBest}}(m, n; k) \leq \ell$  or  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ .

```

1: index  $\leftarrow$  1, ruBest  $\leftarrow$  0.
2: Generate random playing set  $\mathcal{L}$  of cardinality  $\ell$ .
3: Determine the resource utilisation ru of  $\mathcal{L}$ .
4: if (ru = 1) then
5:   print  $L_1(m, n; k) \leq \ell$ . stop.
6: end if
7: if (ru > ruBest) then
8:   ruBest  $\leftarrow$  ru.
9: end if
10: index  $\leftarrow$  index + 1.
11: if (index < Iterations) then
12:   goto (2).
13: else [index  $\geq$  Iterations]
14:   print  $L_{\text{ruBest}}(m, n; k) \leq \ell$ .
15:   print  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ . stop.
16: end if

```

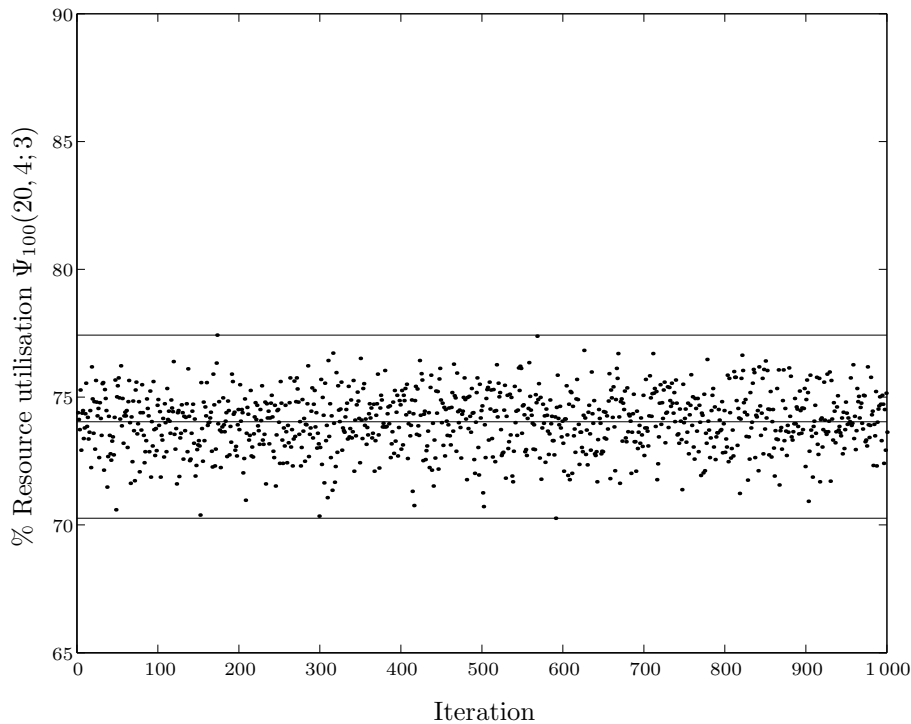


Figure 5.1: Lower bound on the resource utilisation for  $\langle 20, 4; 3 \rangle$  using a hundred 4-sets from  $\mathcal{U}_{20}$  at every iteration of Algorithm 2. The upper, middle and lower solid (—) lines respectively represent the maximum, mean and minimum resource utilisation over all iterations.

## 5.2 Distributed random algorithm

### Description

As a possible improvement to the Classical random algorithm in §5.1, the generation of playing sets (Step (2) in Algorithm 2) may be constrained. Let  $f_{\mathcal{L}}(i)$  denote the frequency of occurrence of the element



$1 \leq i \leq m$  in a playing set  $\mathcal{L}$ . Then the constraint of generating a playing set  $\mathcal{L}$  such that  $f_{\mathcal{L}}(i)$  may only take the value  $\lfloor \frac{m}{n} \rfloor - 1$ ,  $\lfloor \frac{m}{n} \rfloor$  or  $\lfloor \frac{m}{n} \rfloor + 1$  (depending on the divisibility properties of  $m$  by  $n$ ) may be added to Algorithm 2, thus forcing the elements  $1, \dots, m$  to be approximately equally utilised in  $\mathcal{L}$ . This constraint on the generation of  $n$ -sets seems intuitively verifiable, due to the fact that “spreading out”  $n$ -sets (and hence distributing the elements of  $\mathcal{U}_m$  across the  $n$ -sets of  $\mathcal{L}$ ) in a playing set  $\mathcal{L}$  is intuitively expected to yield a greater resource utilisation. For completeness, this alternative algorithm is presented in pseudocode as Algorithm 3 and in C++ code in Appendix A.2.

---

**Algorithm 3** Distributed random algorithm
 

---

**Input:** The lottery parameters  $\langle m, n; k \rangle$ , a playing set cardinality  $\ell$ , the number of iterations **Iterations**.

**Output:**  $L_{\text{ruBest}}(m, n; k) \leq \ell$  or  $\Psi_{\ell}(m, n; k) \geq \text{ruBest}$ .

```

1: index  $\leftarrow$  1, ruBest  $\leftarrow$  0.
2: Generate random playing set  $\mathcal{L}$  of cardinality  $\ell$  such that  $f_{\mathcal{L}}(i) \in \{\lfloor \frac{m}{n} \rfloor - 1, \lfloor \frac{m}{n} \rfloor, \lfloor \frac{m}{n} \rfloor + 1\}$  for all  $1 \leq i \leq m$ .
3: Determine the resource utilisation ru of  $\mathcal{L}$ .
4: if (ru = 1) then
5:   print  $L_1(m, n; k) \leq \ell$ . stop.
6: end if
7: if (ru > ruBest) then
8:   ruBest  $\leftarrow$  ru.
9: end if
10: index  $\leftarrow$  index + 1.
11: if (index < Iterations) then
12:   goto (2)
13: else [index  $\geq$  Iterations]
14:   print  $L_{\text{ruBest}}(m, n; k) \leq \ell$ .
15:   print  $\Psi_{\ell}(m, n; k) \geq \text{ruBest}$ . stop.
16: end if

```

---

## Performance and complexity

The Distributed random algorithm performs marginally better than the Classical random procedure presented in Algorithm 2, in the sense that it yields a higher mean resource utilisation at a fractional increase in the worst order complexity measure. Assuming the choice of an element for inclusion in any  $n$ -set is of worst case complexity  $\mathcal{O}(m)$  (i.e., examining all elements  $1 \leq i \leq m$  in search of a minimum  $f_{\mathcal{L}}(i)$ ), the worst case complexity measure of Algorithm 3 is  $\mathcal{O}(m \ell \tau_{\Psi_{\ell}(m, n; k)} \text{Iterations})$ , with  $\tau_{\Psi_{\ell}(m, n; k)}$  given in (5.1).

**Example 5.2 (continuation of Example 5.1)** *Reconsider the lottery  $\langle 20, 4; 3 \rangle$  (in Example 5.1). Algorithm 3 was initialised with **Iterations** = 1000, generating playing sets of cardinality  $\ell = 100$ . The best resource utilisation was achieved at iteration 715, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{3805}{4845} \approx 78.5346\%$  in comparison to the bound  $\Psi_{100}(20, 4; 3) \geq 77.4247\%$  yielded by Algorithm 2. The graph in Figure 5.2 represents the (percentage) resource utilisation of the conditionally generated random playing sets (of cardinality 100) at every iteration of Algorithm 3. Once again no convergence is observed, as expected. ■*

Although conditions are imposed on the random generation of playing sets in Algorithm 3, there is still no improvement trend in successively generated playing sets during the algorithm implementation.

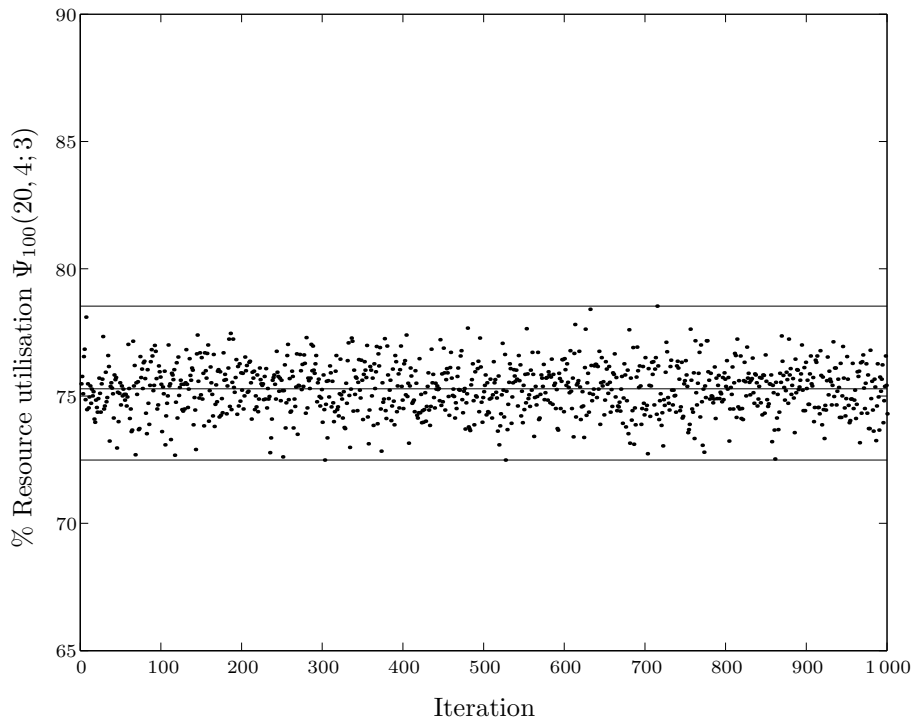


Figure 5.2: Lower bound on the resource utilisation for  $\langle 20, 4; 3 \rangle$  using a hundred 4-sets from  $\mathcal{U}_{20}$  at every iteration of Algorithm 3. The upper, middle and lower solid (—) lines respectively represent the maximum, mean and minimum resource utilisation over all iterations.

## 5.3 Minimal overlapping algorithm

### Description

According to Lemma 3.1, the more elements any two labels [ $n$ -sets] in  $G\langle m, n; k \rangle [\Phi(\mathcal{U}_m, n)]$  have in common, the more vertices of  $G\langle m, n; k \rangle$  [ $n$ -sets] are collectively dominated by the two vertices in question (and hence the resource utilisation is weakened). It is therefore reasonable to assume that, in order to maximise the resource utilised by a given set  $\mathcal{L}$ , the elements ( $n$ -sets) in  $\mathcal{L}$  should be chosen so that the intersection between any two  $n$ -sets in  $\mathcal{L}$  is minimised. Equivalently stated, this implies that the vertices in  $\mathcal{L}$  are forcibly spread out over the topology of  $G\langle m, n; k \rangle$ . This is the principle behind the following heuristic algorithm (presented in pseudocode as Algorithm 4 and given in C++ code in Appendix A.3).

The main components of Algorithm 4 are the maintenance of the frequencies  $f_{\mathcal{L}^{(i)}}(j)$  ( $j = 1, \dots, m$ ) and the overlapping function  $o_{\mathcal{L}^{(i)}}(v_{l_1}, v_{l_2})$  (where  $v_{l_1}, v_{l_2} \in \mathcal{L}^{(i)}$ ). Here the function  $o_{\mathcal{L}^{(i)}}(v_{l_1}, v_{l_2})$  returns the number of common elements shared between the sets  $v_{l_1}, v_{l_2} \in \mathcal{L}^{(i)}$  (i.e.,  $o_{\mathcal{L}^{(i)}}(v_{l_1}, v_{l_2}) = |v_{l_1} \cap v_{l_2}|$  where  $v_{l_1}, v_{l_2} \in \mathcal{L}^{(i)}$ ).

A playing set  $\mathcal{L}^{(i)}$  (of cardinality  $\ell$ ) is constructed from a playing set  $\mathcal{L}^{(i-1)}$  (of cardinality  $\ell - 1$ ) by adding an  $n$ -set  $v_i$  to  $\mathcal{L}^{(i-1)}$  such that  $v_i$  is “as mutually disjoint as possible” from any  $v \in \mathcal{L}^{(i-1)}$  (i.e., for any  $u, v \in \mathcal{L}^{(i)}$ ,  $|u \cap v|$  is minimised). In the event that  $v_i$  shares one (or more) element(s) with any  $v \in \mathcal{L}^{(i-1)}$ , any single element overlapping is preferred above a double element overlapping, which in turn is preferred above a triple element overlapping, *etc.* The described procedure (intuitively) leads to an approximately even distribution of the frequency occurrences  $f_{\mathcal{L}^{(i)}}(j)$  ( $j = 1, \dots, m$ ).

### Performance and complexity

Consider the following example that builds on the previous examination of the lottery  $\langle 20, 4; 3 \rangle$ .

**Algorithm 4** Minimal overlapping algorithm**Input:** The lottery parameters  $\langle m, n, k \rangle$ , a playing set cardinality  $\ell$ .**Output:**  $L_{\text{ruBest}}(m, n, k) \leq \ell$  or  $\Psi_\ell(m, n, k) \geq \text{ruBest}$ .

---

```

1:  $\text{index} \leftarrow 0, \text{ruBest} \leftarrow 0, \mathcal{L}^{(0)} \leftarrow \{\}, \text{ovrlps} \leftarrow 0.$ 
2:  $v_{\text{index}} \leftarrow \{\}, \text{ovrlps} \leftarrow \min_{1 \leq j < \text{index}} \{o_{\mathcal{L}^{(\text{index})}}(v_j, v_{\text{index}})\}.$ 
3:  $v^* \leftarrow \{\text{first element with (minimum } f_{\mathcal{L}^{(\text{index})}}(i) \ \& \ (o_{\mathcal{L}^{(\text{index})}}(v_j, v_{\text{index}} \cup v^*) \leq \text{ovrlps}) \text{ for all } i =$ 
    $1, \dots, m\}.$ 
4: if ( $v^* = \{\}$ ) then [i.e., no such element exist]
5:    $\text{ovrlps} \leftarrow \text{ovrlps} + 1.$  goto (3).
6: else [ $v^* \neq \{\}$ ]
7:    $v_{\text{index}} \leftarrow \{v_{\text{index}} \cup v^*\}.$ 
8: end if
9: if ( $|v_{\text{index}}| = n$ ) then
10:   $\mathcal{L}^{(\text{index}+1)} \leftarrow \{\mathcal{L}^{(\text{index})} \cup v_{\text{index}}\}, \text{index} \leftarrow \text{index} + 1.$ 
11: end if
12: if ( $\text{index} < \ell$ ) then
13:   goto (2).
14: else [ $\text{index} \geq \ell$ ]
15:   goto (7).
16: end if
17:  $\text{ruBest} \leftarrow \text{RU}(\mathcal{L}^{(\ell)}).$ 
18: if ( $\text{ruBest} = 1$ ) then
19:   print  $L_1(m, n, k) \leq \ell.$  stop.
20: else [ $\text{ruBest} < 1$ ]
21:   print  $L_{\text{ruBest}}(m, n, k) \leq \ell.$ 
22:   print  $\Psi_\ell(m, n, k) \geq \text{ruBest}.$  stop.
23: end if

```

---

**Example 5.3 (continuation of Example 5.2)** Reconsider the lottery  $(20, 4, 3)$  of Example 5.2. Algorithm 4 was initialised to generate a playing set of cardinality  $\ell = 100$ . The iterative resource utilisation  $\Psi_i(20, 4, 3)$  obtained by investigating  $\text{RU}(\mathcal{L}^{(i)})$  for every  $i = 1, \dots, \ell$  is presented in Figure 5.3, culminating in the lower bound  $\Psi_{100}(20, 4, 3) \geq \frac{4024}{4845} \approx 83.0547\%$  in comparison to the lower bounds  $\Psi_{100}(20, 4, 3) \geq 77.4247\%$  and  $\Psi_{100}(20, 4, 3) \geq 78.5346\%$  obtained by Algorithms 2 and 3, respectively. ■

Consider the worst case complexity of adding a single element to an  $n$ -set  $v$  for inclusion in a playing set  $\mathcal{L}^{(i)}$ . The algorithm searches through all frequencies of occurrence  $f_{\mathcal{L}^{(i)}}(j)$  ( $j = 1, \dots, m$ ) evaluating every  $o_{\mathcal{L}^{(i)}}(v, u)$  for all  $u \in \mathcal{L}^{(i-1)}$ . This may be performed with  $m(i-1)$  comparisons. This process is executed for every element in  $v$  (*i.e.*,  $n$  times) and sequentially for every  $v \in \mathcal{L}^{(i)}$  (*i.e.*,  $i$  times). The computation of the resource utilisation is suppressed until the end (for the playing set  $\mathcal{L}^{(\ell)}$ ) of Algorithm 4 (and may therefore be considered constant) and is consequently excluded as a complexity consideration. From these observations it is possible to conclude that the worst case complexity of Algorithm 4 is  $\mathcal{O}(mn\ell^2)$ .

## 5.4 Neighbourhood removal algorithm

### Description

It is obvious that the preceding algorithms do not exploit the rich structure of the lottery graph  $G\langle m, n, k \rangle$ . In contrast, the deterministic Neighbourhood removal algorithm (presented in pseudocode as Algorithm 5 and in C++ code in Appendix A.4) progressively constructs a playing set  $\mathcal{L}^{(i)}$  of cardinality  $i$  from the lottery graph  $G\langle m, n, k \rangle$  by greedily adding a vertex  $v_i$  with largest (closed) neighbourhood set to the existing playing set  $\mathcal{L}^{(i-1)}$  of cardinality  $i-1$  ( $i = 1, \dots, \ell$ ) in a naive attempt at exploiting the lottery graph structure. The lottery graph  $G\langle m, n, k \rangle$  is continually pruned during the algorithm implementation in the sense that the closed neighbourhood of vertex  $v_i$  is removed from the remaining

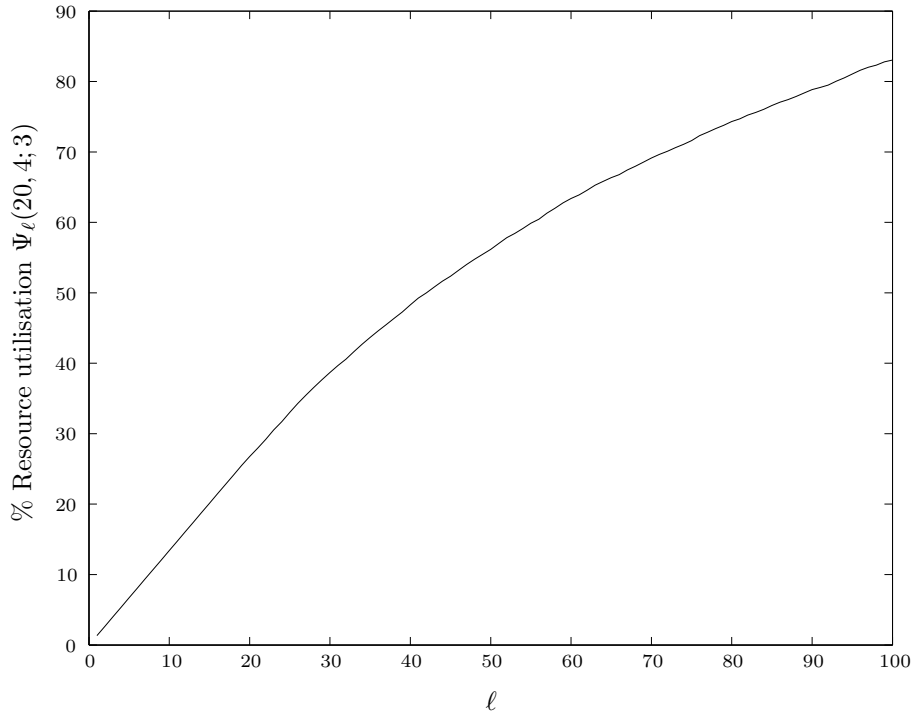


Figure 5.3: Lower bound on the resource utilisation  $\Psi_\ell(20, 4; 3)$  ( $\ell = 1, \dots, 100$ ) at every iteration of Algorithm 4.

lottery subgraph at the  $i$ -th iteration of the algorithm, that is  $V(G\langle m, n; k \rangle) \setminus N[v_i]$  is considered to be the “lottery graph” vertex set during iteration  $i + 1$ . It is clear that the final playing set determined by Algorithm 5 represents a maximal independent set for  $G\langle m, n; k \rangle$  and hence also a minimal dominating set for  $G\langle m, n; k \rangle$ <sup>2</sup>. In Algorithm 5, the function  $\operatorname{argmax}_{u \in V(\mathcal{G})} \deg_{\mathcal{G}}(u)$  returns the first vertex  $u$  having  $\deg_{\mathcal{G}}(u) = \Delta(\mathcal{G})$  for every  $u \in V(\mathcal{G})$ .

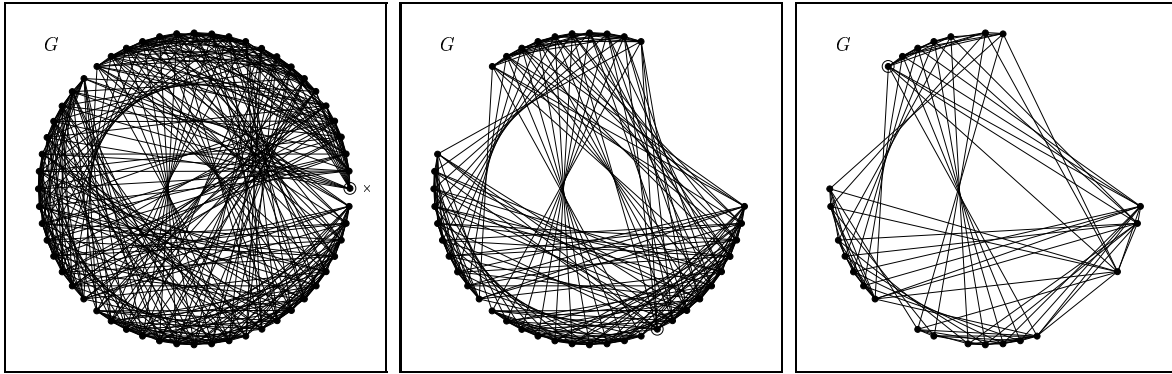
## Performance and complexity

The following small example is presented to illustrate the sequence of steps iterated by Algorithm 5.

**Example 5.4** Consider the lottery graph  $G\langle 8, 3; 2 \rangle$  on  $\binom{8}{3} = 56$  vertices presented in Figure 5.4(a) (starting with the vertex label  $\{1, 2, 3\}$  represented by the symbol “ $\times$ ” on the extreme right and using a counter clockwise lexicographic labelling). The iterative sequence of steps performed by Algorithm 5 on the lottery graph  $G\langle 8, 3; 2 \rangle$  is shown in Figure 5.4. At iteration  $i = 1, \dots, 6$ , the induced subgraph  $G$  (vertex with largest closed neighbourhood is encircled), the constructed playing set  $\mathcal{L}^{(i)}$ , a lower bound on  $\Psi_{|\mathcal{L}^{(i)}|}(8, 3; 2)$  and an upper bound on  $L_\psi(8, 3; 2)$  (for various values of the parameter  $\psi$ ) are presented. The algorithm output yields the upper bound  $L_1(8, 3; 2) \leq 7$ , given by the complete lottery set  $\mathcal{L}^{(7)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\}, \{2, 4, 7\}, \{3, 5, 8\}, \{2, 6, 8\}, \{3, 6, 7\}\}$ . ■

At each iteration of Algorithm 5, a locally optimal decision (of which vertex to include in the playing set) yields a lower bound on  $\Psi_{|\mathcal{L}^{(i)}|}(m, n; k)$ . It is clear that the construction is dependent on the sequence in which vertices (together with their respective neighbourhoods) of  $G$  are removed. The algorithm generally yields better bounds than those obtained by Algorithms 2 and 3, as may be seen in the following example.

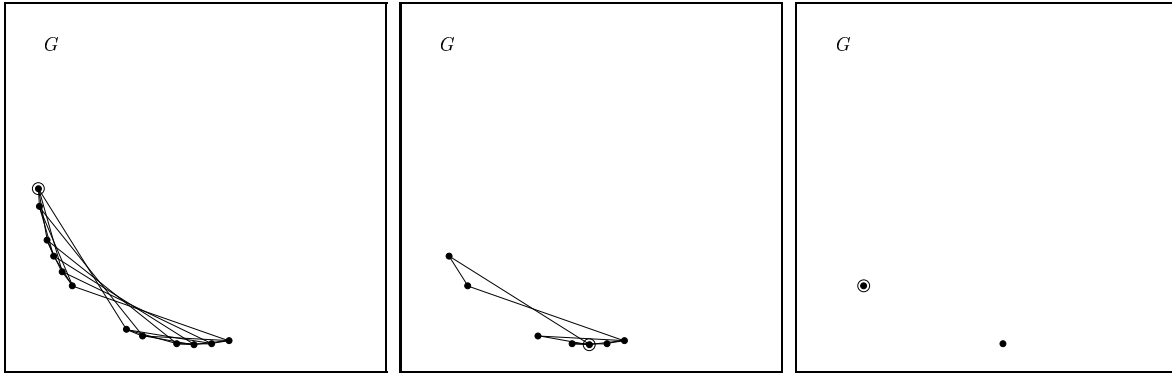
<sup>2</sup>It is known that the maximal independence of a vertex subset implies that the vertex subset is also minimal dominating and *vice versa*. See, for example, [103].



(a)  $\mathcal{L}^{(0)} = \{\};$   
 $\text{ruBest} = 0$

(b)  $\mathcal{L}^{(1)} = \{\{1, 2, 3\}\};$   
 $\Psi_1(8, 3; 2) \geq \text{ruBest} = \frac{16}{56};$   
 $L_\psi(8, 3; 2) = 1 \ (0 < \psi \leq \frac{16}{56})$

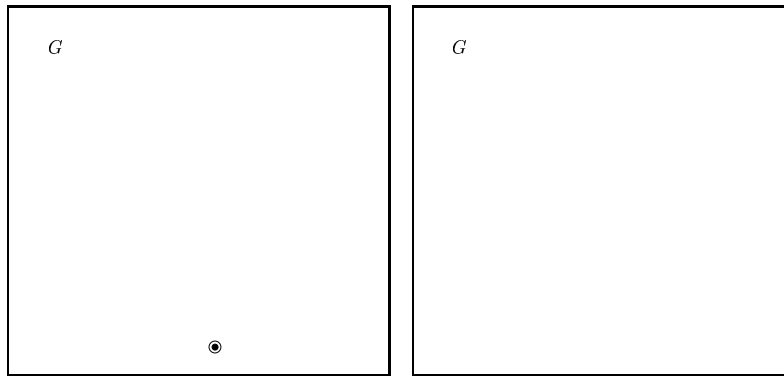
(c)  $\mathcal{L}^{(2)} = \{\{1, 2, 3\}, \{4, 5, 6\}\};$   
 $\Psi_2(8, 3; 2) \geq \text{ruBest} = \frac{32}{56};$   
 $L_\psi(8, 3; 2) \leq 2 \ (\frac{16}{56} < \psi \leq \frac{32}{56})$



(d)  $\mathcal{L}^{(3)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\}\};$   
 $\Psi_3(8, 3; 2) \geq \text{ruBest} = \frac{44}{56};$   
 $L_\psi(8, 3; 2) \leq 3 \ (\frac{32}{56} < \psi \leq \frac{44}{56})$

(e)  $\mathcal{L}^{(4)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\},$   
 $\{2, 4, 7\}\};$   
 $\Psi_4(8, 3; 2) \geq \text{ruBest} = \frac{49}{56};$   
 $L_\psi(8, 3; 2) \leq 4 \ (\frac{44}{56} < \psi \leq \frac{49}{56})$

(f)  $\mathcal{L}^{(5)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\},$   
 $\{2, 4, 7\}, \{3, 5, 8\}\};$   
 $\Psi_5(8, 3; 2) \geq \text{ruBest} = \frac{54}{56};$   
 $L_\psi(8, 3; 2) \leq 5 \ (\frac{49}{56} < \psi \leq \frac{54}{56})$



(g)  $\mathcal{L}^{(6)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\},$   
 $\{2, 4, 7\}, \{3, 5, 8\}, \{2, 6, 8\}\};$   
 $\Psi_6(8, 3; 2) \geq \text{ruBest} = \frac{55}{56};$   
 $L_\psi(8, 3; 2) \leq 6 \ (\frac{54}{56} < \psi \leq \frac{55}{56})$

(h)  $\mathcal{L}^{(7)} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 7, 8\},$   
 $\{2, 4, 7\}, \{3, 5, 8\}, \{2, 6, 8\}, \{3, 6, 7\}\};$   
 $\Psi_7(8, 3; 2) = \text{ruBest} = 1;$   
 $L_\psi(8, 3; 2) \leq 7 \ (\frac{55}{56} < \psi \leq 1)$

Figure 5.4: Step-by-step analysis of Algorithm 5 as applied to the lottery  $\langle 8, 3; 2 \rangle$ .

**Algorithm 5** Neighbourhood removal algorithm**Input:** The lottery graph  $G\langle m, n; k \rangle$ , a playing set cardinality  $\ell$ .**Output:**  $L_{\text{ruBest}}(m, n; k) \leq \ell$  or  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ .

---

```

1: ruBest  $\leftarrow 0$ ,  $\mathcal{L}^{(0)} \leftarrow \{\}$ ,  $G \leftarrow G\langle m, n; k \rangle$ .
2:  $\mathcal{L}^{(|\mathcal{L}|+1)} \leftarrow \mathcal{L}^{(|\mathcal{L}|)} \cup \operatorname{argmax}_{u \in V(G)} \{\deg_G(u)\}$ 
3:  $G \leftarrow$  the subgraph induced by the vertex set  $V(G) \setminus N_G[u]$ .
4: ruBest  $\leftarrow \text{ruBest} + \frac{|N_G[u]|}{\binom{m}{n}}$  [i.e., yielding the lower bound  $\Psi_{|\mathcal{L}^{(|\mathcal{L}|)}|}(m, n; k) \geq \text{ruBest}$ ].
5: if  $( (|\mathcal{L}^{(|\mathcal{L}|)}| < \ell) \text{ and } (V(G) \neq \emptyset) )$  then
6:   goto (2).
7: else  $[ (|\mathcal{L}^{(|\mathcal{L}|)}| = \ell) \text{ or } (V(G) = \emptyset) ]$ 
8:   goto (6).
9: end if
10: if  $(\text{ruBest} = 1)$  then
11:   print  $L_1(m, n; k) \leq \ell$ . stop.
12: else  $[\text{ruBest} < 1]$ 
13:   print  $L_{\text{ruBest}}(m, n; k) \leq \ell$ .
14:   print  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ . stop.
15: end if

```

---

**Example 5.5 (continuation of Example 5.3)** Reconsider the lottery  $\langle 20, 4; 3 \rangle$  from Example 5.3. Using Algorithm 5, Table 5.1 was generated, yielding the improved lower bound  $\Psi_\ell(20, 4; 3) \geq \frac{4293}{4845} \approx 88.6068\%$  in comparison to the lower bounds  $\Psi_{100}(20, 4; 3) \geq 77.4247\%$ ,  $\Psi_{100}(20, 4; 3) \geq 78.5346\%$  and  $\Psi_{100}(20, 4; 3) \geq 83.0547\%$  obtained by Algorithms 2, 3 and 4 respectively. ■

Algorithm 5 takes no global consideration of the order in which vertices are chosen for addition to the playing set  $\mathcal{L}^{(i)}$ . Therefore, one possible disadvantage of this greedy heuristic algorithmic implementation is the fact that the induced subgraph  $G$  (Step (3) in Algorithm 5) may be prone to break up into components<sup>3</sup>. In the specific case of finding an  $L_1(m, n; k)$ -set [a dominating set] for  $\langle m, n; k \rangle$  [ $G\langle m, n; k \rangle$ ], this may unnecessarily force a greater increase in the lottery [dominating] set cardinality<sup>4</sup> towards the end of the implementation.

In order to determine a worst case complexity estimate for Algorithm 5, consider the algorithm after (say)  $i - 1$  iterations. The order of the induced subgraph  $G$  after  $i - 1$  iterations (and therefore the number of vertices to consider for inclusion in  $\mathcal{L}^{(i)}$ ) is given by  $\binom{m}{n} - |\cup_{j=1}^i N_G[v_j]|$ . Each of these possible vertices may have a closed neighbourhood cardinality of at most  $r + 1$  for which the individual resource utilisation must be evaluated, implying that the worst case complexity of Algorithm 5 is given by  $\ell \left( \binom{m}{n} - |\cup_{j=1}^i N_G[v_j]| \right) (r + 1) \tau_{\Psi_\ell(m, n; k)}$  which is  $\mathcal{O} \left( r \ell \binom{m}{n} \tau_{\Psi_\ell(m, n; k)} \right)$ .

## 5.5 Tabu search algorithm

### Methodological background

The method of tabu search, proposed by Glover [89] as an optimisation technique in 1986, originated as a device for implementing the oscillating assignment strategy presented in [90]. In most (combinatorial) optimisation search methods one struggles with an inherent inability to escape from local optima. In this regard, the tabu search methodology may be described as a local search technique guided by the use of adaptive or flexible memory structures [204]. A “memory” of recent decisions forces the search algorithm to explore new areas of the solution space (this is called a *diversification* strategy). Reversals

<sup>3</sup>A probabilistic variation to Algorithm 5 (similar to a proposed algorithm in [202]) would be to remove only a proportion (with some probability, depending on the algorithm progression, for example) of  $N_G[v] \setminus v$  from  $G$ , in an attempt at reducing the formation of components and hence allowing for a possible non-independent dominating set of  $G$  to be found.

<sup>4</sup>To motivate this statement, let  $\mathcal{H}$  be a graph consisting of two vertex-induced components  $\mathcal{H}_1$  and  $\mathcal{H}_2$ . Then it is known that  $\gamma(\mathcal{H}) \leq \gamma(\mathcal{H}_1) + \gamma(\mathcal{H}_2)$ .

$\ell$	Vertex $v_\ell$ added to $\mathcal{L}^{(\ell-1)}$	$ N_G[v_\ell] $	Lower bound on $\Psi_\ell(20, 4; 3)$	$\ell$	Vertex $v_\ell$ added to $\mathcal{L}^{(\ell-1)}$	$ N_G[v_\ell] $	Lower bound on $\Psi_\ell(20, 4; 3)$
1	{1, 2, 3, 4}	65	65/4845	51	{4, 7, 13, 15}	42	2887/4845
2	{1, 5, 6, 7}	65	130/4845	52	{6, 10, 16, 18}	42	2929/4845
3	{1, 8, 9, 10}	65	195/4845	53	{7, 11, 14, 16}	40	2969/4845
4	{1, 11, 12, 13}	65	260/4845	54	{3, 9, 14, 15}	39	3008/4845
5	{1, 14, 15, 16}	65	325/4845	55	{4, 5, 11, 20}	39	3047/4845
6	{1, 17, 18, 19}	65	390/4845	56	{4, 10, 16, 17}	39	3086/4845
7	{2, 5, 8, 11}	65	455/4845	57	{6, 12, 15, 16}	39	3125/4845
8	{2, 6, 9, 12}	65	520/4845	58	{8, 12, 17, 19}	39	3164/4845
9	{2, 7, 10, 13}	65	585/4845	59	{1, 2, 14, 18}	37	3201/4845
10	{2, 14, 17, 20}	65	650/4845	60	{1, 6, 11, 19}	36	3237/4845
11	{3, 5, 9, 13}	65	715/4845	61	{3, 7, 8, 17}	35	3272/4845
12	{3, 6, 8, 14}	65	780/4845	62	{4, 6, 9, 13}	35	3307/4845
13	{3, 7, 11, 15}	65	845/4845	63	{5, 8, 10, 20}	35	3342/4845
14	{3, 10, 12, 16}	65	910/4845	64	{8, 11, 15, 18}	35	3377/4845
15	{4, 5, 10, 14}	65	975/4845	65	{2, 9, 19, 20}	33	3410/4845
16	{4, 6, 11, 16}	65	1040/4845	66	{1, 3, 5, 14}	32	3442/4845
17	{4, 7, 8, 12}	65	1105/4845	67	{2, 4, 14, 16}	32	3474/4845
18	{4, 9, 15, 17}	65	1170/4845	68	{2, 5, 6, 17}	32	3506/4845
19	{4, 13, 18, 20}	65	1235/4845	69	{5, 11, 13, 16}	32	3538/4845
20	{5, 12, 15, 18}	65	1300/4845	70	{1, 3, 13, 16}	30	3568/4845
21	{5, 16, 19, 20}	65	1365/4845	71	{4, 6, 15, 18}	30	3598/4845
22	{6, 10, 15, 19}	65	1430/4845	72	{6, 10, 11, 12}	30	3628/4845
23	{4, 7, 14, 18}	65	1495/4845	73	{7, 15, 17, 20}	30	3658/4845
24	{8, 13, 16, 17}	65	1560/4845	74	{9, 12, 16, 18}	30	3688/4845
25	{9, 10, 11, 20}	61	1621/4845	75	{4, 8, 14, 17}	29	3717/4845
26	{11, 12, 14, 19}	61	1682/4845	76	{1, 10, 13, 15}	28	3745/4845
27	{1, 8, 15, 20}	57	1739/4845	77	{2, 3, 12, 17}	28	3773/4845
28	{2, 3, 16, 18}	57	1796/4845	78	{7, 13, 18, 19}	28	3801/4845
29	{2, 13, 15, 19}	57	1853/4845	79	{5, 6, 9, 14}	27	3828/4845
30	{3, 6, 17, 20}	57	1910/4845	80	{8, 9, 11, 13}	27	3855/4845
31	{7, 9, 16, 19}	57	1967/4845	81	{3, 4, 7, 9}	26	3881/4845
32	{10, 11, 17, 18}	57	2024/4845	82	{2, 11, 15, 20}	25	3906/4845
33	{3, 4, 8, 19}	53	2077/4845	83	{5, 7, 9, 15}	25	3931/4845
34	{5, 7, 12, 17}	53	2130/4845	84	{10, 12, 14, 18}	25	3956/4845
35	{6, 8, 13, 18}	53	2183/4845	85	{11, 16, 17, 19}	25	3981/4845
36	{12, 13, 14, 20}	50	2233/4845	86	{1, 2, 7, 12}	23	4004/4845
37	{1, 7, 10, 20}	47	2280/4845	87	{2, 5, 10, 19}	23	4027/4845
38	{6, 13, 14, 17}	46	2326/4845	88	{4, 5, 8, 16}	23	4050/4845
39	{1, 4, 12, 19}	44	2370/4845	89	{4, 6, 14, 20}	23	4073/4845
40	{1, 9, 11, 17}	44	2414/4845	90	{3, 11, 13, 14}	22	4095/4845
41	{2, 6, 7, 20}	44	2458/4845	91	{9, 10, 12, 17}	22	4117/4845
42	{2, 8, 10, 15}	44	2502/4845	92	{1, 3, 9, 18}	21	4138/4845
43	{7, 8, 14, 19}	44	2546/4845	93	{3, 5, 6, 12}	21	4159/4845
44	{8, 9, 16, 20}	44	2590/4845	94	{1, 5, 8, 13}	20	4179/4845
45	{3, 10, 13, 19}	43	2633/4845	95	{4, 7, 17, 19}	20	4199/4845
46	{3, 12, 18, 20}	43	2676/4845	96	{7, 8, 16, 18}	20	4219/4845
47	{5, 9, 18, 19}	43	2719/4845	97	{14, 18, 19, 20}	20	4239/4845
48	{1, 2, 5, 16}	42	2761/4845	98	{1, 4, 10, 11}	18	4257/4845
49	{2, 4, 11, 18}	42	2803/4845	99	{4, 12, 15, 20}	18	4275/4845
50	{3, 5, 15, 17}	42	2845/4845	100	{5, 13, 17, 18}	18	4293/4845

Table 5.1: Step-by-step analysis of the Algorithm 5 on the lottery  $\langle 20, 4; 3 \rangle$  for determining lower [upper] bounds on  $\Psi_\ell(20, 4; 3)$  ( $\ell = 1, \dots, 100$ ) [ $L_\psi(8, 3; 2)$  ( $0 < \psi \leq 1$ )].

of those recent decisions that yielded locally best candidate solutions are classified as *tabu* (forbidden) and are avoided (or penalised) when making decisions about selecting the next best candidate solution. In contrast to such *recency-based* memory structures, *frequency-based* memory structures store frequency measures relating to the occurrence of attributes in the solutions encountered. To avoid an already traced path of solutions, the procedure records information about moves recently made, employing one or more *tabu lists*. The nature of any tabu list is, of course, problem specific and is usually implemented in the form of a *FIFO (First-In-Last-Out) list*, where new elements added force expulsion of the last elements from the list (therefore maintaining a constant list order). The function of such lists is not to prevent a move from being repeated, but to prevent it from being reversed. Furthermore, the prohibition of

reversal is conditional rather than absolute. In particular, tabu searches proceed with the assumption that poor choices (by accident or design) yield no benefit, except for the purpose of avoiding a path already examined [89].

A key aspect of tabu search is the processing and length of the tabu lists. The length of a tabu list describes how many of the past moves should be remembered and is referred to as a *tabu tenure*. Exact definitions regarding the tabu tenure remain problem specific<sup>5</sup>. Allowing entries in a tabu list to have a pre-emptively large penalty, none of the tabu moves will be considered as long as non-tabu moves exist. The penalties may be allowed to decay as a function of time and/or tabu list length, allowing older tabu moves to be considered in preference to more recent ones (if no move other than a tabu move exists). Although this pre-emptive penalty scheme seems reasonable, cycling solutions may result and this strategy therefore defies the chief purpose of tabu lists. A different resolution, more faithful to the principle of tabu lists, is to incorporate a so-called *aspiration criterion* which overrides the tabu status of a move if the resulting trial solution improves on the best solution obtained thus far (other aspiration criteria also exist [89, 162]). For additional tabu parameters and tabu list management rules, the reader is referred to [89, 190].

Tabu searches typically also employ longer term strategies of *intensification*. Intensification strategies are designed to exploit those good solutions by intensifying the local search around their respective neighbourhoods in the solution space. Additionally, tabu searches usually operate without reference to randomisation and are therefore basically deterministic, although they may also be applied in conjunction with probabilistic methods [162]. Traditionally tabu searches explore the whole neighbourhood of a specific solution in order to select its successor. This may become computationally intensive for large problems (for example, the problem of finding maximum cardinality packing sets for any (large) random graph), although methods exist whereby only a proportion of the candidate solutions is examined. *Candidate list strategies* work on this principle, by considering only a subset of a solution's neighbourhood when looking for a successive solution (see, for example, [199]).

Tabu search algorithms have been employed successfully in the search for dominating sets of near-minimum cardinality in graphs [56], although this optimisation method becomes less attractive as the graph size (and consequently also vertex neighbourhoods) increase. In this context, we are therefore moved to refrain from investigating large (typical) cases of the lottery problem via the tabu search approach, but nevertheless present this optimisation technique as a comparison to the other techniques discussed for small lottery instances.

## Description

The tabu search algorithm implemented for finding lower bounds on  $\Psi_\ell(m, n; k)$  (or alternatively upper bounds on  $L_\psi(m, n; k)$ ) is presented (in pseudocode) as Algorithm 6 and in C++ code in Appendix A.5. Candidate solutions are represented as binary vectors  $\underline{b}$  of length  $\binom{m}{n}$  (the number of vertices in  $G(m, n; k)$ ) and weight<sup>6</sup>  $\ell$ , where the  $i$ -th bit  $b_i$  ( $1 \leq i \leq \binom{m}{n}$ ) takes the value 1 if the  $i$ -th (lexicographically ordered)  $n$ -set is included in the candidate solution and 0 otherwise.

The initial candidate solution is randomly generated, and the following three neighbouring moves were considered for this tabu search implementation:

- **Shift move:** A shift length (between 1 and  $\binom{m}{n} - 1$ ) is randomly chosen (from a uniform distribution), and applied to a candidate solution  $\underline{b}$ , which results in two neighbouring candidates,  $\underline{b}^{(\text{left})}$  and  $\underline{b}^{(\text{right})}$ , for either direction (left and right shift modulo  $\binom{m}{n}$ ). See Figure 5.5(a). The inverse (opposite direction) of the selected move is added to the tabu list.
- **Flip move:** Define  $1 \leq p^{(j)} \leq \binom{m}{n}$  ( $j = 1, \dots, \ell$ ) as the position of the  $j$ -th 1-bit in  $\underline{b}$  (i.e.,  $b_{p^{(j)}} = 1$  for all  $j = 1, \dots, \ell$ ). A bit index  $1 \leq \tilde{i} \leq \ell$ , chosen at random from a uniform distribution, defines the neighbourhood as all the vectors  $\underline{b}^{(i)}$  with  $b_{p^{(\tilde{i})}} = 0$  and  $b_j^{(i)} = 1$  for all  $j = p^{(\tilde{i}-1)} + 1, \dots, p^{(\tilde{i})} - 1, p^{(\tilde{i})} + 1, \dots, p^{(\tilde{i}+1)} - 1$  (i.e., bit  $b_j^{(i)}$  is “flipped” to take the value  $b_j^{(i)} + 1 \pmod{2}$  for all possible

<sup>5</sup>Tabu lists containing only a single element may sometimes yield solutions significantly better than a local optimum (see, for example, a tabu solution strategy to the well-known knapsack problem in [90]).

<sup>6</sup>The weight of a binary vector  $\underline{b}$  is given by the sum of the elements of  $\underline{b}$  (i.e., the number of ones contained in  $\underline{b}$ ).



$j$ ). See Figure 5.5(b). The flip information is recorded in the tabu list and any move reversing a flip in a specific range of the resulting bit flipped position is classified as tabu.

- **Swap move:** Subvectors of a specified size in a candidate solution  $\underline{b}$  are swapped. The neighbouring candidates are defined as all possible combinations of swaps of such subvectors. Incorporating a variation to candidate list strategies, the subvector size was randomly chosen (from a uniform distribution) such that at least 100 and at most 5 000 (in increments of 100) neighbouring candidates may exist. See Figure 5.5(c). The swap information is recorded in the tabu list and any move reversing a swap is classified as tabu.

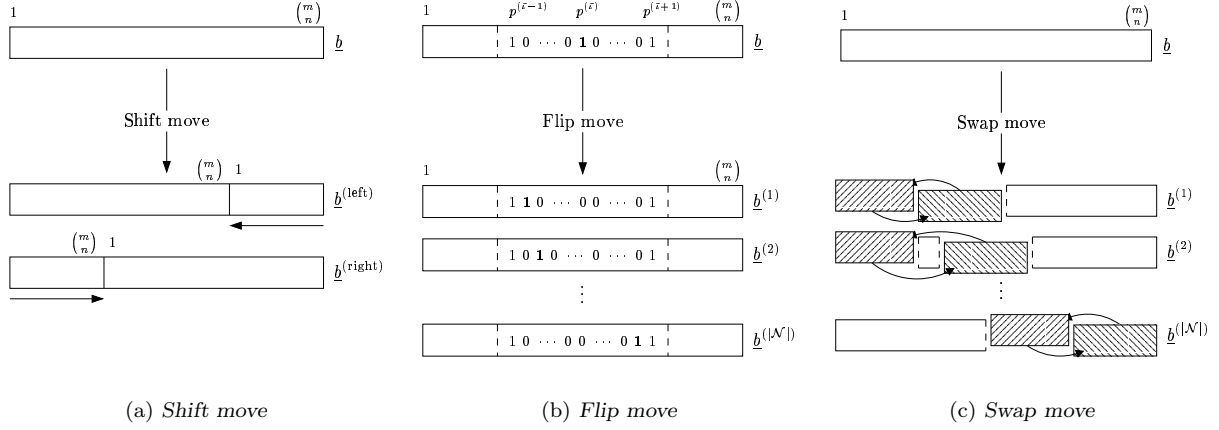


Figure 5.5: An illustration of the three defined moves for the tabu search algorithm.

In Algorithm 6, the specific candidate solution neighbourhood (given any of the defined moves) of a candidate  $\mathcal{L}^*$  is denoted  $\mathcal{N}(\mathcal{L}^*)$ . Also, the function  $\text{RU}(\mathcal{L}_j^{(i)})$  represents the resource utilisation of the  $j$ -th neighbouring candidate solution at iteration  $i$  of the algorithm.

## Performance and complexity

The following example illustrates the consequences and effectiveness of each individual move and investigates some combined attributes of the tabu search algorithm.

**Example 5.6 (continuation of Example 5.5)** *Reconsider the lottery  $\langle 20, 4; 3 \rangle$  of Example 5.5. As aspiration criterion for Algorithm 6, the tabu status of a trial solution was overridden (and hence chosen) if it improved on the best solution obtained up to that particular point in the implementation. Algorithm 6 was further initialised with `Iterations` = 1 000 and `TabuTenure` =  $\lfloor \binom{20}{4} / 20 \rfloor = 242$ , generating playing sets of cardinality  $\ell = 100$  and was executed for the following different cases:*

- Case 1:** *The selection of neighbouring moves was restricted to the Shift move with the best resource utilisation obtained at iteration 98, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{3769}{4845} \approx 77.7915\%$ . See Figure 5.6(a).*
- Case 2:** *The selection of neighbouring moves was restricted to the Flip move with the best resource utilisation obtained at iteration 701, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4158}{4845} \approx 85.8204\%$ . See Figure 5.6(b).*
- Case 3:** *The selection of neighbouring moves was restricted to the Swap move with the best resource utilisation obtained at iteration 96 (with an average of 2 600 possible neighbouring candidate solutions per iteration), yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{3806}{4845} \approx 78.5552\%$ . See Figure 5.6(c).*

**Algorithm 6** Tabu search algorithm

**Input:** The lottery parameters  $\langle m, n, k \rangle$ , a playing set cardinality  $\ell$ , number of iterations **Iterations**, the tabu tenure **TabuTenure**.

**Output:**  $L_{\text{ruBest}}(m, n, k) \leq \ell$  or  $\Psi_\ell(m, n, k) \geq \text{ruBest}$ .

```

1: index  $\leftarrow$  1, ruBest  $\leftarrow$  0,  $\mathcal{L}^{(0)} \leftarrow$  an  $\ell$ -set consisting of random  $n$ -sets in  $\Phi(\mathcal{U}_m, n)$  (using uniform distribution).
2: Select neighbouring move (according to prespecified distribution).
3: for all  $(\mathcal{L}_j^{(\text{index}+1)} \in \mathcal{N}(\mathcal{L}^{(\text{index})}))$  do
4:   Update  $\text{RU}(\mathcal{L}_j^{(\text{index}+1)})$ .
5: end for
6:  $j^* \leftarrow \text{argmax}_{\mathcal{L}_j^{(\text{index}+1)} \in \mathcal{N}(\mathcal{L}^{(\text{index})})} \text{RU}(\mathcal{L}_j^{(\text{index}+1)})$  [i.e., determine neighbouring candidate with largest resource utilisation, taking into consideration aspiration criteria and tabu considerations].
7: if  $(\text{RU}(\mathcal{L}_{j^*}^{(\text{index}+1)}) > \text{ruBest})$  then
8:   ruBest  $\leftarrow$   $\text{RU}(\mathcal{L}_{j^*}^{(\text{index}+1)})$ .
9: end if
10: if (ruBest = 1) then
11:   print  $L_1(m, n, k) \leq \ell$ . stop.
12: end if
13:  $\mathcal{L}^{(\text{index}+1)} \leftarrow \mathcal{L}_{j^*}^{(\text{index}+1)}$  and add tabu move to tabu list.
14: index  $\leftarrow$  index + 1.
15: if (index  $\leq$  Iterations) then
16:   goto (2).
17: else [index > Iterations]
18:   print  $L_{\text{ruBest}}(m, n, k) \leq \ell$ .
19:   print  $\Psi_\ell(m, n, k) \geq \text{ruBest}$ . stop.
20: end if

```

**Case 4:** A uniform distribution between all three neighbouring moves was utilised with the best resource utilisation at iteration 1000, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{3840}{4845} \approx 79.2570\%$ . See Figure 5.6(d).

**Case 5:** A 0.5%, 99% and 0.5% weighted distribution between the respective Shift, Flip and Swap neighbouring moves was utilised with the best resource utilisation obtained at iteration 902, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4169}{4845} \approx 86.0475\%$ . See Figure 5.6(e).

From the results obtained, it is clear that the Flip move (Case 2) represents a good intensification strategy, while the Shift and Swap moves (Cases 1 and 3) may represent possible diversification strategies. This hypothesis is validated by the incorporation of a weighted distribution of both strategies (evident in Case 5) yielding the marginal improvement in resource utilised. Although the best lower bound on the resource utilisation  $\Psi_{100}(20, 4; 3)$  obtained by Algorithm 6 is slightly weaker than that obtained by Algorithm 5, this is not the case in general, as will be argued later. ■

It is assumed that a tabu tenure of  $\lambda$  is used throughout the following complexity considerations. Additionally, the generation of any neighbouring candidate is considered constant due to the operations performed (AND, OR and XOR) being bit-related.

First consider the worst case complexity of determining the next locally optimal non-tabu candidate solution when only the Shift move is utilised as a possible neighbouring move. Trivially only two possible neighbouring candidates exist. For both these candidates, a search through  $\lambda$  tabu moves is performed (in a possible worst case or due to the incorporation of aspiration criteria) and their respective resource utilisation is determined. Therefore, the worst case complexity of the Shift move is  $\mathcal{O}(\lambda \tau_{\Psi_\ell(m, n, k)})$ . Secondly, consider the worst case complexity given that only the Flip move is applied to generate neigh-

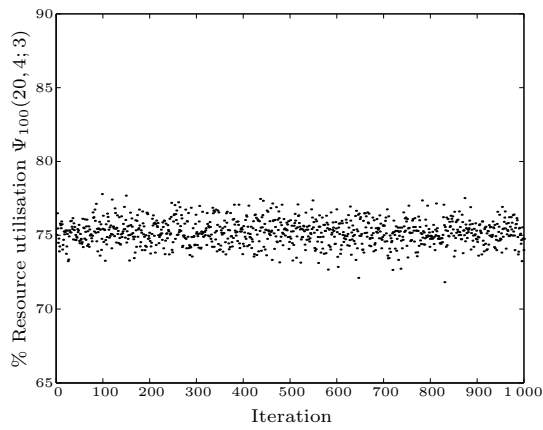
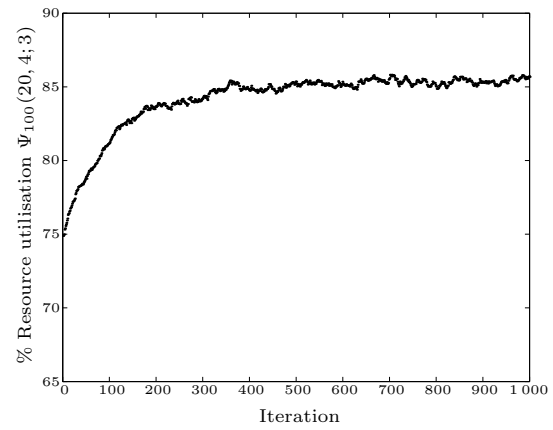
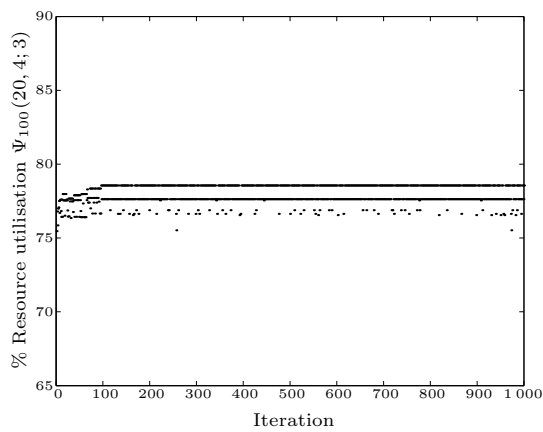
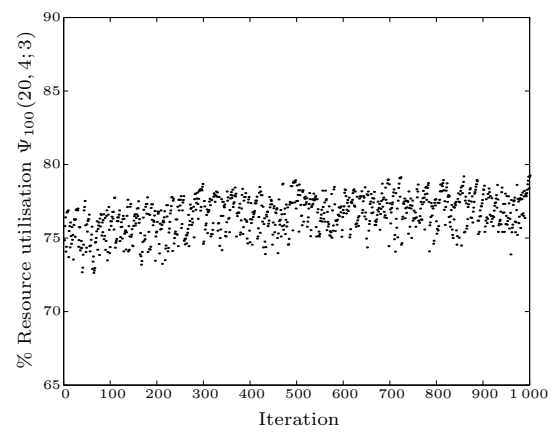
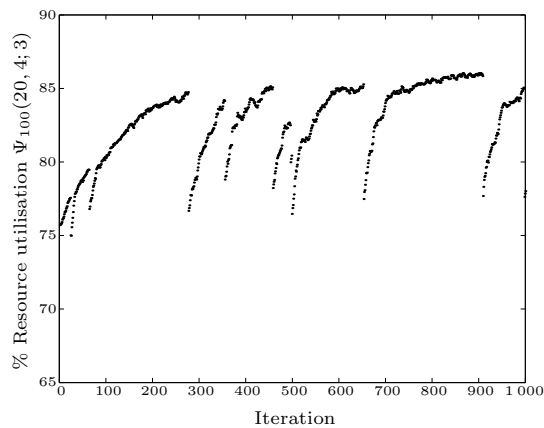
(a) *Shift move*(b) *Flip move*(c) *Swap move*(d) *Evenly distributed combined moves*(e) *Weighted distribution combined moves*

Figure 5.6: Lower bound on the resource utilisation for  $(20, 4; 3)$  at every iteration of Algorithm 6 using the (a) Shift move, (b) Flip move, (c) Swap move and (d) a uniformly distributed and (e) weighted (0.5% Shift, 99% Flip, 0.5% Swap) random combination of the Shift, Flip and Swap moves.

```

Program Evolution
Begin
   $t \leftarrow 0$ 
  Initialise  $P(t)$ 
  Evaluate Fitness( $P(t)$ )
  While (Fitness( $P(t)$ ) < Tolerance) Do
    Begin
       $t \leftarrow t + 1$ 
      Select  $P(t)$  from  $P(t - 1)$ 
      Change  $P(t)$  [Crossover & Mutation]
      Evaluate Fitness( $P(t)$ )
    End
  End

```

Figure 5.7: Typical structure of an evolution program and also the form of a genetic algorithm in pseudocode [161, 162].

bouring candidate solutions. In a (highly unlikely, yet possible) worst case scenario, the binary vector  $\underline{b}$  may be completely unbalanced (*i.e.*, the distribution of 1-bit values are clustered at either or both ends). Then, only one specific set of  $\binom{m}{n} - (\ell + 1)$  neighbouring candidates exists for which their respective resource utilisation requires evaluation (each  $\mathcal{O}(\tau_{\Psi_\ell(m,n;k)})$ ) as well as the search through  $\lambda$  possible tabu moves. It therefore follows that the worst case complexity of the Flip move is  $\mathcal{O}(\lambda \binom{m}{n} \tau_{\Psi_\ell(m,n;k)})$ . Finally, we consider the worst case complexity when only the Swap move is employed as a possible tabu search move for generating neighbouring candidate solutions. The maximum number of neighbouring candidates in a possible worst case is 5000 (therefore constant), and similar complexity arguments to when the Flip move is used, holds for the Swap neighbouring move. We are able to deduce that the worst case complexity of the tabu search Swap move is therefore  $\mathcal{O}(\lambda \tau_{\Psi_\ell(m,n;k)})$ .

## 5.6 Genetic algorithm

### Methodological background

Charles R Darwin (1809–1882), known as the father of evolution, published an article in 1859, entitled *On the Origin of Species* [114], in which he described his observations of nature during an expedition visiting an island [226]. Many of his successors attached the phrase “survival of the fittest” to his theory, which combined the process of natural selection<sup>7</sup> with that of evolution.

Based on Darwin’s evolution theory, genetic algorithms (or evolution programs) may be described as learning methods simulating biological evolution, where individuals of a species procreate in order to produce better offspring. The underlying terminology used to describe the components of a genetic algorithm, was accordingly derived from genetics. For example, individual solutions are sometimes referred to as **chromosomes**, consisting of **genes** from a certain **genepool**. A selection of chromosomes form part of a **generation** (or **population**). This general notion of evolution is captured in pseudocode in Figure 5.7, where the parameter  $t$  represents time and  $P(t)$  represents a population of chromosomes at time  $t$ .

Similar to natural processes, a measure of fitness is used to distinguish between good and bad solution candidates when a genetic algorithmic approach is taken to solve a combinatorial optimisation problem approximately. This (problem specific) fitness measure incorporates a mechanism for defining a ranking between any two individuals with respect to the optimisation problem objective. Candidate selection for

<sup>7</sup>The process of natural selection describes the way in which any population of individuals (or candidates) adapts to its surrounding environment. The characteristics of the fitter individuals are necessarily carried over to successive generations. As the environment progressively changes, fitter offspring are produced from fitter individuals, hence the phrase “survival of the fittest.”

transition operations between successive generations of approximate solutions to the optimisation problem is usually performed stochastically relative to candidate fitness (*i.e.*, fitter candidates are paired for transition operations). This choice of pairing fitter solution candidates for certain transition operations serves as a filtering process in which a successive generation of solution candidates consists of fitter candidates<sup>8</sup>. Two basic genetic transition operators, propagation (also called **crossover**) and **mutation**, represent the process of evolution. The crossover operator yields, as output, new solution candidates (called children) inheriting properties of candidates taking part in the process (called parents). Children typically replace parent candidates in order to maintain a constant population cardinality at every iteration of the genetic algorithm. The mutation operator, on the other hand, is specifically incorporated to perturb a solution generation away from local optima (in the solution space) when an essentially stagnant population of candidate solutions is encountered. Mutation therefore typically involves the exploration of different (possibly new) areas of the solution space. It should be evident that mutations might also lead to less fit candidate solutions (as with the case of the crossover operator), although such weakening mutations are usually corrected by similar mutation and/or crossovers with fitter candidates during successive generations. Both definitions of the genetic operators as well as the fitness measure are problem specific and therefore vary from application to application. The importance of suitably defined crossover and mutation operators should be noted, as these may well distinguish between efficient and inefficient genetic algorithms.

Traditionally, genetic algorithms are used in multi-dimensional optimisation problems with large solution spaces in which the following two (possibly contradicting) goals are present: (i) an effective search through the *whole* solution space (or at least a representative part thereof) and (ii) the progressive improvement of *specific* good solutions<sup>9</sup>. Efficient genetic algorithms maintain a good balance between scouting (of the solution space) and local improvement (of successive specific solutions).

Two variations of a genetic algorithm were implemented in this dissertation, the main difference being in the way the crossover procedure between chromosomes was performed. The first implementation (classical genetic algorithm, in §5.6.1) utilises no intelligent method of exchanging genes between parent/candidate solutions. In contrast, the second implementation (intelligent genetic algorithm, in §5.6.2) gives preference to fitter offspring by incorporating an evaluation procedure to decide which offspring will be produced during propagation.

### 5.6.1 Classical genetic algorithm

#### Description

Genetic algorithms may be employed in search of (upper and lower) bounds on  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  (respectively). Candidates in a population are represented by playing sets of cardinality  $\ell$  and therefore consist of subsets from  $\Phi(\mathcal{U}_m, n)$ . The chromosome population is initialised with a collection of  $\ell$  randomly selected  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ .

The number of vertices in the lottery graph  $G\langle m, n; k \rangle$  dominated by a playing set lends itself as an intuitive and realistic fitness measure. Chromosomes are paired (for the genetic crossover procedure) at random (using a population fitness distribution) relative to their individual fitness (*i.e.*, the fitter [weaker] candidates in a population possess the propensity to pair with other fit [less fit] candidates). The classical genetic crossover procedure is defined as a single exchange of *any* individual gene between two paired candidate solutions (say Parent A and Parent B). More specifically, *any* gene from Parent A [B] is exchanged with *any* gene from Parent B [A], generating a pool of 2 candidate offspring (say Child A and Child B). Child A [B] necessarily replaces Parent A [B] in the genetic algorithm crossover procedure. The genetic mutation procedure alters a random proportion of genes in a random proportion of solution candidates (using a uniform distribution). An illustrative example of the crossover [mutation] operation performed on chromosomes for  $\langle 7, 3; 2 \rangle$  is shown in Figure 5.8(a) [(b)].

This process is iterated for a (user) prespecified number of generations or until a certain fitness tolerance

<sup>8</sup>Due to the stochastic nature of the selection procedure and criteria, it may well occur (although the chances are small relative to fitter candidates) that less fit offspring result from paired transition candidates.

<sup>9</sup>Similar to the *intensification* and *diversification* strategies incorporated in tabu searches.

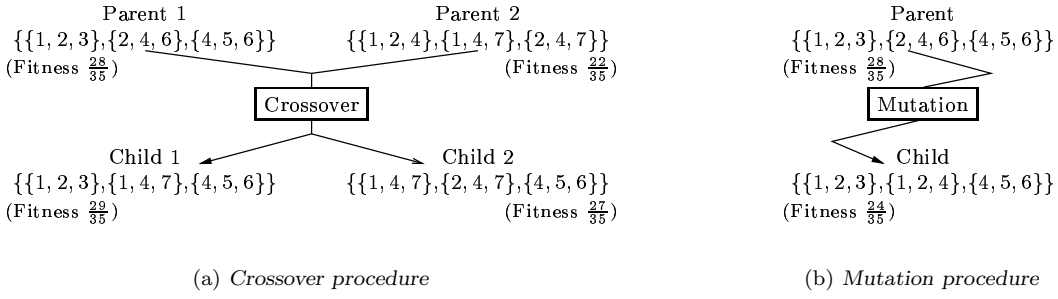


Figure 5.8: Illustration of (a) the crossover procedure performed on two parent chromosomes yielding two (fitter) children (Child 1 [2] is obtained from Parent 1 [2] by the exchange of gene 2 [3] from Parent 2 [1] with gene 2 [1] from Parent 1 [2]) and (b) the mutation procedure on a single chromosome yielding a (weaker) child (the Child is obtained from the Parent by replacing gene 2 with the random gene  $\{1, 2, 4\}$  for  $\langle 7, 3; 2 \rangle$ ).

is obtained (for this specific implementation, the tolerance may be given by some minimally acceptable average population fitness or if one candidate represents a  $L_1(m, n; k)$ -set for  $\langle m, n; k \rangle$ ).

## Performance and complexity

Consider the complexity of performing the classical crossover procedure between any two individual candidates of a population  $\mathcal{M}$ . A single gene exchange is considered in the generation of child candidates, which may be performed in  $|\mathcal{M}|$  operations. For each of the 2 future offspring, the resource utilisation is calculated, leading to a complexity of  $\mathcal{O}(\text{Generations}|\mathcal{M}|\tau_{\Psi_\ell(m,n;k)})$ . Consider, on the other hand, the complexity of performing the mutation procedure on a generation of candidate solutions. A random selection of  $\text{gMutate} \times \ell$  genes in  $\text{cMutate} \times |\mathcal{M}|$  individuals is mutated (replaced by random  $n$ -sets). The choice of candidates and relevant genes may each be considered of constant complexity. This leads to the conclusion that the mutation procedure is  $\mathcal{O}(\ell \text{gMutate} \times |\mathcal{M}|\text{cMutate})$  per generation. With both procedures being performed independently (and hence being of complexity  $\mathcal{O}(\text{Generations} (|\mathcal{M}|\tau_{\Psi_\ell(m,n;k)} + \ell \text{gMutate}|\mathcal{M}|\text{cMutate}))$ ), we are able to deduce that the worst case complexity of the Classical genetic algorithm is  $\mathcal{O}(\text{Generations}|\mathcal{M}|\tau_{\Psi_\ell(m,n;k)})$ .

The implemented classical genetic algorithm was found to exhibit far slower convergence than that of the intelligent genetic algorithm — possibly due to the inferior change in resource utilisation incurred by not exchanging specifically bad genes for better ones that would improve the fitness of the chromosome the most. Therefore, the discussion involving genetic algorithms in this section will henceforth only focus on the intelligent genetic algorithm. However, the C++ code for the classical genetic algorithm is presented in Appendix A.6.

### 5.6.2 Intelligent genetic algorithm

#### Description

As stated, the only difference between the Intelligent and Classical genetic algorithm approach, is in the way the crossover procedure is performed. For the intelligent genetic crossover procedure, a single exchange of *all* individual genes between two paired candidate solutions (say Parent A and Parent B), is considered. More specifically, *all* genes from Parent A [B] are exchanged with *all* genes from Parent B [A], generating a pool of  $\ell^2$  candidate offspring. These  $\ell^2$  offspring are then evaluated (according to the fitness of the resulting chromosome) with the choice of the fittest offspring (Child A [B]) replacing the parent (Parent A [B]).

**Algorithm 7** (Classical/intelligent) Genetic algorithm

**Input:** The lottery parameters  $\langle m, n; k \rangle$ , a playing set cardinality  $\ell$ , a candidate population  $\mathcal{M}$ , the percentage of chromosomes to mutate per generation `cMutate`, the percentage of genes to mutate per chromosome `gMutate`, number of generations `Generations`.

**Output:**  $L_{\text{ruBest}}(m, n; k) \leq \ell$  or  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ .

```

1: index  $\leftarrow$  1, ruBest  $\leftarrow$  0.
2: for all  $j = 1, \dots, |\mathcal{M}|$  do
3:    $\mathcal{L}_j^{(0)} \leftarrow \ell$ -set consisting of random  $n$ -sets in  $\Phi(\mathcal{U}_m, n)$  (using uniform distribution).
4: end for
5: for all  $j = 1, \dots, |\mathcal{M}|$  do
6:   Update  $\text{RU}(\mathcal{L}_j^{(\text{index})})$ .
7: end for
8: if  $(\text{RU}(\mathcal{L}_j^{(\text{index})}) > \text{ruBest})$  then
9:    $\text{index}^* \leftarrow \text{argmax}_{j \in \{1, \dots, |\mathcal{M}|\}} \text{RU}(\mathcal{L}_j^{(\text{index})})$ ,  $\text{ruBest} \leftarrow \text{RU}(\mathcal{L}_j^{(\text{index}^*)})$ .
10: end if
11: if ( $\text{ruBest} = 1$ ) then
12:   print  $L_1(m, n; k) \leq \ell$ . stop.
13: end if
14: Perform transition procedures: (Classical/intelligent) Crossover & Mutation (according to the parameters gMutate and cMutate).
15: index  $\leftarrow$  index + 1.
16: if (index < Generations) then
17:   goto (2).
18: else [index = Generations]
19:   print  $L_{\text{ruBest}}(m, n; k) \leq \ell$ .
20:   print  $\Psi_\ell(m, n; k) \geq \text{ruBest}$ . stop.
21: end if

```

Algorithm 7 presents the description of the (classical/intelligent) genetic algorithm in pseudocode, while the corresponding C++ code is given in Appendix A.7. In Algorithm 7, chromosome  $i$  of a population  $\mathcal{M}$  and its resource utilisation is represented by  $\mathcal{L}^{(i)}$  and  $\text{RU}(\mathcal{L}^{(i)})$ , respectively. The mutation procedure proportions (as mentioned in the description of the Classical genetic algorithm in §5.6.1) are given respectively by the parameters `gMutate` and `cMutate` in Algorithm 7.

### Performance and complexity

Consider the complexity of performing the intelligent crossover procedure between any two individual candidates of a population  $\mathcal{M}$ . The number of operations required to perform every possible gene exchange for every solution candidate is  $\ell^2|\mathcal{M}|$ . For each of the possible future offspring, the resource utilisation is calculated in order to extract those children that are fittest, leading to a complexity  $\mathcal{O}(\text{Generations } \ell^2|\mathcal{M}|\tau_{\Psi_\ell(m, n; k)})$ . Because there is no change in the way mutation of solution candidates is concerned, we deduce that the worst case complexity of the Intelligent genetic algorithm is  $\mathcal{O}(\text{Generations } \ell^2|\mathcal{M}|\tau_{\Psi_\ell(m, n; k)})$ .

One of the (most probable) disadvantages of the intelligent genetic algorithm implementation in determining lower bounds on the resource utilisation  $\Psi_\ell(m, n; k)$ , as opposed to the tabu search optimisation heuristic of Algorithm 6, is the crossover procedure that makes an enormous contribution to the computational complexity. Tabu search implementations for finding dominating sets in graphs are known to become less attractive as the order of the graphs increase [56]. This constraint may be even worse in the case of Algorithm 7.

In the following example, the impact of individual parameter variations is considered. This sheds light on the effectiveness of genetic algorithms in the context of maximising resource utilisation or finding incomplete lottery sets.

**Example 5.7 (continuation of Example 5.6)** Reconsider the lottery  $\langle 20, 4; 3 \rangle$  of Example 5.6. Three independent parameter variations were considered (for every case the parameter **Generations** was initialised to be 200):

**Case 1:** Algorithm 7 was initialised with a random population consisting of  $|\mathcal{M}| = 20$  candidate solutions, each representing a playing set  $\mathcal{L}_j^{(0)}$  ( $j = 1, \dots, 20$ ) of  $\ell = 100$  genes (4-sets). In every generation, **cMutate** = 5% of the population candidates are subjected to mutation. The parameter **gMutate** was initialised to take the values (i) 1%, (ii) 2%, (iii) 5%, (iv) 10% and (v) 25% respectively. The maximum population fitness (and hence the best resource utilisation) per generation is presented in Figure 5.9(a). The best resource utilisation was obtained with a gene mutation percentage of **gMutate** = 2%, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4250}{4845} \approx 87.7193\%$ .

**Case 2:** The parameter **cMutate** was initialised to take the values (i) 5%, (ii) 10%, (iii) 25% and (iv) 50% respectively. Algorithm 7 was further randomly initialised with a population consisting of  $|\mathcal{M}| = 20$  candidate solutions, each representing a playing set  $\mathcal{L}_j^{(0)}$  ( $j = 1, \dots, 20$ ) of  $\ell = 100$  genes (4-sets). Every candidate subjected to mutation, had **gMutate** = 5% of its respective genes mutated. The maximum population fitness (and hence the best resource utilisation) per generation is presented in Figure 5.9(b). The best resource utilisation was obtained with a candidate mutation percentage of **cMutate** = 10%, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4261}{4845} \approx 87.9463\%$ .

**Case 3:** The number of candidate solutions  $|\mathcal{M}|$  in a population was varied to take the values (i) 6, (ii) 10 and (iii) 20. A single chromosome (**cMutate** =  $\frac{1}{|\mathcal{M}|}$ ) of the  $|\mathcal{M}|$  candidates in the population, were subjected to a mutation of **gMutate** = 5% genes. The maximum population fitness (and hence the best resource utilisation) per generation is presented in Figure 5.9(c). The best resource utilisation was obtained with a candidate population of  $|\mathcal{M}| = 20$ , yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4260}{4845} \approx 87.9257\%$ .

Using these results, the following additional case is considered.

**Case 4:** Algorithm 7 was initialised with **gMutate** = 2%, **cMutate** = 10% in a population of  $|\mathcal{M}| = 20$  candidate solutions, thus utilising the best empirically obtained parameter values from Cases 1–3. Figure 5.9(d) represents the minimum, mean and maximum generation fitness (or equivalently the resource utilisation  $\Psi_{100}(20, 4; 3)$ ) as a function of the population generation. The best resource utilisation was obtained in generation 99, yielding the lower bound  $\Psi_{100}(20, 4; 3) \geq \frac{4277}{4845} \approx 88.2766\%$ .

Although the best lower bound on the resource utilisation  $\Psi_{100}(20, 4; 3)$  obtained by Algorithm 7 is slightly weaker than that obtained by Algorithm 5, this is not the case in general as will be argued later. ■

## 5.7 Comparison of algorithms for small lotteries

Consider the small lotteries  $\langle m, n; k \rangle$  where  $1 \leq k < n < m \leq 10$  for which the complete lottery numbers  $L_1(m, n; k)$  are given in Table 3.1. Algorithms 2–7 were applied to determine lower bounds on  $\Psi_\ell(m, n; k)$  in these cases for  $2 \leq \ell \leq L_1(m, n; k)$ , as presented in Table 5.2. Algorithms 2–4 and 6 [7] were each initialised to perform 1 000 iterations [generations], retaining the best resource utilisation obtained (no additional initialisation is required for Algorithm 5). In all cases Algorithms 6 and 7 perform at least as well as the other four algorithms, often yielding strictly better results than these four alternatives. And of Algorithms 6 and 7, only Algorithm 7 was able to achieve a value of  $\Psi_\ell(m, n; k) = 1$  for  $\ell = L_1(m, n; k)$  in *all cases* considered.



		$\ell$	
		<b>2</b>	<b>3</b>
ALGORITHM	2	$\frac{14}{15}$	1
	3	$\frac{14}{15}$	1
	4	$\frac{14}{15}$	1
	5	$\frac{14}{15}$	1
	6	$\frac{14}{15}$	1
	7	$\frac{14}{15}$	1

- (a) Lower bounds on  $\Psi_\ell(6, 2; 1) = \Psi_\ell(6, 4; 3)$   
 $(2 \leq \ell \leq 3 = L_1(6, 2; 1) = L_1(6, 4; 3))$

		$\ell$	
		<b>2</b>	<b>3</b>
ALGORITHM	2	$\frac{18}{21}$	1
	3	$\frac{18}{21}$	1
	4	$\frac{18}{21}$	1
	5	$\frac{18}{21}$	1
	6	$\frac{18}{21}$	1
	7	$\frac{18}{21}$	1

- (b) Lower bounds on  $\Psi_\ell(7, 2; 1) = \Psi_\ell(7, 5; 4)$   
 $(2 \leq \ell \leq 3 = L_1(7, 2; 1) = L_1(7, 5; 4))$

		$\ell$		
		<b>2</b>	<b>3</b>	<b>4</b>
ALGORITHM	2	$\frac{26}{35}$	$\frac{32}{35}$	1
	3	$\frac{26}{35}$	$\frac{32}{35}$	1
	4	$\frac{26}{35}$	$\frac{31}{35}$	$\frac{34}{35}$
	5	$\frac{26}{35}$	$\frac{31}{35}$	$\frac{34}{35}$
	6	$\frac{26}{35}$	$\frac{32}{35}$	1
	7	$\frac{26}{35}$	$\frac{32}{35}$	1

- (c) Lower bounds on  $\Psi_\ell(7, 3; 2) = \Psi_\ell(7, 4; 3)$   
 $(2 \leq \ell \leq 4 = L_1(7, 3; 2) = L_1(7, 4; 3))$

		$\ell$		
		<b>2</b>	<b>3</b>	<b>4</b>
ALGORITHM	2	$\frac{22}{28}$	$\frac{27}{28}$	1
	3	$\frac{22}{28}$	$\frac{27}{28}$	1
	4	$\frac{22}{28}$	$\frac{27}{28}$	1
	5	$\frac{22}{28}$	$\frac{27}{28}$	1
	6	$\frac{22}{28}$	$\frac{27}{28}$	1
	7	$\frac{22}{28}$	$\frac{27}{28}$	1

- (d) Lower bounds on  $\Psi_\ell(8, 2; 1) = \Psi_\ell(8, 6; 5)$   
 $(2 \leq \ell \leq 4 = L_1(8, 2; 1) = L_1(8, 6; 5))$

		$\ell$			
		<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
ALGORITHM	2	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{50}{56}$	$\frac{54}{56}$
	3	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{50}{56}$	$\frac{54}{56}$
	4	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{49}{56}$	$\frac{54}{56}$
	5	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{49}{56}$	$\frac{54}{56}$
	6	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{50}{56}$	1
	7	$\frac{32}{56}$	$\frac{44}{56}$	$\frac{50}{56}$	1

- (e) Lower bounds on  $\Psi_\ell(8, 3; 2) = \Psi_\ell(8, 5; 4)$   
 $(2 \leq \ell \leq 5 = L_1(8, 3; 2) = L_1(8, 5; 4))$

		$\ell$				
		<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
ALGORITHM	2	$\frac{34}{70}$	$\frac{47}{70}$	$\frac{57}{70}$	$\frac{63}{70}$	$\frac{68}{70}$
	3	$\frac{34}{70}$	$\frac{47}{70}$	$\frac{60}{70}$	$\frac{64}{70}$	$\frac{67}{70}$
	4	$\frac{34}{70}$	$\frac{43}{70}$	$\frac{52}{70}$	$\frac{57}{70}$	$\frac{62}{70}$
	5	$\frac{34}{70}$	$\frac{47}{70}$	$\frac{60}{70}$	$\frac{63}{70}$	$\frac{66}{70}$
	6	$\frac{34}{70}$	$\frac{47}{70}$	$\frac{60}{70}$	$\frac{64}{70}$	1
	7	$\frac{34}{70}$	$\frac{47}{70}$	$\frac{60}{70}$	$\frac{64}{70}$	1

- (f) Lower bounds on  $\Psi_\ell(8, 4; 3)$   
 $(2 \leq \ell \leq 6 = L_1(8, 4; 3))$

Table 5.2: A comparison between the performances of Algorithms 2–7 in the determination of lower bounds on the resource utilisation number  $\Psi_\ell(m, n; k)$  ( $2 \leq \ell \leq L_1(m, n; k)$ ) for the small lotteries  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$ . Boldfaced values of  $\ell$  represent exact values of  $L_1(m, n; k)$  [19, 44, 133].

		$\ell$		
		2	3	4
ALGORITHM	2	$\frac{26}{36}$	$\frac{33}{36}$	1
	3	$\frac{26}{36}$	$\frac{33}{36}$	1
	4	$\frac{26}{36}$	$\frac{33}{36}$	1
	5	$\frac{26}{36}$	$\frac{33}{36}$	1
	6	$\frac{26}{36}$	$\frac{33}{36}$	1
	7	$\frac{26}{36}$	$\frac{33}{36}$	1

(g) Lower bounds  $\Psi_\ell(9, 2; 1) = \Psi_\ell(9, 7; 6)$   
 $(2 \leq \ell \leq 4 = L_1(9, 2; 1) = L_1(9, 7; 6))$

		$\ell$	
		2	3
ALGORITHM	2	$\frac{83}{84}$	1
	3	$\frac{83}{84}$	1
	4	$\frac{83}{84}$	1
	5	$\frac{83}{84}$	1
	6	$\frac{83}{84}$	1
	7	$\frac{83}{84}$	1

(h) Lower bounds on  $\Psi_\ell(9, 3; 1) = \Psi_\ell(9, 6; 4)$   
 $(2 \leq \ell \leq 3 = L_1(9, 3; 1) = L_1(9, 6; 4))$

		$\ell$						
		2	3	4	5	6	7	
ALGORITHM	2	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{65}{84}$	$\frac{73}{84}$	$\frac{79}{84}$	$\frac{81}{84}$	
	3	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{65}{84}$	$\frac{73}{84}$	$\frac{78}{84}$	$\frac{81}{84}$	
	4	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{64}{84}$	$\frac{71}{84}$	$\frac{78}{84}$	$\frac{79}{84}$	
	5	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{64}{84}$	$\frac{71}{84}$	$\frac{78}{84}$	$\frac{79}{84}$	
	6	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{65}{84}$	$\frac{75}{84}$	$\frac{80}{84}$	1	
	7	$\frac{38}{84}$	$\frac{57}{84}$	$\frac{65}{84}$	$\frac{75}{84}$	$\frac{80}{84}$	1	

(i) Lower bounds on  $\Psi_\ell(9, 3; 2) = \Psi_\ell(9, 6; 5)$   
 $(2 \leq \ell \leq 7 = L_1(9, 3; 2) = L_1(9, 6; 5))$

		$\ell$								
		2	3	4	5	6	7	8	9	
ALGORITHM	2	$\frac{42}{126}$	$\frac{63}{126}$	$\frac{80}{126}$	$\frac{93}{126}$	$\frac{100}{126}$	$\frac{109}{126}$	$\frac{114}{126}$	$\frac{119}{126}$	
	3	$\frac{42}{126}$	$\frac{63}{126}$	$\frac{80}{126}$	$\frac{93}{126}$	$\frac{101}{126}$	$\frac{106}{126}$	$\frac{112}{126}$	$\frac{117}{126}$	
	4	$\frac{42}{126}$	$\frac{48}{126}$	$\frac{65}{126}$	$\frac{82}{126}$	$\frac{90}{126}$	$\frac{100}{126}$	$\frac{108}{126}$	$\frac{112}{126}$	
	5	$\frac{42}{126}$	$\frac{63}{126}$	$\frac{80}{126}$	$\frac{93}{126}$	$\frac{103}{126}$	$\frac{110}{126}$	$\frac{116}{126}$	$\frac{120}{126}$	
	6	$\frac{42}{126}$	$\frac{63}{126}$	$\frac{80}{126}$	$\frac{93}{126}$	$\frac{104}{126}$	$\frac{115}{126}$	$\frac{121}{126}$	$\frac{125}{126}$	
	7	$\frac{42}{126}$	$\frac{63}{126}$	$\frac{80}{126}$	$\frac{93}{126}$	$\frac{104}{126}$	$\frac{115}{126}$	$\frac{121}{126}$	1	

(j) Lower bounds on  $\Psi_\ell(9, 4; 3) = \Psi_\ell(9, 5; 4)$   
 $(2 \leq \ell \leq 9 = L_1(9, 4; 3) = L_1(9, 5; 4))$

		$\ell$			
		2	3	4	5
ALGORITHM	2	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1
	3	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1
	4	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1
	5	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1
	6	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1
	7	$\frac{30}{45}$	$\frac{39}{45}$	$\frac{44}{45}$	1

(k) Lower bounds on  $\Psi_\ell(10, 2; 1) = \Psi_\ell(10, 8; 7)$   
 $(2 \leq \ell \leq 5 = L_1(10, 2; 1) = L_1(10, 8; 7))$

		$\ell$	
		2	3
ALGORITHM	2	$\frac{116}{120}$	1
	3	$\frac{116}{120}$	1
	4	$\frac{116}{120}$	1
	5	$\frac{116}{120}$	1
	6	$\frac{116}{120}$	1
	7	$\frac{116}{120}$	1

(l) Lower bounds on  $\Psi_\ell(10, 3; 1) = \Psi_\ell(10, 7; 5)$   
 $(2 \leq \ell \leq 3 = L_1(10, 3; 1) = L_1(10, 7; 5))$

Table 5.2 (continued): A comparison between the performances of Algorithms 2–7 in the determination of lower bounds on the resource utilisation number  $\Psi_\ell(m, n; k)$  ( $2 \leq \ell \leq L(m, n; k)$ ) for the small lotteries  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$ . Boldfaced values of  $\ell$  represent exact values of  $L_1(m, n; k)$  [19, 44, 133].

		$\ell$						
		2	3	4	5	6	7	8
ALGORITHM	2	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{91}{120}$	$\frac{100}{120}$	$\frac{106}{120}$	$\frac{112}{120}$
	3	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{92}{120}$	$\frac{100}{120}$	$\frac{107}{120}$	$\frac{111}{120}$
	4	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{90}{120}$	$\frac{100}{120}$	$\frac{108}{120}$	$\frac{111}{120}$
	5	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{92}{120}$	$\frac{102}{120}$	$\frac{107}{120}$	$\frac{112}{120}$
	6	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{92}{120}$	$\frac{102}{120}$	$\frac{110}{120}$	1
	7	$\frac{44}{120}$	$\frac{66}{120}$	$\frac{80}{120}$	$\frac{92}{120}$	$\frac{102}{120}$	$\frac{110}{120}$	1

(m) Lower bounds on  $\Psi_\ell(10, 3; 2) = \Psi_\ell(10, 7; 6)$   
 $(2 \leq \ell \leq 8 = L_1(10, 3; 2) = L_1(10, 7; 6))$

		$\ell$	
		2	3
ALGORITHM	2	$\frac{194}{210}$	1
	3	$\frac{194}{210}$	1
	4	$\frac{194}{210}$	1
	5	$\frac{194}{210}$	1
	6	$\frac{194}{210}$	1
	7	$\frac{194}{210}$	1

(n) Lower bounds on  $\Psi_\ell(10, 4; 2) = \Psi_\ell(10, 6; 4)$   
 $(2 \leq \ell \leq 3 = L_1(10, 4; 2) = L_1(10, 6; 4))$

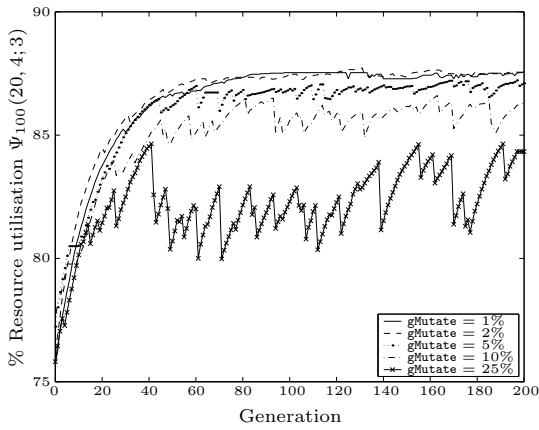
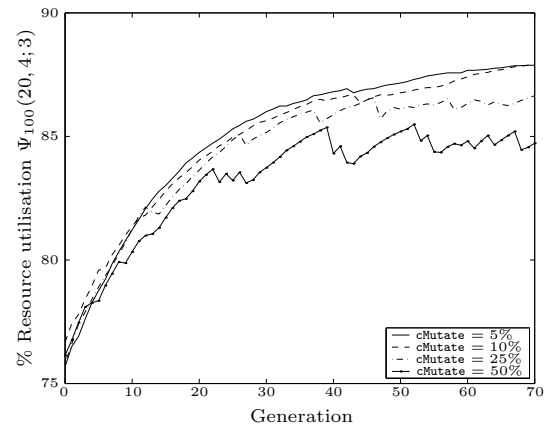
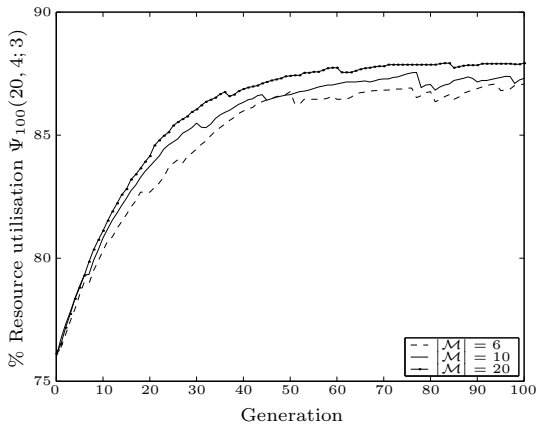
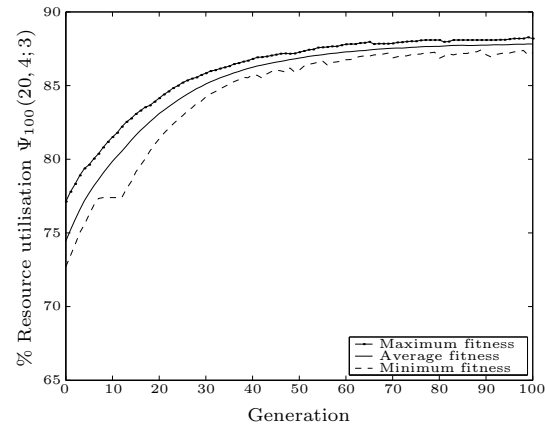
		$\ell$													
		2	3	4	5	6	7	8	9	10	11	12	13	14	
ALGORITHM	2	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{100}{210}$	$\frac{117}{210}$	$\frac{131}{210}$	$\frac{145}{210}$	$\frac{156}{210}$	$\frac{169}{210}$	$\frac{178}{210}$	$\frac{184}{210}$	$\frac{189}{210}$	$\frac{191}{210}$	$\frac{200}{210}$	
	3	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{100}{210}$	$\frac{121}{210}$	$\frac{134}{210}$	$\frac{148}{210}$	$\frac{161}{210}$	$\frac{169}{210}$	$\frac{174}{210}$	$\frac{182}{210}$	$\frac{188}{210}$	$\frac{193}{210}$	$\frac{201}{210}$	
	4	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{96}{210}$	$\frac{117}{210}$	$\frac{127}{210}$	$\frac{141}{210}$	$\frac{155}{210}$	$\frac{167}{210}$	$\frac{177}{210}$	$\frac{185}{210}$	$\frac{193}{210}$	$\frac{198}{210}$	$\frac{201}{210}$	
	5	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{96}{210}$	$\frac{117}{210}$	$\frac{138}{210}$	$\frac{151}{210}$	$\frac{164}{210}$	$\frac{175}{210}$	$\frac{185}{210}$	$\frac{193}{210}$	$\frac{198}{210}$	$\frac{202}{210}$	$\frac{204}{210}$	
	6	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{100}{210}$	$\frac{125}{210}$	$\frac{139}{210}$	$\frac{156}{210}$	$\frac{170}{210}$	$\frac{180}{210}$	$\frac{191}{210}$	$\frac{198}{210}$	$\frac{201}{210}$	$\frac{206}{210}$	$\frac{208}{210}$	
	7	$\frac{50}{210}$	$\frac{75}{210}$	$\frac{100}{210}$	$\frac{125}{210}$	$\frac{139}{210}$	$\frac{156}{210}$	$\frac{170}{210}$	$\frac{181}{210}$	$\frac{193}{210}$	$\frac{199}{210}$	$\frac{204}{210}$	$\frac{208}{210}$	1	

(o) Lower bounds on  $\Psi_\ell(10, 4; 3) = \Psi_\ell(10, 6; 5)$  ( $2 \leq \ell \leq 14 = L_1(10, 4; 3) = L_1(10, 6; 5)$ )

		$\ell$													
		2	3	4	5	6	7	8	9	10	11	12	13	14	
ALGORITHM	2	$\frac{52}{252}$	$\frac{78}{252}$	$\frac{104}{252}$	$\frac{126}{252}$	$\frac{144}{252}$	$\frac{162}{252}$	$\frac{176}{252}$	$\frac{185}{252}$	$\frac{198}{252}$	$\frac{203}{252}$	$\frac{212}{252}$	$\frac{221}{252}$	$\frac{230}{252}$	
	3	$\frac{52}{252}$	$\frac{78}{252}$	$\frac{104}{252}$	$\frac{126}{252}$	$\frac{144}{252}$	$\frac{159}{252}$	$\frac{170}{252}$	$\frac{184}{252}$	$\frac{197}{252}$	$\frac{203}{252}$	$\frac{210}{252}$	$\frac{220}{252}$	$\frac{232}{252}$	
	4	$\frac{52}{252}$	$\frac{74}{252}$	$\frac{96}{252}$	$\frac{115}{252}$	$\frac{134}{252}$	$\frac{150}{252}$	$\frac{166}{252}$	$\frac{181}{252}$	$\frac{196}{252}$	$\frac{208}{252}$	$\frac{224}{252}$	$\frac{232}{252}$	$\frac{236}{252}$	
	5	$\frac{52}{252}$	$\frac{78}{252}$	$\frac{104}{252}$	$\frac{130}{252}$	$\frac{156}{252}$	$\frac{170}{252}$	$\frac{184}{252}$	$\frac{198}{252}$	$\frac{209}{252}$	$\frac{220}{252}$	$\frac{229}{252}$	$\frac{235}{252}$	$\frac{240}{252}$	
	6	$\frac{52}{252}$	$\frac{78}{252}$	$\frac{104}{252}$	$\frac{130}{252}$	$\frac{156}{252}$	$\frac{170}{252}$	$\frac{188}{252}$	$\frac{203}{252}$	$\frac{214}{252}$	$\frac{224}{252}$	$\frac{229}{252}$	$\frac{236}{252}$	$\frac{241}{252}$	
	7	$\frac{52}{252}$	$\frac{78}{252}$	$\frac{104}{252}$	$\frac{130}{252}$	$\frac{156}{252}$	$\frac{170}{252}$	$\frac{192}{252}$	$\frac{203}{252}$	$\frac{220}{252}$	$\frac{226}{252}$	$\frac{234}{252}$	$\frac{239}{252}$	1	

(p) Lower bounds on  $\Psi_\ell(10, 5; 4)$  ( $2 \leq \ell \leq 14 = L(10, 5; 4)$ )

Table 5.2 (continued): A comparison between the performances of Algorithms 2–7 in the determination of lower bounds on the resource utilisation number  $\Psi_\ell(m, n; k)$  ( $2 \leq \ell \leq L(m, n; k)$ ) for the small lotteries  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$ . Boldfaced values of  $\ell$  represent exact values of  $L_1(m, n; k)$  [19, 44, 133].

(a) Gene mutation parameter ( $gMutate$ )(b) Candidate mutation parameter ( $cMutate$ )(c) Population parameter ( $|\mathcal{M}|$ )

(d) Best parameters utilised

Figure 5.9: Influence on the resource utilisation  $\Psi_{100}(20, 4; 3)$  due to a change in the genetic algorithm parameters: (a) gene mutation parameter  $gMutate$ , (b) candidate mutation parameter  $cMutate$ , (c) population parameter  $|\mathcal{M}|$  and (d) utilising the best genetic parameters ( $gMutate = 2\%$ ,  $cMutate = 10\%$  and  $|\mathcal{M}| = 20$ ) for the lottery  $\langle 20, 4; 3 \rangle$ .

## 5.8 Comparison of algorithms for larger lotteries

This section is devoted to considering the performance of Algorithms 2–7 on more realistic values of the lottery parameters  $\langle m, n; k \rangle$ . A whole lottery class  $\langle m, 5; 2 \rangle$  (for  $5 \leq m \leq 30$ ) is considered in §5.8.1, while in §5.8.2 the focus is on the specific lottery  $\langle 49, 6; 3 \rangle$ , which seems to be the single most popular lottery scheme world wide (according to Table 1.1).

### 5.8.1 The lottery class $\langle m, 5; 2 \rangle$

Algorithms 2–6 [7] were implemented with the intention of finding lottery sets for  $\langle m, 5; 2 \rangle$ , where  $5 \leq m \leq 25$ , in order to give the reader a better impression of the relative performance of the algorithms described in §§5.1–5.5 [§5.6]. The cardinality of a smallest lottery set cardinality determined with 1 000 iterations [generations] by all of the specific algorithms (hence upper bounds on  $L(m, 5; 2)$ ) are summarised in Table 5.3.

From Table 5.3, it is evident that Algorithm 7 outperforms Algorithms 2–6. For completeness, the best

		$m$															
Bounds on $L_1(m, 5; 2)$		5	6	7	8	9	10	11	12	13	14	15	16	17			
Classical random algorithm (2)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>	4	5			
Distributed random algorithm (3)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	4	4			
Minimal overlapping algorithm (4)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>	4			
Neighbourhood removal algorithm (5)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>	4			
Tabu search algorithm (6)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>	4			
(Intelligent) Genetic algorithm (7)		<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>	4			

		$m$															
Bounds on $L_1(m, 5; 2)$		18	19	20	21	22	23	24	25	26	27	28	29	30			
Classical random algorithm (2)		6	8	10	13	13	16	19	20	25	29	30	33	36			
Distributed random algorithm (3)		6	8	8	11	13	16	17	20	22	26	28	32	35			
Minimal overlapping algorithm (4)		<b>4</b>	<b>4</b>	<b>4</b>	9	9	12	14	16	18	21	22	26	28			
Neighbourhood removal algorithm (5)		<b>4</b>	<b>4</b>	<b>4</b>	9	9	11	15	18	18	22	22	26	27			
Tabu search algorithm (6)		<b>4</b>	<b>4</b>	<b>4</b>	<b>6</b>	<b>6</b>	8	11	15	16	21	21	25	26			
(Intelligent) Genetic algorithm (7)		<b>4</b>	<b>4</b>	<b>4</b>	<b>6</b>	<b>6</b>	9	11	12	15	16	18	20	22			

Table 5.3: A comparison between upper bounds on  $L_1(m, 5; 2)$ , where  $5 \leq m \leq 30$ , as determined by Algorithms 2–7. Boldfaced entries represent exact values of  $L_1(m, 5; 2)$ .

known lower and upper bounds on  $L_1(m, n; k)$  for  $\langle m, n; k \rangle$  where  $2 \leq k < n \leq 6$  and  $3 \leq m \leq 50$  (available from lottery and covering number repository cites on the Internet [19, 44, 133, 237]) are presented in Tables B.2 and B.3.

### 5.8.2 The lottery $\langle 49, 6; 3 \rangle$

In this subsection the most widely used lottery, namely the lottery  $\langle 49, 6; 3 \rangle$  (see Table 1.1), is considered. The lottery graph  $G\langle 49, 6; 3 \rangle$  on 13 983 816 vertices is 260 623-regular and has a density of  $\frac{6\,061}{325\,205} \approx 0.01864$  containing  $|E(G\langle 49, 6; 3 \rangle)| = 1\,822\,252\,038\,684$  edges. Due to the complexities of Algorithms 5–7 for this specific case, only Algorithms 2–4 could be used to determine lower bounds on  $\Psi_\ell(49, 6; 3)$ . These lower bounds are presented in Figure 5.10, together with the best known bound of  $L(49, 6; 3) \leq 163$ , available from Internet repository tables [19].

From Figure 5.10, it is evident that Algorithms 2 and 3 perform relatively weaker than Algorithm 4, even in a realistic case such as  $\langle 49, 6; 3 \rangle$ . For this specific case, Algorithm 4 performs on average between 3% and 4% better than Algorithms 2 and 3. The best lower bound on the resource utilisation, namely  $\Psi_{163}(49, 6; 3) \geq \frac{13\,752\,983}{13\,983\,816} \approx 98.3493\%$ , was obtained by Algorithm 4 (as opposed to the best lower bounds of  $\Psi_{163}(49, 6; 3) \geq 96.1337\%$  and  $\Psi_{163}(49, 6; 3) \geq 96.7481\%$  obtained by Algorithms 2 and 3, respectively), with the generated playing set given in Table B.1(a).

It is expected that Algorithms 5–7 may yield much better results, although this claim could not be validated, due to the extremely high computational complexity of these algorithms. This restriction, however, may be alleviated partially with the aid of parallelisation programming techniques.

## 5.9 Chapter summary

In this chapter, seven algorithms were investigated in order to obtain lower and upper bounds on respectively the resource utilisation number  $\Psi_\ell(m, n; k)$  and the lottery number  $L_\psi(m, n; k)$ . These algorithms may be classified into three categories, using iterative (i) independent solution generation techniques, (ii) solution construction techniques and (iii) modification of solution candidates techniques.

Algorithm 3 (Distributed random algorithm) achieves a marginal improvement on the performance of Algorithm 2 (Classical random algorithm), although neither algorithm utilises the structure of the lottery graph in terms of the way in which solution candidates are determined. As a possible improvement,

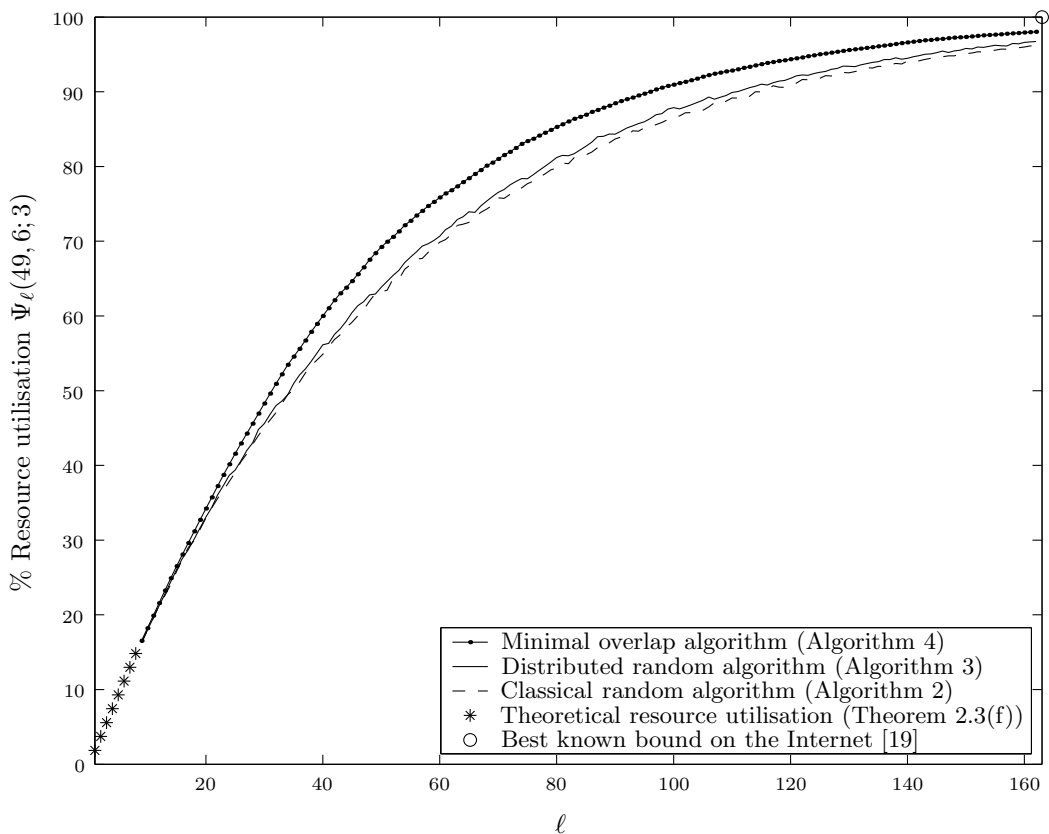


Figure 5.10: Lower bounds on the resource utilisation  $\Psi_\ell(49, 6; 3)$  obtained as a function of  $\ell$ , using Algorithms 2–4.

Algorithm 4 (Minimal overlapping algorithm) incorporates a mechanism for constructing interdependent playing sets. From the results obtained, no significant general improvement occurs for small values of  $\ell > 2$ , although improvements upon the performance of Algorithms 2 and 3 are observed in larger, more realistic lottery cases.

Algorithm 5 (Neighbourhood removal algorithm) represents an effective local optimisation strategy for relatively small values of  $\ell$ . The algorithm progress tends to deteriorate as the order of the vertex-induced subgraph decreases. The locally optimal decision to remove a vertex with largest (closed) neighbourhood, leaves the vertex-induced subgraph disposed to the formation of separate components and hence unnecessarily decreases [increases] the bounds on  $\Psi_\ell(m, n; k)$  [ $L_\psi(m, n; k)$ ].

The performance of the optimisation techniques of Algorithms 6 (Tabu search algorithm) and 7 (Intelligent genetic algorithm) as compared to those of the first four algorithms, are visible in the bounds obtained in Table 5.2. More specifically, Algorithm 7 is the only algorithm that found  $L_1(m, n; k)$ -sets for all the lottery numbers  $L_1(m, n; k)$  where  $1 \leq k < n < m \leq 10$  (presented in Table 3.1). Algorithm 7 is therefore a potentially convincing method for finding (near-) optimal complete lottery sets. The applicability of this algorithm is, however, diminished by the computational intensity of the crossover procedure.

For completeness, the best known lower and upper bounds on  $L_1(m, n; k)$  (where  $2 \leq k < n \leq 6$  and  $3 \leq m \leq 50$ ), taken from covering and lottery repository tables on the Internet [19, 44, 133, 237], are given in Table B.2.



## Chapter 6

# Optimal solution characterisations

“Each problem that I solved became a rule which served afterwards to solve other problems.”  
*René Descartes* (1596–1650)

In cases where the incomplete lottery number  $L_\psi(m, n; k)$  is known, it is possible to characterise all possible  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets. This chapter is devoted to describing two similar enumeration methods (each using a different method of structure representation) for determining all structurally different  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets for  $\langle m, n; k \rangle$ . The first enumeration method (in §6.1.1) employs a vector representation to capture the overlapping structure of any playing set for  $\langle m, n; k \rangle$ , while the second enumeration method (in 6.1.2) utilises a simple graph together with an automorphism testing program. Both enumeration methods may also be used if  $L_\psi(m, n; k)$  is not known in order to attempt establishing the incomplete lottery number, although the computational complexity prohibits implementation for large values of  $m, n, k$  and  $\ell$ . Some general properties of the characterisation sequences are derived in §6.2 (for variations in the parameters  $m, n$  and  $k$ ). The chapter also extends the computational results performed in Chapter 3 (§3.4) in the sense that all structurally different  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (where  $1 \leq k < n < m \leq 10$  such that  $m + k \geq 2n$  and  $n \leq \lfloor \frac{m}{2} \rfloor$ , and  $2 \leq L_1(m, n; k) \leq 6, 7$ ) are determined in §6.3. This is followed by an exhibition of new complete lottery numbers and improved (upper and lower) bounds in §6.4, found by the characterisation techniques in §6.1. The chapter closes with a chapter summary in §6.5.

### 6.1 Characterisation of $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set structures

In order to describe the number of structurally different<sup>1</sup>  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets in this Chapter, we require the following additional notation.

**Definition 6.1 (The incomplete lottery characterisation number  $\eta_\psi(m, n; k)$ )** *Let the set  $\mathcal{L}_\psi = \{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \dots, \mathcal{L}^{(\eta_\psi(m, n; k))}\}$  be composed of all structurally different  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets for the lottery  $\langle m, n; k \rangle$  (where  $1 \leq k \leq n \leq m$ ,  $0 < \psi \leq 1$  and  $1 \leq \ell \leq L_1(m, n; k)$ ). Then, we refer to  $\eta_\psi(m, n; k) = |\mathcal{L}_\psi|$  as the incomplete lottery characterisation number for  $\langle m, n; k \rangle$ . ■*

#### 6.1.1 Lottery tree method

In this section the objective is to find (i) the number of structurally different  $n$ -set overlapping structures of  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$  as well as all  $\eta_\psi(m, n; k)$  actual  $L_\psi(m, n; k)$ -set structures and (ii) the

<sup>1</sup>By *structurally different* sets it is meant that the  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets differ in their  $n$ -set overlapping structure: two  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets may be different, but share the same overlapping structure in terms of their  $n$ -sets, as explained in Chapter 3.



number of different  $n$ -set overlapping structures of  $\Psi_\ell(m, n; k)$ -sets for  $\langle m, n; k \rangle$  as well as all possible  $\Psi_\ell(m, n; k)$  characterisations for small feasible values of  $1 \leq k \leq n \leq m \leq 10$ ,  $L_\psi(m, n; k)$  and  $\ell$ . We define  $\eta_\psi(m, n; k)$  to indicate the number of structurally different playing sets of minimum cardinality yielding a resource utilisation of at least  $0 < \psi \leq 1$  and use the same  $\vec{X}$ -vector notation outlined in §3.2 to distinguish between the structural difference of  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets.

To this end, suppose that  $\mathcal{L}_\ell = \{T_1, T_2, \dots, T_\ell\}$  is a playing set of cardinality  $\ell$  for the lottery  $\langle m, n; k \rangle$ . One method of enumerating all  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set structures for  $\langle m, n; k \rangle$ , consists of constructing a rooted tree (henceforth referred to as the *lottery tree*) of evolving overlapping structures, whose nodes represent overlap specifications similar to that shown in Figure 3.7(b). Level  $i$  of the lottery tree contains all possible (non-isomorphic) overlapping  $n$ -set structures of cardinality  $i$  and is constructed from the nodes on level  $i - 1$  of the lottery tree by appending  $2^{i-1}$  integers to (*i.e.*, doubling) each of the existing vectors  $\vec{X}^{(i-1)}$ . These integer appendices represent all possible (new)  $n$ -set overlappings with respect to the existing overlappings  $\{T_1, T_2, \dots, T_{i-1}\}$  (represented by nodes on level  $i - 1$  of the lottery tree) when adding the  $i$ -th  $n$ -set,  $T_i$  (in such a manner that  $|T_i \cap T_j| < n$  for any two  $n$ -sets  $T_i$  and  $T_j$ ).

The lottery tree has  $\ell + 1$  levels in total. The first level of the tree consists of the node  $\vec{X}^{(1)} = (m - n, n)$  only (the root), while the nodes  $\vec{X}^{(\ell)}$  on level  $\ell$  of the lottery tree represent potential  $\Psi_\ell(m, n; k)$ -set structures or  $L_\psi(m, n; k)$ -set structures of cardinality  $\ell$  for  $\langle m, n; k \rangle$ . An  $(\ell + 1)$ -st level of nodes is added to the tree (in such a manner that  $|T_{\ell+1} \cap T_j| \leq n$  for all  $j \leq \ell$ ) in order to carry out a so-called domination test<sup>2</sup> (*i.e.*, to test which of the nodes on level  $\ell$  actually represent valid  $L_\psi(m, n; k)$ -sets of cardinality  $\ell$ ). This domination test is achieved by testing whether *all* nodes on level  $\ell + 1$  of the tree overlap in at least  $k$  positions with at least one  $n$ -set of the existing  $\ell$   $n$ -set overlapping structure (represented by its parent node  $\vec{X}^{(\ell)}$ ) in the tree (*i.e.*,  $|T_i \cap T_{\ell+1}| \geq k$  for some  $i \in \{1, \dots, \ell\}$ ). If this were the case, then the  $n$ -set overlapping structure represented by the parent node  $\vec{X}^{(\ell)}$  would constitute an  $L_1(m, n; k)$ -set for  $\langle m, n; k \rangle$ . However, if there exists at least one node on level  $\ell + 1$  of the tree whose corresponding final  $n$ -set overlaps in fewer than  $k$  positions with *all*  $n$ -sets of the parent node overlapping structure, the parent node does not represent a complete lottery set (although possibly an incomplete lottery set) and hence  $\vec{X}^{(\ell)}$  will yield a probability,  $\psi_{\vec{X}^{(\ell)}} < 1$  (say), of winning a  $k$ -prize by means of a playing set whose structure conforms to the vector  $\vec{X}^{(\ell)}$ . In order to quantify this probability, define the set  $\mathcal{D}_T$  [ $\mathcal{D}_F$ ] as all those structures  $\vec{X}^{(\ell+1)}$  on level  $\ell + 1$  with parent node  $\vec{X}^{(\ell)}$  in the lottery tree that yield a **True** [**False**] result in the domination test (hence  $\mathcal{D}_F = \emptyset$  for any  $L_1(m, n; k)$ -set). The probability  $\psi_{\vec{X}^{(\ell)}}$  of a parent node  $\vec{X}^{(\ell)}$  yielding a  $k$ -prize in the lottery  $\langle m, n; k \rangle$  is then given by

$$\psi_{\vec{X}^{(\ell)}} = \frac{\sum_{\vec{X}^{(\ell+1)} \in \mathcal{D}_T} \mathcal{M}(\vec{X}^{(\ell+1)})}{\sum_{\vec{X}^{(\ell+1)} \in \{\mathcal{D}_T \cup \mathcal{D}_F\}} \mathcal{M}(\vec{X}^{(\ell+1)})}, \quad (6.1)$$

where  $\mathcal{M}(\vec{X}^{(\ell)})$  is given in (3.4). The enumeration procedure described for generating a lottery tree for  $\langle m, n; k \rangle$  is given in pseudocode as Algorithm 8. For clarity of the description of the above mentioned enumeration method and parameters, construction of lottery trees for the lotteries  $\langle 5, 3; 2 \rangle$  and  $\langle 7, 3; 2 \rangle$  are reconsidered in the following examples.

**Example 6.1 (continuation of Example 3.3)** *Reconsider the lottery  $\langle 5, 3; 2 \rangle$  of Example 3.3. It is known that  $L_1(5, 3; 2) = 2$ . Using the above mentioned method of enumerating all the possible  $L_1(5, 3; 2)$ -set structures for  $\langle 5, 3; 2 \rangle$ , the lottery tree in Figure 6.1 is obtained. The table in Figure 6.1 contains the relevant domination test information, as obtained from the nodes on level three of the lottery tree. By utilisation of (6.1) the vector  $\vec{X}^{(2)} = (1, 1, 1, 2)$  yields a probability of  $\psi_{\vec{X}^{(2)}} = \frac{60+120+120+120+60+60}{60+120+120+60+120+60+60} = \frac{9}{10}$  of winning a 2-prize, while the corresponding probability is  $\psi_{\vec{X}^{(2)}} = 1$  for the vector  $\vec{X}^{(2)} = (0, 2, 2, 1)$ . From this it is concluded that the only  $L_\psi(5, 3; 2)$ -set structure for  $\frac{9}{10} < \psi \leq 1$  is given by the vector  $\vec{X}^{(2)} = (0, 2, 2, 1)$ , implying that  $\eta_\psi(5, 3; 2) = 1$  (*i.e.*, a unique minimum cardinality solution to the incomplete lottery problem in this case). However,  $\eta_\psi(5, 3; 2) = 2$  if  $\frac{7}{10} < \psi \leq \frac{9}{10}$  with the additional*

<sup>2</sup>The reader should note that the domination test carried out in the lottery tree, refers to the domination of the lottery set structures in the lottery graph  $G(m, n; k)$ , and not to the conventional domination of the lottery tree itself.

**Algorithm 8**  $L_\psi(m, n; k)$ -set characterisation algorithm**Input:** The lottery parameters  $\langle m, n; k \rangle$ , a playing set cardinality  $\ell$ .**Output:**  $L_{\text{ruBest}}(m, n; k) = \ell$  or  $\Psi_\ell(m, n; k) = \text{ruBest}$  and  $\eta_{\text{ruBest}}(m, n; k) = \text{structures}$ .

```

1: level  $\leftarrow$  1, structures  $\leftarrow$  0, ruBest  $\leftarrow$  0, initialise root node  $\vec{X}^{(1)}$ .
2: Generate 1st lexicographic child (i.e., double parent vector  $\vec{X}^{(\text{level})}$ ).
3: level  $\leftarrow$  level + 1.
4: if (level  $\leq$   $\ell$ ) then
5:   Use pruning rules [permutation checking] to eliminate duplicate  $\vec{X}^{(\text{level})}$  structures.
6:   if ( $\vec{X}^{(\text{level})}$  is eliminated) then
7:     goto (18).
8:   end if
9:   goto (2).
10: else [(level =  $\ell$  + 1)]
11:   Determine  $\psi_{\vec{X}^{(\ell)}}$  (i.e., evaluate and sum all contributions of  $\vec{X}^{(\ell+1)}$ ).
12:   if ( $\psi_{\vec{X}^{(\ell)}} >$  ruBest) then
13:     ruBest  $\leftarrow$   $\psi_{\vec{X}^{(\ell)}}$ , structures  $\leftarrow$  0.
14:   else if ( $\psi_{\vec{X}^{(\ell)}} =$  ruBest) then
15:     structures  $\leftarrow$  structures + 1.
16:   end if
17: end if
18: while ((level > 1) and (next lexicographic sibling  $\vec{X}^{(\text{level})}$  does not exist)) do
19:   level  $\leftarrow$  level - 1.
20: end while
21: if (level = 1) then
22:   print  $L_{\text{ruBest}}(m, n; k) = \ell$ .
23:   print  $\Psi_\ell(m, n; k) = \text{ruBest}$ .
24:   print  $\eta_{\text{ruBest}}(m, n; k) = \text{structures}$ . stop.
25: else [((level > 1) and (next lexicographic sibling  $\vec{X}^{(\text{level})}$  exists))]
26:   Generate next lexicographic sibling  $\vec{X}^{(\text{level})}$ .
27:   goto (2).
28: end if

```

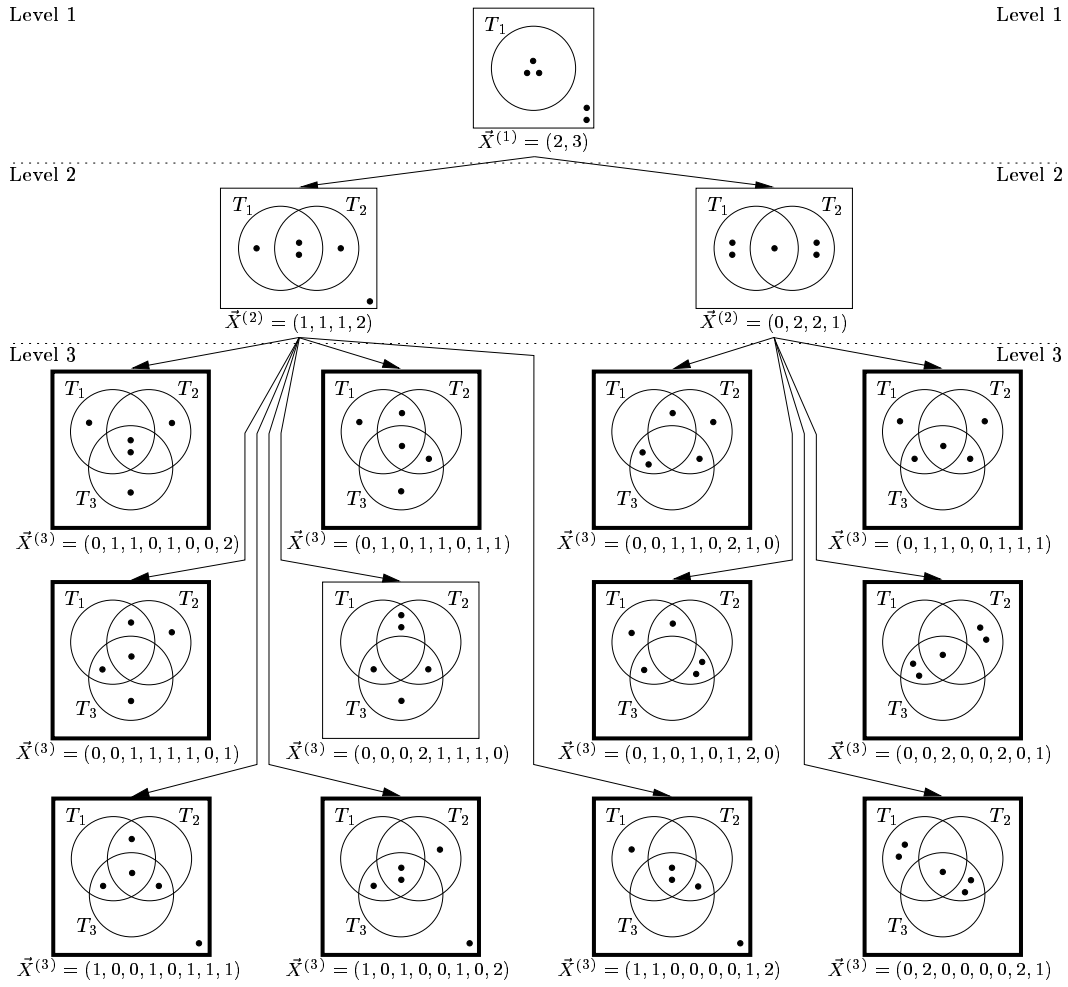
minimum cardinality characterisation being the vector  $\vec{X}^{(2)} = (1, 1, 1, 2)$ . Any 3-set taken from  $\mathcal{U}_5$  necessarily yields a probability  $\psi_{\vec{X}^{(1)}} = \frac{7}{10}$  of winning a 2-prize (where  $\vec{X}^{(1)} = (2, 3)$  is given by the root of the lottery tree for  $\langle 5, 3; 2 \rangle$  in Figure 6.1), implying that  $\eta_\psi(5, 3; 2) = 1$  for all  $0 < \psi \leq \frac{7}{10}$ . ■

**Example 6.2 (continuation of Example 3.4)** Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 3.4. Figure 6.2(a)–(c) contains the only three nodes on level two of the lottery tree. Figure 6.2(d) [(e), (f)] contains all possible 3-set overlappings  $\vec{X}^{(3)}$  when an additional (winning) 3-set,  $T_3$ , is added to the construction  $\vec{X}^{(2)} = (3, 1, 1, 2)$  [ $\vec{X}^{(2)} = (2, 2, 2, 1)$ ,  $\vec{X}^{(2)} = (1, 3, 3, 0)$ ] on level two of the lottery tree. Each of these constructions is accompanied by its respective (i) multiplicity (using (3.4)) and (ii) domination test results in order to determine the probabilities  $\psi_{\vec{X}^{(2)}}$ . From these results it is possible to deduce that  $L_\psi(7, 3; 2) = 2$  if  $\frac{13}{35} < \psi \leq \frac{26}{35}$  and that

$$\eta_\psi(7, 3; 2) = \begin{cases} 3 & \text{if } \frac{13}{35} < \psi \leq \frac{19}{35} \text{ (vectors } \vec{X}^{(2)} = (3, 1, 1, 2), \vec{X}^{(2)} = (2, 2, 2, 1) \text{ and } \vec{X}^{(2)} = (1, 3, 3, 0)) \\ 2 & \text{if } \frac{19}{35} < \psi \leq \frac{22}{35} \text{ (vectors } \vec{X}^{(2)} = (2, 2, 2, 1) \text{ and } \vec{X}^{(2)} = (1, 3, 3, 0)) \\ 1 & \text{if } \frac{22}{35} < \psi \leq \frac{26}{35} \text{ (vector } \vec{X}^{(2)} = (1, 3, 3, 0)). \end{cases}$$

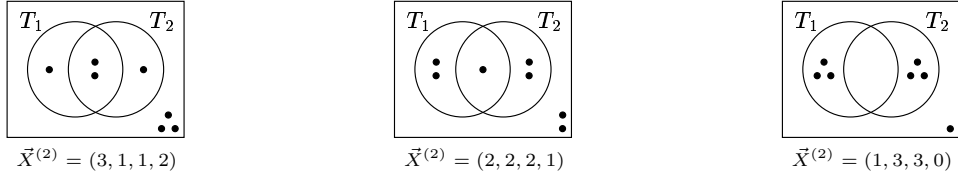
Moreover, it is clear that  $\Psi_2(7, 3; 2) = \frac{26}{35}$ , yielded by the vector  $\vec{X}^{(2)} = (1, 3, 3, 0)$  (recall this result from Example 3.1). With the root of the lottery tree ( $\vec{X}^{(1)} = (4, 3)$ ) acting as the only possible 3-set from  $\mathcal{U}_7$ ,  $\eta_\psi(7, 3; 2) = 1$  for all  $0 < \psi \leq \frac{13}{35}$ . ■

The previous example focussed on the determination of  $\eta_\psi(7, 3; 2)$  when  $L_\psi(7, 3; 2) = 2$  for  $\langle 7, 3; 2 \rangle$ .

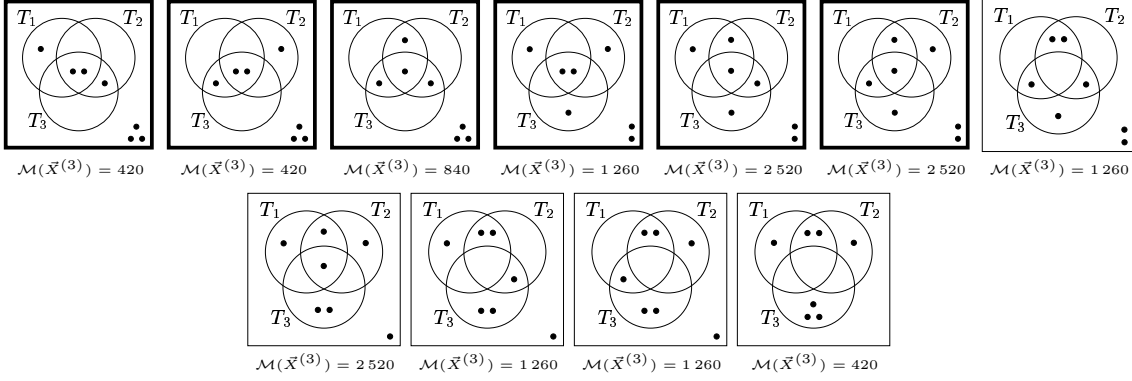


Potential $L_\psi(5, 3; 2)$ -set structure	Domination test candidate	Result	$\mathcal{M}(\vec{X}^{(3)})$
 $\vec{X}^{(2)} = (1, 1, 1, 2)$	$\vec{X}^{(3)} = (0, 1, 1, 0, 1, 0, 0, 2)$	True	60
	$\vec{X}^{(3)} = (0, 0, 1, 1, 1, 1, 0, 1)$	True	120
	$\vec{X}^{(3)} = (0, 1, 0, 1, 1, 0, 1, 1)$	True	120
	$\vec{X}^{(3)} = (0, 0, 0, 2, 1, 1, 1, 0)$	False	60
	$\vec{X}^{(3)} = (1, 0, 0, 1, 0, 1, 1, 1)$	True	120
	$\vec{X}^{(3)} = (1, 0, 1, 0, 0, 1, 0, 2)$	True	60
	$\vec{X}^{(3)} = (1, 1, 0, 0, 0, 0, 1, 2)$	True	60
 $\vec{X}^{(2)} = (0, 2, 2, 1)$	$\vec{X}^{(3)} = (0, 0, 1, 1, 0, 2, 1, 0)$	True	60
	$\vec{X}^{(3)} = (0, 1, 0, 1, 0, 1, 2, 0)$	True	60
	$\vec{X}^{(3)} = (0, 1, 1, 0, 0, 1, 1, 1)$	True	120
	$\vec{X}^{(3)} = (0, 0, 2, 0, 0, 2, 0, 1)$	True	30
	$\vec{X}^{(3)} = (0, 2, 0, 0, 0, 0, 2, 1)$	True	30

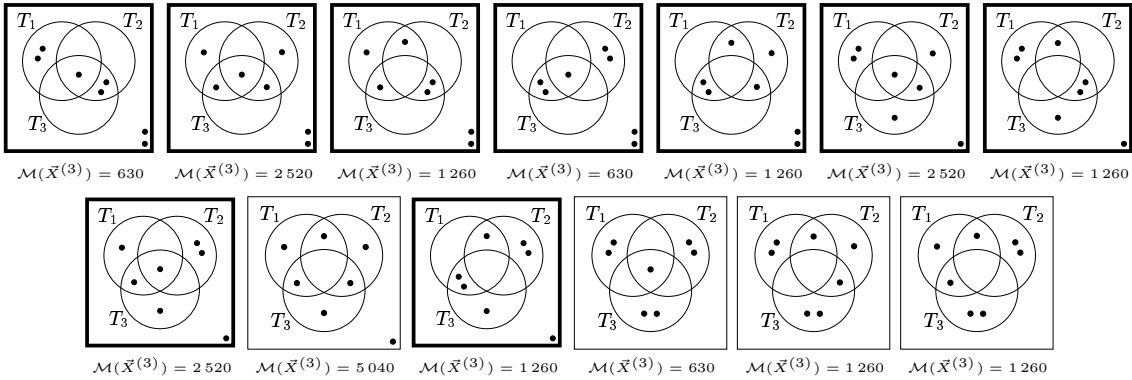
Figure 6.1: Complete lottery tree and domination test results for the lottery  $\langle 5, 3; 2 \rangle$ . Bold emboxed structures indicate that the parent node is an  $L_1(5, 3; 2)$ -set structure and regular emboxed structures are witness to the fact that the parent node is an  $L_\psi(m, n; k)$ -set structure with  $\psi < 1$ . The results of the domination test performed on the nodes on level two ( $\vec{X}^{(2)}$ ) and multiplicities of the nodes on level three ( $\vec{X}^{(3)}$ ) of the lottery tree are presented in the table beneath the tree. It is concluded that  $\eta_\psi(5, 3; 2) = 2$  if  $\frac{7}{10} < \psi \leq \frac{9}{10}$  and that  $\eta_\psi(5, 3; 2) = 1$  if  $\frac{9}{10} < \psi \leq 1$ . With the root of the lottery tree ( $\vec{X}^{(1)} = (2, 3)$ ) acting as the only possible individual 3-set from  $\mathcal{U}_5$ ,  $\eta_\psi(5, 3; 2) = 1$  for all  $0 < \psi \leq \frac{7}{10}$ .



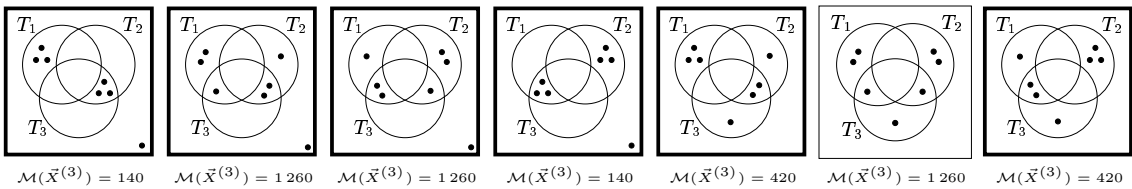
(a) Lexicographic first node on level 2 of the lottery tree for the lottery  $\langle 7, 3; 2 \rangle$       (b) Lexicographic second node on level 2 of the lottery tree for the lottery  $\langle 7, 3; 2 \rangle$       (c) Lexicographic third (last) node on level 2 of the lottery tree for the lottery  $\langle 7, 3; 2 \rangle$



(d) All 11 nodes used in the domination test for the lexicographic first node,  $\vec{X}^{(2)} = (3, 1, 1, 2)$ , on level 3 of the lottery tree for  $\langle 7, 3; 2 \rangle$ , yielding a probability (according to (6.1)) of  $\psi_{\vec{X}^{(2)}} = \frac{7980}{14700} = \frac{19}{35}$  of winning a 2-prize



(e) All 13 nodes used in the domination test for the lexicographic second node,  $\vec{X}^{(2)} = (2, 2, 2, 1)$ , on level 3 of the lottery tree for  $\langle 7, 3; 2 \rangle$ , yielding a probability (according to (6.1)) of  $\psi_{\vec{X}^{(2)}} = \frac{13860}{22050} = \frac{22}{35}$  of winning a 2-prize



(f) All 7 nodes used in the domination test for the lexicographic third (last) node,  $\vec{X}^{(2)} = (1, 3, 3, 0)$ , on level 3 of the lottery tree for  $\langle 7, 3; 2 \rangle$ , yielding a probability (according to (6.1)) of  $\psi_{\vec{X}^{(2)}} = \frac{3640}{4900} = \frac{26}{35}$  of winning a 2-prize

Figure 6.2: Fragmented lottery tree for  $\langle 7, 3; 2 \rangle$  up to level three (Example 6.2). Subfigure (a) [(b), (c)] contains the lexicographic first [second, third (last)] node on level two of the tree. The relevant nodes used in the domination test (in level three of the tree) are given in subfigure (d) [(e), (f)], together with their respective domination test results (bold [regular] embossed structures denote that  $\vec{X}^{(3)} \in \mathcal{D}_T$  [ $\vec{X}^{(3)} \in \mathcal{D}_F$ ]) and multiplicities.

$\ell$	$\vec{X}^{(\ell)}$	$\psi_{\vec{X}^{(\ell)}}$	$\eta_{\psi_{\vec{X}^{(\ell)}}}(7, 3; 2)$	$L_{\psi_{\vec{X}^{(\ell)}}}(7, 3; 2)$
1	(4, 3)	$\frac{13}{35}$	<b>1</b>	<b>1</b>
2	(3, 1, 1, 2)	$\frac{19}{35}$	<b>3</b>	<b>2</b>
	(2, 2, 2, 1)	$\frac{22}{35}$	<b>2</b>	
	(1, 3, 3, 0)	$\frac{26}{35}$	<b>1</b>	
	(0, 2, 2, 0, 2, 0, 0, 1)	$\frac{27}{35}$	<b>5</b>	
3	(1, 1, 0, 2, 2, 0, 1, 0)	$\frac{28}{35}$	<b>4</b>	<b>3</b>
	(1, 1, 1, 1, 1, 1, 1, 0)	$\frac{28}{35}$		
	(0, 2, 1, 1, 2, 0, 1, 0)	$\frac{31}{35}$	<b>2</b>	
	(0, 1, 1, 2, 3, 0, 0, 0)	$\frac{32}{35}$	<b>1</b>	
	(0, 0, 0, 1, 1, 1, 1, 0, 2, 0, 0, 1, 0, 0, 0, 0)	$\frac{33}{35}$	<b>7</b>	
4	(0, 0, 0, 2, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0)	$\frac{33}{35}$		4
	(0, 1, 0, 1, 0, 0, 1, 1, 2, 0, 0, 0, 1, 0, 0, 0, 0)	$\frac{33}{35}$		
	(0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0)	$\frac{34}{35}$	<b>4</b>	
	(0, 1, 0, 2, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0)	$\frac{34}{35}$		
	(0, 0, 0, 2, 0, 1, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0)	$\frac{35}{35}$	<b>2</b>	
	(0, 0, 0, 1, 0, 1, 1, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0)	$\frac{35}{35}$		

Table 6.1: All structurally different  $\eta_{\psi}(7, 3; 2)$   $L_{\psi}(7, 3; 2)$ -sets ( $0 < \psi \leq 1$ ) for  $\langle 7, 3; 2 \rangle$ , as given in (6.2). Vertical lines between columns 3 and 4 (read top to bottom for every value of  $\ell$ ) depict the span of structures  $\vec{X}^{(\ell)}$  that result in the value for  $\eta_{\psi_{\vec{X}^{(\ell)}}}(7, 3; 2)$ . Boldfaced entries indicate previously undetermined results.

As an extension to this exploration, the following example wraps up the discussion by determining all  $\eta_{\psi}(7, 3; 2)$   $L_{\psi}(7, 3; 2)$ -sets where  $0 < \psi \leq 1$ , using the enumeration method described in §6.1.1.

**Example 6.3 (continuation of Example 6.2)** Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 6.2. Using the enumeration method (Algorithm 8), the value of  $\eta_{\psi}(7, 3; 2)$  for all  $0 < \psi \leq 1$ , is given by

$$\eta_{\psi}(7, 3; 2) = \left\{ \begin{array}{l} 1 \text{ if } 0 < \psi \leq \frac{13}{35} \\ 3 \text{ if } \frac{13}{35} < \psi \leq \frac{19}{35} \\ 2 \text{ if } \frac{19}{35} < \psi \leq \frac{22}{35} \\ 1 \text{ if } \frac{22}{35} < \psi \leq \frac{26}{35} \\ \hline 5 \text{ if } \frac{26}{35} < \psi \leq \frac{27}{35} \\ 4 \text{ if } \frac{27}{35} < \psi \leq \frac{28}{35} \\ 2 \text{ if } \frac{28}{35} < \psi \leq \frac{31}{35} \\ 1 \text{ if } \frac{31}{35} < \psi \leq \frac{32}{35} \\ \hline 7 \text{ if } \frac{32}{35} < \psi \leq \frac{33}{35} \\ 4 \text{ if } \frac{33}{35} < \psi \leq \frac{34}{35} \\ 2 \text{ if } \frac{34}{35} < \psi \leq 1 \end{array} \right\} \begin{array}{l} L_{\psi}(7, 3; 2) = 1 \\ \left. \begin{array}{l} L_{\psi}(7, 3; 2) = 2 \\ L_{\psi}(7, 3; 2) = 3 \\ L_{\psi}(7, 3; 2) = 4 \end{array} \right\} \end{array} \quad (6.2)$$

The corresponding  $\eta_{\psi}(7, 3; 2)$  non-isomorphic structures for  $\langle 7, 3; 2 \rangle$  are given in Table 6.1. ■

One problem that may occur with the implementation of Algorithm 8 is in the calculation of  $\psi_{\vec{X}^{(\ell)}}$  in (6.1) for a vector  $\vec{X}^{(\ell)}$ . It is well-documented that problems may arise when dealing with large numbers and fractions in a programming language [62]. Large numbers may not always be manageable, while fractions may cause a loss of precision<sup>3</sup>. To counter this computational drawback, a different approach to the formula in (6.1) was used to calculate  $\psi_{\vec{X}^{(\ell)}}$ . This probability was computed in Algorithm 8 by rather

<sup>3</sup>When considering large numbers, the maximum size of a positive integer that may be used in C++ (for example) is

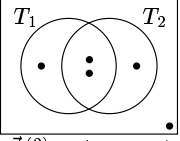
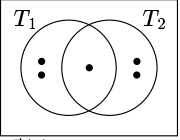
Potential $L_\psi(5, 3; 2)$ -set structure	Domination test candidate	Domination result	Contribution of $\vec{X}^{(3)}$ to $\psi_{\vec{X}^{(2)}}$
 $\vec{X}^{(2)} = (1, 1, 1, 2)$	$\vec{X}^{(3)} = (0, 1, 1, 0, 1, 0, 2)$	True	$\binom{(1,1,1,2)}{(0,1,1,0)} / \binom{5}{3} = \binom{1}{1} \binom{1}{1} / 10 = \frac{1}{10}$
	$\vec{X}^{(3)} = (0, 0, 1, 1, 1, 1, 0, 1)$	True	$\binom{(1,1,1,2)}{(0,0,1,1)} / \binom{5}{3} = \binom{1}{1} \binom{2}{1} / 10 = \frac{2}{10}$
	$\vec{X}^{(3)} = (0, 1, 0, 1, 1, 0, 1, 1)$	True	$\binom{(1,1,1,2)}{(0,1,0,1)} / \binom{5}{3} = \binom{1}{1} \binom{2}{1} / 10 = \frac{2}{10}$
	$\vec{X}^{(3)} = (0, 0, 0, 2, 1, 1, 1, 0)$	False	—
	$\vec{X}^{(3)} = (1, 0, 0, 1, 0, 1, 1, 1)$	True	$\binom{(1,1,1,2)}{(1,0,0,1)} / \binom{5}{3} = \binom{1}{1} \binom{2}{1} / 10 = \frac{2}{10}$
	$\vec{X}^{(3)} = (1, 0, 1, 0, 0, 1, 0, 2)$	True	$\binom{(1,1,1,2)}{(1,0,1,0)} / \binom{5}{3} = \binom{1}{1} \binom{1}{1} / 10 = \frac{1}{10}$
 $\vec{X}^{(2)} = (0, 2, 2, 1)$	$\vec{X}^{(3)} = (0, 0, 1, 1, 0, 2, 1, 0)$	True	$\binom{(0,2,2,1)}{(0,0,1,1)} / \binom{5}{3} = \binom{2}{1} \binom{1}{1} / 10 = \frac{2}{10}$
	$\vec{X}^{(3)} = (0, 1, 0, 1, 0, 1, 2, 0)$	True	$\binom{(0,2,2,1)}{(0,1,0,1)} / \binom{5}{3} = \binom{2}{1} \binom{1}{1} / 10 = \frac{2}{10}$
	$\vec{X}^{(3)} = (0, 1, 1, 0, 0, 1, 1, 1)$	True	$\binom{(0,2,2,1)}{(0,1,1,0)} / \binom{5}{3} = \binom{2}{2} \binom{2}{2} / 10 = \frac{4}{10}$
	$\vec{X}^{(3)} = (0, 0, 2, 0, 0, 2, 0, 1)$	True	$\binom{(0,2,2,1)}{(0,0,2,0)} / \binom{5}{3} = \binom{2}{2} / 10 = \frac{1}{10}$
	$\vec{X}^{(3)} = (0, 2, 0, 0, 0, 0, 2, 1)$	True	$\binom{(0,2,2,1)}{(0,2,0,0)} / \binom{5}{3} = \binom{2}{2} / 10 = \frac{1}{10}$

Table 6.2: Alternative approach (incorporated in Algorithm 8) toward determining the winning probabilities of  $\vec{X}^{(2)}$  for  $\langle 5, 3; 2 \rangle$  in Example 6.4. All contributions (column 4) for a specific  $\vec{X}^{(2)}$  are summed, according to (6.3), in order to obtain  $\psi_{\vec{X}^{(2)}}$ .

fixing the  $(\ell + 1)$ -st  $n$ -set,  $T_{\ell+1}$ , and counting the different possible ways of choosing the overlappings in  $\{T_1, T_2, \dots, T_\ell\}$  when the elements of  $T_{\ell+1}$  are removed. The summed contribution of each node yields

$$\psi_{\vec{X}^{(\ell)}} = \sum_{\vec{X}^{(\ell+1)} \in \mathcal{D}_T} \left( \prod_{i=0}^{2^\ell - 1} \binom{x_i^{(\ell)}}{x_i^{(\ell+1)}} \right) / \binom{m}{n}, \quad (6.3)$$

where  $\ell = L_\psi(m, n; k)$ . The following example is presented in an attempt to illustrate the above alternative approach to the computation of  $\psi_{\vec{X}^{(\ell)}}$ , according to (6.3).

**Example 6.4 (continuation of Example 6.1)** Reconsider the lottery  $\langle 5, 3; 2 \rangle$  of Example 6.1. The (only) two vectors  $\vec{X}^{(2)} = (1, 1, 1, 2)$  and  $\vec{X}^{(2)} = (0, 2, 2, 1)$  respectively yield probabilities  $\psi_{\vec{X}^{(2)}} = \frac{9}{10}$  and  $\psi_{\vec{X}^{(2)}} = 1$  of winning a 2-prize. These probabilities may be determined by considering the values in Table 6.2. All contributions (column 4) for a specific  $\vec{X}^{(2)}$  are summed, according to (6.3), in order to obtain  $\psi_{\vec{X}^{(2)}}$ . So, for  $\vec{X}^{(2)} = (1, 1, 1, 2)$ ,  $\psi_{\vec{X}^{(2)}} = \frac{1+2+2+2+1+1}{10} = \frac{9}{10}$ , while  $\psi_{\vec{X}^{(2)}} = \frac{2+2+4+1+1}{10} = 1$  for  $\vec{X}^{(2)} = (0, 2, 2, 1)$ . ■

The lottery tree has an extremely large width-wise growth rate. By introducing a suitable (lexicographic) order in which new vectors on level  $i$  are inserted, duplicate structures in every lottery subtree (with roots on level  $i - 1$ ) may be eliminated, causing a reduction in the width-wise growth rate of and hence the time required for construction of the lottery tree. This procedure is referred to as pruning of the lottery tree. Alternative pruning rules are also possible with the incorporation of a (lexicographic) permutation testing method in which similar  $n$ -set overlapping structures may be avoided (yielding similar overlapping structures) down different branches of the lottery tree (refer back to the discussion of the non-unique vector  $\vec{X}^{(4)}$  in Example 3.2). The following example illustrates why pruning of a lottery tree is desirable.

**Example 6.5 (continuation of Example 3.2)** Reconsider the lottery  $\langle 14, 6; 3 \rangle$  in Example 3.2. Figure 6.3 represents the complete first and second levels of the corresponding lottery tree, together with

$4294976295 < 13!$  (using the standard data type `unsigned long int`). These limits may vary for different programming languages, although relatively small factorial computations commonly supercede the capacity of the standard data type available. Similarly, when dealing with fractions, rounding errors and hence a loss of precision may easily occur (for example,  $\frac{10}{3} = 3.\dot{3}$ , although a computer will only allocate a fixed amount of memory to store such a value and hence only an approximation to  $\frac{10}{3}$ ).

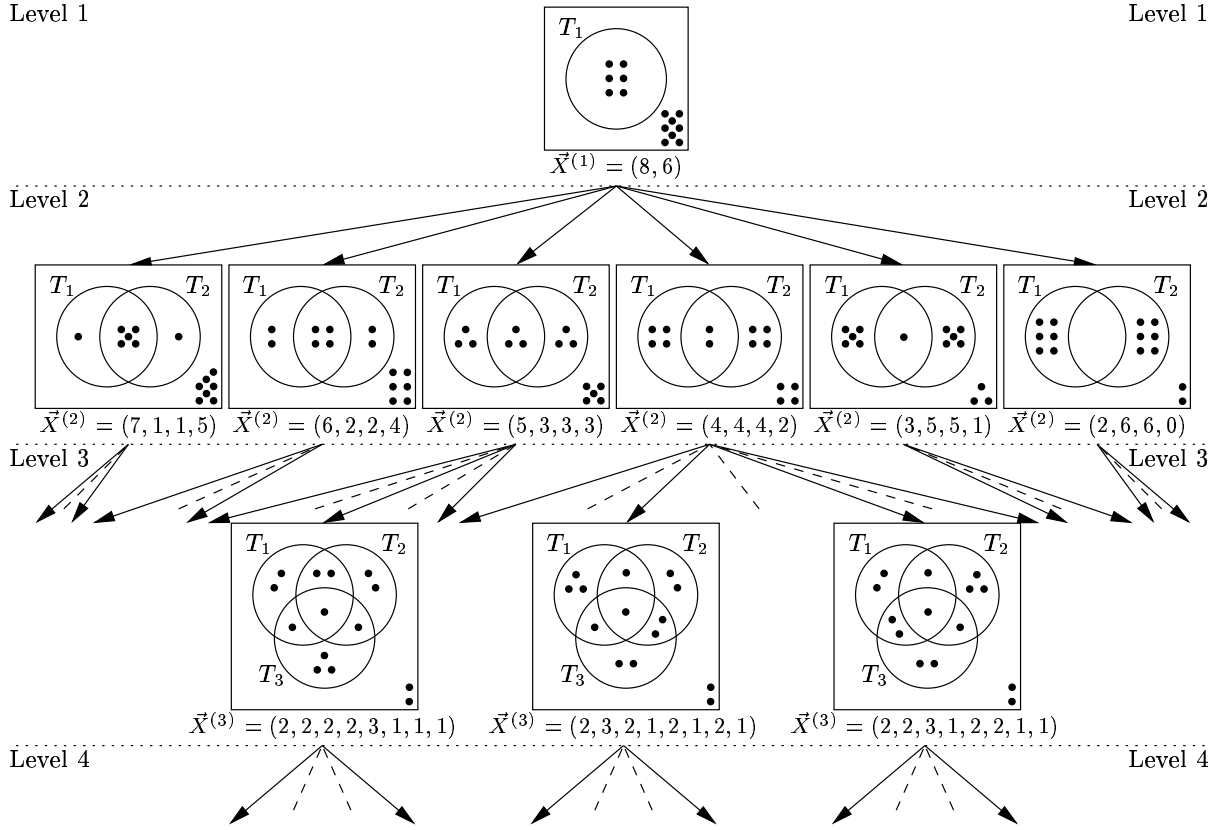


Figure 6.3: Partial lottery tree for the lottery  $\langle 14, 6; 3 \rangle$  displaying the generation of the vectors  $\vec{X}^{(i)}$  as well as possible equivalent permutation structures that occur during the lottery tree generation. Only one of these duplicate structures need be considered from level four downwards in the lottery tree.

the respective vectors  $\vec{X}^{(i)}$  ( $i = 1, 2$ ). Each of the three vectors  $\vec{X}^{(3)}$  on level three of the lottery tree represents equivalent overlapping structures, although they result from different branches down the lottery tree. Only one of these duplicate structures has to be considered from level four downwards in the lottery tree. ■

Pruning (width-wise growth rate reduction) is restricted to levels three to  $L_\psi(m, n; k)$  of the lottery tree (and therefore all reductions are excluded on level  $L_\psi(m, n; k) + 1$ , where the domination test is performed). Reviewing Example 6.1, it is observed that the vector pair  $\vec{X}^{(3)} = (0, 0, 1, 1, 1, 1, 0, 1)$  and  $\vec{X}^{(3)} = (0, 1, 0, 1, 1, 0, 1, 1)$  from the leftmost branch (or similarly the vector pair  $\vec{X}^{(3)} = (0, 0, 1, 1, 0, 2, 1, 0)$  and  $\vec{X}^{(3)} = (0, 1, 0, 1, 0, 1, 2, 0)$  from the rightmost branch) are duplicates. Although the vector pairs  $\vec{X}^{(3)} = (0, 0, 0, 2, 1, 1, 1, 0)$  (yielding a **False** domination test) and  $\vec{X}^{(3)} = (0, 0, 1, 1, 0, 2, 1, 0)$  (yielding a **True** domination test) are also duplicates (resulting from different branches in the lottery tree), both must be included in the domination test, because duplicate  $\vec{X}^{(L_\psi(m, n; k) + 1)}$ -structures may belong to either  $\mathcal{D}_T$  or  $\mathcal{D}_F$ , depending on the specific subtree from which it originates. It is therefore concluded that the enforcement of pruning rules on level  $L_\psi(m, n; k) + 1$  of the lottery tree should be prohibited. The following two additional rules for specifically determining  $L_1(m, n; k)$ -sets for  $\langle m, n; k \rangle$  (obtained by recursive application of the specific case at level  $\ell = L_1(m, n; k)$ ) may be implemented at any level  $1 \leq \ell \leq L_1(m, n; k)$  to prune the lottery tree, thereby speeding up the domination test process:

- (1) If there are more than  $n$  elements from  $U_m$  not contained in any  $n$ -set of the overlapping structure  $\vec{X}^{(\ell)}$ , any winning  $n$ -set containing  $n$  of the unused elements will cause  $\vec{X}^{(\ell)}$  (by definition) to not represent an  $L_1(m, n; k)$ -set for  $\langle m, n; k \rangle$ . In other words, if  $x_{(000\dots 0)_2}^{(\ell)} \geq (L_1(m, n; k) - \ell + 1)n$ ,

then the structure corresponding to the vector  $\vec{X}^{(\ell)}$  will *not* yield any  $L_1(m, n; k)$ -sets down the subtree having  $\vec{X}^{(\ell)}$  as root and hence may be omitted from the lottery tree.

- (2) If the number of elements that are in at most one  $n$ -set of the structure corresponding to  $\vec{X}^{(\ell)}$  are added and found to be at least  $n$ , there exists an  $n$ -set having no  $k$ -intersection with any of the  $n$ -sets in the structure  $\vec{X}^{(\ell)}$  and hence the structure does not represent an  $L_1(m, n; k)$ -set for  $\langle m, n; k \rangle$ . In other words, if  $\min\{x_{(100\dots 0)_2}^{(\ell)}, k-1\} + \dots + \min\{x_{(000\dots 1)_2}^{(\ell)}, k-1\} + x_{(000\dots 0)_2}^{(\ell)} \geq (L_1(m, n; k) - \ell + 1)n$ , then the structure corresponding to the vector  $\vec{X}^{(\ell)}$  will *not* yield any  $L_1(m, n; k)$ -sets down the subtree having  $\vec{X}^{(\ell)}$  as root and hence may be omitted from the lottery tree.

The following example is presented to illustrate (i) the benefit of enforcing the pruning rules outlined above, and (ii) the amount of CPU time invested in the determination of the  $L_1(15, 6; 3)$ -set [ $L_1(16, 6; 3)$ -set,  $L_1(17, 6; 3)$ -set] structure characterisations for  $\langle 15, 6; 3 \rangle$  [ $\langle 16, 6; 3 \rangle$ ,  $\langle 17, 6; 3 \rangle$ ] (the largest case that could be investigated in this dissertation)].

**Example 6.6** Consider the lottery  $\langle 15, 6; 3 \rangle$  [ $\langle 16, 6; 3 \rangle$ ,  $\langle 17, 6; 3 \rangle$ ] with known complete lottery number  $L_1(15, 6; 3) = 4$  [ $L_1(16, 6; 3) = 5$ ,  $L_1(17, 6; 3) = 6$ ]. The characterisation algorithm (Algorithm 8) was used to determine all the possible structurally different  $L_1(15, 6; 3)$ -set [ $L_1(16, 6; 3)$ -set,  $L_1(17, 6; 3)$ -set] structures for  $\langle 15, 6; 3 \rangle$  [ $\langle 16, 6; 3 \rangle$ ,  $\langle 17, 6; 3 \rangle$ ]. These structures are shown graphically in Figure 6.4(a)(i)–(iv) [(v)–(xi), (xii)–(xiv)]. It is concluded that  $\eta_1(15, 6; 3) = 4$  [ $\eta_1(16, 6; 3) = 7$ ,  $\eta_1(17, 6; 3) = 3$ ]. Algorithmic statistics involved with the characterisation of the four  $L_1(15, 6; 3)$ -set [seven  $L_1(16, 6; 3)$ -set, three  $L_1(17, 6; 3)$ -set] structures are shown in Figures 6.4(b) and (c). ■

### 6.1.2 nauty tree method

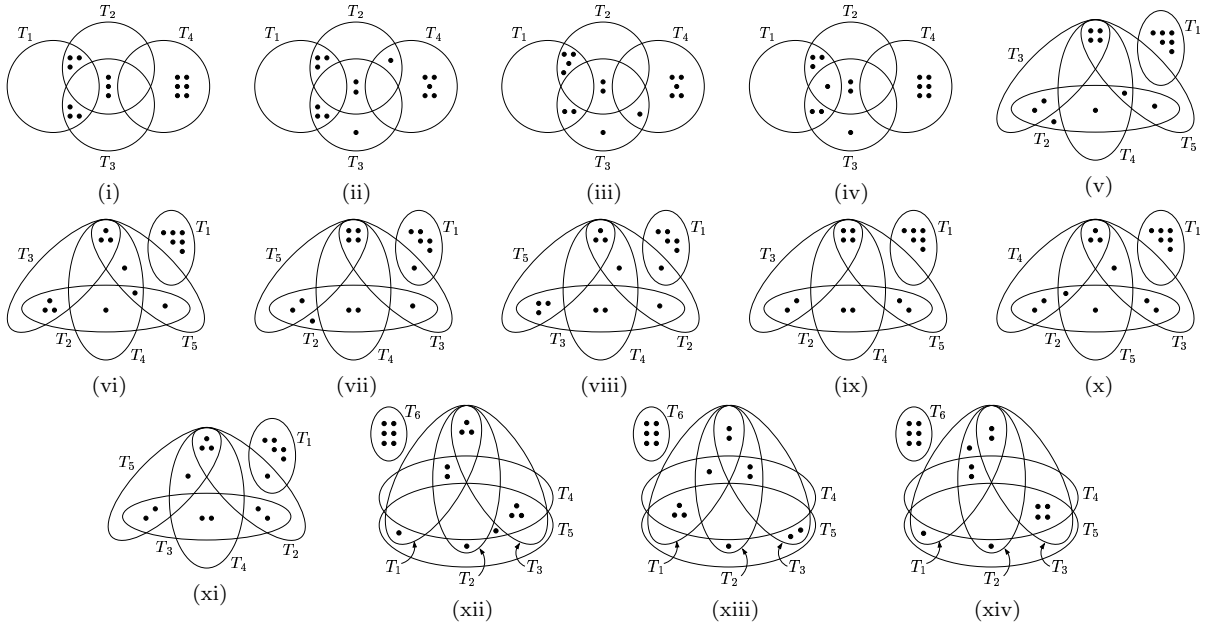
The computational complexity of the lottery tree characterisation procedure (described in §6.1.1) quickly becomes prohibitive, even for small playing set cardinalities. At every level (from level  $\ell$ , say) in the lottery tree, an additional  $2^\ell$  elements are added to the  $\vec{X}$ -vector representation (of the playing set  $\mathcal{L}$ ). This exponential increase in the number of elements to search through in order to determine all possible overlappings of an additional  $n$ -set added to  $\mathcal{L}$ , incurs an exponential amount of additional computation time at each new level of the lottery tree. One resolution to this computational increase may be to investigate the following alternative enumeration method.

Consider a bipartite graph  $\aleph_{\mathcal{L}}$  (referred to as a **nauty** graph) on  $(m + |\mathcal{L}|)$  vertices that is partitioned into two sets: the first partition contains (as labelled vertices) the elements of  $\mathcal{U}_m = \{1, \dots, m\}$ , while the second partition contains the elements of the playing set  $\mathcal{L} = \{\mathcal{T}_1, \dots, \mathcal{T}_{|\mathcal{L}|}\}$ . There exists an edge between  $i \in \mathcal{U}_m$  and  $\mathcal{T}_j \in \mathcal{L}$  if  $i \in \mathcal{T}_j$  for all  $j \in \{1, \dots, |\mathcal{L}|\}$ . Given any two playing sets  $\mathcal{L}$  and  $\mathcal{L}'$  (or equivalently  $\aleph_{\mathcal{L}}$  and  $\aleph_{\mathcal{L}'}$ ), a computer program called **nauty** [160] is able to distinguish between two structurally similar playing sets. **nauty** uses a search tree technique (as described in [158]) to determine the automorphism group of any given graph and also assists in the process of graph isomorphism testing by being able to produce canonically labelled isomorphisms of a graph<sup>4</sup>. To clarify the above mentioned setup and approach using **nauty**, the following example is presented.

**Example 6.7 (continuation of Example 6.3)** Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 6.3, together with three  $L_1(7, 3; 2)$ -sets  $\mathcal{L} = \{\{1, 2, 3\}, \{1, 5, 7\}, \{2, 5, 7\}, \{3, 4, 6\}\}$ ,  $\mathcal{L}' = \{\{1, 3, 5\}, \{1, 6, 7\}, \{2, 3, 4\}, \{2, 4, 5\}\}$  and  $\mathcal{L}'' = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{5, 6, 7\}\}$ . The associated **nauty** graph representations ( $\aleph_{\mathcal{L}}$ ,  $\aleph_{\mathcal{L}'}$  and  $\aleph_{\mathcal{L}''}$ ) of these  $L_1(7, 3; 2)$ -sets are given in Figure 6.7(a)–(c). The canonically relabelled graphs for  $\aleph_{\mathcal{L}}$  and  $\aleph_{\mathcal{L}'}$  were found to be the same (hence  $\aleph_{\mathcal{L}} \simeq \aleph_{\mathcal{L}'}$ , where the set of isomorphism pairs  $\{(v, \phi(v)) : v \in V(\aleph_{\mathcal{L}}), \phi(v) \in V(\aleph_{\mathcal{L}'})\}$  is given by  $\{(1, 3), (2, 5), (3, 1), (4, 7), (5, 4), (6, 6), (7, 1), (T_1, T'_1), (T_2, T'_2), (T_3, T'_3), (T_4, T'_4)\}$ ), although different from that determined for  $\aleph_{\mathcal{L}''}$ . Hence, it is

<sup>4</sup>Informally, the canonical labelling of a graph  $\mathcal{G}$  is a unique vertex labelling of  $\mathcal{G}$  that is invariant with respect to isomorphism (i.e., ordering of vertices and edges of  $\mathcal{G}$ ). As a result, two graphs exhibit the same canonical labelling if and only if they are isomorphic. See the **nauty** User's Guide [160] for a proof of this result.





(a) (i)–(iv) The only four  $L_1(15, 6; 3)$ –set overlapping structures for  $\langle 15, 6; 3 \rangle$ , corresponding to the result  $\eta_1(15, 6; 3) = 4$ ;  
(v)–(xi) The only seven  $L_1(16, 6; 3)$ –set overlapping structures for  $\langle 16, 6; 3 \rangle$ , corresponding to the result  $\eta_1(16, 6; 3) = 7$ ;  
(xii)–(xiv) The only three  $L_1(17, 6; 3)$ –set overlapping structures for  $\langle 17, 6; 3 \rangle$ , corresponding to the result  $\eta_1(17, 6; 3) = 3$

Branch in lottery tree	Execution time (dd:hh:mm:ss)		
	$\langle 15, 6; 3 \rangle$	$\langle 16, 6; 3 \rangle$	$\langle 17, 6; 3 \rangle$
1	00:00:00:01	00:00:01:16	09:23:49:21
2	00:00:00:01	00:00:04:12	37:18:24:19
3	00:00:00:00	00:00:01:44	12:15:37:44
4	00:00:00:00	00:00:00:02	00:00:51:49
5	00:00:00:00	00:00:00:00	00:00:00:01
6	00:00:00:00	00:00:00:00	00:00:00:01
	00:00:00:02	00:00:07:14	60:10:43:15

(b) Execution time for branch generation and traversal of the lottery tree for  $\langle 15, 6; 3 \rangle$ ,  $\langle 16, 6; 3 \rangle$  and  $\langle 17, 6; 3 \rangle$ , when pruning rules are enforced

Level in lottery tree	Number of nodes					
	$\langle 15, 6; 3 \rangle$		$\langle 16, 6; 3 \rangle$		$\langle 17, 6; 3 \rangle$	
	Pruning rules enforced	Pruning rules restrained	Pruning rules enforced	Pruning rules restrained	Pruning rules enforced	Pruning rules restrained
1	1	1	1	1	1	1
2	6	6	6	6	6	6
3	68	238	71	245	72	248
4	2384	12949	2643	14343	2795	15152
5	–	–	319813	1965928	368800	2269307
6	–	–	–	–	129820402	831947054

(c) The number of nodes on each level of the lottery tree for  $\langle 15, 6; 3 \rangle$ ,  $\langle 16, 6; 3 \rangle$  and  $\langle 17, 6; 3 \rangle$ , when pruning rules are either enforced (columns 2, 4 and 6) or relaxed (columns 3, 5 and 7)

Figure 6.4: Algorithmic statistics involved with the characterisation of the four  $L_1(15, 6; 3)$ –set, seven  $L_1(16, 6; 3)$ –set and three  $L_1(17, 6; 3)$ –set structures for the lotteries  $\langle 15, 6; 3 \rangle$ ,  $\langle 16, 6; 3 \rangle$  and  $\langle 17, 6; 3 \rangle$ , respectively (the largest case that could be investigated in this dissertation being  $\langle 17, 6; 3 \rangle$ ) using the characterisation algorithm (Algorithm 8). Execution of the algorithm was performed on an AMD Thunderbird 1.4 GHz processor with 512 MB memory.

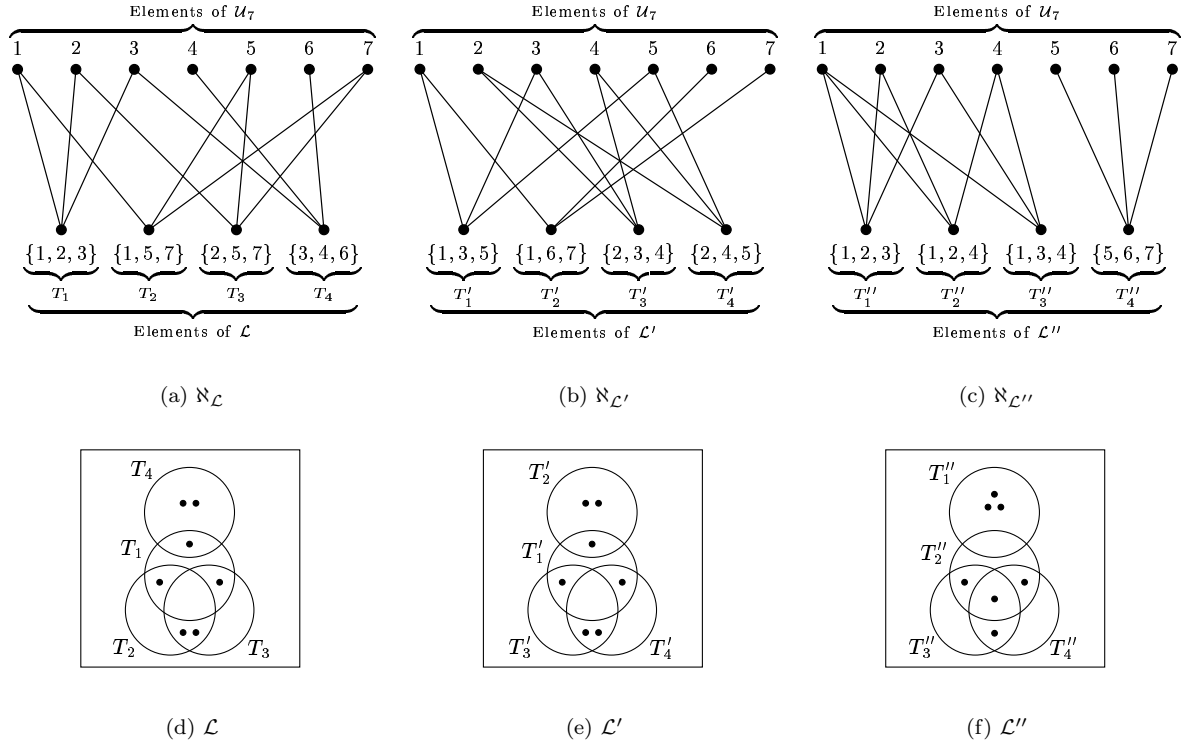


Figure 6.5: Graph representations of the  $L_1(7, 3; 2)$ -sets (a)  $\mathcal{L} = \{\{1, 2, 3\}, \{1, 5, 7\}, \{2, 5, 7\}, \{3, 4, 6\}\}$  (given by  $\aleph_{\mathcal{L}}$ ), (b)  $\mathcal{L}' = \{\{1, 3, 5\}, \{1, 6, 7\}, \{2, 3, 4\}, \{2, 4, 5\}\}$  (given by  $\aleph_{\mathcal{L}'}$ ) and (c)  $\mathcal{L}'' = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{5, 6, 7\}\}$  (given by  $\aleph_{\mathcal{L}''}$ ). With the use of *nauty*, the  $L_1(7, 3; 2)$ -sets  $\mathcal{L}$  and  $\mathcal{L}'$  were determined to be structurally similar (hence  $\aleph_{\mathcal{L}} \simeq \aleph_{\mathcal{L}'}$ ), while the  $L_1(7, 3; 2)$ -set  $\mathcal{L}''$  is structurally different from both  $\mathcal{L}$  and  $\mathcal{L}'$  (hence  $\aleph_{\mathcal{L}''} \not\cong \aleph_{\mathcal{L}}$  and  $\aleph_{\mathcal{L}''} \not\cong \aleph_{\mathcal{L}'}$ ). For comparison, subfigure (d) [(e), (f)] captures the overlapping structure of  $\mathcal{L}$  [ $\mathcal{L}'$ ,  $\mathcal{L}''$ ] with  $\vec{X}^{(4)} = (0, 0, 0, 1, 0, 1, 2, 0, 2, 1, 0, 0, 0, 0, 0, 0)$  [ $\vec{X}^{(4)} = (0, 0, 2, 1, 0, 1, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0)$ ,  $\vec{X}^{(4)} = (0, 0, 0, 1, 0, 1, 1, 1, 3, 0, 0, 0, 0, 0, 0, 0)$ ], as described in §3.2.

possible to conclude that the  $L_1(7, 3; 2)$ -sets  $\mathcal{L}$  and  $\mathcal{L}'$  are structurally similar, while  $\mathcal{L}''$  is structurally different from both  $\mathcal{L}$  and  $\mathcal{L}'$ . For completeness and comparison, Figure 6.7(d)–(f) captures the overlapping structure of the three  $L_1(7, 3; 2)$ -sets. ■

The number of times that any element of  $\mathcal{U}_m$  is used per playing set, on average, when considering all possible playing sets of cardinality  $\ell$  for  $\langle m, n; k \rangle$ , is  $\frac{\ell \times n}{m}$ . It is therefore reasonable to assume that some element in  $\mathcal{U}_m$  (say  $m$ , without loss of generality) is used at most  $\lfloor \frac{\ell \times n}{m} \rfloor$  times. By letting  $\ell' = \ell - \lfloor \frac{\ell \times n}{m} \rfloor$ , any playing set of cardinality  $\ell$  in  $\langle m, n; k \rangle$  may be constructed from (i) a playing set of cardinality  $\ell'$  containing  $n$ -sets from  $\Phi(\mathcal{U}_{m-1}, n)$ , together with (ii) a playing set of cardinality  $(\ell - \ell') = \lfloor \frac{\ell \times n}{m} \rfloor$  containing  $n$ -sets from  $\Phi(\mathcal{U}_m, n)$ . By using a parent–child search tree approach (similar to that of the lottery tree enumeration method in §6.1.1) with the use of *nauty* to eliminate possible *nauty* graph isomorphisms, it is possible to generate all structurally different playing (and hence also incomplete lottery) sets of cardinality  $\ell$  for  $\langle m, n; k \rangle$ , by using the following two–phase construction method to construct a so–called *nauty* tree:

**Phase 1:** Every node up to level  $i \leq \ell'$  of the *nauty* tree represents a *nauty* graph  $\aleph_{\mathcal{L}^{(i)}}^{(1)}$  of some playing set  $\mathcal{L}^{(i)} = \{T_1, T_2, \dots, T_i\}$  in  $\langle m-1, n; k \rangle$ , obtained from level  $i-1$  by adding the  $i$ -th  $n$ -set,  $T_i$ , in all possible non-isomorphic ways to  $\aleph_{\mathcal{L}^{(i-1)}}^{(1)}$  (where  $\mathcal{L}^{(i-1)} = \{T_1, T_2, \dots, T_{i-1}\}$ );

**Phase 2:** Every node on level  $\ell' < j \leq \ell$  of the *nauty* tree represents a *nauty* graph  $\aleph_{\mathcal{L}^{(j)}}^{(2)}$  of some playing set  $\mathcal{L}^{(j)} = \mathcal{L}^{(\ell')} \cup \{T_{\ell'+1}, T_{\ell'+2}, \dots, T_j\}$  in  $\langle m, n; k \rangle$ , obtained from level  $j-1$  by adding the  $j$ -th  $n$ -

set,  $T_j$ , in all possible non-isomorphic ways to  $\aleph_{\mathcal{L}^{(j-1)}}^{(2)}$  (where  $\mathcal{L}^{(j-1)} = \mathcal{L}^{(\ell')} \cup \{T_{\ell'+1}, T_{\ell'+2}, \dots, T_{j-1}\}$  and  $\mathcal{L}^{(\ell')}$  is the root of the subtree starting at level  $\ell'$ ).

No overlapping structure is captured by the **nauty** representation of a playing set, and hence the determination of the probability of winning a  $k$ -prize in  $\langle m, n; k \rangle$  is manually performed by searching through all  $\binom{m}{n}$   $n$ -sets in  $\Phi(\mathcal{U}_m, n)$ . Although it is possible to construct an overlapping structure  $\vec{X}^{(\ell)}$  from any given **nauty** graph  $\aleph_{\mathcal{L}^{(\ell)}}^{(2)}$  for the determination of  $\psi_{\vec{X}^{(\ell)}}$  (refer back to Example 3.2), this is, however, avoided, due to the computational intensity of manipulating vectors with exponential growth (as mentioned before). Instead, the unambiguous notation  $\psi_{\mathcal{L}^{(\ell)}}$  is used to represent the resource utilised by the set  $\mathcal{L}^{(\ell)}$  of the **nauty** graph  $\aleph_{\mathcal{L}^{(\ell)}}^{(2)}$ .

The following example shows a fragmented construction of the **nauty** tree for  $\langle 7, 3; 2 \rangle$ , clarifying the above mentioned definitions and concepts.

**Example 6.8 (continuation of Example 6.7)** *Reconsider the lottery  $\langle 7, 3; 2 \rangle$  of Example 6.7. Using the above mentioned enumeration method the **nauty** tree up to level three was constructed for  $\langle 7, 3; 2 \rangle$ . All possible non-isomorphic **nauty** graphs of order 9 (i.e.,  $\ell' = 2$  with  $\mathcal{L}^{(1)}$ ), corresponding to the **nauty** graph  $\aleph_{\mathcal{L}^{(1)}}^{(1)}$ , consisting of the single 3-set  $\{1, 2, 3\} \in \Phi(\mathcal{U}_6, 3)$  were generated in Phase 1. Figure 6.6(a)–(c) contains the only three non-isomorphic **nauty** graphs  $\aleph_{\mathcal{L}^{(2)}}^{(1)}$  on level two of the **nauty** tree. This was followed by the generation of all possible non-isomorphic **nauty** graphs of order 10 by adding one 3-set in  $\Phi(\mathcal{U}_7, 3)$  to  $\mathcal{L}^{(2)}$  in all possible ways (up to graph isomorphism which was determined by **nauty**) in Phase 2. Figure 6.6(d)–(e) contains the only ten non-isomorphic **nauty** graphs  $\aleph_{\mathcal{L}^{(3)}}^{(2)}$  on level three of the **nauty** tree (no additional non-isomorphic **nauty** graphs were obtained down the subtree with root in Figure 6.6(c)). **nauty** discarded 30 [49] **nauty** graphs  $\aleph_{\mathcal{L}^{(2)}}^{(1)}$  [ $\aleph_{\mathcal{L}^{(3)}}^{(2)}$ ] (being isomorphic to another node on that specific level) on level two [three] of the **nauty** tree for  $\langle 7, 3; 2 \rangle$ .  $\mathcal{L}^{(3)} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{5, 6, 7\}\}$  was the only set of cardinality 3 that yielded a maximum resource utilisation of  $\frac{32}{35}$ , implying that  $L_{\frac{32}{35}}(7, 3; 2) = 3$ ,  $\Psi_3(7, 3; 2) = \frac{32}{35}$  and  $\eta_{\frac{32}{35}}(7, 3; 2) = 1$ . Furthermore, the values of  $\eta_\psi(7, 3; 2)$  ( $\frac{26}{35} < \psi \leq \frac{32}{35}$ ) in (6.2) may be verified by only considering the bold emboxed nodes in Figure 6.6(d)–(e). ■*

Apart from the fact that **nauty** is capable of removing all isomorphic graphs  $\aleph_{\mathcal{L}^{(\ell)}}^{(\bullet)}$  (in both phases of the **nauty** tree construction), the same pruning rules as described in §6.1.1 may be used. This, however, requires keeping additional non-**nauty** data containing the values for  $x_{(00\dots 0)_2}$ ,  $x_{(00\dots 1)_2}$ ,  $\dots$ ,  $x_{(10\dots 0)_2}$  in  $\vec{X}^{(\ell)}$ . Of course, trade-offs exist between using only **nauty** as opposed to calculating some values of  $\vec{X}^{(\ell)}$ .

## 6.2 General properties of the parameter $\eta_\psi(m, n; k)$

Before proceeding to find values for the parameter  $\eta_\psi(m, n; k)$ , its boundedness (and hence existence) is established.

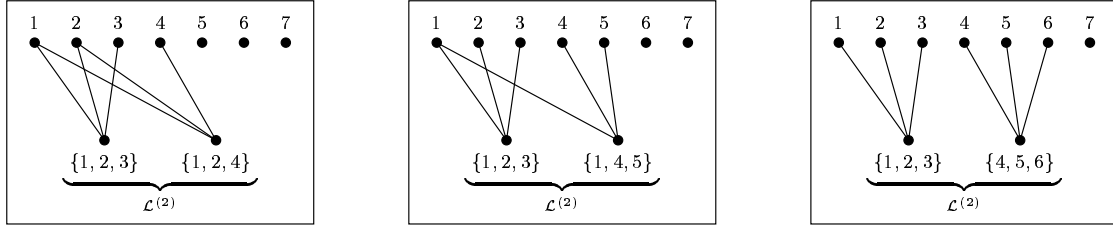
**Theorem 6.1 (Boundedness of  $\eta_\psi(m, n; k)$ )** *The parameter  $\eta_\psi(m, n; k)$  is finite. In fact,  $\eta_\psi(m, n; k) = 1$  if  $L_\psi(m, n; k) = 1$ . Furthermore,*

$$1 \leq \eta_\psi(m, n; k) \leq n \binom{m}{n}^{L_\psi(m, n; k) - 2}$$

for all  $1 \leq k \leq n \leq m$  and all  $0 < \psi \leq 1$ , if  $L_\psi(m, n; k) \geq 2$ .

### Proof

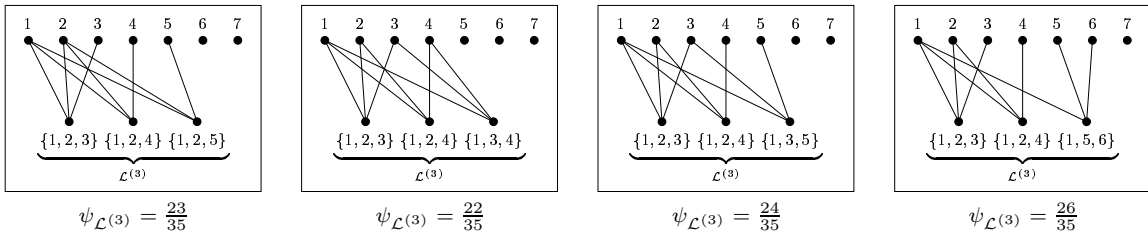
It is clear that there is only one way in which to choose a single  $n$ -set from  $\mathcal{U}_m$ , up to structure isomorphism. Hence  $\eta_\psi(m, n; k) = 1$  when  $L_\psi(m, n; k) = 1$ . Therefore suppose that  $L_\psi(m, n; k) \geq 2$ . In this case an upper bound on  $\eta_\psi(m, n; k)$  is obtained by counting the number of structurally different playing sets,  $\mathcal{L}$ , of cardinality  $L_\psi(m, n; k)$  for  $\langle m, n; k \rangle$  (not all of these playing sets will be incomplete lottery sets). There is only one way in which to choose the first  $n$ -set from  $\mathcal{U}_m$  in the construction of  $\mathcal{L}$ ,



(a) First (lexicographic) node  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$  on level 2 of the nauty tree for  $\langle 7, 3; 2 \rangle$  (Phase 1)

(b) Second (lexicographic) node  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$  on level 2 of the nauty tree for  $\langle 7, 3; 2 \rangle$  (Phase 1)

(c) Third (lexicographic) and last node  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$  on level 2 of the nauty tree for  $\langle 7, 3; 2 \rangle$  (Phase 1)

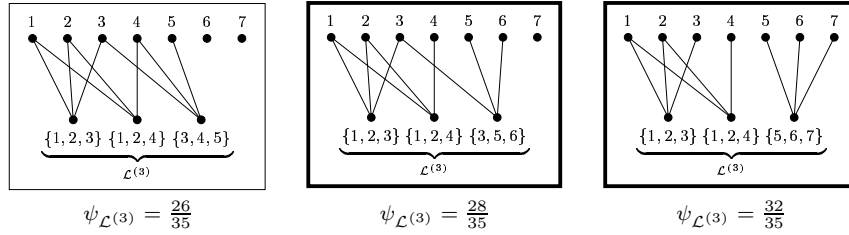


$$\psi_{\mathcal{L}^{(3)}} = \frac{23}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{22}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{24}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{26}{35}$$

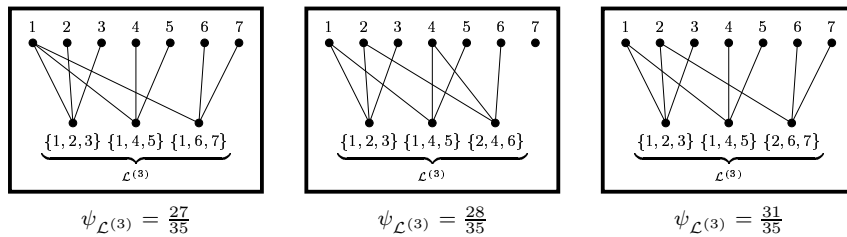


$$\psi_{\mathcal{L}^{(3)}} = \frac{26}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{28}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{32}{35}$$

(d) All 7 non-isomorphic nauty graphs  $\aleph_{\mathcal{L}^{(3)}}^{(2)}$ , obtained from the nauty subtree with root  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$ , where  $\mathcal{L}^{(2)} = \{\{1, 2, 3\}, \{1, 2, 4\}\}$  (Phase 2)



$$\psi_{\mathcal{L}^{(3)}} = \frac{27}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{28}{35}$$

$$\psi_{\mathcal{L}^{(3)}} = \frac{31}{35}$$

(e) All 3 non-isomorphic nauty graphs  $\aleph_{\mathcal{L}^{(3)}}^{(2)}$ , obtained from the nauty subtree with root  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$ , where  $\mathcal{L}^{(2)} = \{\{1, 2, 3\}, \{1, 4, 5\}\}$  (Phase 2)

Figure 6.6: Fragmented nauty tree for  $\langle 7, 3; 2 \rangle$  up to level three with root  $\aleph_{\mathcal{L}^{(1)}}^{(1)}$ , where  $\mathcal{L}^{(1)} = \{\{1, 2, 3\}\}$  (Example 6.8). Subfigure (a) [(b), (c)] contains the first [second, third (last)] non-isomorphic node  $\aleph_{\mathcal{L}^{(2)}}^{(2)}$  on level two of the nauty tree. All possible non-isomorphic nauty graphs  $\aleph_{\mathcal{L}^{(3)}}^{(2)}$  on level three of the nauty tree (obtained from the subtree with root shown in subfigure (a)–(b)) is shown in subfigure (d)–(e). No additional non-isomorphic nauty graphs were obtained from the subtree with root in subfigure (c). Bold boxed nodes show that  $\psi_{\mathcal{L}^{(3)}} > \Psi_2(7, 3; 2) = \frac{26}{35}$  and were used to verify that  $L_\psi(7, 3; 2) = 3$  ( $\frac{26}{35} < \psi \leq \frac{32}{35}$ ),  $\Psi_3(7, 3; 2) = \frac{32}{35}$  and the values of  $\eta_\psi(7, 3; 2)$  ( $\frac{26}{35} < \psi \leq \frac{32}{35}$ ) in (6.2).

up to structure isomorphism. The second  $n$ -set in  $\mathcal{L}$  may be chosen in at most  $n$  structurally different ways (by either having  $\max\{2n - m, 0\}$ ,  $\max\{2n - m, 0\} + 1, \dots, n - 2$  or  $n - 1$  elements in common with the first  $n$ -set). All  $L_\psi(m, n; k) - 2$  remaining  $n$ -sets in  $\mathcal{L}$  may be chosen in at most  $\binom{m}{n}$  structurally different ways, yielding the desired result. ■

The following result is stated (without proof) as a direct consequence of Theorem 3.3(e) and of the implicit relationship between  $\langle m, n; k \rangle$ ,  $G\langle m, n; k \rangle$  and  $\eta_\psi(m, n; k)$ .

**Corollary 6.1**  $\eta_\psi(m, n; k) = \eta_\psi(m, m - n; m + k - 2n)$  for all  $1 \leq k \leq n < m$  such that  $m + k > 2n$  and  $0 < \psi \leq 1$ .

In order to establish growth properties for the parameter  $\eta_\psi(m, n; k)$  with respect to its arguments, we require the notion of a so-called *jump sequence* for each of the arguments. Informally, a jump sequence for one of the parameters  $m$ ,  $n$ ,  $k$  or  $\psi$  is an increasing sequence of values for this parameter at which the value of the incomplete lottery number  $L_\psi(m, n; k)$  changes as this parameter increases, if the other three parameters remain constant. More formally, define an  $m$ -jump sequence as the increasing sequence  $n = m_1^{(n, k, \psi)}, m_2^{(n, k, \psi)}, m_3^{(n, k, \psi)}, \dots$  of those integers  $m_{i+1}^{(n, k, \psi)}$  satisfying

$$L_\psi(m_{i+1}^{(n, k, \psi)}, n; k) > L_\psi(m_i^{(n, k, \psi)}, n; k),$$

but for which

$$L_\psi(m_{i+1}^{(n, k, \psi)} - 1, n; k) = L_\psi(m_i^{(n, k, \psi)}, n; k)$$

in cases where  $m_i^{(n, k, \psi)}$  and  $m_{i+1}^{(n, k, \psi)}$  are non-adjacent integers. The notions of an  $n$ -jump sequence, a  $k$ -jump sequence and a  $\psi$ -jump sequence may be defined similarly. However, the  $\psi$ -jump sequence is of course an increasing sequence of real numbers bounded from below by zero and from above by 1, rather than a sequence of integers. It will be shown that, whilst the  $m$ -jump sequence is infinite, the  $n$ -jump sequence, the  $k$ -jump sequence and the  $\psi$ -jump sequence are all finite, since  $1 \leq k \leq n \leq m$  and  $0 < \psi \leq 1$ , where  $m$  is fixed. Interesting saw-tooth growth patterns for the parameter  $\eta_\psi(m, n; k)$  with respect to each of its four arguments will also be established, where the respective jump sequences separate saw-teeth, as illustrated in Figure 6.7 for the lottery  $\langle 15, 6; 3 \rangle$ , where  $\psi$  varies over the range  $(0, 1]$ . Graphs of similar saw-teeth growth patterns, but where saw-teeth are separated by values of the  $m$ -jump sequence instead of values of the  $\psi$ -jump sequence, are presented in Figure 6.8.

Note that, for  $n = k = 1$ , the  $m$ -jump sequence is simply the sequence of all positive multiples of  $\lceil \frac{1}{\psi} \rceil$  since  $L_\psi(m, 1; 1) = \lceil \psi m \rceil$  (see Theorem 2.3(b)). On the other hand, for  $n = k$  and  $\psi = 1$ , the  $m$ -jump sequence is the sequence of all integers exceeding  $n - 1$ , since  $L_1(m, n; n) = \binom{m}{n}$  (see Theorem 2.3(b)). These two cases represent two extreme growth patterns of the characterisation sequence in  $\{\eta_\psi(m, n; k)\}_{m=n}^\infty$  with respect to increasing  $m$ : In the former case, the saw-teeth all have maximum width (namely  $\lceil \frac{1}{\psi} \rceil$ ), the saw-teeth are as blunt as possible (as will be shown), the  $m$ -jump sequence never reaches maximum density and the growth in saw-tooth height is positive and constant. In the latter case, the saw-teeth all have minimum width (namely 1, in other words the maximum density is achieved by the  $m$ -jump sequence right from the start), the saw-teeth have become so sharp (infinitely sharp) that the individual teeth themselves have become indistinguishable, and the growth in saw-tooth height is positive and binomial.

Before determining exact values of  $\eta_\psi(m, n; k)$ , we first explore some properties of the jump sequences. In particular, the question of the boundedness of the jump sequences (the number of saw-teeth) is explored for all parameters  $m$ ,  $n$ ,  $k$  and  $\psi$ , whereafter a derivation of asymptotic lower bounds on  $L_\psi(m, n; k)$  follows for  $k = 1, 2$ . Finally, an investigation into the density of the jump sequences (the widths of the saw-teeth) as well as the relevant characterisation parameter growth properties (the shapes of the saw-teeth) is launched.

We first establish the following boundedness properties of the jump sequences.

**Theorem 6.2 (Properties of the jump sequences)**

(a) The  $m$ -jump sequence is infinite, for all  $1 \leq k \leq n$  and  $0 < \psi \leq 1$ .

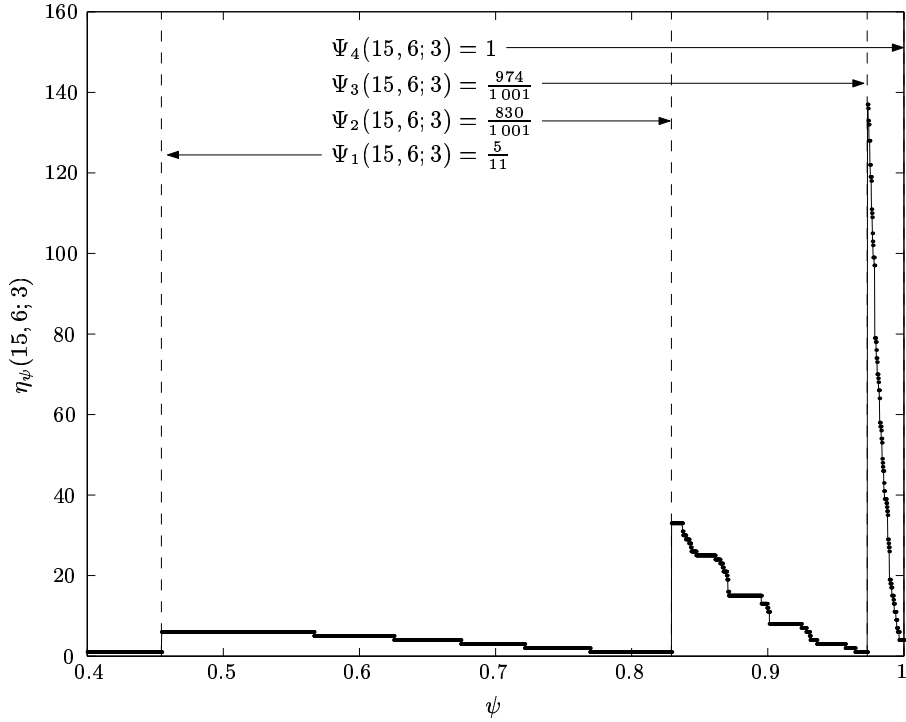


Figure 6.7: The resource utilisation characterisation number  $\eta_\psi(15, 6; 3)$ , as a function of  $0.4 \leq \psi \leq 1$ , in increments of  $1/\binom{15}{5} = 1/5005$  ( $\eta_\psi(15, 6; 3) = 1$  for all  $0 < \psi < 0.4$ ).

- (b) The  $n$ -jump sequence is finite, for all  $1 \leq k \leq m$  and  $0 < \psi \leq 1$ .
- (c) The  $k$ -jump sequence is finite, for all  $1 \leq n \leq m$  and  $0 < \psi \leq 1$ .
- (d) The  $\psi$ -jump sequence is finite, for all  $1 \leq k \leq n \leq m$ .

### Proof

(a) By contradiction. First, suppose the  $m$ -jump sequence is finite for some values of  $1 \leq k \leq n$  and  $\psi = 1$ . Then  $L_1(m, n; k)$  must be bounded from above as  $m \rightarrow \infty$ . According to Corollary 2.2(d) and Theorem 2.2(c), we have that  $L_1(m, n; k) \geq L_1(m, n; 1) \geq \lfloor \frac{m}{n} \rfloor$ , which grows without bound (as a function of  $m$ , for fixed  $1 \leq k \leq n$ ). This contradicts the assumption of boundedness of  $L_1(m, n; k)$  as  $m \rightarrow \infty$ , hence the result follows.

Now suppose the  $m$ -jump sequence is finite for some values of  $1 \leq k \leq n$  and  $0 < \psi < 1$ . Then  $L_\psi(m, n; k)$  must be bounded from above as  $m \rightarrow \infty$ . Recalling Theorem 2.3(g), the incomplete lottery number  $L_\psi(m, n; 1) = \ell$  is the smallest integer satisfying

$$\binom{m - \ell n}{n} \leq \binom{m}{n} (1 - \psi).$$

Now,  $\ell$  is bounded below by any  $\tilde{\ell} \in \mathbb{N}$  satisfying

$$\frac{(m - \tilde{\ell} n - n + 1)^n}{n!} \leq \binom{m}{n} (1 - \psi). \quad (6.4)$$

The value

$$\tilde{\ell}^* = \underbrace{\frac{m}{n}}_{\text{linear term}} - \underbrace{\frac{\log_n(n! \binom{m}{n} (1 - \psi))}{n}}_{\text{asymptotically sub-linear term}} + \underbrace{\left(\frac{1}{n} - 1\right)}_{\text{constant term}} \quad (6.5)$$

yields equality in (6.4) and hence  $\tilde{\ell}^* \leq \tilde{\ell} \leq \ell$ . However,

$$\lim_{m \rightarrow \infty} \frac{\log_n(n! \binom{m}{n} (1 - \psi)) / n}{m/n} = \lim_{m \rightarrow \infty} \left( \frac{\Gamma'(m+1)}{m!} - \frac{\Gamma'(m-n+1)}{(m-n)!} \right) = 0 \quad (6.6)$$

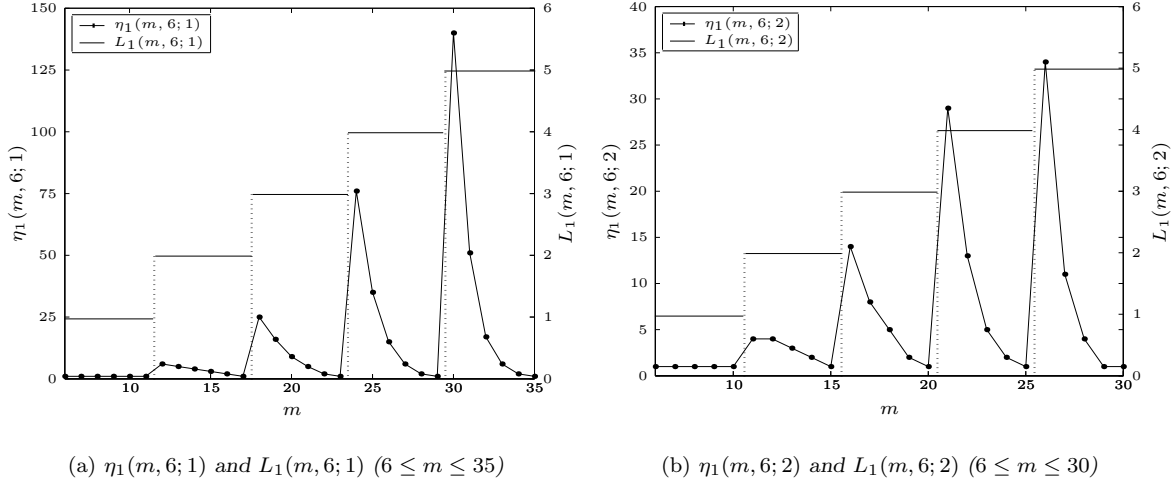


Figure 6.8: The number of different  $L_1(m, 6; k)$ -set structures  $\eta_1(m, 6; k)$  for  $\langle m, 6; k \rangle$  where (a)  $6 \leq m \leq 35$  and  $k = 1$ , (b)  $6 \leq m \leq 30$  and  $k = 2$ . In (a) the  $m$ -jump sequence  $m_i^{(6,1,1)}$  is given by 12, 18, 24, 30, 36. In (b) the  $m$ -jump sequence  $m_i^{(6,2,1)}$  is given by 11, 16, 21, 26, 31.

by virtue of l'Hospital's rule, where  $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$  is the well-known gamma function. Therefore the lower bound  $\tilde{\ell}^*$  in the inequality chain  $\tilde{\ell}^* \leq \tilde{\ell} \leq \ell = L_\psi(m, n; 1) \leq L_\psi(m, n; k)$  grows without bound (as a function of  $m$ , for fixed  $1 \leq k \leq n$  and fixed  $0 < \psi < 1$ ), contradicting the assumption of boundedness of  $L_\psi(m, n; k)$  as  $m \rightarrow \infty$ .

(b)–(c) Since  $1 \leq k \leq n \leq m$ ,  $m$  is fixed and  $0 < \psi \leq 1$  for any  $n$ -jump sequence or any  $k$ -jump sequence, these (strictly increasing) sequences of integers must be finite.

(d) The  $\psi$ -jump sequence is given by  $\{\Psi_\ell(m, n; k)\}_{\ell=1}^{L_\psi(m, n; k)}$  for all  $1 \leq k \leq n \leq m$ . This sequence is finite by virtue of the boundedness of  $L_\psi(m, n; k)$  in Theorem 2.2. ■

Before the densities of the jump sequences are investigated, the following intermediate result is established, for which some additional notation is required. Suppose  $\mathcal{F}(m)$  and  $\mathcal{G}(m)$  are functions of  $m$ . Then  $\mathcal{F}(m)$  is said to grow asymptotically at least as fast as  $\mathcal{G}(m)$  as  $m \rightarrow \infty$ , denoted  $\mathcal{F}(m) = \Omega(\mathcal{G}(m))$ , if there exist constants  $c > 0$  and  $m_0 \in \mathbb{N}$  such that  $0 \leq c\mathcal{G}(m) \leq \mathcal{F}(m)$  for all  $m \geq m_0$ .

**Proposition 6.1 (Asymptotic lower bounds on the [in]complete lottery number)**

(a)  $L_\psi(m, n; 1) = \Omega(m)$  for any fixed  $n \geq 1$  and fixed  $0 < \psi \leq 1$ .

(b)  $L_1(m, n; 2) = \Omega(m^2)$  for any fixed  $n \geq 2$ .

**Proof**

(a) First, consider the (special) case of the complete lottery problem where  $\psi = 1$ . In this case it is known that  $L_1(m, n; 1) = \lfloor \frac{m}{n} \rfloor = \Omega(m)$  (see Corollary 2.2(e)).

Now, consider the (general) case of the incomplete lottery problem where  $0 < \psi < 1$ . From (6.6) it follows that for any  $0 < \epsilon \leq 1$ , there exists a  $\delta > 0$ , such that

$$\log_n \left( n! \binom{m}{n} (1 - \psi) \right) \leq \epsilon m,$$

for all  $m \geq \delta$ . Therefore,

$$m - \log_n \left( n! \binom{m}{n} (1 - \psi) \right) \geq (1 - \epsilon) m \geq 0$$

for all  $m \geq \delta$ . Consequently, from (6.5), we have that  $n\tilde{\ell}^* + n - 1 = \Omega(m)$ , or equivalently that  $L_\psi(m, n; 1) = \tilde{\ell}^* = \Omega(m)$  for any fixed  $n$  and fixed  $0 < \psi < 1$ .

(b) In 1964 Hanani, *et al.* [99] proved that

$$L_1(m, n; 2) \geq \frac{m(m-n+1)}{n(n-1)^2}$$

for all  $m \geq n$ , and additionally showed that this bound is asymptotically best possible, in the sense that

$$\lim_{m \rightarrow \infty} L_1(m, n; 2) \frac{n(n-1)^2}{m(m-n+1)} = 1.$$

Hence  $L_1(m, n; 2) = \Omega(m^2)$ . ■

The following two results are direct consequences of Proposition 6.1 and show that the saw-teeth observed in the growth pattern of  $\eta_\psi(m, n; k)$  do not grow wider as  $m \rightarrow \infty$ .

**Corollary 6.2 (Density of the  $m$ -jump sequence for the [in]complete lottery problem)**

(a) The density of the  $m$ -jump sequence is asymptotically non-decreasing for all  $1 \leq n$  and  $0 < \psi \leq 1$  in the lottery  $\langle m, n; 1 \rangle$ , in the sense that there exists an  $i^{(n, k, \psi)} \in \mathbb{N}$  such that  $m_{i+1}^{(n, k, \psi)} - m_i^{(n, k, \psi)} \leq m_i^{(n, k, \psi)} - m_{i-1}^{(n, k, \psi)}$  for all  $i \geq i^{(n, k, \psi)}$ .

(b) The density of the  $m$ -jump sequence is asymptotically increasing for all  $2 \leq k \leq n$  in the lottery  $\langle m, n; k \rangle$ , in the sense that there exists an  $i^{(n, k, 1)} \in \mathbb{N}$  such that  $m_{i+1}^{(n, k, 1)} = m_i^{(n, k, 1)} + 1$  for all  $i \geq i^{(n, k, 1)}$ . After this point  $i^{(n, k, 1)}$  the  $m$ -jump sequence is said to have reached maximum density.

**Proof**

(a) From Proposition 6.1(a) it follows that  $L_\psi(m, n; 1) = \Omega(m)$  for all  $n \geq 1$  and  $0 < \psi \leq 1$  in the lottery  $\langle m, n; 1 \rangle$ . Hence, the result follows.

(b) For all  $2 \leq k \leq n \leq m$  it follows from Proposition 6.1(b) that

$$L_1(m, n; k) \geq L_1(m, n; 2) = \Omega(m^2).$$

Hence the result follows. ■

We are now in a position to consider the growth properties of the characterisation parameter  $\eta_\psi(m, n; k)$  for  $\langle m, n; k \rangle$ , *i.e.*, the shapes of the saw-teeth in the growth pattern of  $\eta_\psi(m, n; k)$ .

**Theorem 6.3 (Growth properties of  $\eta_\psi(m, n; k)$ )**

(a)  $\eta_\psi(m, n; k) \geq \eta_\psi(m, n; k')$  for all  $k_i^{(m, n, \psi)} \leq k \leq k' < k_{i+1}^{(m, n, \psi)}$  and all  $0 < \psi \leq 1$ , where  $1 \leq k \leq k' \leq n \leq m$ , whenever  $k_i^{(m, n, \psi)}$  and  $k_{i+1}^{(m, n, \psi)}$  are non-adjacent integers.

(b)  $\eta_\psi(m, n; k) \geq \eta_{\psi'}(m, n; k)$  for all  $\Psi_i(m, n; k) \leq \psi \leq \psi' < \Psi_{i+1}(m, n; k)$  and all  $i \in \{1, \dots, L_1(m, n; k) - 1\}$ , where  $1 \leq k \leq n \leq m$ .

**Proof**

(a) If  $k_i^{(m, n, \psi)} \leq k \leq k' < k_{i+1}^{(m, n, \psi)}$ , then clearly  $L_\psi(m, n; k) = L_\psi(m, n; k')$ , in which case any  $L_\psi(m, n; k')$ -set for  $\langle m, n; k' \rangle$  is also an  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$ , since  $k' \geq k$ . Hence, the desired result follows.

(b) If  $\Psi_i(m, n; k) \leq \psi \leq \psi' < \Psi_{i+1}(m, n; k)$ , then clearly  $L_\psi(m, n; k) = L_{\psi'}(m, n; k)$ , in which case any  $L_{\psi'}(m, n; k)$ -set for  $\langle m, n; k \rangle$  is also an  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$ , since  $\psi' \geq \psi$ . Hence, the desired result follows. ■

The author was unable to prove the corresponding results for the  $m$ -jump sequence and  $n$ -jump sequence in the general incomplete lottery problem context, but the following is conjectured, based on considerable numerical (albeit circumstantial) evidence (see [39] for a proof of the conjecture for the special case where  $\psi = 1$ ):



**Conjecture 6.1 (Growth properties of  $\eta_\psi(m, n; k)$ )**

(a)  $\eta_\psi(m, n; k) \geq \eta_\psi(m', n; k)$  for all  $m_i^{(n, k, \psi)} \leq m \leq m' < m_{i+1}^{(n, k, \psi)}$  and all  $0 < \psi \leq 1$ , where  $1 \leq k \leq n \leq m \leq m'$ , whenever  $m_i^{(n, k, \psi)}$  and  $m_{i+1}^{(n, k, \psi)}$  are non-adjacent integers.

(b)  $\eta_\psi(m, n; k) \leq \eta_\psi(m, n'; k)$  for all  $n_i^{(m, k, \psi)} \leq n \leq n' < n_{i+1}^{(m, k, \psi)}$  and all  $0 < \psi \leq 1$ , where  $1 \leq k \leq n \leq n' \leq m$ , whenever  $n_i^{(m, k, \psi)}$  and  $n_{i+1}^{(m, k, \psi)}$  are non-adjacent integers. ■

Note that  $\eta_\psi(m, n; k)$  exhibits exactly opposite growth properties to those of  $L_\psi(m, n; k)$  with respect to the arguments  $m, n, k$  and  $\psi$ , as may be seen by comparing the growth properties in Theorem 2.2 and the above theorem and conjecture.

It is also possible to determine explicitly the values of  $\eta_\psi(m, n; k)$  for a few simple combinations of the arguments  $m, n, k$  and  $\psi$ . However, some additional notation is required before it is possible to prove these results. Let  $\zeta_\ell(m, n)$  denote the number of different ways in which a set of  $m$  elements may be covered<sup>5</sup> by  $\ell$  distinct sets, each of cardinality  $n$  (excluding permutations/symmetries). It seems a hard problem to find a closed form formula for  $\zeta_\ell(m, n)$ . Hence, Table 6.3 contains the values for  $\zeta_\ell(m, n)$  where  $\ell \in \{3, 4, 5\}$ , within the ranges for  $m$  and  $n$  necessary for the following results. These values were tabulated using the characterisation techniques described in §6.1, only counting those structures having  $x_0^{(\ell)} = 0$  in their respective  $\vec{X}^{(\ell)}$ -vector representations.

**Theorem 6.4 (Special values of  $\eta_\psi(m, n; k)$ )** Suppose  $1 \leq k \leq n \leq m$ . Then

(a)  $\eta_\psi(m, n; k) = 1$  for all  $0 < \psi \leq 1$ , if  $m + k \leq 2n$ .

(b)  $\eta_\psi(m, n; k) = 1$  for all  $0 < \psi \leq \Psi_1(m, n; k)$ .

(c)  $\eta_\psi(m, n; n) = \sum_{i=0}^{m-n} \zeta_{\lceil \psi \binom{m}{n} \rceil}(m - i, n)$  for all  $0 < \psi \leq 1$ .

**Proof**

(a) It follows by Theorem 3.4(a) that  $L_1(m, n; k) = 1$  if and only if  $m + k \leq 2n$ . Hence  $L_\psi(m, n; k) = 1$  for all  $0 < \psi \leq 1$  if  $m + k \leq 2n$ , by Theorem 2.2(d). Clearly there is only one way, up to structure isomorphism, in which a playing set (consisting of one  $n$ -set) for  $\langle m, n; k \rangle$  may be formed. Hence, the desired result follows.

(b) By definition,  $L_\psi(m, n; k) = 1$  for all  $0 < \psi \leq \Psi_1(m, n; k)$ , yielding the desired result, in the same way as in (a).

(c) It is clear that any  $\lceil \psi \binom{m}{n} \rceil$  distinct  $n$ -subsets from  $\mathcal{U}_m$  form an  $L_\psi(m, n; k)$ -set for  $\langle m, n; n \rangle$ . The number of structurally different  $L_\psi(m, n; k)$ -sets for  $\langle m, n; n \rangle$  is given by  $\sum_{i=0}^{m-n} \zeta_{\lceil \psi \binom{m}{n} \rceil}(m - i, n)$ . ■

In the special case where  $\psi = 1$  the characterisation number  $\eta_1(m, n; k)$  may be determined analytically for  $L_1(m, n; k) = 1, 2, 3$  (recall that these values of the complete lottery number were characterised in Theorem 3.4).

**Theorem 6.5 (Special values of  $\eta_1(m, n; k)$  for small  $L_1(m, n; k)$ )**

(a) When  $L_1(m, n; k) = 1$ ,  $\eta_1(m, n; k) = 1$  for all  $1 \leq k \leq n \leq m$ .

(b) When  $L_1(m, n; k) = 2$ ,

$$\eta_1(m, n; k) = \begin{cases} 3n - 2k + 2 - m, & \text{if } m \geq 2n \\ n - 2k + 2, & \text{if } m < 2n. \end{cases} \quad (6.7)$$

(c) When  $L_1(m, n; k) = 3$ ,

$$\eta_1(m, n; k) = \begin{cases} \sum_{i=0}^{n-3k+2} \zeta_3(m - i, n), & \text{if } m \geq 2n \\ \sum_{i=0}^{5n-3k-2m+2} \zeta_3(m - i, m - n), & \text{if } m < 2n. \end{cases} \quad (6.8)$$

<sup>5</sup>The set  $\mathcal{U}_m$  of  $m$  elements is said to be covered by  $\ell$  distinct subsets  $l_1, \dots, l_\ell$  of  $\mathcal{U}_m$ , each of cardinality  $n$ , if  $\mathcal{U}_m \setminus \cup_{i=1}^{\ell} l_i = \emptyset$ .

		$n$									
		3	4	5	6	7	8	9	10	11	12
$m$	4	1	0	0	0	0	0	0	0	0	0
	5	3	1	0	0	0	0	0	0	0	0
	6	3	4	1	0	0	0	0	0	0	0
	7	3	5	4	1	0	0	0	0	0	0
	8	1	6	6	4	1	0	0	0	0	0
	9	1	4	9	7	4	1	0	0	0	0
	10	0	3	8	11	7	4	1	0	0	0
	11	0	1	7	12	12	7	4	1	0	0
	12	0	1	4	13	15	13	7	4	1	0
	13	0	0	3	9	18	17	13	7	4	1
	14	0	0	1	7	16	22	18	13	7	4
	15	0	0	1	4	14	23	25	19	13	7
	16	0	0	0	3	9	23	28	27	19	13
	17	0	0	0	1	7	17	31	32	28	19
	18	0	0	0	1	4	14	28	38	35	29
	19	0	0	0	0	3	9	24	38	43	37
	20	0	0	0	0	1	7	17	37	46	47

(a) Values for  $\zeta_3(m, n)$  for  $4 \leq m \leq 20$  and  $3 \leq n \leq 12$ 

		$n$									
		3	4	5	6	7	8	9	10	11	12
$m$	4	1	0	0	0	0	0	0	0	0	0
	5	5	1	0	0	0	0	0	0	0	0
	6	15	8	1	0	0	0	0	0	0	0
	7	17	29	9	1	0	0	0	0	0	0
	8	14	55	42	10	1	0	0	0	0	0
	9	8	62	103	49	10	1	0	0	0	0
	10	4	55	169	150	53	10	1	0	0	0
	11	1	34	194	308	180	54	10	1	0	0
	12	1	19	173	470	447	198	55	10	1	0
	13	0	9	122	528	825	550	206	55	10	1
	14	0	4	78	489	1169	1192	619	210	55	10
	15	0	1	40	379	1316	1996	1493	655	211	55
	16	0	1	20	259	1256	2715	2873	1712	674	212
	17	0	0	9	152	1027	3050	4509	3641	1844	682
	18	0	0	4	86	751	2964	5924	6446	4240	1921
	19	0	0	1	41	493	2530	6626	9586	8225	4647
	20	0	0	1	20	297	1967	6533	12244	13589	9699

(b) Values for  $\zeta_4(m, n)$  for  $4 \leq m \leq 20$  and  $3 \leq n \leq 12$ 

		$n$									
		3	4	5	6	7	8	9	10	11	12
$m$	4	0	0	0	0	0	0	0	0	0	0
	5	6	1	0	0	0	0	0	0	0	0
	6	37	14	1	0	0	0	0	0	0	0
	7	94	122	20	1	0	0	0	0	0	0
	8	115	484	231	23	1	0	0	0	0	0
	9	96	975	1344	324	24	1	0	0	0	0
	10	54	1265	4154	2513	377	25	1	0	0	0
	11	27	1129	8022	10911	3588	403	25	1	0	0
	12	10	781	10608	29769	20390	4342	413	25	1	0
	13	4	425	10491	55011	74172	30100	4757	417	25	1
	14	1	205	8151	74737	183102	138418	37826	4958	418	25
	15	1	81	5259	78776	328247	436612	209043	42880	5038	419
	16	0	33	2876	67748	451522	997900	809358	271532	45675	5070
	17	0	11	1395	48987	499822	1735400	2281778	1241432	317643	47037
	18	0	4	600	30785	461374	2403632	4882934	4185588	1653819	347033
	19	0	1	244	17048	365962	2749496	8262393	10741695	6473710	1988996
	20	0	1	88	8557	254805	2679218	11454075	21738877	19446274	8786966

(c) Values for  $\zeta_5(m, n)$  for  $4 \leq m \leq 20$  and  $3 \leq n \leq 12$ Table 6.3: Values for  $\zeta_\ell(m, n)$  for  $4 \leq m \leq 20$  and  $3 \leq n \leq 12$  where (a)  $\ell = 3$ , (b)  $\ell = 4$  and (c)  $\ell = 5$ .

**Proof**

(a) The only possible  $\vec{X}^{(1)}$ -vector representation is given by  $\vec{X}^{(1)} = (m - n, n)$ , yielding the desired result.

(b) Each feasible value of  $x_{(00)_2}^{(2)}$  in the  $\vec{X}^{(2)}$ -vector of an  $L_1(m, n; k)$ -set corresponds to a different overlapping structure. Because  $\max\{m - 2n, 0\} \leq x_{(00)_2}^{(2)} \leq n - 2k + 1$ , the result follows immediately.

(c) Each feasible value for  $x_{(000)_2}^{(3)}$  in the construction of  $\mathcal{L}^{(3)}$  in the proof of Theorem 3.4(c) corresponds to  $\zeta_3(m - x_{(000)_2}^{(3)}, n)$  different overlapping  $N$ -set structures for  $\mathcal{L}^{(3)}$ . Note that  $\zeta_3(m, n) = 0$  if  $m > 3n$ . We have  $\max\{m - 3n, 0\} \leq x_{(000)_2}^{(3)} \leq n - 3k + 2$  if  $m \geq 2n$ . Therefore, the first equality in (6.8) follows. For the case  $m < 2n$ , the complementary complete lottery problem  $\langle m', n'; k' \rangle \equiv \langle m, m - n; m + k - 2n \rangle$  (by virtue of the isomorphism result in Theorem 3.3(e)) may be used to obtain the second equality in (6.8), similar to the argument of Theorem 3.4(c). ■

Note that the interesting relationship

$$\sum_{i=0}^{m-n} \zeta_\ell(m - i, n) = \sum_{i=0}^n \zeta_\ell(m - i, m - n) \text{ if } \ell > 1$$

follows from the isomorphism result of Theorem 3.3(e) and Corollary 6.1. Another interesting result is that the characterisations of  $L_\psi(m, m - n; m + k - 2n)$ -sets are given by the mirror images of the  $\vec{X}$ -vectors characterising  $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$ , as dictated by the following theorem.

**Theorem 6.6** *If an  $L_\psi(m, n; k)$ -set for  $\langle m, n; k \rangle$  conforms to the overlapping structure*

$$\vec{X}^{(L_\psi)} = \left( x_0^{(L_\psi)}, x_1^{(L_\psi)}, \dots, x_{2^{L_\psi-2}}^{(L_\psi)}, x_{2^{L_\psi-1}}^{(L_\psi)} \right),$$

for some  $1 \leq k \leq n < m$  satisfying  $m + k > 2n$ , then the set corresponding to the overlapping structure

$$\vec{X}^{(L_\psi)} = \left( x_{2^{L_\psi-1}}^{(L_\psi)}, x_{2^{L_\psi-2}}^{(L_\psi)}, \dots, x_1^{(L_\psi)}, x_0^{(L_\psi)} \right)$$

is an  $L_\psi(m, m - n; m + k - 2n)$ -set for  $\langle m, m - n; m + k - 2n \rangle$ .

**Proof**

Consider a two-dimensional tabular representation similar to that in Figure 3.7(a), but for an  $L_\psi(m, n; k)$ -set,  $\mathcal{L}$ , for  $\langle m, n; k \rangle$ , consisting of  $L_\psi(m, n; k)$  rows denoting the  $n$ -sets in  $\mathcal{L}$  and  $m$  columns denoting the elements of  $\mathcal{U}_m$ , in which the  $(i, j)$ -th cell contains a cross if  $j \in \mathcal{U}_m$  is an element of the  $i$ -th  $n$ -set of  $\mathcal{L}$ . Then the complement of the tabular representation (where crosses are replaced by empty spaces and *vice versa*) represents the corresponding  $L_\psi(m, m - n; m + k - 2n)$ -set for  $\langle m, m - n; m + k - 2n \rangle$ . For any specific element of  $\mathcal{U}_m$ , a cross in its column indicates that the element is present in some  $n$ -set of  $\mathcal{L}$ . These crosses correspond to 1-bits in the binary index of the  $\vec{X}^{(L_\psi)}$ -vector capturing the overlapping  $n$ -set structure of  $\mathcal{L}$ . Thus the corresponding element in the vector  $\vec{X}^{(L_\psi)}$  for the  $L_\psi(m, m - n; m + k - 2n)$ -set may be obtained by taking the complement of each of the bits in the binary indices of that element. Therefore the  $\vec{X}^{(L_\psi)}$ -vector for the  $L_\psi(m, m - n; m + k - 2n)$ -set is the  $\vec{X}^{(L_\psi)}$ -vector for the set  $\mathcal{L}$  in reverse order. ■

In Chapter 3 a characterisation of the single  $\Psi_2(m, n; k)$ -set ( $\eta_{\Psi_2(m, n; k)}(m, n; k) = 1$ ) was presented for all lotteries  $\langle m, n; k \rangle$  where  $L_1(m, n; k) \geq 3$ . Note that all  $\eta_{\Psi_2(m, n; k)}(m, n; k)$   $\Psi_2(m, n; k)$ -sets for  $\langle m, n; k \rangle$  were also characterised when  $L_1(m, n; k) = 2$ , because in this case a  $\Psi_2(m, n; k)$ -set is also an  $L_1(m, n; k)$ -set (see Theorem 3.4(b)).

It is therefore only necessary to tabulate resource utilisations,  $\Psi_\ell(m, n; k)$ , and document their corresponding  $\Psi_\ell(m, n; k)$ -set structures, for  $\ell \geq 3$ . However, due to the complexity of the characterisation procedures described in §6.1, it is only possible to tabulate values of  $\Psi_\ell(m, n; k)$  for  $\ell \leq 5$  (as stated previously). The ranges of the parameters  $m$ ,  $n$  and  $k$  in these tabulations generally follow the ranges of Li & Van Rees [136] in cases where the resource utilisations are non-trivial (*i.e.*, entire rows and



$m$	$\ell$	$n$			
		3	4	5	6
7	3	(91.4286, 1 <sup>1</sup> )	—	—	—
	4	(100.000, 2 <sup>†</sup> )	—	—	—
	5	—	—	—	—
8	3	(78.5714, 1 <sup>2</sup> )	—	—	—
	4	(89.2857, 1 <sup>3</sup> )	—	—	—
	5	(100.000, 1 <sup>†</sup> )	—	—	—
9	3	(67.8571, 1 <sup>4</sup> )	—	—	—
	4	(77.3810, 1 <sup>5</sup> )	—	—	—
	5	(89.2857, 1 <sup>6</sup> )	—	—	—
10	3	(55.0000, 1 <sup>7</sup> )	(100.000, 3 <sup>†</sup> )	—	—
	4	(66.6667, 1 <sup>8</sup> )	—	—	—
	5	(76.6667, 1 <sup>9</sup> )	—	—	—
11	3	(45.4545, 1 <sup>10</sup> )	(100.000, 1 <sup>†</sup> )	—	—
	4	(58.1818, 1 <sup>11</sup> )	—	—	—
	5	(66.6667, 1 <sup>12</sup> )	—	—	—
12	3	(38.1818, 1 <sup>13</sup> )	(100.000, 1 <sup>†</sup> )	—	—
	4	(50.9091, 1 <sup>14</sup> )	—	—	—
	5	(58.6364, 1 <sup>15</sup> )	—	—	—
13	3	(32.5175, 1 <sup>16</sup> )	(91.0490, 1 <sup>17</sup> )	(100.000, 7 <sup>†</sup> )	—
	4	(43.3566, 1 <sup>18</sup> )	(97.7622, 1 <sup>19</sup> )	—	—
	5	(51.3986, 1 <sup>20</sup> )	(100.000, 8 <sup>†</sup> )	—	—
14	3	(28.0220, 1 <sup>21</sup> )	(82.4176, 1 <sup>22</sup> )	(100.000, 4 <sup>†</sup> )	—
	4	(37.3626, 1 <sup>23</sup> )	(93.6064, 1 <sup>24</sup> )	—	—
	5	(45.6044, 1 <sup>25</sup> )	(100.000, 1 <sup>†</sup> )	—	—
15	3	(24.3956, 1 <sup>26</sup> )	(74.5055, 1 <sup>27</sup> )	(100.000, 2 <sup>†</sup> )	—
	4	(32.5275, 1 <sup>28</sup> )	(89.4505, 1 <sup>29</sup> )	—	—
	5	(40.6593, 1 <sup>30</sup> )	(94.1392, 1 <sup>31</sup> )	—	—
16	3	(21.4286, 1 <sup>32</sup> )	(67.4176, 1 <sup>33</sup> )	(100.000, 1 <sup>†</sup> )	(100.000, 14 <sup>†</sup> )
	4	(28.5714, 1 <sup>34</sup> )	(85.9341, 1 <sup>35</sup> )	—	—
	5	(35.7143, 1 <sup>36</sup> )	(89.6154, 1 <sup>37</sup> )	—	—
17	3	(18.9706, 1 <sup>38</sup> )	(61.1345, 1 <sup>39</sup> )	(97.9800, 1 <sup>40</sup> )	(100.000, 8 <sup>†</sup> )
	4	(25.2941, 1 <sup>41</sup> )	(78.4874, 1 <sup>42</sup> )	(100.000, 9 <sup>†</sup> )	—
	5	(31.6176, 1 <sup>43</sup> )	(85.2521, 1 <sup>44</sup> )	—	—
18	3	(16.9118, 1 <sup>45</sup> )	(55.5882, 1 <sup>46</sup> )	(94.7479, 1 <sup>47</sup> )	(100.000, 5 <sup>†</sup> )
	4	(22.5490, 1 <sup>48</sup> )	(71.7647, 1 <sup>49</sup> )	(100.000, 4 <sup>†</sup> )	—
	5	(28.1863, 1 <sup>50</sup> )	(81.1765, 1 <sup>51</sup> )	—	—
19	3	(15.1703, 1 <sup>52</sup> )	(50.6966, 1 <sup>53</sup> )	(90.8411, 1 <sup>54</sup> )	(100.000, 2 <sup>†</sup> )
	4	(20.2270, 1 <sup>55</sup> )	(65.7379, 1 <sup>56</sup> )	(100.000, 1 <sup>†</sup> )	—
	5	(25.2838, 1 <sup>57</sup> )	(77.2962, 1 <sup>58</sup> )	—	—
20	3	(13.6842, 1 <sup>59</sup> )	(46.3777, 1 <sup>60</sup> )	(86.6099, 1 <sup>61</sup> )	(100.000, 1 <sup>†</sup> )
	4	(18.2456, 1 <sup>62</sup> )	(60.3509, 1 <sup>63</sup> )	(100.000, 1 <sup>†</sup> )	—
	5	(22.8070, 1 <sup>64</sup> )	(73.5810, 1 <sup>65</sup> )	—	—

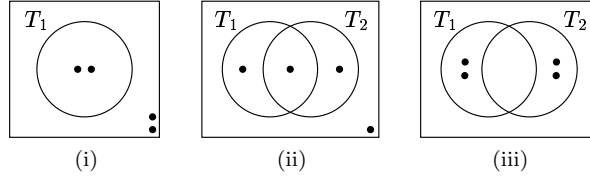
Table 6.4: Values of  $(100\Psi_\ell(m, n; 2), \eta_{\Psi_\ell}(m, n; 2))$  for all  $7 \leq m \leq 20$ ,  $3 \leq n \leq 6$  and  $\ell = 3, \dots, \min\{5, L_1(m, n; 2)\}$ . Associated  $\vec{X}^{(\ell)}$ -vector structure encodings are given by superscript in Appendix C. <sup>†</sup>These characterisations may be found in Theorem 3.4.

$m$	$\ell$	$n$			
		4	5	6	7
7	3	(91.4286, 1 <sup>1</sup> )	—	—	—
	4	(100.000, 2 <sup>†</sup> )	—	—	—
	5	—	—	—	—
8	3	(67.1429, 1 <sup>2</sup> )	—	—	—
	4	(85.7143, 1 <sup>3</sup> )	—	—	—
	5	(91.4286, 2 <sup>4</sup> )	—	—	—
9	3	(50.0000, 1 <sup>5</sup> )	—	—	—
	4	(63.4921, 1 <sup>6</sup> )	—	—	—
	5	(73.8095, 4 <sup>7</sup> )	—	—	—
10	3	(35.7143, 2 <sup>8</sup> )	—	—	—
	4	(47.6190, 1 <sup>9</sup> )	—	—	—
	5	(59.5238, 1 <sup>10</sup> )	—	—	—
11	3	(26.3636, 4 <sup>11</sup> )	(91.3420, 1 <sup>12</sup> )	—	—
	4	(35.1515, 3 <sup>13</sup> )	(99.1342, 1 <sup>14</sup> )	—	—
	5	(43.9394, 3 <sup>15</sup> )	(100.000, 49 <sup>†</sup> )	—	—
12	3	(20.0000, 5 <sup>16</sup> )	(79.5455, 2 <sup>17</sup> )	—	—
	4	(26.6667, 6 <sup>18</sup> )	(90.9091, 1 <sup>19</sup> )	—	—
	5	(33.3333, 8 <sup>20</sup> )	(97.6010, 1 <sup>21</sup> )	—	—
13	3	(15.5245, 5 <sup>22</sup> )	(15.7895, 1 <sup>23</sup> )	—	—
	4	(20.6993, 11 <sup>24</sup> )	(79.9534, 1 <sup>25</sup> )	—	—
	5	(25.8741, 18 <sup>26</sup> )	(91.1422, 1 <sup>27</sup> )	—	—
14	3	(12.2877, 5 <sup>28</sup> )	(59.0410, 1 <sup>29</sup> )	(99.5005, 1 <sup>30</sup> )	—
	4	(16.3836, 14 <sup>31</sup> )	(70.3297, 1 <sup>32</sup> )	(100.000, 26 <sup>†</sup> )	—
	5	(20.4795, 33 <sup>33</sup> )	(81.1688, 1 <sup>34</sup> )	—	—
15	3	(9.89011, 5 <sup>35</sup> )	(50.0500, 1 <sup>36</sup> )	(97.3027, 1 <sup>37</sup> )	—
	4	(13.1868, 15 <sup>38</sup> )	(60.7393, 1 <sup>39</sup> )	(100.000, 4 <sup>†</sup> )	—
	5	(16.4835, 47 <sup>40</sup> )	(71.4286, 1 <sup>41</sup> )	—	—
16	3	(8.07692, 5 <sup>42</sup> )	(41.6209, 1 <sup>43</sup> )	(93.2567, 1 <sup>44</sup> )	—
	4	(10.7692, 16 <sup>45</sup> )	(52.1978, 2 <sup>46</sup> )	(98.8761, 1 <sup>47</sup> )	—
	5	(13.4615, 60 <sup>48</sup> )	(61.9505, 1 <sup>49</sup> )	(100.000, 7 <sup>†</sup> )	—
17	3	(6.68067, 5 <sup>50</sup> )	(34.9548, 1 <sup>51</sup> )	(87.8798, 1 <sup>52</sup> )	(100.000, 7 <sup>†</sup> )
	4	(8.90756, 16 <sup>53</sup> )	(44.8610, 3 <sup>54</sup> )	(93.5520, 1 <sup>55</sup> )	—
	5	(11.1345, 67 <sup>56</sup> )	(53.6037, 2 <sup>57</sup> )	(98.5456, 1 <sup>58</sup> )	—
18	3	(5.58824, 5 <sup>59</sup> )	(29.6218, 1 <sup>60</sup> )	(81.8197, 1 <sup>61</sup> )	(100.000, 4 <sup>†</sup> )
	4	(7.45098, 16 <sup>62</sup> )	(38.6555, 2 <sup>63</sup> )	(87.8744, 1 <sup>64</sup> )	—
	5	(9.31373, 70 <sup>65</sup> )	(46.4286, 4 <sup>66</sup> )	(95.8791, 1 <sup>67</sup> )	—
19	3	(4.72136, 5 <sup>68</sup> )	(25.3096, 1 <sup>69</sup> )	(72.6338, 1 <sup>70</sup> )	(100.000, 3 <sup>†</sup> )
	4	(6.29515, 16 <sup>71</sup> )	(33.4365, 1 <sup>72</sup> )	(81.7890, 1 <sup>73</sup> )	—
	5	(7.86894, 71 <sup>74</sup> )	(40.3251, 6 <sup>75</sup> )	(89.9491, 1 <sup>76</sup> )	—
20	3	(4.02477, 5 <sup>77</sup> )	(21.7879, 1 <sup>78</sup> )	(64.4737, 1 <sup>79</sup> )	(100.000, 1 <sup>†</sup> )
	4	(5.36636, 16 <sup>80</sup> )	(29.0506, 1 <sup>81</sup> )	(75.8101, 1 <sup>82</sup> )	—
	5	(6.70795, 72 <sup>83</sup> )	(35.1522, 6 <sup>84</sup> )	(83.7900, 1 <sup>85</sup> )	—

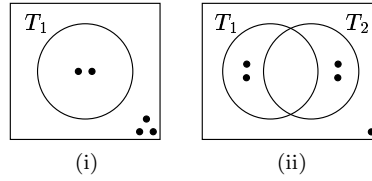
Table 6.5: Values of  $(100\Psi_\ell(m, n; 3), \eta_{\Psi_\ell}(m, n; 3))$  for all  $7 \leq m \leq 20$ ,  $4 \leq n \leq 7$  and  $\ell = 3, \dots, \min\{5, L_1(m, n; 3)\}$ . Associated  $\vec{X}^{(\ell)}$ -vector structure encodings are given by superscript in Appendix C. <sup>†</sup>These characterisations may be found in Theorem 3.4.

$m$	$\ell$	$n$			
		5	6	7	8
8	3	(78.5714, 1 <sup>1</sup> )	—	—	—
	4	(89.2857, 1 <sup>2</sup> )	—	—	—
	5	(100.000, 1 <sup>†</sup> )	—	—	—
9	3	(50.0000, 1 <sup>3</sup> )	(100.000, 5 <sup>†</sup> )	—	—
	4	(63.4921, 1 <sup>4</sup> )	—	—	—
	5	(73.8095, 4 <sup>5</sup> )	—	—	—
10	3	(30.9524, 3 <sup>6</sup> )	(100.000, 3 <sup>†</sup> )	—	—
	4	(41.2698, 3 <sup>7</sup> )	—	—	—
	5	(51.5873, 1 <sup>8</sup> )	—	—	—
11	3	(20.1299, 7 <sup>9</sup> )	(91.3420, 1 <sup>10</sup> )	—	—
	4	(26.8398, 15 <sup>11</sup> )	(99.1342, 1 <sup>12</sup> )	—	—
	5	(33.5498, 29 <sup>13</sup> )	(100.000, 49 <sup>†</sup> )	—	—
12	3	(13.6364, 10 <sup>14</sup> )	(73.4848, 1 <sup>15</sup> )	—	—
	4	(18.1818, 40 <sup>16</sup> )	(90.4762, 1 <sup>17</sup> )	—	—
	5	(22.7273, 222 <sup>18</sup> )	(95.0216, 2 <sup>19</sup> )	—	—
13	3	(9.55711, 13 <sup>20</sup> )	(58.3916, 1 <sup>21</sup> )	—	—
	4	(12.7428, 70 <sup>22</sup> )	(73.0769, 1 <sup>23</sup> )	—	—
	5	(15.9285, 772 <sup>24</sup> )	(81.5851, 2 <sup>25</sup> )	—	—
14	3	(6.89311, 14 <sup>26</sup> )	(45.6543, 1 <sup>27</sup> )	—	—
	4	(9.19081, 105 <sup>28</sup> )	(57.7090, 1 <sup>29</sup> )	—	—
	5	(11.4885, 1660 <sup>30</sup> )	(68.6314, 1 <sup>31</sup> )	—	—
15	3	(5.09491, 15 <sup>32</sup> )	(35.6643, 1 <sup>33</sup> )	(91.8415, 1 <sup>34</sup> )	—
	4	(6.79321, 129 <sup>35</sup> )	(45.4146, 1 <sup>36</sup> )	(98.8345, 1 <sup>37</sup> )	—
	5	(8.49151, 2662 <sup>38</sup> )	(55.8442, 1 <sup>39</sup> )	(100.000, 26 <sup>†</sup> )	—
16	3	(3.84615, 15 <sup>40</sup> )	(27.5724, 3 <sup>41</sup> )	(81.2587, 1 <sup>42</sup> )	—
	4	(5.12821, 145 <sup>43</sup> )	(35.8641, 1 <sup>44</sup> )	(92.9196, 1 <sup>45</sup> )	—
	5	(6.41026, 3493 <sup>46</sup> )	(44.1683, 1 <sup>47</sup> )	(97.6923, 1 <sup>48</sup> )	—
17	3	(2.95734, 15 <sup>49</sup> )	(21.6225, 4 <sup>50</sup> )	(71.7195, 1 <sup>51</sup> )	—
	4	(3.94312, 153 <sup>52</sup> )	(28.5391, 1 <sup>53</sup> )	(83.0780, 1 <sup>54</sup> )	—
	5	(4.92889, 4068 <sup>55</sup> )	(35.1729, 1 <sup>56</sup> )	(92.9761, 1 <sup>57</sup> )	—
18	3	(2.31092, 15 <sup>58</sup> )	(17.1784, 5 <sup>59</sup> )	(62.2926, 1 <sup>60</sup> )	(99.2321, 1 <sup>61</sup> )
	4	(3.08123, 157 <sup>62</sup> )	(22.9045, 1 <sup>63</sup> )	(73.3095, 1 <sup>64</sup> )	(100.000, 22 <sup>†</sup> )
	5	(3.85154, 4388 <sup>65</sup> )	(28.2428, 2 <sup>66</sup> )	(85.1056, 1 <sup>67</sup> )	—
19	3	(1.83179, 15 <sup>68</sup> )	(13.8103, 5 <sup>69</sup> )	(53.0146, 1 <sup>70</sup> )	(96.1843, 1 <sup>71</sup> )
	4	(2.44238, 158 <sup>72</sup> )	(18.4137, 3 <sup>73</sup> )	(64.0053, 1 <sup>74</sup> )	(100.000, 5 <sup>†</sup> )
	5	(3.05298, 4551 <sup>75</sup> )	(22.8844, 1 <sup>76</sup> )	(75.2620, 1 <sup>77</sup> )	—
20	3	(1.47059, 15 <sup>78</sup> )	(11.2229, 5 <sup>79</sup> )	(44.9174, 1 <sup>80</sup> )	(91.1090, 1 <sup>81</sup> )
	4	(1.96078, 159 <sup>82</sup> )	(14.9639, 6 <sup>83</sup> )	(55.4386, 1 <sup>84</sup> )	(100.000, 1 <sup>†</sup> )
	5	(2.45098, 4619 <sup>85</sup> )	(18.7049, 1 <sup>86</sup> )	(65.6192, 1 <sup>87</sup> )	—

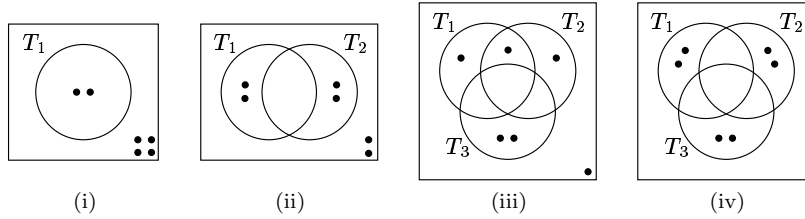
Table 6.6: Values of  $(100\Psi_\ell(m, n; 4), \eta_{\Psi_\ell}(m, n; 4))$  for all  $8 \leq m \leq 20$ ,  $5 \leq n \leq 8$  and  $\ell = 3, \dots, \min\{5, L_1(m, n; 4)\}$ . Associated  $\vec{X}^{(\ell)}$ -vector structure encodings are given by superscript in Appendix C. <sup>†</sup>These characterisations may be found in Theorem 3.4.



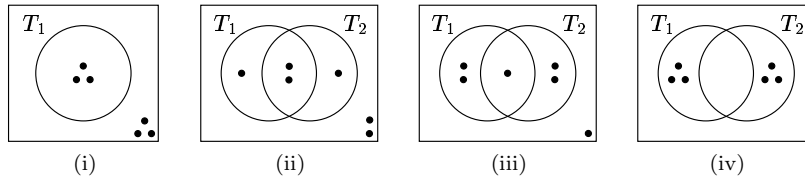
- (a) (i)  $L_\psi(4, 2; 1) = 1$  for  $0 < \psi \leq \frac{5}{6} = \Psi_1(4, 2; 1)$ ;  $\eta_{\frac{5}{6}}(4, 2; 1) = 1$   
(ii)–(iii)  $L_\psi(4, 2; 1) = 2$  for  $\frac{5}{6} < \psi \leq 1 = \Psi_2(4, 2; 1)$ ;  $\eta_1(4, 2; 1) = 2$



- (b) (i)  $L_\psi(5, 2; 1) = 1$  for  $0 < \psi \leq \frac{7}{10} = \Psi_1(5, 2; 1)$ ;  $\eta_{\frac{7}{10}}(5, 2; 1) = 1$   
(ii)  $L_\psi(5, 2; 1) = 2$  for  $\frac{7}{10} < \psi \leq 1 = \Psi_2(5, 2; 1)$ ;  $\eta_1(5, 2; 1) = 1$



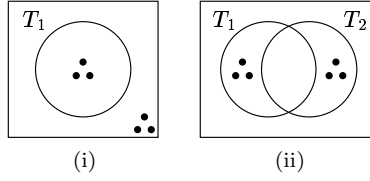
- (c) (i)  $L_\psi(6, 2; 1) = 1$  for  $0 < \psi \leq \frac{9}{15} = \Psi_1(6, 2; 1)$ ;  $\eta_{\frac{9}{15}}(6, 2; 1) = 1$   
(ii)  $L_\psi(6, 2; 1) = 2$  for  $\frac{9}{15} < \psi \leq \frac{14}{15} = \Psi_2(6, 2; 1)$ ;  $\eta_{\frac{14}{15}}(6, 2; 1) = 1$   
(iii)–(iv)  $L_\psi(6, 2; 1) = 3$  for  $\frac{14}{15} < \psi \leq 1 = \Psi_3(6, 2; 1)$ ;  $\eta_1(6, 2; 1) = 2$



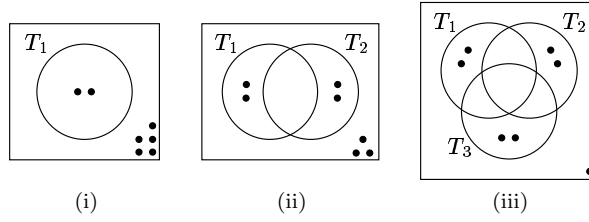
- (d) (i)  $L_\psi(6, 3; 1) = 1$  for  $0 < \psi \leq \frac{19}{20} = \Psi_1(6, 3; 1)$ ;  $\eta_{\frac{19}{20}}(6, 3; 1) = 1$   
(ii)–(iv)  $L_\psi(6, 3; 1) = 2$  for  $\frac{19}{20} < \psi \leq 1 = \Psi_2(6, 3; 1)$ ;  $\eta_1(6, 3; 1) = 3$

Figure 6.9: Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

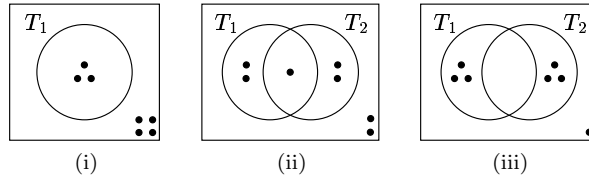




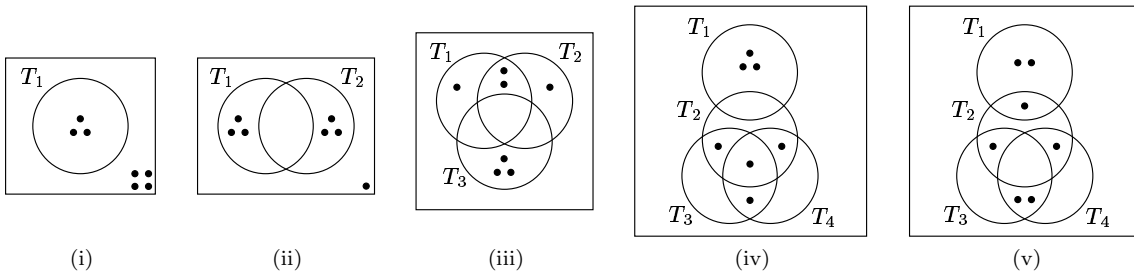
- (e) (i)  $L_\psi(6, 3; 2) = 1$  for  $0 < \psi \leq \frac{10}{20} = \Psi_1(6, 3; 2)$ ;  $\eta_{\frac{10}{20}}(6, 3; 2) = 1$   
(ii)  $L_\psi(6, 3; 2) = 2$  for  $\frac{10}{20} < \psi \leq 1 = \Psi_2(6, 3; 2)$ ;  $\eta_1(6, 3; 2) = 1$



- (f) (i)  $L_\psi(7, 2; 1) = 1$  for  $0 < \psi \leq \frac{11}{21} = \Psi_1(7, 2; 1)$ ;  $\eta_{\frac{11}{21}}(7, 2; 1) = 1$   
(ii)  $L_\psi(7, 2; 1) = 2$  for  $\frac{11}{21} < \psi \leq \frac{18}{21} = \Psi_2(7, 2; 1)$ ;  $\eta_{\frac{18}{21}}(7, 2; 1) = 1$   
(iii)  $L_\psi(7, 2; 1) = 3$  for  $\frac{18}{21} < \psi \leq 1 = \Psi_3(7, 2; 1)$ ;  $\eta_1(7, 2; 1) = 1$

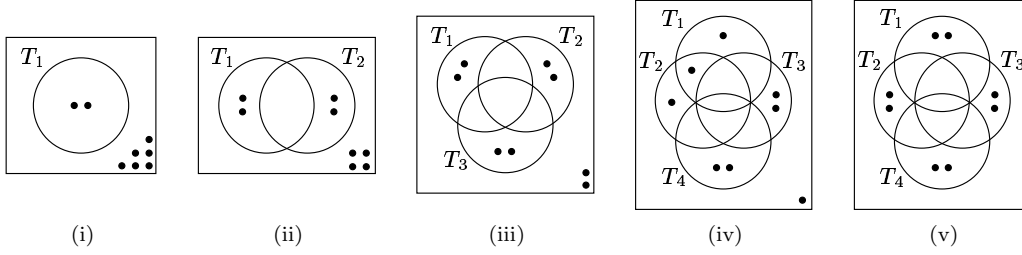


- (g) (i)  $L_\psi(7, 3; 1) = 1$  for  $0 < \psi \leq \frac{31}{35} = \Psi_1(7, 3; 1)$ ;  $\eta_{\frac{31}{35}}(7, 3; 1) = 1$   
(ii)–(iii)  $L_\psi(7, 3; 1) = 2$  for  $\frac{31}{35} < \psi \leq 1 = \Psi_2(7, 3; 1)$ ;  $\eta_1(7, 3; 1) = 2$

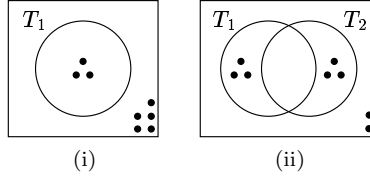


- (h) (i)  $L_\psi(7, 3; 2) = 1$  for  $0 < \psi \leq \frac{13}{35} = \Psi_1(7, 3; 2)$ ;  $\eta_{\frac{13}{35}}(7, 3; 2) = 1$   
(ii)  $L_\psi(7, 3; 2) = 2$  for  $\frac{13}{35} < \psi \leq \frac{26}{35} = \Psi_2(7, 3; 2)$ ;  $\eta_{\frac{26}{35}}(7, 3; 2) = 1$   
(iii)  $L_\psi(7, 3; 2) = 3$  for  $\frac{26}{35} < \psi \leq \frac{32}{35} = \Psi_3(7, 3; 2)$ ;  $\eta_{\frac{32}{35}}(7, 3; 2) = 1$   
(iv)–(v)  $L_\psi(7, 3; 2) = 4$  for  $\frac{32}{35} < \psi \leq 1 = \Psi_4(7, 3; 2)$ ;  $\eta_1(7, 3; 2) = 2$

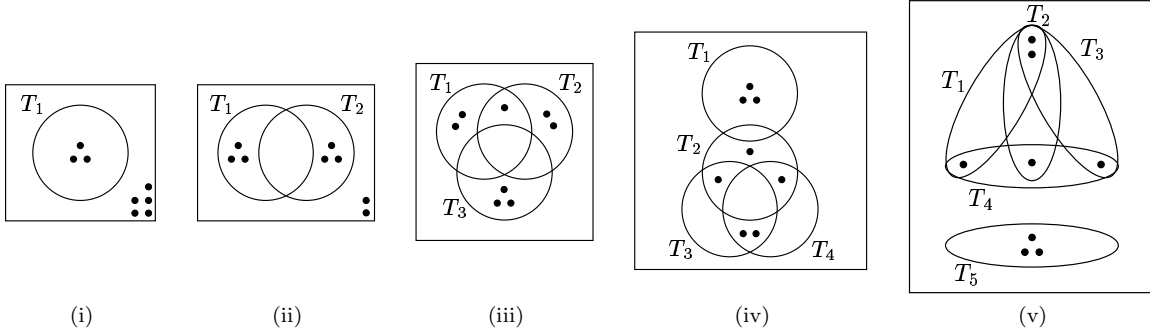
Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.



- (i) (i)  $L_\psi(8, 2; 1) = 1$  for  $0 < \psi \leq \frac{13}{28} = \Psi_1(8, 2; 1)$ ;  $\eta_{\frac{13}{28}}(8, 2; 1) = 1$   
(ii)  $L_\psi(8, 2; 1) = 2$  for  $\frac{13}{28} < \psi \leq \frac{22}{28} = \Psi_2(8, 2; 1)$ ;  $\eta_{\frac{22}{28}}(8, 2; 1) = 1$   
(iii)  $L_\psi(8, 2; 1) = 3$  for  $\frac{22}{28} < \psi \leq \frac{27}{28} = \Psi_3(8, 2; 1)$ ;  $\eta_{\frac{27}{28}}(8, 2; 1) = 1$   
(iv)–(v)  $L_\psi(8, 2; 1) = 4$  for  $\frac{27}{28} < \psi \leq 1 = \Psi_4(8, 2; 1)$ ;  $\eta_1(8, 2; 1) = 2$

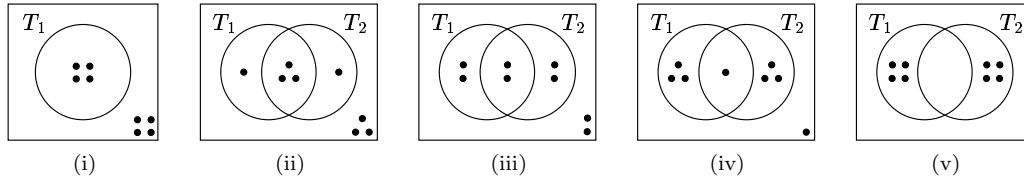


- (j) (i)  $L_\psi(8, 3; 1) = 1$  for  $0 < \psi \leq \frac{46}{56} = \Psi_1(8, 3; 1)$ ;  $\eta_{\frac{46}{56}}(8, 3; 1) = 1$   
(ii)  $L_\psi(8, 3; 1) = 2$  for  $\frac{46}{56} < \psi \leq 1 = \Psi_2(8, 3; 1)$ ;  $\eta_1(8, 3; 1) = 1$

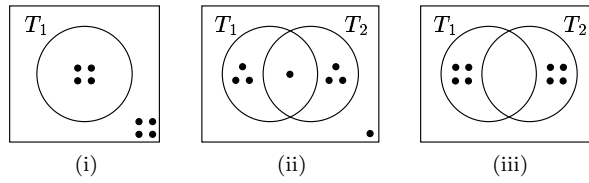


- (k) (i)  $L_\psi(8, 3; 2) = 1$  for  $0 < \psi \leq \frac{16}{56} = \Psi_1(8, 3; 2)$ ;  $\eta_{\frac{16}{56}}(8, 3; 2) = 1$   
(ii)  $L_\psi(8, 3; 2) = 2$  for  $\frac{16}{56} < \psi \leq \frac{32}{56} = \Psi_2(8, 3; 2)$ ;  $\eta_{\frac{32}{56}}(8, 3; 2) = 1$   
(iii)  $L_\psi(8, 3; 2) = 3$  for  $\frac{32}{56} < \psi \leq \frac{44}{56} = \Psi_3(8, 3; 2)$ ;  $\eta_{\frac{44}{56}}(8, 3; 2) = 1$   
(iv)  $L_\psi(8, 3; 2) = 4$  for  $\frac{44}{56} < \psi \leq \frac{50}{56} = \Psi_4(8, 3; 2)$ ;  $\eta_{\frac{50}{56}}(8, 3; 2) = 1$   
(v)  $L_\psi(8, 3; 2) = 5$  for  $\frac{50}{56} < \psi \leq 1 = \Psi_5(8, 3; 2)$ ;  $\eta_1(8, 3; 2) = 1$

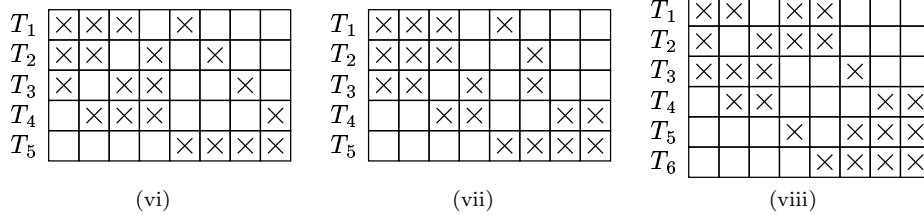
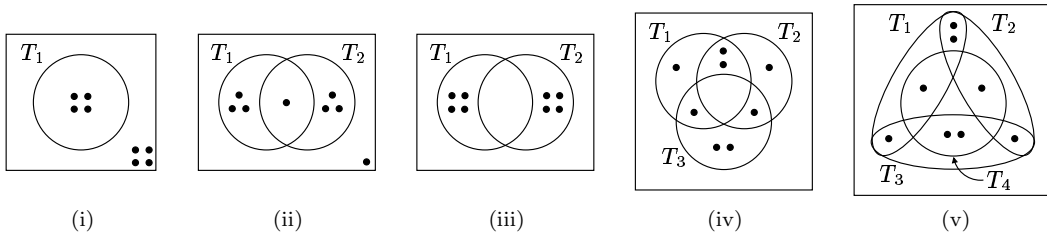
Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.



- (l) (i)  $L_\psi(8, 4; 1) = 1$  for  $0 < \psi \leq \frac{69}{70} = \Psi_1(8, 4; 1); \eta_{\frac{69}{70}}(8, 4; 1) = 1$   
(ii)–(v)  $L_\psi(8, 4; 1) = 2$  for  $\frac{69}{70} < \psi \leq 1 = \Psi_2(8, 4; 1); \eta_1(8, 4; 1) = 4$

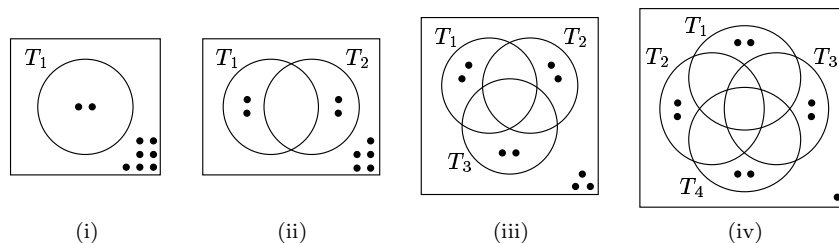


- (m) (i)  $L_\psi(8, 4; 2) = 1$  for  $0 < \psi \leq \frac{53}{70} = \Psi_1(8, 4; 2); \eta_{\frac{53}{70}}(8, 4; 2) = 1$   
(ii)–(iii)  $L_\psi(8, 4; 2) = 2$  for  $\frac{53}{70} < \psi \leq 1 = \Psi_2(8, 4; 2); \eta_1(8, 4; 2) = 2$

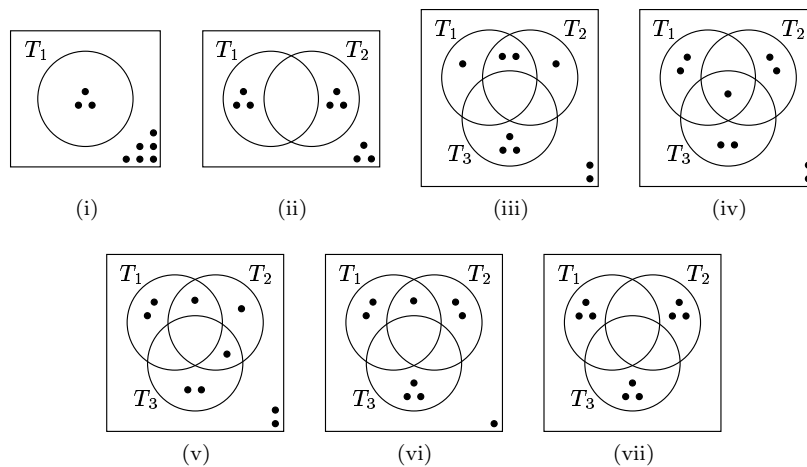


- (n) (i)  $L_\psi(8, 4; 3) = 1$  for  $0 < \psi \leq \frac{17}{70} = \Psi_1(8, 4; 3); \eta_{\frac{17}{70}}(8, 4; 3) = 1$   
(ii)–(iii)  $L_\psi(8, 4; 3) = 2$  for  $\frac{17}{70} < \psi \leq \frac{34}{70} = \Psi_2(8, 4; 3); \eta_{\frac{34}{70}}(8, 4; 3) = 2$   
(iv)  $L_\psi(8, 4; 3) = 3$  for  $\frac{34}{70} < \psi \leq \frac{47}{70} = \Psi_3(8, 4; 3); \eta_{\frac{47}{70}}(8, 4; 3) = 1$   
(v)  $L_\psi(8, 4; 3) = 4$  for  $\frac{47}{70} < \psi \leq \frac{60}{70} = \Psi_4(8, 4; 3); \eta_{\frac{60}{70}}(8, 4; 3) = 1$   
(vi)–(vii)  $L_\psi(8, 4; 3) = 5$  for  $\frac{60}{70} < \psi \leq \frac{64}{70} = \Psi_5(8, 4; 3); \eta_{\frac{64}{70}}(8, 4; 3) = 2$   
(viii)  $L_\psi(8, 4; 3) = 6$  for  $\frac{64}{70} < \psi \leq 1 = \Psi_6(8, 4; 3); \eta_1(8, 4; 3) = 1$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

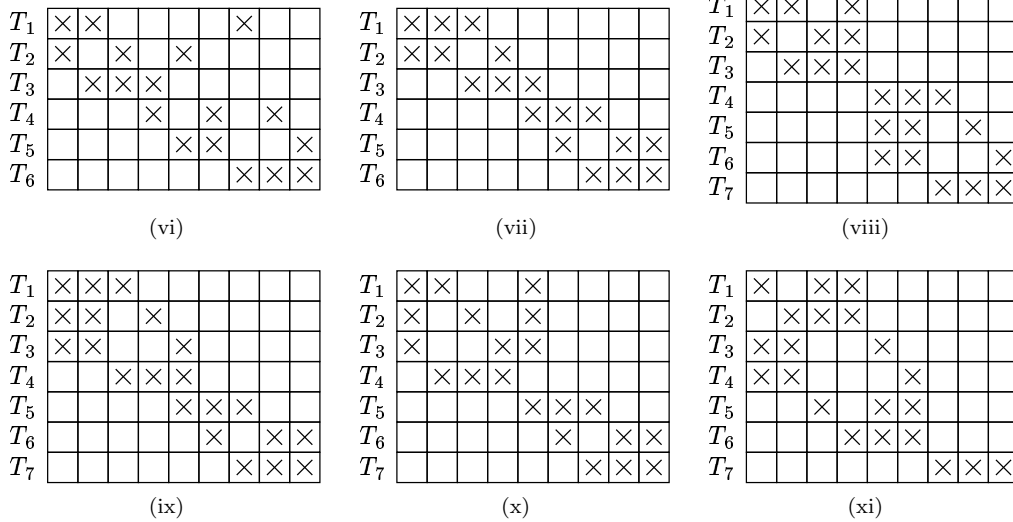
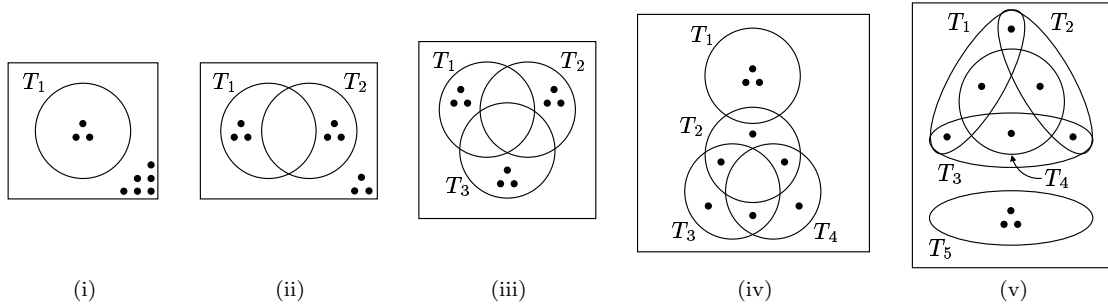


- (o) (i)  $L_\psi(9, 2; 1) = 1$  for  $0 < \psi \leq \frac{15}{36} = \Psi_1(9, 2; 1); \eta_{\frac{15}{36}}(9, 2; 1) = 1$   
 (ii)  $L_\psi(9, 2; 1) = 2$  for  $\frac{15}{36} < \psi \leq \frac{26}{36} = \Psi_2(9, 2; 1); \eta_{\frac{26}{36}}(9, 2; 1) = 1$   
 (iii)  $L_\psi(9, 2; 1) = 3$  for  $\frac{26}{36} < \psi \leq \frac{33}{36} = \Psi_3(9, 2; 1); \eta_{\frac{33}{36}}(9, 2; 1) = 1$   
 (iv)  $L_\psi(9, 2; 1) = 4$  for  $\frac{33}{36} < \psi \leq 1 = \Psi_4(9, 2; 1); \eta_1(9, 2; 1) = 1$

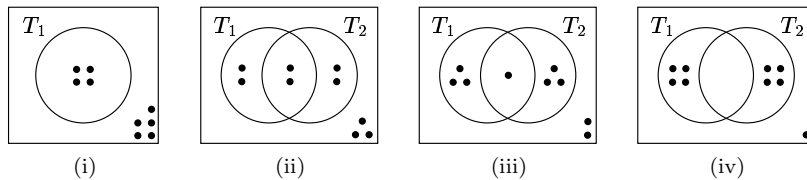


- (p) (i)  $L_\psi(9, 3; 1) = 1$  for  $0 < \psi \leq \frac{64}{84} = \Psi_1(9, 3; 1); \eta_{\frac{64}{84}}(9, 3; 1) = 1$   
 (ii)  $L_\psi(9, 3; 1) = 2$  for  $\frac{64}{84} < \psi \leq \frac{83}{84} = \Psi_2(9, 3; 1); \eta_{\frac{83}{84}}(9, 3; 1) = 1$   
 (iii)–(vii)  $L_\psi(9, 3; 1) = 3$  for  $\frac{83}{84} < \psi \leq 1 = \Psi_3(9, 3; 1); \eta_1(9, 3; 1) = 5$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

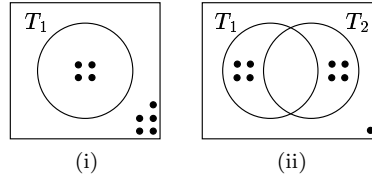


- (q) (i)  $L_\psi(9, 3; 2) = 1$  for  $0 < \psi \leq \frac{19}{84} = \Psi_1(9, 3; 2)$ ;  $\eta_{\frac{19}{84}}(9, 3; 2) = 1$   
(ii)  $L_\psi(9, 3; 2) = 2$  for  $\frac{19}{84} < \psi \leq \frac{38}{84} = \Psi_2(9, 3; 2)$ ;  $\eta_{\frac{38}{84}}(9, 3; 2) = 1$   
(iii)  $L_\psi(9, 3; 2) = 3$  for  $\frac{38}{84} < \psi \leq \frac{57}{84} = \Psi_3(9, 3; 2)$ ;  $\eta_{\frac{57}{84}}(9, 3; 2) = 1$   
(iv)  $L_\psi(9, 3; 2) = 4$  for  $\frac{57}{84} < \psi \leq \frac{65}{84} = \Psi_4(9, 3; 2)$ ;  $\eta_{\frac{65}{84}}(9, 3; 2) = 1$   
(v)  $L_\psi(9, 3; 2) = 5$  for  $\frac{65}{84} < \psi \leq \frac{75}{84} = \Psi_5(9, 3; 2)$ ;  $\eta_{\frac{75}{84}}(9, 3; 2) = 1$   
(vi)–(vii)  $L_\psi(9, 3; 2) = 6$  for  $\frac{75}{84} < \psi \leq \frac{80}{84} = \Psi_6(9, 3; 2)$ ;  $\eta_{\frac{80}{84}}(9, 3; 2) = 2$   
(viii)–(xi)  $L_\psi(9, 3; 2) = 7$  for  $\frac{80}{84} < \psi \leq 1 = \Psi_7(9, 3; 2)$ ;  $\eta_1(9, 3; 2) = 4$

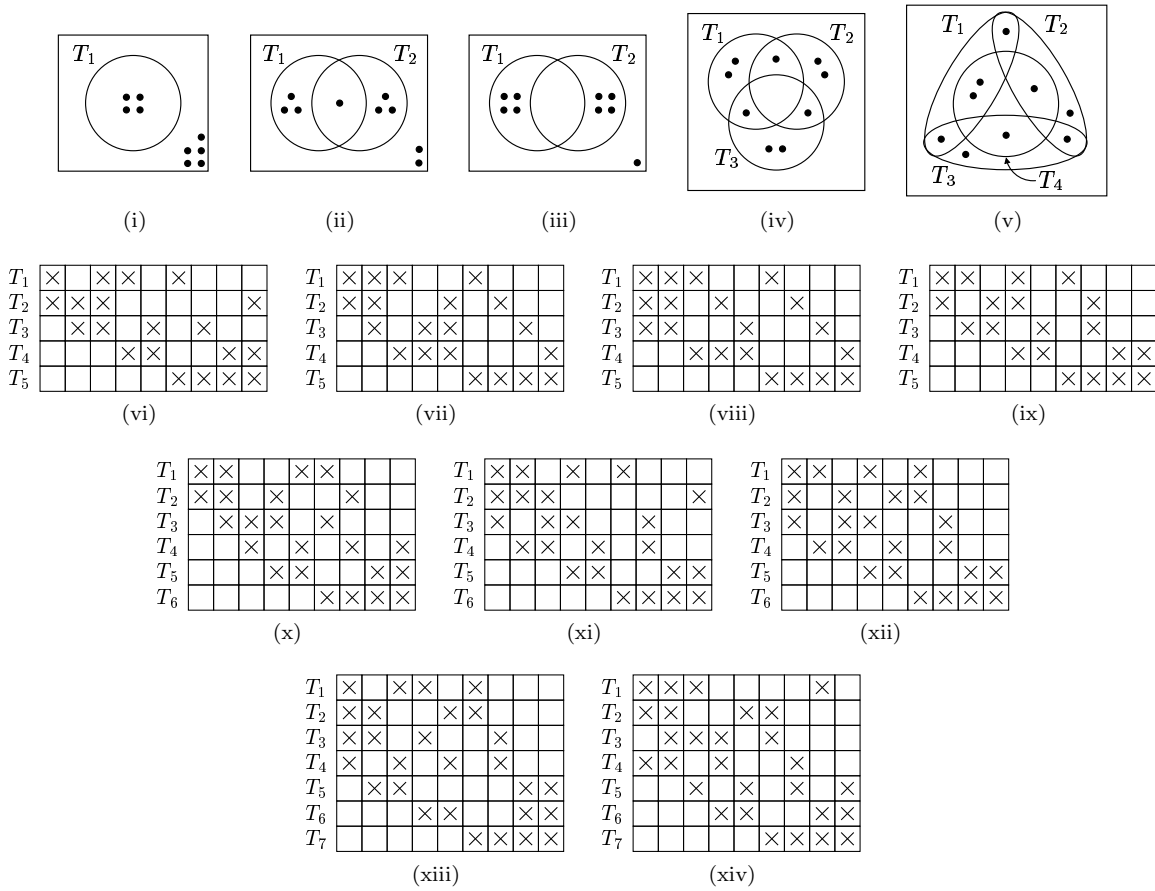


- (r) (i)  $L_\psi(9, 4; 1) = 1$  for  $0 < \psi \leq \frac{121}{126} = \Psi_1(9, 4; 1)$ ;  $\eta_{\frac{121}{126}}(9, 4; 1) = 1$   
(ii)–(iv)  $L_\psi(9, 4; 1) = 2$  for  $\frac{121}{126} < \psi \leq 1 = \Psi_2(9, 4; 1)$ ;  $\eta_1(9, 4; 1) = 3$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

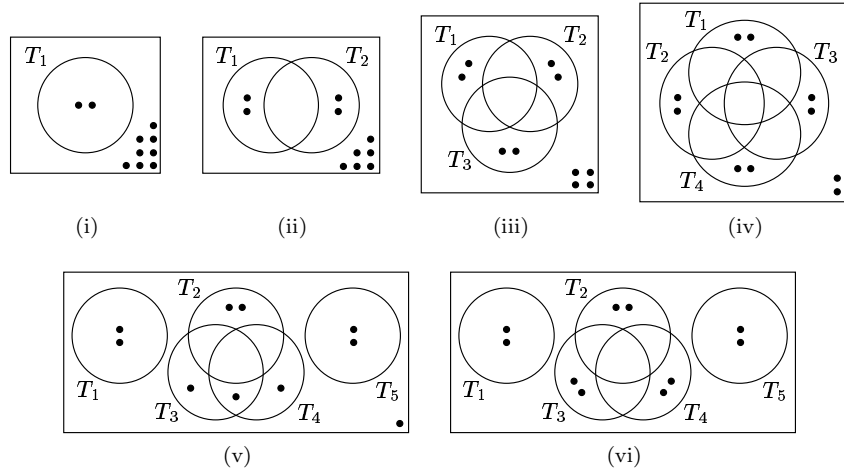


- (s) (i)  $L_\psi(9, 4; 2) = 1$  for  $0 < \psi \leq \frac{81}{126} = \Psi_1(9, 4; 2)$ ;  $\eta_{\frac{81}{126}}(9, 4; 2) = 1$   
(ii)  $L_\psi(9, 4; 2) = 2$  for  $\frac{81}{126} < \psi \leq 1 = \Psi_2(9, 4; 2)$ ;  $\eta_1(9, 4; 2) = 1$

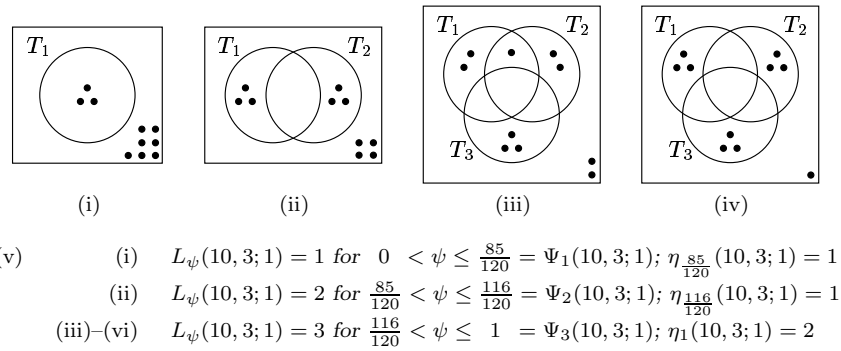


- (t) (i)  $L_\psi(9, 4; 3) = 1$  for  $0 < \psi \leq \frac{21}{126} = \Psi_1(9, 4; 3)$ ;  $\eta_{\frac{21}{126}}(9, 4; 3) = 1$   
(ii)–(iii)  $L_\psi(9, 4; 3) = 2$  for  $\frac{21}{126} < \psi \leq \frac{42}{126} = \Psi_2(9, 4; 3)$ ;  $\eta_{\frac{42}{126}}(9, 4; 3) = 2$   
(iv)  $L_\psi(9, 4; 3) = 3$  for  $\frac{42}{126} < \psi \leq \frac{63}{126} = \Psi_3(9, 4; 3)$ ;  $\eta_{\frac{63}{126}}(9, 4; 3) = 1$   
(v)  $L_\psi(9, 4; 3) = 4$  for  $\frac{63}{126} < \psi \leq \frac{80}{126} = \Psi_4(9, 4; 3)$ ;  $\eta_{\frac{80}{126}}(9, 4; 3) = 1$   
(vi)–(ix)  $L_\psi(9, 4; 3) = 5$  for  $\frac{80}{126} < \psi \leq \frac{93}{126} = \Psi_5(9, 4; 3)$ ;  $\eta_{\frac{93}{126}}(9, 4; 3) = 4$   
(x)–(xii)  $L_\psi(9, 4; 3) = 6$  for  $\frac{93}{126} < \psi \leq \frac{104}{126} = \Psi_6(9, 4; 3)$ ;  $\eta_{\frac{104}{126}}(9, 4; 3) = 3$   
(xiii)–(xiv)  $L_\psi(9, 4; 3) = 7$  for  $\frac{104}{126} < \psi \leq \frac{115}{126} = \Psi_7(9, 4; 3)$ ;  $\eta_{\frac{115}{126}}(9, 4; 3) = 2$   
 $L_\psi(9, 4; 3) \in \{8, 9\}$  for  $\frac{115}{126} < \psi \leq 1$ ;  $\eta_\psi(9, 4; 3)$  unknown for  $\frac{115}{126} < \psi \leq 1$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

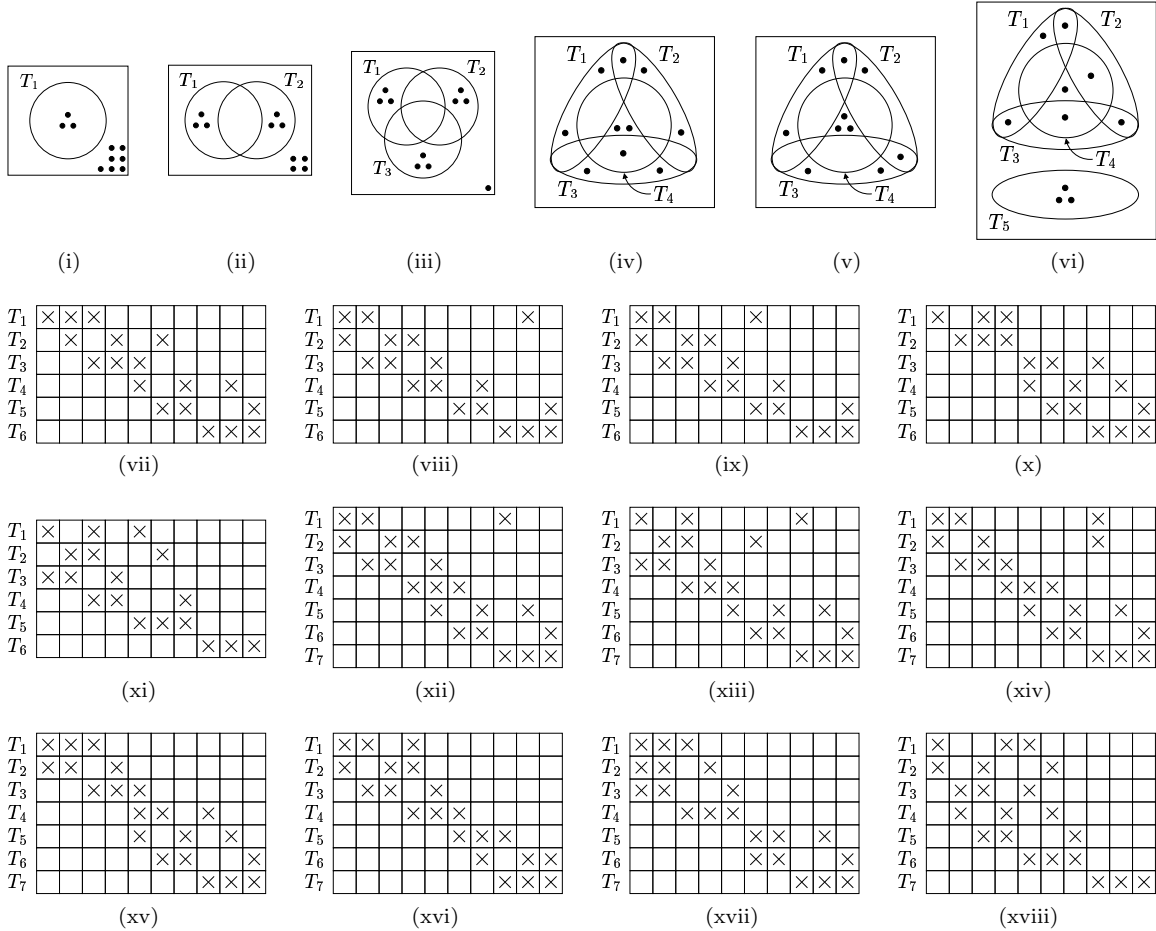


- (u)
- (i)  $L_\psi(10, 2; 1) = 1$  for  $0 < \psi \leq \frac{17}{45} = \Psi_1(10, 2; 1)$ ;  $\eta_{\frac{17}{45}}(10, 2; 1) = 1$
  - (ii)  $L_\psi(10, 2; 1) = 2$  for  $\frac{17}{45} < \psi \leq \frac{30}{45} = \Psi_2(10, 2; 1)$ ;  $\eta_{\frac{30}{45}}(10, 2; 1) = 1$
  - (iii)  $L_\psi(10, 2; 1) = 3$  for  $\frac{30}{45} < \psi \leq \frac{39}{45} = \Psi_3(10, 2; 1)$ ;  $\eta_{\frac{39}{45}}(10, 2; 1) = 1$
  - (iv)  $L_\psi(10, 2; 1) = 4$  for  $\frac{39}{45} < \psi \leq \frac{44}{45} = \Psi_4(10, 2; 1)$ ;  $\eta_{\frac{44}{45}}(10, 2; 1) = 1$
  - (v)–(vi)  $L_\psi(10, 2; 1) = 5$  for  $\frac{44}{45} < \psi \leq 1 = \Psi_5(10, 2; 1)$ ;  $\eta_1(10, 2; 1) = 2$



- (v)
- (i)  $L_\psi(10, 3; 1) = 1$  for  $0 < \psi \leq \frac{85}{120} = \Psi_1(10, 3; 1)$ ;  $\eta_{\frac{85}{120}}(10, 3; 1) = 1$
  - (ii)  $L_\psi(10, 3; 1) = 2$  for  $\frac{85}{120} < \psi \leq \frac{116}{120} = \Psi_2(10, 3; 1)$ ;  $\eta_{\frac{116}{120}}(10, 3; 1) = 1$
  - (iii)–(vi)  $L_\psi(10, 3; 1) = 3$  for  $\frac{116}{120} < \psi \leq 1 = \Psi_3(10, 3; 1)$ ;  $\eta_1(10, 3; 1) = 2$

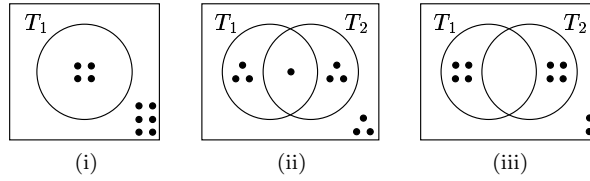
Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.



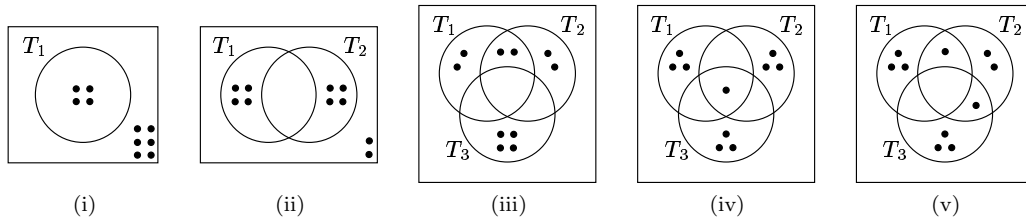
- (w)
- (i)  $L_\psi(10, 3; 2) = 1$  for  $0 < \psi \leq \frac{22}{120} = \Psi_1(10, 3; 2)$ ;  $\eta_{\frac{22}{120}}(10, 3; 2) = 1$
  - (ii)  $L_\psi(10, 3; 2) = 2$  for  $\frac{22}{120} < \psi \leq \frac{44}{120} = \Psi_2(10, 3; 2)$ ;  $\eta_{\frac{44}{120}}(10, 3; 2) = 1$
  - (iii)  $L_\psi(10, 3; 2) = 3$  for  $\frac{44}{120} < \psi \leq \frac{66}{120} = \Psi_3(10, 3; 2)$ ;  $\eta_{\frac{66}{120}}(10, 3; 2) = 1$
  - (iv)–(v)  $L_\psi(10, 3; 2) = 4$  for  $\frac{66}{120} < \psi \leq \frac{80}{120} = \Psi_4(10, 3; 2)$ ;  $\eta_{\frac{80}{120}}(10, 3; 2) = 2$
  - (vi)  $L_\psi(10, 3; 2) = 5$  for  $\frac{80}{120} < \psi \leq \frac{92}{120} = \Psi_5(10, 3; 2)$ ;  $\eta_{\frac{92}{120}}(10, 3; 2) = 1$
  - (vii)–(xi)  $L_\psi(10, 3; 2) = 6$  for  $\frac{92}{120} < \psi \leq \frac{102}{120} = \Psi_6(10, 3; 2)$ ;  $\eta_{\frac{102}{120}}(10, 3; 2) = 5$
  - (xii)–(xviii)  $L_\psi(10, 3; 2) = 7$  for  $\frac{102}{120} < \psi \leq \frac{110}{120} = \Psi_7(10, 3; 2)$ ;  $\eta_{\frac{110}{120}}(10, 3; 2) = 7$
  - $L_\psi(10, 3; 2) = 8$  for  $\frac{110}{120} < \psi \leq 1 = \Psi_8(10, 3; 2)$ ;  $\eta_1(10, 3; 2)$  is unknown

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

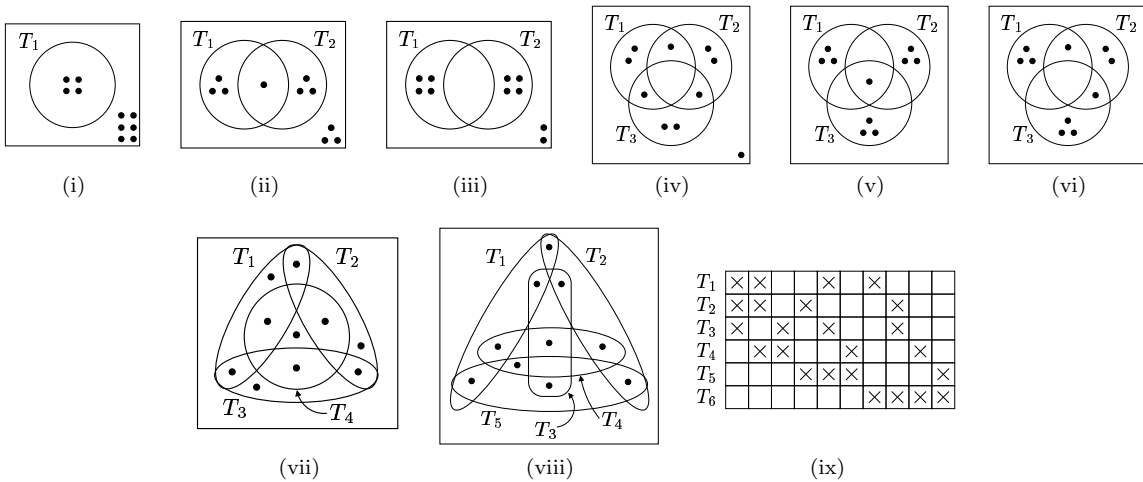




- (x) (i)  $L_\psi(10, 4; 1) = 1$  for  $0 < \psi \leq \frac{195}{210} = \Psi_1(10, 4; 1)$ ;  $\eta_{\frac{195}{210}}(10, 4; 1) = 1$   
 (ii)–(iii)  $L_\psi(10, 4; 1) = 2$  for  $\frac{195}{210} < \psi \leq 1 = \Psi_2(10, 4; 1)$ ;  $\eta_1(10, 4; 1) = 2$

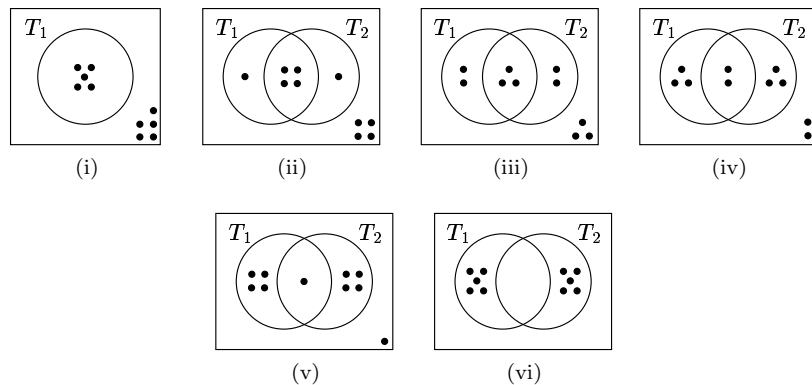


- (y) (i)  $L_\psi(10, 4; 2) = 1$  for  $0 < \psi \leq \frac{115}{210} = \Psi_1(10, 4; 2)$ ;  $\eta_{\frac{115}{210}}(10, 4; 2) = 1$   
 (ii)  $L_\psi(10, 4; 2) = 2$  for  $\frac{115}{210} < \psi \leq \frac{194}{210} = \Psi_2(10, 4; 2)$ ;  $\eta_{\frac{194}{210}}(10, 4; 2) = 1$   
 (iii)–(v)  $L_\psi(10, 4; 2) = 3$  for  $\frac{194}{210} < \psi \leq 1 = \Psi_3(10, 4; 2)$ ;  $\eta_1(10, 4; 2) = 3$

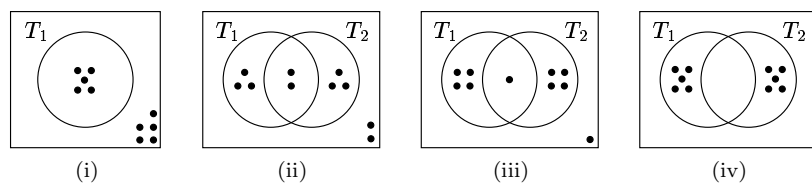


- (z) (i)  $L_\psi(10, 4; 3) = 1$  for  $0 < \psi \leq \frac{25}{210} = \Psi_1(10, 4; 3)$ ;  $\eta_{\frac{25}{210}}(10, 4; 3) = 1$   
 (ii)–(iii)  $L_\psi(10, 4; 3) = 2$  for  $\frac{25}{210} < \psi \leq \frac{50}{210} = \Psi_2(10, 4; 3)$ ;  $\eta_{\frac{50}{210}}(10, 4; 3) = 2$   
 (iv)–(vi)  $L_\psi(10, 4; 3) = 3$  for  $\frac{50}{210} < \psi \leq \frac{75}{210} = \Psi_3(10, 4; 3)$ ;  $\eta_{\frac{75}{210}}(10, 4; 3) = 3$   
 (vii)  $L_\psi(10, 4; 3) = 4$  for  $\frac{75}{210} < \psi \leq \frac{100}{210} = \Psi_4(10, 4; 3)$ ;  $\eta_{\frac{100}{210}}(10, 4; 3) = 1$   
 (viii)  $L_\psi(10, 4; 3) = 5$  for  $\frac{100}{210} < \psi \leq \frac{125}{210} = \Psi_5(10, 4; 3)$ ;  $\eta_{\frac{125}{210}}(10, 4; 3) = 1$   
 (ix)  $L_\psi(10, 4; 3) = 6$  for  $\frac{125}{210} < \psi \leq \frac{139}{210} = \Psi_6(10, 4; 3)$ ;  $\eta_{\frac{139}{210}}(10, 4; 3) = 1$   
 $L_\psi(10, 4; 3) \in \{7, \dots, 14\}$  for  $\frac{139}{210} < \psi \leq 1$ ;  $\eta_\psi(10, 4; 3)$  unknown for  $\frac{139}{210} < \psi \leq 1$

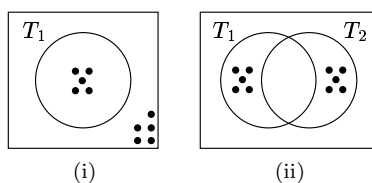
Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.



- (aa) (i)  $L_\psi(10, 5; 1) = 1$  for  $0 < \psi \leq \frac{251}{252} = \Psi_1(10, 5; 1)$ ;  $\eta_{\frac{251}{252}}(10, 5; 1) = 1$   
(ii)–(vi)  $L_\psi(10, 5; 1) = 2$  for  $\frac{251}{252} < \psi \leq 1 = \Psi_2(10, 5; 1)$ ;  $\eta_1(10, 5; 1) = 5$

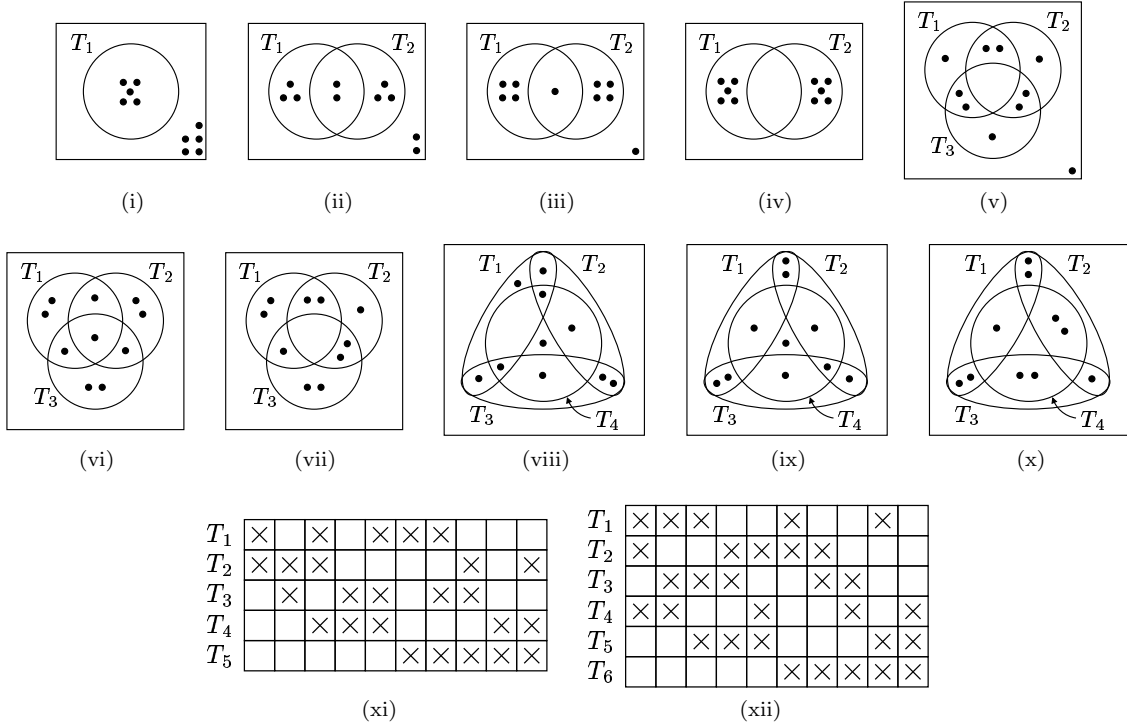


- (bb) (i)  $L_\psi(10, 5; 2) = 1$  for  $0 < \psi \leq \frac{226}{252} = \Psi_1(10, 5; 2)$ ;  $\eta_{\frac{226}{252}}(10, 5; 2) = 1$   
(ii)–(iv)  $L_\psi(10, 5; 2) = 2$  for  $\frac{226}{252} < \psi \leq 1 = \Psi_2(10, 5; 2)$ ;  $\eta_1(10, 5; 2) = 3$



- (cc) (i)  $L_\psi(10, 5; 3) = 1$  for  $0 < \psi \leq \frac{126}{252} = \Psi_1(10, 5; 3)$ ;  $\eta_{\frac{126}{252}}(10, 5; 3) = 1$   
(ii)  $L_\psi(10, 5; 3) = 2$  for  $\frac{126}{252} < \psi \leq 1 = \Psi_2(10, 5; 3)$ ;  $\eta_1(10, 5; 3) = 1$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.



- (dd)
- (i)  $L_\psi(10, 5; 4) = 1$  for  $0 < \psi \leq \frac{26}{252} = \Psi_1(10, 5; 4); \eta_{\frac{26}{252}}(10, 5; 4) = 1$
  - (ii)–(iv)  $L_\psi(10, 5; 4) = 2$  for  $\frac{26}{252} < \psi \leq \frac{52}{252} = \Psi_2(10, 5; 4); \eta_{\frac{52}{252}}(10, 5; 4) = 3$
  - (v)–(vii)  $L_\psi(10, 5; 4) = 3$  for  $\frac{52}{252} < \psi \leq \frac{78}{252} = \Psi_3(10, 5; 4); \eta_{\frac{78}{252}}(10, 5; 4) = 3$
  - (viii)–(x)  $L_\psi(10, 5; 4) = 4$  for  $\frac{78}{252} < \psi \leq \frac{104}{252} = \Psi_4(10, 5; 4); \eta_{\frac{104}{252}}(10, 5; 4) = 3$
  - (xi)  $L_\psi(10, 5; 4) = 5$  for  $\frac{104}{252} < \psi \leq \frac{130}{252} = \Psi_5(10, 5; 4); \eta_{\frac{130}{252}}(10, 5; 4) = 1$
  - (xii)  $L_\psi(10, 5; 4) = 6$  for  $\frac{130}{252} < \psi \leq \frac{156}{252} = \Psi_6(10, 5; 4); \eta_{\frac{156}{252}}(10, 5; 4) = 1$
- $L_\psi(10, 5; 4) \in \{7, \dots, 14\}$  for  $\frac{156}{252} < \psi \leq 1; \eta_\psi(10, 5; 4)$  unknown for  $\frac{156}{252} < \psi \leq 1$

Figure 6.9 (continued): Graphical representations of all  $\eta_\psi(m, n; k)$  characterisations of  $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ , where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $L_\psi(m, n; k) \leq 6, 7$  obtained by the enumeration methods described in §6.1.

**Proof of Theorem 6.7**

By contradiction. Suppose that  $L_1(18, 6; 3) = 6$ . Then it follows, by Lemma 6.1, that any  $L_1(18, 6; 3)$ -set must contain exactly one disjoint 6-set from  $\mathcal{U}_{18}$ . If, in the construction technique outlined above, a column involving the disjoint 6-set is deleted, the resulting complete lottery set for  $\langle 17, 6; 3 \rangle$  will have no disjoint 6-set, which contradicts the characterisation of minimal complete lottery sets for  $\langle 17, 6; 3 \rangle$  (see Figure 6.4(a)). It is therefore possible to conclude that  $6 < L_1(18, 6; 3) \leq 7$ , which yields the desired result. ■

As an extension to §6.3, the enumeration methods described in §6.1 were used (either directly or indirectly) to find new complete lottery numbers  $L_1(m, n; k) \leq 7$  for which  $m > 20$  (by either verifying design optimality, or by upper/lower bound improvement) and/or improvements on current bounds available on Internet lottery repositories [19, 44, 133, 237]. Table 6.7 summarises all the results using the enumeration methods in §6.1 for  $\langle m, n; k \rangle$ , apart from the two lower bound improvements

$$\begin{aligned} 6 &\leq L_1(17, 7; 4) \leq 9 \quad \text{and} \\ 7 &\leq L_1(19, 6; 3) \leq 9. \end{aligned}$$

Previous best lower bounds on these lotteries were given by  $L_1(17, 7; 4) \geq 5$  and  $L_1(19, 6; 3) \geq 6$  respectively [19, 44, 133, 237].

## 6.5 Chapter summary

In this chapter two possible enumeration methods for determining all structurally different  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -sets for  $\langle m, n; k \rangle$  were described. The only difference between the two methods is the way in which the  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set structure is represented and manipulated.

In §6.2 the characterisation number  $\eta_\psi(m, n; k)$  was studied in depth. First the existence and boundedness of  $\eta_\psi(m, n; k)$  was established (in Theorem 6.1), followed by a presentation of some growth properties of the parameter. This led to the definition of so-called jump sequences, defining the intervals at which the parameter  $\eta_\psi(m, n; k)$  was seen to display a saw-tooth growth behaviour. Theorem 6.5 established  $\eta_1(m, n; k)$  for small values of  $L_1(m, n; k) = 1, 2, 3$  (these complete lottery numbers were characterised in Chapter 3). An interesting mirror (isomorphism) result relating the optimal solution  $\vec{X}$ -vectors for  $\langle m, n; k \rangle$  to those for the lottery  $\langle m, m - n; m + k - 2n \rangle$  was also established in §6.2.

An  $n$ -set addition to any playing set  $\mathcal{L}$  (of cardinality  $\ell$ ) only incurs a linear (unit) increase in the order of  $\aleph_{\mathcal{L}}$  (for the enumeration method described in §6.1.2), as opposed to the exponential ( $2^\ell$ ) increase in the vector  $\vec{X}^{(\ell)}$  when using the enumeration method described in §6.1.1. It is therefore suggested that the lottery tree method may be used when determining  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set characterisations for smaller values of  $\ell$ , while the `nauty` tree method should be used when dealing with larger values of  $\ell$ . This, however, is not the only motivation for using one enumeration method above the other. The value of the parameter  $m$  also plays a crucial role in this distinction, seeing that `nauty` has to select the canonically first labelling of  $\aleph_{\mathcal{L}^{(\ell)}}$  from all possible permutations of  $(m + |\mathcal{L}^{(\ell)}|)$  vertices in  $\aleph_{\mathcal{L}^{(\ell)}}$ . In contrast, the computational effect of a change/increase in the parameter  $m$  when using the lottery tree enumeration method (§6.1.1), is not as large.

An analysis of small lotteries  $\langle m, n; k \rangle$  (where  $1 \leq k < n < m \leq 10$ ) was conducted in §6.3, leading to the determination of all the  $\eta_\psi(m, n; k)$  structurally different  $\{L_\psi(m, n; k), \Psi_\ell(m, n; k)\}$ -set (where  $1 \leq k < n < m \leq 10$  satisfying  $m + k > 2n$ ,  $L_1(m, n; k) > 1$  and  $1 \leq L_\psi(m, n; k) \leq 6, 7$ ), as represented in Figure 6.9. As a summary of the results in §6.3, Table 6.8 contains the values of the lottery characterisation numbers  $\eta_1(m, n; k)$  for  $\langle m, n; k \rangle$  (where  $1 \leq k \leq n \leq m \leq 10$ ).

The enumeration methods described in §6.1 were used to establish optimality of, or improve upon the best known bounds available on the Internet [19, 44, 133, 237] for various lotteries. More specifically, previously known upper bounds on 27 complete lottery numbers were determined to be optimal while 28 upper bounds on  $L_1(m, n; k)$  were found to be suboptimal and were subsequently improved, rendering a total of 55 new complete lottery numbers. In addition, 46 upper bounds on  $L_1(m, n; k)$  were also improved, using a decomposition result from [38]. These results are captured in Table 6.7.

$L_1(15, 7; 4) = 5$	$L_1(23, 6; 2) = 4$	$L_1(30, 6; 2) = 5$	$L_1(46, 8; 2) = 6$
$L_1(16, 7; 4) = 6$	$L_1(24, 6; 2) = 4$	$L_1(31, 6; 2) = 7$	$L_1(50, 8; 2) = 7$
$L_1(18, 6; 3) = 7^\dagger$	$L_1(25, 6; 2) = 4$	$L_1(32, 6; 2) = 7$	$L_1(51, 8; 2) = 7$
$L_1(21, 7; 3) = 3$	$L_1(26, 6; 2) = 5$	$L_1(33, 6; 2) = 7$	$L_1(52, 8; 2) = 7$
$L_1(21, 6; 2) = 4$	$L_1(27, 6; 2) = 5$	$L_1(43, 8; 2) = 6$	$L_1(53, 8; 2) = 7$
$L_1(22, 6; 2) = 4$	$L_1(28, 6; 2) = 5$	$L_1(44, 8; 2) = 6$	$L_1(54, 8; 2) = 7$
$L_1(22, 7; 3) = 6$	$L_1(29, 6; 2) = 5$	$L_1(45, 8; 2) = 6$	

(a) Lotteries  $\langle m, n; k \rangle$  for which upper bounds on  $L_1(m, n; k) \leq 7$  in [19, 44, 133, 237] were determined to be optimal via the enumeration techniques described in §6.1

New result	Previous bound	New result	Previous bound	New result	Previous bound	New result	Previous bound
$L_1(23, 7; 3) = 6$	$\leq 7$	$L_1(48, 8; 2) = 6$	$\leq 7$	$L_1(53, 9; 2) = 6$	$\leq 25$	$L_1(58, 9; 2) = 7$	$\leq 18$
$L_1(39, 8; 2) = 6$	$\leq 6$	$L_1(48, 9; 2) = 5$	$\leq 16$	$L_1(54, 9; 2) = 6$	$\leq 27$	$L_1(59, 9; 2) = 7$	$\leq 18$
$L_1(40, 8; 2) = 5$	$\leq 6$	$L_1(49, 8; 2) = 6$	$\leq 7$	$L_1(55, 8; 2) = 7$	$\leq 9$	$L_1(60, 9; 2) = 7$	$\leq 18$
$L_1(41, 8; 2) = 5$	$\leq 6$	$L_1(49, 9; 2) = 6$	$\leq 18$	$L_1(55, 9; 2) = 6$	$\leq 29$	$L_1(61, 9; 2) = 7$	$\leq 18$
$L_1(42, 8; 2) = 5$	$\leq 6$	$L_1(50, 9; 2) = 6$	$\leq 20$	$L_1(56, 8; 2) = 7$	$\leq 9$	$L_1(62, 9; 2) = 7$	$\leq 18$
$L_1(45, 10; 3) = 6$	$\leq 15$	$L_1(51, 9; 2) = 6$	$\leq 22$	$L_1(56, 9; 2) = 6$	$\leq 18$	$L_1(63, 9; 2) = 7$	$\leq 19$
$L_1(47, 8; 2) = 6$	$\leq 7$	$L_1(52, 9; 2) = 6$	$\leq 23$	$L_1(57, 9; 2) = 7$	$\leq 18$	$L_1(64, 9; 2) = 7$	$\leq 19$

(b) Lotteries  $\langle m, n; k \rangle$  for which the upper bounds in [19, 44, 133, 237] were found to be suboptimal. These bounds were improved to optimality via the enumeration techniques described in §6.1

New bound	Previous bound	New bound	Previous bound	New bound	Previous bound	New bound	Previous bound
$L_1(21, 8; 4) \leq 7$	$\leq 8$	$L_1(50, 10; 3) \leq 11$	$\leq 15$	$L_1(69, 9; 2) \leq 8$	$\leq 20$	$L_1(80, 9; 2) \leq 12$	$\leq 22$
$L_1(38, 9; 3) \leq 7$	$\leq 10$	$L_1(51, 10; 3) \leq 12$	$\leq 15$	$L_1(70, 9; 2) \leq 8$	$\leq 20$	$L_1(81, 9; 2) \leq 13$	$\leq 22$
$L_1(39, 9; 3) \leq 9$	$\leq 10$	$L_1(52, 7; 2) \leq 13$	$\leq 14$	$L_1(71, 9; 2) \leq 8$	$\leq 20$	$L_1(82, 9; 2) \leq 13$	$\leq 22$
$L_1(45, 7; 2) \leq 8$	$\leq 12$	$L_1(52, 10; 3) \leq 13$	$\leq 15$	$L_1(72, 9; 2) \leq 8$	$\leq 20$	$L_1(83, 9; 2) \leq 14$	$\leq 22$
$L_1(46, 7; 2) \leq 9$	$\leq 12$	$L_1(53, 10; 3) \leq 14$	$\leq 15$	$L_1(73, 9; 2) \leq 10$	$\leq 20$	$L_1(84, 9; 2) \leq 14$	$\leq 22$
$L_1(46, 10; 3) \leq 7$	$\leq 15$	$L_1(60, 10; 3) \leq 21$	$\leq 23$	$L_1(74, 9; 2) \leq 10$	$\leq 20$	$L_1(85, 9; 2) \leq 15$	$\leq 22$
$L_1(47, 7; 2) \leq 10$	$\leq 12$	$L_1(61, 10; 3) \leq 22$	$\leq 24$	$L_1(75, 9; 2) \leq 10$	$\leq 20$	$L_1(86, 9; 2) \leq 15$	$\leq 22$
$L_1(47, 10; 3) \leq 9$	$\leq 15$	$L_1(62, 10; 3) \leq 23$	$\leq 24$	$L_1(76, 9; 2) \leq 10$	$\leq 20$	$L_1(87, 9; 2) \leq 16$	$\leq 22$
$L_1(48, 7; 2) \leq 10$	$\leq 12$	$L_1(65, 9; 2) \leq 8$	$\leq 20$	$L_1(77, 9; 2) \leq 11$	$\leq 21$	$L_1(88, 9; 2) \leq 16$	$\leq 22$
$L_1(48, 10; 3) \leq 9$	$\leq 15$	$L_1(66, 9; 2) \leq 8$	$\leq 20$	$L_1(78, 9; 2) \leq 11$	$\leq 21$	$L_1(89, 9; 2) \leq 17$	$\leq 22$
$L_1(49, 7; 2) \leq 11$	$\leq 12$	$L_1(67, 9; 2) \leq 8$	$\leq 20$	$L_1(79, 9; 2) \leq 12$	$\leq 22$	$L_1(90, 9; 2) \leq 17$	$\leq 22$
$L_1(49, 10; 3) \leq 10$	$\leq 15$	$L_1(68, 9; 2) \leq 8$	$\leq 20$				

(c) Lotteries  $\langle m, n; k \rangle$  for which upper bounds on  $L_1(m, n; k)$  [19, 44, 133, 237] were improved via the enumeration techniques described in §6.1

Table 6.7: Lotteries  $\langle m, n; k \rangle$  from Internet repositories [19, 44, 133, 237] for which (a) upper bounds on  $L_1(m, n; k) \leq 7$  were found to be optimal; (b) upper bounds on  $L_1(m, n; k) \leq 7$  were found to be suboptimal (the corresponding suboptimal upper bounds on  $L_1(m, n; k)$  are given and the optimal values of these lottery numbers are listed); and (c) upper bounds on  $L_1(m, n; k)$  were improved using a decomposition result in [38] (the corresponding previous best upper bounds on  $L_1(m, n; k)$  are also given). <sup>†</sup>See Theorem 6.7.

<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="2" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td></tr> <tr><td rowspan="2" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(a) <math>\eta_1(2, n; k)</math></p>			$n$				1	2	$k$	1	1	1	2	-	1	<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="3" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td></tr> <tr><td rowspan="3" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(b) <math>\eta_1(3, n; k)</math></p>			$n$					1	2	3	$k$	1	1	1	1	2	-	1	1	3	-	-	1	<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="4" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td></tr> <tr><td rowspan="4" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(c) <math>\eta_1(4, n; k)</math></p>			$n$						1	2	3	4	$k$	1	1	2	1	1	2	-	1	1	1	3	-	-	1	1	4	-	-	-	1																																																																																																																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2																																																																																																																																																																																																																																																						
$k$	1	1	1																																																																																																																																																																																																																																																						
	2	-	1																																																																																																																																																																																																																																																						
		$n$																																																																																																																																																																																																																																																							
		1	2	3																																																																																																																																																																																																																																																					
$k$	1	1	1	1																																																																																																																																																																																																																																																					
	2	-	1	1																																																																																																																																																																																																																																																					
	3	-	-	1																																																																																																																																																																																																																																																					
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4																																																																																																																																																																																																																																																				
$k$	1	1	2	1	1																																																																																																																																																																																																																																																				
	2	-	1	1	1																																																																																																																																																																																																																																																				
	3	-	-	1	1																																																																																																																																																																																																																																																				
	4	-	-	-	1																																																																																																																																																																																																																																																				
<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="5" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td></tr> <tr><td rowspan="5" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(d) <math>\eta_1(5, n; k)</math></p>			$n$							1	2	3	4	5	$k$	1	1	2	1	1	1	2	-	1	1	1	1	3	-	-	1	1	1	4	-	-	-	1	1	5	-	-	-	-	1	<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="6" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">6</td></tr> <tr><td rowspan="6" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(e) <math>\eta_1(6, n; k)</math></p>			$n$								1	2	3	4	5	6	$k$	1	1	2	3	1	1	1	2	-	1	1	1	1	1	3	-	-	1	2	1	1	4	-	-	-	1	1	1	5	-	-	-	-	1	1	6	-	-	-	-	-	1																																																																																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5																																																																																																																																																																																																																																																			
$k$	1	1	2	1	1	1																																																																																																																																																																																																																																																			
	2	-	1	1	1	1																																																																																																																																																																																																																																																			
	3	-	-	1	1	1																																																																																																																																																																																																																																																			
	4	-	-	-	1	1																																																																																																																																																																																																																																																			
	5	-	-	-	-	1																																																																																																																																																																																																																																																			
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6																																																																																																																																																																																																																																																		
$k$	1	1	2	3	1	1	1																																																																																																																																																																																																																																																		
	2	-	1	1	1	1	1																																																																																																																																																																																																																																																		
	3	-	-	1	2	1	1																																																																																																																																																																																																																																																		
	4	-	-	-	1	1	1																																																																																																																																																																																																																																																		
	5	-	-	-	-	1	1																																																																																																																																																																																																																																																		
	6	-	-	-	-	-	1																																																																																																																																																																																																																																																		
<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="7" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">7</td></tr> <tr><td rowspan="7" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(f) <math>\eta_1(7, n; k)</math></p>			$n$									1	2	3	4	5	6	7	$k$	1	1	1	2	1	1	1	1	2	-	1	2	2	1	1	1	3	-	-	1	2	1	1	1	4	-	-	-	1	1	1	1	5	-	-	-	-	1	1	1	6	-	-	-	-	-	1	1	7	-	-	-	-	-	-	1	<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="8" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">8</td></tr> <tr><td rowspan="8" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(g) <math>\eta_1(8, n; k)</math></p>			$n$										1	2	3	4	5	6	7	8	$k$	1	1	2	1	4	1	1	1	1	2	-	1	1	2	1	1	1	1	3	-	-	1	1	1	1	1	1	4	-	-	-	1	1	1	1	1	5	-	-	-	-	1	2	1	1	6	-	-	-	-	-	1	1	1	7	-	-	-	-	-	-	1	1	8	-	-	-	-	-	-	-	1																																																																																
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7																																																																																																																																																																																																																																																	
$k$	1	1	1	2	1	1	1	1																																																																																																																																																																																																																																																	
	2	-	1	2	2	1	1	1																																																																																																																																																																																																																																																	
	3	-	-	1	2	1	1	1																																																																																																																																																																																																																																																	
	4	-	-	-	1	1	1	1																																																																																																																																																																																																																																																	
	5	-	-	-	-	1	1	1																																																																																																																																																																																																																																																	
	6	-	-	-	-	-	1	1																																																																																																																																																																																																																																																	
	7	-	-	-	-	-	-	1																																																																																																																																																																																																																																																	
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8																																																																																																																																																																																																																																																
$k$	1	1	2	1	4	1	1	1	1																																																																																																																																																																																																																																																
	2	-	1	1	2	1	1	1	1																																																																																																																																																																																																																																																
	3	-	-	1	1	1	1	1	1																																																																																																																																																																																																																																																
	4	-	-	-	1	1	1	1	1																																																																																																																																																																																																																																																
	5	-	-	-	-	1	2	1	1																																																																																																																																																																																																																																																
	6	-	-	-	-	-	1	1	1																																																																																																																																																																																																																																																
	7	-	-	-	-	-	-	1	1																																																																																																																																																																																																																																																
	8	-	-	-	-	-	-	-	1																																																																																																																																																																																																																																																
<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="9" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">9</td></tr> <tr><td rowspan="9" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">9</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(h) <math>\eta_1(9, n; k)</math></p>			$n$											1	2	3	4	5	6	7	8	9	$k$	1	1	1	5	3	1	1	1	1	1	2	-	1	4	1	3	1	1	1	1	3	-	-	1	?	1	1	1	1	1	4	-	-	-	1	?	5	1	1	1	5	-	-	-	-	1	4	1	1	1	6	-	-	-	-	-	1	1	1	1	7	-	-	-	-	-	-	1	1	1	8	-	-	-	-	-	-	-	1	1	9	-	-	-	-	-	-	-	-	1	<table style="border-collapse: collapse; margin: auto;"> <tr><td colspan="2"></td><td colspan="10" style="text-align: center;"><math>n</math></td></tr> <tr><td colspan="2"></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">9</td><td style="border: 1px solid black; padding: 2px;">10</td></tr> <tr><td rowspan="10" style="vertical-align: middle;"><math>k</math></td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">4</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">?</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">9</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 2px;">10</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">-</td><td style="border: 1px solid black; padding: 2px;">1</td></tr> </table> <p style="text-align: center;">(i) <math>\eta_1(10, n; k)</math></p>			$n$												1	2	3	4	5	6	7	8	9	10	$k$	1	1	2	2	2	5	1	1	1	1	1	2	-	1	?	3	3	1	1	1	1	1	3	-	-	1	?	1	2	1	1	1	1	4	-	-	-	1	?	3	1	1	1	1	5	-	-	-	-	1	?	2	1	1	1	6	-	-	-	-	-	1	?	1	1	1	7	-	-	-	-	-	-	1	2	1	1	8	-	-	-	-	-	-	-	1	1	1	9	-	-	-	-	-	-	-	-	1	1	10	-	-	-	-	-	-	-	-	-	1
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8	9																																																																																																																																																																																																																																															
$k$	1	1	1	5	3	1	1	1	1	1																																																																																																																																																																																																																																															
	2	-	1	4	1	3	1	1	1	1																																																																																																																																																																																																																																															
	3	-	-	1	?	1	1	1	1	1																																																																																																																																																																																																																																															
	4	-	-	-	1	?	5	1	1	1																																																																																																																																																																																																																																															
	5	-	-	-	-	1	4	1	1	1																																																																																																																																																																																																																																															
	6	-	-	-	-	-	1	1	1	1																																																																																																																																																																																																																																															
	7	-	-	-	-	-	-	1	1	1																																																																																																																																																																																																																																															
	8	-	-	-	-	-	-	-	1	1																																																																																																																																																																																																																																															
	9	-	-	-	-	-	-	-	-	1																																																																																																																																																																																																																																															
		$n$																																																																																																																																																																																																																																																							
		1	2	3	4	5	6	7	8	9	10																																																																																																																																																																																																																																														
$k$	1	1	2	2	2	5	1	1	1	1	1																																																																																																																																																																																																																																														
	2	-	1	?	3	3	1	1	1	1	1																																																																																																																																																																																																																																														
	3	-	-	1	?	1	2	1	1	1	1																																																																																																																																																																																																																																														
	4	-	-	-	1	?	3	1	1	1	1																																																																																																																																																																																																																																														
	5	-	-	-	-	1	?	2	1	1	1																																																																																																																																																																																																																																														
	6	-	-	-	-	-	1	?	1	1	1																																																																																																																																																																																																																																														
	7	-	-	-	-	-	-	1	2	1	1																																																																																																																																																																																																																																														
	8	-	-	-	-	-	-	-	1	1	1																																																																																																																																																																																																																																														
	9	-	-	-	-	-	-	-	-	1	1																																																																																																																																																																																																																																														
	10	-	-	-	-	-	-	-	-	-	1																																																																																																																																																																																																																																														

Table 6.8: Lottery characterisation numbers  $\eta_1(m, n; k)$  for the lotteries  $\langle m, n; k \rangle$ ,  $1 \leq k \leq n \leq m \leq 10$ . A question mark (?) indicates that the lottery characterisation number  $\eta_1(m, n; k)$  is not known.



# Chapter 7

## Conclusions

“The Moving Finger writes; and, having writ,  
Moves on: nor all your Piety nor Wit  
Shall lure it back to cancel half a Line.  
Not all your Tears wash out a Word of it.”  
*Edward Fitzgerald (1809–1883)*

This chapter consists of two sections. In the first (§7.1) a brief summary of the work contained in this dissertation is given, while in the second (§7.2) possible improvements to the work presented in this study as well as some ideas with respect to further work are outlined.

### 7.1 Dissertation summary

In Chapter 1 a brief summary of the history of lotteries, regarding their origins and progressive changes worldwide, was given. A collection of known lottery parameters in use across the world was presented in Table 1.1. Both the newly defined incomplete lottery and resource utilisation problems were presented formally, together with a brief literature survey of known research on the covering, packing and complete lottery problems. To the author’s knowledge, no reference to the incomplete lottery or resource utilisation problem exists in the combinatorial literature, rendering these problem formulations and the subsequent study thereof a novel contribution of this dissertation to the body of literature on lotteries.

The existence and some basic properties of the incomplete lottery and resource utilisation numbers,  $L_\psi(m, n; k)$  and  $\Psi_\ell(m, n; k)$  respectively, were derived in Chapter 2. A binary programming solution approach to both combinatorial problems was considered, although such an approach was concluded to be practically infeasible, even for relatively small values of the parameters  $m$ ,  $n$ ,  $k$  and  $\ell$ . From these problem formulations, the conclusion was drawn that the incomplete lottery and resource utilisation problems, being more general than the well-known complete lottery problem (in the sense that the problems are formulated in an incomplete or partial domination fashion), may be of considerably greater difficulty to solve than the complete lottery problem. Some basic values of the incomplete lottery number  $L_\psi(m, n; k)$  and resource utilisation number  $\Psi_\ell(m, n; k)$  was also established in Chapter 2.

The underlying methodological approach in this dissertation towards the incomplete lottery and resource utilisation problems was based on a graph theoretic interpretation of lottery schemes. Some basic concepts from graph theory and how the above two combinatorial optimisation problems translate to within this field, were presented in Chapter 3. This led to the definition of the so-called lottery graph  $G\langle m, n; k \rangle$ , not previously encountered in the literature on lotteries, thus achieving Objective I in §1.4. An algorithm for the symmetric representation of  $G\langle m, n; k \rangle$ , revealing the inherent symmetry of the lottery graph, was presented and used to draw lottery graphs  $G\langle m, n; k \rangle$  for  $1 \leq k \leq n \leq m \leq 10$  in Figure 3.11. Furthermore, a characterisation of exactly when the complete lottery number  $L_1(m, n; k) = 1, 2$  or  $3$  was



also established. Using graph domination theory, upper bounds on and explicit values of the complete lottery number  $L_1(m, n; k)$  for  $1 \leq k \leq n \leq m \leq 10$  were found, as presented in Table 3.1.

The combinatorial literature survey performed on the covering, packing and complete lottery problems in §1.3, was intensified in Chapter 4, to focus specifically on known analytic bounds on the complete lottery number  $L_1(m, n; k)$  derived from results on the lower domination parameter,  $\gamma(\mathcal{G})$ , of a graph  $\mathcal{G}$ . The relative merit of these bounds were presented in Table 4.1, where only the best bounds on  $L_1(m, n; k)$  for  $6 \leq m \leq 50$  and  $1 \leq k \leq n \leq 6$  were displayed. A number of bounds were also presented in a comparative fashion for the lottery class  $\langle m, 6; 3 \rangle$ , where  $6 \leq m \leq 50$ , in Table 4.2, in partial fulfilment of Objective II in §1.4.

Seven algorithmic solution approaches for the determination of lower and upper bounds on respectively the resource utilisation number  $\Psi_\ell(m, n; k)$  and incomplete lottery number  $L_\psi(m, n; k)$  were described in Chapter 5. Each of these algorithms was evaluated in terms of its performance relative to the other algorithms and its worst case complexity measure, rounding off the requirement stipulated by Objective II in §1.4. Chapter 5 also extends toward partially accomplishing Objective III. More specifically, upper bounds (via Algorithms 2–7) on  $L_1(m, 5; 2)$ , for  $5 \leq m \leq 25$ , were presented in Table 5.3, while lower bounds on  $\Psi_\ell(m, n; k)$  for  $1 \leq k < n < m \leq 10$  and  $2 \leq \ell \leq L_1(m, n; k)$ , via Algorithms 2–7 were presented in Table 5.2. For the sake of completeness, the best known upper and lower bounds on  $L_1(m, n; k)$ , gathered from Internet covering and lottery number repository sites, were also included in Table B.2.

Another novel contribution of this dissertation is the interest expressed in the number of structurally different  $L_\psi(m, n; k)$ -sets and  $\Psi_\ell(m, n; k)$ -sets for  $\langle m, n; k \rangle$ , leading to the description of two enumeration methods for finding and thus characterising all distinct  $L_\psi(m, n; k)$ -set and  $\Psi_\ell(m, n; k)$ -set structures for  $\langle m, n; k \rangle$  (performed in Chapter 6). The characterisation procedure involved the use of a newly defined lottery/nauty tree and the lottery characterisation number  $\eta_\psi(m, n; k)$  for  $\langle m, n; k \rangle$ . Only small instances of  $\eta_\psi(m, n; k)$  were investigated, due to the computationally intensive nature of the enumeration methods. An in-depth study of the parameter jump sequences involved in the growth patterns of the characterisation number  $\eta_\psi(m, n; k)$  was also performed, wrapping up the requirements of Objective III in §1.4.

The following section is devoted to achieving Objective IV in §1.4.

## 7.2 Possible future work

The author was excited by a number of possible future avenues of investigation during the course of conducting the research contained in this dissertation. Some of these possible improvements to or elaborations on the work presented in the previous chapters are outlined in this section. More specifically, fifteen questions are posed as a challenge to possible future investigators.

The method described for finding all possible distinct  $L_\psi(m, n; k)$ -set characterisations utilised the notion of a so-called lottery tree, as described in §6.1.1. Although the pruning rules affected a considerable improvement in the execution time of the characterisation algorithm (Algorithm 8), as is evident from the table in Figure 6.4(c), other pruning rules certainly also exist.

**Question 7.1** Consider an overlapping  $n$ -set structure  $\vec{X}^{(\ell)}$  of cardinality  $\ell < L_\psi(m, n; k)$  in the lottery tree. Is it possible to predetermine whether or not the descendants of  $\vec{X}^{(\ell)}$  would yield any  $L_\psi(m, n; k)$ -set structure characterisations further down in the lottery tree? ■

If Question 7.1 may be answered in the affirmative,  $\vec{X}^{(\ell)}$  and its descendants may be removed from the lottery tree in cases where it represents a node not capable of producing an  $L_\psi(m, n; k)$ -set structure further down in the lottery tree, thereby avoiding further traversals down its branch of “dead” descendants. For small values of  $\ell$  (i.e., high up in the lottery tree) it may well be very difficult to answer the above question in the affirmative (due to the considerable structural changes affected to  $\vec{X}^{(\ell)}$  in subsequent levels of the lottery tree). However, if it were possible to answer this question in the affirmative (especially for small values of  $\ell$ ), one would be able to speed up the characterisation procedure significantly.

The following question relates to which of the two characterisation techniques described in §6.1 utilises a more computationally efficient representation of playing set structures.

**Question 7.2** *Does the fact that you only have to search through  $(m + L_\psi(m, n; k))$  elements (and hence using the automorphism software package `nauty`) improve the performance of finding all  $\eta_\psi(m, n; k)$   $L_\psi(m, n; k)$ -set structures for  $\langle m, n; k \rangle$ ? Furthermore, to what extent may this improvement be exploited to establish unknown  $L_\psi(m, n; k)$  and  $L_\psi(m, n; k)$ -set structures?* ■

One important factor to include in the investigation of this computational efficiency (as mentioned in Chapter 6), would be the respective values of the parameters  $m$  and  $\ell$  at which one characterisation method (and its playing set structure representation) is better suited than the other.

Next, consider the resource utilisation problem. The establishment of this problem most probably embodies the single component of this dissertation from which the largest amount of subsequent research could evolve. More specifically, the following question is posed.

**Question 7.3** *Is it possible to formulate improved theoretical bounds on the resource utilisation number  $\Psi_\ell(m, n; k)$ , or even algorithmically determine explicit resource utilisation numbers for larger values of  $m, n, k$  and  $\ell$  in certain special cases?* ■

This may well be pursued by the establishment of either different heuristic algorithmic solution approaches to those presented in Chapter 5, or by considering modifications to the existing optimisation techniques. One possible improvement of the intelligent genetic algorithm implementation (Algorithm 7), for example, might be to incorporate a candidate list protocol (similar to candidate list strategies used in conjunction with the tabu search optimisation method). Such a protocol definition would be able to intensify locally optimal candidate solutions by considering only a selection of all possible gene exchanges between parent candidates in the genetic crossover procedure (as opposed to considering *all* possible gene exchanges), thus reducing computational complexity of the method.

The next question involves the parallelisation of the algorithms in Chapter 5. In particular, Algorithms 6 and 7 are computationally intensive, mainly due to the extremely high number of resource utilisation calculations that have to be performed. However, both algorithms lend themselves to a parallel implementation, especially on a MOSIX cluster system (such as the one described in Chapter 5). Somewhat imprecisely speaking, algorithms may be implemented using a client–server approach, where client systems are designed specifically to determine the computationally intensive components of an algorithm (for example, determining the resource utilisation of a playing set) and a single server system handing out jobs to any available clients (for example, the passing of a playing set to a client system for the determination of its resource utilisation). For instance, if a number  $\kappa$  of client systems are utilised (by a server system) to perform the resource utilisation of  $\kappa$  playing sets (in parallel), the result may potentially speed up the execution time by the factor  $\kappa$ , as opposed to the (serial) determination of the resource utilisation of  $\kappa$  playing sets by any stand-alone system or process, as performed by the algorithms in Chapter 5. Of course, a more specific protocol will need to be investigated, which includes the effective management of memory and hardware resources (process migration slows down execution time and is performed whenever hardware access is required [12]). Recently such investigations have been initiated with the development of a tool called `autoson` [159] which schedules single or multi user execution of processes across a network of UNIX workstations (the flavours of UNIX systems supported may include (a mixture of): Solaris One and Solaris Two on Sun workstations; IRIX on Silicon Graphics mips workstations; OSF1 on Dec Alpha workstations; ULTRIX on Decstations; Hewlett–Packard HP–UX computers; the Linux operating system).

Another parallelisation approach may be to utilise a distributed system over the Internet. This resolution is motivated by a paper by Atkins, *et al.* [10] where the so-called “RSA–129 challenge”<sup>1</sup> was solved with the use of approximately 1 600 computers worldwide. The factorisation problem was subdivided into numerous small parts and sent to Internet volunteers for calculations on their computers, in their own

<sup>1</sup>The “RSA–129 challenge” consisted of the factorisation of a 129–digit integer into two prime factors of 64 and 65 digits respectively. The RSA cipher (named after its creators: R Rivest, A Shamir and L Adleman) was invented in 1977 [210], and its security is based on the presumed intractability of prime factorisation.

time<sup>2</sup>. Once results were received from volunteers, data integrity checks were performed and a huge factor base was constructed, thereby eventually solving the factorisation problem via the quadratic sieve method. The time of factorisation was diminished from an estimated  $4 \times 10^{13}$  years (as calculated by Rivest in [85]) to a mere 8 months (from August 1993 to April 2, 1994). The following question is therefore posed.

**Question 7.4** *May significant improvements in execution times be achieved by implementing the heuristic algorithmic solution approaches of Chapter 5 (in particular the tabu search and genetic algorithm) or the characterisation algorithm of Chapter 6, incorporating the parallel distribution of a client-server system, by using a computer cluster configuration or via Internet distribution?* ■

The genetic algorithmic approach towards determining upper bounds on  $L_\psi(m, n; k)$  and lower bounds on  $\Psi_\ell(m, n; k)$  may also be adapted to incorporate dynamic elements, similar to those present in the tabu search implementation (such as, using a variable tabu tenure). Here, for example, one could use a dynamic mutation parameter that may be used to deter the solution process away from local optima in the following way: When the average population fitness converges, the mutation parameter may be increased (albeit the number of elements to alter per chromosome, or the number of chromosomes altogether, or a combination of both) to ensure that more diverse candidates are formed in an attempt at avoiding so-called inbreeding. Another interesting dynamic and practical technique intuitive to genetic algorithms is to allow the formation of so-called species within a certain population (also referred to as *speciation*). When fitness convergence reaches some minimum threshold, the genetic propagation of chromosomes may be performed across different species, thereby hopefully perturbing current solutions away from locality. These dynamic genetic algorithm alterations lead to the following question.

**Question 7.5** *What effect would the incorporation of dynamic mutation and/or crossover procedures have on algorithmic solutions to the incomplete lottery or resource utilisation problems of Definitions 1.2 and 1.3?* ■

The next question investigates an extension of the domination test performed at level  $L_\psi(m, n; k) + 1$  in the lottery tree (as described in §6.1.1). More specifically, the question inquires about the particular  $L_\psi(m, n; k)$ -set structure amongst the  $\eta_\psi(m, n; k)$  incomplete lottery set structures that guarantees a  $k$ -prize for the lottery  $\langle m, n; k \rangle$ , that would additionally yield the greatest probability of winning a  $(k+i)$ -prize ( $i = 1, \dots, n-k-1$ ) in  $\langle m, n; k+i \rangle$ . Obviously it would be more beneficial if a participant were able to choose from a selection of playing sets for  $\langle m, n; k \rangle$ , in such a way that the probability of winning more than just a  $k$ -prize is maximised. The following more general question therefore arises.

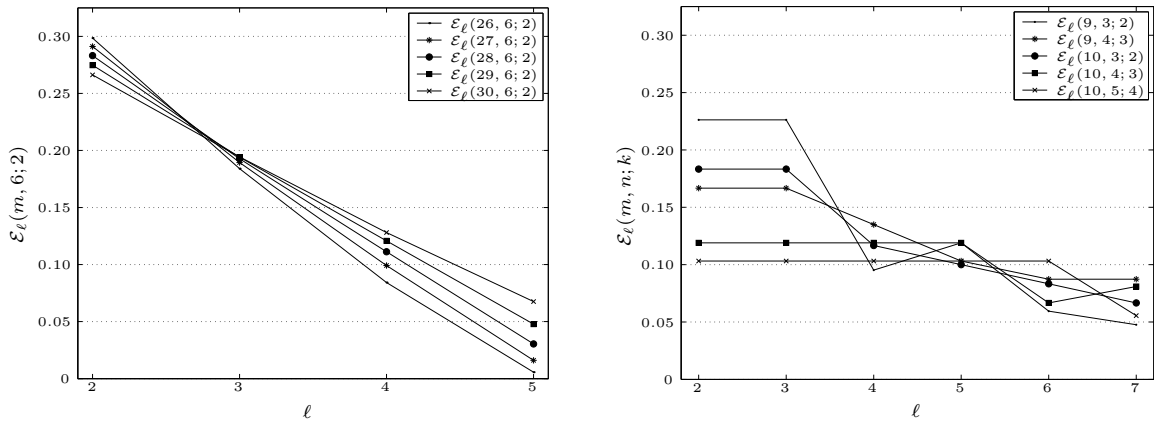
**Question 7.6** *Suppose the set  $\mathcal{S}$  contains all  $\eta_\psi(m, n; k)$   $L_\psi(m, n; k)$ -sets for  $\langle m, n; k \rangle$ . Which  $s^* \in \mathcal{S}$  yields a maximal value for the resource utilised in the lottery  $\langle m, n; k+i \rangle$  by a subset of  $\Phi(\mathcal{U}_m, n)$  of cardinality  $L_\psi(m, n; k)$ , where  $i = 1, \dots, n-k-1$ ?* ■

Question 7.6 may, in fact, be generalised. Throughout this dissertation, the incomplete lottery problem was attempted for a specific case of the lottery parameters  $m$ ,  $n$ ,  $k$  and  $\psi$ . However, in practical situations it may occur that a participant of the lottery scheme  $\langle m, n; k \rangle$  wishes to maximise his/her chance of winning a collection of prizes according to a specified set of weighting preferences, rather than a specific  $k$ -prize. Hence, the following question arises.

**Question 7.7** *Suppose  $\mathcal{L}$  is a playing set of cardinality  $\ell$  for the lottery  $\langle m, n; k \rangle$ . Let  $\rho_{\mathcal{L}}^i$  denote the probability of the playing set  $\mathcal{L}$  winning an  $i$ -prize ( $k \leq i \leq n$ ) in the lottery  $\langle m, n; i \rangle$ . Also, let  $\epsilon_i, \epsilon_{i+1}, \dots, \epsilon_{n-1}$  be real numbers (greater than or equal to zero), such that  $\sum_{i=k}^n \epsilon_i = 1$ , capturing the lottery participant's preference of winning an  $i$ -prize, for all  $i = k, \dots, n$ . What playing set  $\mathcal{L}$  would maximise*

$$\mathcal{W}_{\mathcal{L}} = \epsilon_k \rho_{\mathcal{L}}^k + \epsilon_{k+1} \rho_{\mathcal{L}}^{k+1} + \dots + \epsilon_{n-1} \rho_{\mathcal{L}}^{n-1} = \sum_{i=k}^{n-1} \epsilon_i \rho_{\mathcal{L}}^i,$$

<sup>2</sup>The Internet advertisement posted by Atkins, *et al.* [10] read, "Your donations of idle cycles on your PC's, workstations, supercomputers and fax machines may not be tax deductible, but they are truly a charitable donation."



(a) Values for the ticket efficiency number  $\mathcal{E}_\ell(m, 6; 2)$  for  $1 < \ell \leq 5 = L_1(m, 6; 2)$  and  $26 \leq m \leq 30$ .

(b) Values for the ticket efficiency number  $\mathcal{E}_\ell(m, n; k)$  for small lotteries where  $1 < \ell \leq 7$  and  $m = 9, 10$ .

Figure 7.1: Investigation into the ticket efficiency number  $\mathcal{E}_\ell(m, n; k)$  for the lotteries (a)  $\langle m, 6; 2 \rangle$  where  $26 \leq m \leq 35$  and  $2 \leq \ell \leq 5 = L_1(m, 6; 2)$  and (b)  $\langle m, n; k \rangle$  where  $m = 9, 10$  and  $2 \leq \ell \leq 7$  for the indicated values of  $n$  and  $k$ .

giving the participant the best chance of winning a combination of  $i$ -prizes as determined by the preferences  $\epsilon_i$ , where  $i = k, \dots, n-1$ ? ■

Practical situations may sometimes occur in which a participant of a lottery is interested to know how much he/she will gain (in terms of the probability of winning a  $k$ -prize) from buying one additional ticket ( $n$ -set). Consider a playing set  $\mathcal{L}^{(\ell)}$  (of cardinality  $\ell$ ) that yields a resource utilisation of  $\Psi_\ell(m, n; k)$  for the lottery  $\langle m, n; k \rangle$ . By a unit increase in the cardinality of  $\mathcal{L}^{(\ell)}$  (whether by adding a specific  $n$ -set from  $\Phi(\mathcal{U}_m, n)$  to  $\mathcal{L}^{(\ell)}$ , or by constructing a (completely) new set  $\mathcal{L}^{(\ell+1)}$  of cardinality  $\ell+1$ ), what is the maximal difference that may be witnessed in the resource utilisation? In order to be able to study this problem, define the ticket efficiency number  $\mathcal{E}_\ell(m, n; k)$  as

$$\mathcal{E}_\ell(m, n; k) = \Psi_\ell(m, n; k) - \Psi_{\ell-1}(m, n; k). \quad (7.1)$$

Graphs of the parameter  $\mathcal{E}_\ell(m, n; k)$  against  $\ell$  are shown in Figure 7.1 for specific values of the parameters  $m$ ,  $n$  and  $k$ . The following question is posed.

**Question 7.8** What conclusion may be drawn (if any) about the efficiency of incomplete lottery sets by studying the ticket efficiency number  $\mathcal{E}_\ell(m, n; k)$  in (7.1), or a normalised version thereof, where  $1 \leq k \leq n \leq m$  as  $\ell$  varies within the range  $1 < \ell \leq L_1(m, n; k)$ ? ■

It is intuitively expected that the value of  $\mathcal{E}_\ell(m, n; k)$ , for any fixed  $m$ ,  $n$  and  $k$ , will decrease as the cardinality of the  $L_\psi(m, n; k)$ -sets increase (see, for example, Figure 7.1(a) or Table 5.1 for  $\langle 20, 4; 3 \rangle$  containing lower bounds on  $\Psi_\ell(20, 4; 3)$  as a function of  $\ell$ ), implying that  $\mathcal{E}_\ell(m, n; k) \geq \mathcal{E}_{\ell+1}(m, n; k)$ . However, this is not always the case, as may be seen in Figure 7.1(b) for the lotteries  $\langle 9, 3; 2 \rangle$  and  $\langle 10, 4; 3 \rangle$ , where  $\mathcal{E}_4(9, 3; 2) \leq \mathcal{E}_5(9, 3; 2)$  and  $\mathcal{E}_6(10, 4; 3) \leq \mathcal{E}_7(10, 4; 3)$ . An understanding of this phenomenon perhaps requires some knowledge of the role of the relative divisibility between the parameters  $m$ ,  $n$ ,  $k$  and  $\ell$  or familiarity with the finer aspects of set theory. It would also be interesting to establish growth patterns for  $\mathcal{E}_\ell(m, n; k)$  with respect to variations in the parameters  $m$ ,  $n$ ,  $k$  and  $\ell$  (similar to those established for  $L_\psi(m, n; k)$ ,  $\Psi_\ell(m, n; k)$  and  $\eta_\psi(m, n; k)$ ).

The complete lottery problem may be viewed in an alternative, generalised context to that posed in Definition 1.1. Suppose the governing body of a lottery is allowed to select a winning  $t$ -set randomly from  $\mathcal{U}_m$ , while a participant constructs a playing set consisting of  $n$ -sets from  $\mathcal{U}_m$  (recall the discussion

in Chapter 1, footnote 3). A  $k$ -prize is awarded if at least one of the participant's  $n$ -sets has  $k$  elements in common with the winning  $t$ -set. This leads to the following more general, four-parameter complete lottery problem definition.

**Definition 7.1 (The four-parameter, complete lottery problem)** *Define a four-parameter, complete lottery set for  $\langle m, n, t; k \rangle$  as a subset  $\mathcal{L} \subseteq \Phi(\mathcal{U}_m, n)$  with the property that, for any element  $\phi_t \in \Phi(\mathcal{U}_m, t)$ , there exists an element  $l \in \mathcal{L}$  such that  $\Phi(\phi_t, k) \cap \Phi(l, k) \neq \emptyset$ . The four-parameter, complete lottery problem is: What is the smallest possible cardinality of a four-parameter, complete lottery set  $\mathcal{L}$ ? Denote the answer to this question by the four-parameter, complete lottery number  $L(m, n, t; k)$ .* ■

A four-parameter, complete lottery set  $\mathcal{L}$  of minimum cardinality  $L(m, n, t; k)$  is called an  $L(m, n, t; k)$ -set for the lottery  $\langle m, n, t; k \rangle$ . In the case where  $n = t$ , the four-parameter, complete lottery number  $L(m, n, n; k)$  is equivalent to the complete lottery number  $L_1(m, n; k)$ , considered in this dissertation. With Definition 7.1 in mind, the following question is posed.

**Question 7.9** *Is there a significant advantage (especially when considering upper bounds on the complete lottery number) in viewing the lottery problem in the generalised, four-parameter, complete context of Definition 7.1?* ■

The answer to the above question seems to be “yes,” especially in view of the following recursive result [13].

**Proposition 7.1** *Let  $m, n, t, k, m_1, m_2, t_1$  and  $t_2$  be positive integers such that  $k \leq t \leq n \leq m$ ,  $m_1 + m_2 = m$  and  $t_1 + t_2 = t + 1$ . Then  $L(m, n, t; k) \leq L(m_1, n, t_1; k) + L(m_2, n, t_2; k)$ .* ■

Of course, a generalisation of the four-parameter, complete lottery problem (in Definition 7.1) to an incomplete context (as considered for  $\langle m, n; k \rangle$  in this dissertation) is also possible. The decomposition result of Proposition 7.1 spawns the following inquiry as to whether larger lotteries are essentially comprised of various smaller lotteries.

**Question 7.10** *Let  $m = m_1 + m_2$ . Is it possible only to consider optimisation of playing sets that yield maximum resource utilisation for  $\langle m_1, n; k \rangle$  and  $\langle m_2, n; k \rangle$  separately when searching for  $L_\psi(m, n; k)$ -sets in  $\langle m, n; k \rangle$ ?* ■

A further generalisation of the incomplete lottery problem may also be explored, when considering the following alternative definition, which is motivated by work presented in [66] and by Bertolo, *et al.* [23] on the concept of the *general covering number*.

**Definition 7.2 (The  $\{\lambda, \psi\}$ -lottery problem)** *Define a  $\{\lambda, \psi\}$ -lottery set for  $\langle m, n; k \rangle$  as a subset  $\mathcal{L} \subseteq \Phi(\mathcal{U}_m, n)$  with the property that there exists some subset  $\mathcal{V}_\psi \subseteq \Phi(\mathcal{U}_m, n)$  of cardinality at least  $\lceil \psi \binom{m}{n} \rceil$  such that, for any element  $\phi_n \in \mathcal{V}_\psi$ , there exists at least  $\lambda \geq 1$  elements  $\{l_1, \dots, l_\lambda\} \in \mathcal{L}$  such that  $\Phi(\phi_n, k) \cap \Phi(l_i, k) \neq \emptyset$  (for all  $i = 1, \dots, \lambda$ ). The  $\{\lambda, \psi\}$ -lottery problem is: What is the smallest possible cardinality of a  $\{\lambda, \psi\}$ -lottery set  $\mathcal{L}$ ? Denote the answer to this question by the  $\{\lambda, \psi\}$ -lottery number  $L_\psi^\lambda(m, n; k)$ .* ■

A minimum cardinality  $\{\lambda, \psi\}$ -lottery set may be called an  $L_\psi^\lambda(m, n; k)$ -set for  $\langle m, n; k \rangle$ , while the  $\{\lambda, \psi\}$ -lottery characterisation number  $\eta_\psi^\lambda(m, n; k)$  would denote all structurally different  $L_\psi^\lambda(m, n; k)$ -sets. In this definition of the lottery problem, the participant is interested in *at least*  $\lambda \geq 1$   $k$ -prizes (as opposed to the minimum constraint in Definition 1.2 of having at least 1  $n$ -set from the participant's playing set). In the case where  $\lambda = 1$ , the  $\{1, \psi\}$ -lottery problem is equivalent to the incomplete lottery problem for  $\langle m, n; k \rangle$ , considered in this dissertation. A similar generalisation of the resource utilisation parameter  $\Psi_\ell(m, n; k)$  may also be formulated using Definition 7.2. Of course, a generalisation involving a combination of Definitions 7.1 and 7.2 may also be interesting. It is obvious why the following proposition, relating the parameters  $L_\psi(m, n; k)$  and  $L_\psi^\lambda(m, n; k)$ , holds.

**Proposition 7.2** *Let  $m, n, k$  be positive integers such that  $1 \leq k \leq n \leq m$  for the lottery  $\langle m, n; k \rangle$ . For any  $0 < \psi \leq 1$  and  $\lambda \geq 1$ ,  $L_\psi(m, n; k) \leq L_\psi^\lambda(m, n; k)$ . ■*

In view of this new lottery definition, the following question arises.

**Question 7.11** *Is there any practical or combinatorial advantage in viewing the incomplete lottery and resource utilisation problems in the generalised, incomplete context as in Definition 7.2? ■*

Due to substantial evidence gathered in analytical arguments and algorithmic computations when determining certain values of  $L_\psi(m, n; k)$  (in Chapter 2), the following is conjectured.

**Conjecture 7.1**  $L_\psi(m, n; k) = \Omega(m^k)$ . ■

This conjecture is motivated by the fact that  $L_\psi(m, n; 1) = \Omega(m)$  (Corollary 2.2) and  $L_\psi(m, n; n) = \Omega(m^n)$  (Theorem 2.3). Furthermore, in Proposition 6.1(b) it was shown that  $L_1(m, n; k) = \Omega(m^2)$  for all  $k \geq 2$ . This bound is, however, considered conservative in the sense that it is expected to hold for general values of  $\psi$  (implying that  $L_\psi(m, n; 2) = \Omega(m^2)$ ).

**Question 7.12** *Is it possible to prove Conjecture 7.1, or in the event that the claim is false, produce a counter example and give a valid asymptotic lower bound? ■*

In Chapter 6 the growth properties of the incomplete lottery characterisation number  $\eta_\psi(m, n; k)$  was determined for variations of the parameters  $k$  and  $\psi$ . However, the author was unable to establish the growth properties of  $\eta_\psi(m, n; k)$  when the parameters  $m$  and  $n$  are varied, hence the formulation of Conjecture 6.1. This gives rise to the following question.

**Question 7.13** *Is it possible to prove Conjecture 6.1? ■*

The lottery graph  $G\langle m, n; k \rangle$  has a very symmetric structure (as already noted in §3.3). Another interesting question surrounding the symmetry of  $G\langle m, n; k \rangle$  realises in studying the core of  $G\langle m, n; k \rangle$ . However, before continuing some preliminary theory and notation regarding graph cores are required.

For any two graphs  $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$  and  $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$ , a function  $h : V_{\mathcal{G}} \mapsto V_{\mathcal{H}}$  mapping the vertices of  $\mathcal{G}$  to the vertices of  $\mathcal{H}$  such that  $h(v)$  and  $h(u)$  are adjacent in  $\mathcal{H}$  whenever  $v$  and  $u$  are adjacent in  $\mathcal{G}$ , is called a **homomorphism** from  $\mathcal{G}$  to  $\mathcal{H}$  [92]. The existence of a homomorphism from  $\mathcal{G}$  to  $\mathcal{H}$  is denoted by  $\mathcal{G} \rightsquigarrow \mathcal{H}$ . For example, the function  $h^* : V(\mathcal{G}_5) \mapsto V(C_3) = \{u_1^*, u_2^*, u_3^*\}$  where  $h^{-1}(u_1^*) = \{u_1, u_7\}$ ,  $h^{-1}(u_2^*) = \{u_2, u_4, u_5\}$  and  $h^{-1}(u_3^*) = \{u_3, u_6, u_8\}$  is a homomorphism from  $\mathcal{G}_5$  (in Figure 3.3(b)) to  $C_3$ , so that  $\mathcal{G}_5 \rightsquigarrow C_3$ . It is easy to see that every isomorphism is a homomorphism, although the converse is not always true. A graph  $\mathcal{G}$  is called a **core** if any homomorphism from  $\mathcal{G}$  onto itself is necessarily a bijection<sup>3</sup> [104]. A vertex-induced subgraph  $\mathcal{G}^\bullet$  of  $\mathcal{G}$  (which is minimal with respect to graph inclusion) is called a **core of  $\mathcal{G}$**  if  $\mathcal{G}^\bullet$  is a core and  $\mathcal{G} \rightsquigarrow \mathcal{G}^\bullet$ . A core  $\mathcal{G}^\bullet$  of a graph  $\mathcal{G}$  is called **trivial** if  $\mathcal{G} \simeq \mathcal{G}^\bullet$ . For the graph  $K_8$  in Figure 3.1(b),  $K_8^\bullet \simeq K_8$  is a trivial core, while  $G_5^\bullet \simeq C_3$  is a non-trivial core with a homomorphism from  $\mathcal{G}_5$  to  $C_3$  given by  $h^* : V(\mathcal{G}_5) \mapsto V(C_3)$  for the graph  $\mathcal{G}_5$  in Figure 3.3(b). It is possible to show, by means of a simple example, that  $\overline{\mathcal{G}^\bullet} \not\rightsquigarrow \mathcal{G}^\bullet$  for a graph  $\mathcal{G}$ , in general.

Hell & Nešetřil [104] showed the following decision problem (for a general graph  $\mathcal{G}$  that is not bi-partite) to be **NP**-complete: “Given a graph  $\mathcal{G}$  and a homomorphism from  $\mathcal{G}$  to  $\mathcal{H}$ , is  $\mathcal{G}$  not a core?” However, known properties of the cores of the class of vertex-transitive graphs (which includes the subclass of lottery graphs  $G\langle m, n; k \rangle$ ) render the search for  $G\langle m, n; k \rangle^\bullet$  more attractive. These properties are summarised in Theorem 7.1.

<sup>3</sup>This definition of a graph core differs from that given by Morgan & Slater [177] in 1980. Instead, the core of a graph  $\mathcal{G}$  was defined there to be a path  $\mathcal{P}$ , with the property of minimising  $d_{\mathcal{G}}(\mathcal{P}) = \sum_{v \in V(\mathcal{G})} d_{\mathcal{G}}(v, \mathcal{P})$ , where  $d_{\mathcal{G}}(v, \mathcal{P})$  denotes the distance from any vertex  $v \in V(\mathcal{G})$  to the path  $\mathcal{P}$ .

Lottery graph $G\langle m, n; k \rangle$	$L_1(m, n; k)$ $= \gamma(G\langle m, n; k \rangle)$	$\binom{m}{n}$	Core of lottery graph $G\langle m, n; k \rangle^\bullet$	$\gamma(G\langle m, n; k \rangle^\bullet)$	$ V_G^\bullet $
$G\langle 4, 2; 1 \rangle$	2	6	$\simeq K_3$	1	3
$G\langle 5, 2; 1 \rangle$	2	10	$\simeq G\langle 5, 2; 1 \rangle$	2	10
$G\langle 6, 2; 1 \rangle$	3	15	$\simeq K_5$	1	5
$G\langle 6, 3; 1 \rangle$	2	20	$\simeq K_{10}$	1	10
$G\langle 6, 3; 2 \rangle$	2	20	$\simeq G\langle 6, 3; 2 \rangle$	2	20
$G\langle 7, 2; 1 \rangle$	3	21	$\simeq G\langle 7, 2; 1 \rangle$	3	21
$G\langle 7, 3; 1 \rangle$	2	35	$\simeq G\langle 7, 3; 1 \rangle$	2	35
$G\langle 7, 3; 2 \rangle$	4	35	$\simeq G\langle 7, 3; 2 \rangle$	4	35
$G\langle 8, 2; 1 \rangle$	4	28	$\simeq K_7$	1	7
$G\langle 8, 4; 1 \rangle$	2	70	$\simeq K_{35}$	1	35
$G\langle 9, 2; 1 \rangle$	4	36	$\simeq G\langle 9, 2; 1 \rangle$	4	36
$G\langle 10, 2; 1 \rangle$	5	45	$\simeq K_9$	1	9

Table 7.1: The core of the lottery graph  $G\langle m, n; k \rangle$ , denoted by  $G\langle m, n; k \rangle^\bullet$ , for some small values of the parameters  $1 \leq k < n < m \leq 10$  and  $n \leq \lfloor \frac{m}{2} \rfloor$ .

**Theorem 7.1 (The core of a vertex-transitive graph [92])**

Suppose the simple graph  $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$  is vertex-transitive with core given by  $\mathcal{G}^\bullet = (V_{\mathcal{G}}^\bullet, E_{\mathcal{G}}^\bullet)$ . Then

- (1)  $\mathcal{G}^\bullet$  is unique (up to isomorphism), vertex-induced and vertex-transitive;
- (2)  $|V_{\mathcal{G}}^\bullet|$  divides  $|V_{\mathcal{G}}|$ ; and
- (3)  $\mathcal{G}^\bullet$  is a trivial core if  $|V_{\mathcal{G}}|$  is prime. ■

It is possible, by using parent-child search tree algorithms (for example) to

- (1) determine all possible vertex-induced subgraphs  $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$  of  $G\langle m, n; k \rangle$  such that  $|\mathcal{H}|$  divides  $\binom{m}{n}$ ; and
- (2) determine whether  $V(G\langle m, n; k \rangle) \rightsquigarrow V_{\mathcal{H}}$  (i.e., there exists a mapping from the elements of  $\Phi(\mathcal{U}_m, n)$  in  $G\langle m, n; k \rangle$  to  $V_{\mathcal{H}}$ ).

The core,  $G\langle m, n; k \rangle^\bullet$ , of some lottery graphs could be determined (see Table 7.1). Figure 7.2 shows the construction of a search tree referred to in (1) above, employed to determine  $G\langle 5, 2; 1 \rangle^\bullet$  as an example. What is interesting to note from  $G\langle m, n; k \rangle^\bullet$ , however, is the apparent relationship between  $\gamma(G\langle m, n; k \rangle)$  and  $\gamma(G\langle m, n; k \rangle^\bullet)$ . The circumstantial evidence of this relationship leads to the final two questions.

**Question 7.14** *Is it true, for the lottery  $\langle m, n; k \rangle$ , that  $\gamma(G\langle m, n; k \rangle^\bullet) |V_G| = \gamma(G\langle m, n; k \rangle) |V_G^\bullet|$ , where  $G\langle m, n; k \rangle = (V_G, E_G)$  [ $G\langle m, n; k \rangle^\bullet = (V_G^\bullet, E_G^\bullet)$ ] denotes the lottery graph [core]? In terms of the lottery graph parameters, this question may be reformulated as follows. Is it true that*

$$L_1(m, n; k) = \frac{\overbrace{\gamma(G\langle m, n; k \rangle^\bullet)}^{\frac{1}{c}}}{|V_G^\bullet|} \binom{m}{n} \quad (7.2)$$

for all  $1 \leq k \leq n \leq m$ ? ■

If it were possible to answer Question 7.14 in the affirmative, then instead of attempting to find a dominating set for  $G\langle m, n; k \rangle$  (or equivalently, determining the complete lottery number  $L_1(m, n; k)$ ), one could focus on determining  $\gamma(G\langle m, n; k \rangle^\bullet)$  of the core of the lottery graph  $G\langle m, n; k \rangle$  (which is potentially much smaller than  $G\langle m, n; k \rangle$  itself) and utilise (7.2) to yield the exact value of  $L_1(m, n; k)$ .

**Question 7.15** *Is the core  $G\langle m, n; k \rangle^\bullet$  of the lottery graph  $G\langle m, n; k \rangle$  either a trivial core or isomorphic to  $K_c$  for some  $c \geq 3$ ? ■*

In fact, if this were the case, then it would be possible to map disjoint sets of  $\binom{m}{n}/c$  vertices in  $G\langle m, n; k \rangle$  to  $G\langle m, n; k \rangle^\bullet$ .

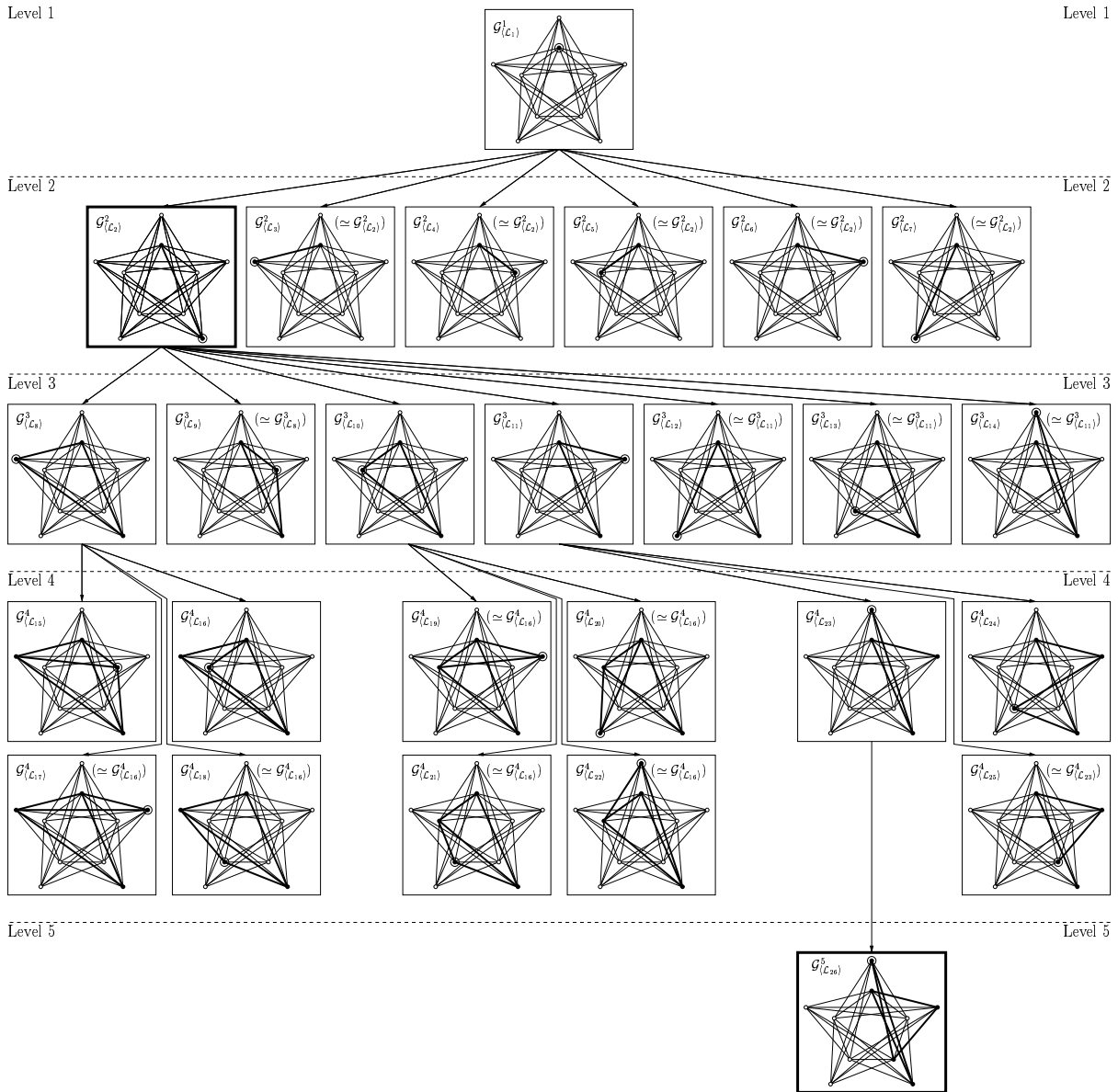


Figure 7.2: A graphical example of a parent–child search tree algorithm for determining a core,  $G\langle 5, 2; 1 \rangle^\bullet$ , for  $G\langle 5, 2; 1 \rangle$ . Level  $i$  of the search tree contains all possible non-isomorphic selections (when using *nauty* to discard isomorphisms) of  $i$  vertices from  $G\langle 5, 2; 1 \rangle$ . These vertex-induced subgraphs (of order  $i$  in  $G\langle 5, 2; 1 \rangle$ ) are obtained from the parent vertex by adding one possible vertex to the parent vertex-induced subgraph. The dark coloured vertices (and edges) in  $G\langle 5, 2; 1 \rangle$  denote the vertex-induced subgraph, while the encircled vertex depicts the new vertex addition from the parent vertex-induced subgraph. For completeness, the top left-hand corner of each node contains a label of the form  $\mathcal{G}_{\mathcal{L}_t}^s$ . This indicates the lottery graph  $G\langle 5, 2; 1 \rangle$  with a (vertex-induced) subgraph of order  $s = |\mathcal{L}_t|$  induced by the elements of  $\mathcal{L}_t$ . In some instances, *nauty* was able to find isomorphic representations of vertex-induced subgraphs, which are labelled in the top right-hand corner of each node in the search tree. No further search is therefore necessary down this branch of the search tree. Only the bold embossed nodes in the search tree ( $\mathcal{G}_{\mathcal{L}_2}^2$  and  $\mathcal{G}_{\mathcal{L}_{26}}^5$  in this case) represent unique, regular, vertex-induced subgraphs of order  $s (> 1)$  such that  $s \mid \binom{5}{2} = 10$  that need to be considered for homomorphism testing (i.e., whether  $V(G\langle 5, 2; 1 \rangle) \rightsquigarrow V(\mathcal{G}_{\mathcal{L}_t}^{|\mathcal{L}_t|})$  for  $t = 2$  and 26).





# References

- [1] AB SVENSKA SPEL, *Förstasidan*, [Online], [cited 2002, March 7], Available from: <http://www.svenskaspel.se/>
- [2] ACMEC, Sport Toto, [Online], [cited 2002, March 8], Available from: <http://www.sportstoto.com.my/>
- [3] RB ALLAN, R LASKAR & S HEDETNIEMI, *A note on total domination*, Discrete Mathematics, **49** (1984), pp. 7–13.
- [4] ALBERTA LOTTERY FUND, *Alberta Lottery Fund*, [Online], [cited 2003, November 2003], Available from: <http://albertalotteryfund.ca/>
- [5] AMTEC.NET, *Lotería de Concepción*, [Online], [cited 2002, March 7], Available from: <http://www.loteria.cl/>
- [6] AN POST NATIONAL LOTTERY COMPANY, *Lotto Results*, [Online], [cited 2002, November 5], Available from: <http://www.lotto.ie/>
- [7] ARIZONA LOTTERY, *Arizona Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.arizonalottery.com/>
- [8] VI ARNAUTOV, *Estimation of the exterior stability number of a graph by means of the minimal degree of the vertices* (Russian), Prikladnaya Matematika i Programirovanie, **11** (1974), pp. 3–8, 126.
- [9] ARTAGON, *Perlas*, [Online], [cited 2002, April 16], Available from: <http://www.olifeja.lt/>
- [10] D ATKINS, M GRAFF, AK LENSTRA & PC LEYLAND, *The Magic Words are Squeamish Ossifrage*, pp. 263–277 in J PIEPRZYK & R SAFAVI–NAINI (EDS): *Advances in Cryptology – ASIACRYPT ’94*, Lecture Notes in Computer Science, **917**, (1995).
- [11] ATLANTIC LOTTERY CORPORATION INC., *Atlantic Lottery Corporation*, [Online], [cited 2002, March 7], Available from: <http://www.alc.ca/>
- [12] A BARAK, *MOSIX*, [Online], [cited 2002, November 25], Available from: <http://www.mosix.org/>
- [13] JA BATE, *A Generalized Covering Problem*, PhD dissertation, University of Manitoba, Canada, (1978).
- [14] JA BATE & RG STANTON, *A survey of  $(2, 2, L, V)$  designs*, Congressus Numerantium, **31** (1981), pp. 3–15.
- [15] JA BATE & RG STANTON, *Some algorithmic results on  $(2, 3, 3, v)$  designs and  $(2, 4, 4, v)$  designs*, Utilitas Mathematica, **20** (1981), pp. 195–220.
- [16] JA BATE, PC LI & GHJ VAN REES, *Finally  $C(19, 6, 2) = 15$* , Congressus Numerantium, **141** (2002), pp. 95–102.
- [17] JA BATE & GHJ VAN REES, *Lotto Designs*, Journal of Combinatorial Mathematics and Combinatorial Computing, **28** (1998), pp. 15–39.

- [18] BELGIAN NATIONAL LOTTERY, *Belgian National Lottery*, [Online], [cited 2002, March 6], Available from: <http://www.loterie-nationale.be/> or <http://www.nationale-loterij.be/>
- [19] R BELIC, *Lotto Systems and Toto Systems to win Wheel Game*, [Online], [cited 2002, April 23], Available from: <http://www.xs4all.nl/~rbelic/>
- [20] C BERGE, *Graph Theory*, American Mathematical Monthly, **71** (1964), pp. 471–480.
- [21] C BERGE, *Graphs and Hypergraphs*, North-Holland, Amsterdam, (1973).
- [22] C BERGE, *Theory of Graphs and its Applications*, Methuen, London, (1962).
- [23] R BERTOLO, I BLUSKOV & H. HÄMÄLÄINEN, *Upper Bounds on the General Covering Number  $C_\lambda(v, k, t, m)$* , preprint (2003).
- [24] THE BIBLE GATEWAY, *Bible Gateway*, [Online], [cited 2002, March 27], Available from: <http://www.biblegateway.net/>
- [25] L BIEGLER, A RAGUNATHAN, Y LANG, V DE SMEDT, M GALATI, F MARGOT, M SALTZMAN & V AUSTEL, *COmputational INfrastructure for Operations Research*, [Online], [cited 2004, September 10], Available from: <http://oss.software.ibm.com/developerworks/opensource/coin/>
- [26] I BLUSKOV, M GREIG & K HEINRICH, *Infinite Classes of Covering Numbers*, Canadian Mathematical Bulletin, **43**(4) (2000), pp. 385–396.
- [27] I BLUSKOV & H HÄMÄLÄINEN, *New Upper Bounds on the Minimum Size of Covering Designs*, Journal of Combinatorial Designs, **6**(1) (1998), pp. 21–41.
- [28] I BLUSKOV & K HEINRICH, *General Upper Bounds on the Minimum Size of Covering Designs*, Journal of Combinatorial Theory Series A, **86**(2) (1999), pp. 205–213.
- [29] RE BRADLEY, *Euler and the Genoese Lottery*, [Online], [cited 2002, March 2], NEC RESEARCH INSTITUTE, *Research Index [NEC Research Institute, Citeseer, Computer Science]*, Available from: <http://www.citeseer.nj.nec.com/cs/>
- [30] BRAINYMEDIA.COM, *Famous Quotes and Quotations at BrainyQuote*, [Online], [cited 2004, December 14], Available from <http://www.brainyquote.com/>
- [31] RC BRIGHAM & RD DUTTON, *A Compilation of Relations between Graph Invariants*, Networks, **15**(1) (1985), pp. 73–107.
- [32] BRITISH COLUMBIA LOTTERY CORPORATION, *BC Lottery*, [Online], [cited 2002, March 7], Available from: <http://www.bclc.com/>
- [33] AE BROUWER, *Block Designs*, pp. 693–745 in RL GRAHAM, M GRÖTSCHEL & L LOVÁSZ (EDS): *Handbook of Combinatorics – Volume 1*, The MIT Press, North-Holland, (1995).
- [34] AE BROUWER, *On the packing of quadruples without common triples*, Ars Combinatoria, **5** (1978), pp. 3–6.
- [35] AE BROUWER, *Packing and Covering of  $\binom{k}{t}$ -sets*, pp. 89–97 in A SCHRIJVER (ED): *Packing and Covering in Combinatorics*, Mathematisch Centre Tracts, **106**, Mathematisch Centrum, Amsterdam, (1979).
- [36] AE BROUWER, *Some lotto numbers from an extension of Turán’s theorem*, Mathematisch Centrum Report, Amsterdam, (1981).
- [37] AE BROUWER & M VOORHOEVE, *Turán theory and the Lotto problem*, pp. 99–105 in A SCHRIJVER (ED): *Packing and Covering in Combinatorics*, Mathematisch Centre Tracts, **106**, Mathematisch Centrum, Amsterdam, (1979).
- [38] AP BURGER, WR GRÜNDLINGH & JH VAN VUUREN, *On the Optimality of Belic’s Lottery Designs*, to appear in Journal of Combinatorial Mathematics and Combinatorial Computing.

- 
- [39] AP BURGER, WR GRÜNDLINGH & JH VAN VUUREN, *Towards a Characterisation of Lottery Set Overlapping Structures*, to appear in *Ars Combinatoria*.
- [40] CAMELOT GROUP PLC, *The Official National Lottery Web Site*, [Online], [cited 2002, March 7], Available from: <http://www.national-lottery.co.uk/>
- [41] P.J. CAMERON, *Combinatorics: Topics, Techniques, Algorithms*, Cambridge University Press, London, (1994), pp. 43–44.
- [42] M. CAPOBIANCO & J.C. MOLLUZZO, *Examples and Counterexamples in Graph Theory*, Elsevier North–Holland, Inc., New York, (1978), pp. 89–101.
- [43] Y. CARO & Y. RODITTY, *On the vertex–independence number and star decomposition of graphs*, *Ars Combinatoria*, **20** (1985), pp. 167–180.
- [44] CENTER FOR COMMUNICATIONS RESEARCH, *CCR La Jolla Home Page*, [Online], [cited 2002, April 20], Available from: <http://www.ccrwest.org/>
- [45] G. CHARTRAND & O.E. OELLERMANN, *Applied Algorithmic Graph Theory*, McGraw–Hill, New York, (1993).
- [46] B. CHEN & S. ZHOU, *Domination number and neighbourhood conditions*, *Discrete Mathematics*, **195** (1999), pp. 81–91.
- [47] A. CHESNEAU, *La Française des jeux*, [Online], [cited 2002, March 6], Available from: <http://www.francaise-des-jeux.fr/>
- [48] CHINA LOTTERY, *China Lottery*, [Online], [cited 2002, March 11], Available from: <http://www.cp168.com/>
- [49] W.E. CLARK, D.C. FISHER, B. SHEKHTMAN & S. SUEN, *Upper bounds for the domination number of a graph*, *Congressus Numerantium*, **132** (1998), pp. 99–123, or [Online], [cited 2002, March 2], NEC RESEARCH INSTITUTE, *Research Index [NEC Research Institute, Citeseer, Computer Science]*, Available from: <http://www.citeseer.nj.nec.com/cs/>
- [50] C.J. COLBOURN, *Winning the Lottery*, pp. 578–584 in C.J. COLBOURN (ED): *The CRC Handbook of Combinatorial Designs*, CRC Press, Boca Raton, (1996).
- [51] CONNECTICUT LOTTERY CORPORATION, *CT Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.ctlottery.org/>
- [52] D. CVETKOVIĆ, P. ROWLINSON & S. SIMIĆ, *Eigenspaces of graphs*, Cambridge University Press, Cambridge, (1997), pp. 171–177.
- [53] DANSK TIPSTJENESTE AS, *Dansk Tipstjeneste*, [Online], [cited 2002, March 6], Available from: <http://www.tips.dk/>
- [54] DATA CONVERSION SOLUTIONS, INC., *Colorado Lottery and Powerball Drawing Results*, [Online], [cited 2002, March 8], Available from: <http://www.colotto.com/>
- [55] DATACALL SERVICES PTY LTD., *Irish Lottery Syndicates*, [Online], [cited 2002, March 6], Available from: <http://www.irishlotto.net/>
- [56] R. DAVIES & G.F. ROYLE, *Graph domination, tabu search and the football pool problem*, *Discrete Mathematics*, **74** (1997), pp. 217–228.
- [57] J. DAVIS, *Official Site For Garfield And Friends*, [Online], [cited 2003, November 1], Available from: <http://www.garfield.com/>
- [58] DC LOTTERY, *DC Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.dclottery.com/>
- [59] D. DE CAEN, *Extension of a theorem of Moon and Moser on complete subgraphs*, *Ars Combinatoria*, **16** (1983), pp. 5–10.

- [60] D DE CAEN, DL KREHER, SP RADZISZOWSKI & WH MILLS, *On the covering of  $t$ -sets with  $(t + 1)$ -sets:  $C(9, 5, 4)$  and  $C(10, 6, 5)$* , *Discrete Mathematics*, **92**(1–3) (1991), pp. 65–77.
- [61] G DE L'HOPITAL, *L'analyse des infiniment petits pour l'intelligence des lignes courbes*, 1696.
- [62] HM DEITEL & PJ DEITEL, *C++ How to program*, Prentice–Hall International, Inc., London, (1994).
- [63] DELOTTO, *Lotto*, [Online], [cited 2002, March 7], Available from: <http://www.lotto.nl/>
- [64] DEPARTAMENTO DE ADMINISTRAÇÃO DE LOTERIAS, *Caixa Econômica Federal*, [Online], [cited 2002, March 6], Available from: <http://www.caixa.gov.br/>
- [65] JW DI PAOLA, *Block Designs and Graph Theory*, *Journal of Combinatorial Theory*, **1** (1966), pp. 132–148.
- [66] JH DINITZ & DR STINSON (EDS), *Contemporary Design Theory: A Collection of Surveys*, John Wiley & Sons, Inc., New York, (1992).
- [67] DIRECCION NACIONAL DE LOTERIAS Y QUINIELAS, *Loteria Uruguay*, [Online], [cited 2002, March 11], Available from: <http://www.loteria.gub.uy/>
- [68] F DROESBEKE & M LORÉA, *On the lotto problem*, *European Journal of Operational Research*, **11**(1) (1982), pp. 21–25.
- [69] DYNAMIC SITE FRAMEWORK, *Pennsylvania Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.palottery.com/>
- [70] EJECUTIVO NACIONAL MINISTERIO DE DESARROLLO SOCIAL Y MEDIO, *Lotería Nacional Sociedad del Estado de la República Argentina*, [Online], [cited 2002, March 9], Available from: <http://www.loteria-nacional.gov.ar/>
- [71] P ERDÖS & H HANANI, *On a limit theorem in combinatorial analysis*, *Publicationes Mathematicae Debrecen*, **10** (1963), pp. 10–13.
- [72] P ERDÖS & J SPENCER, *Probabilistic Methods in Combinatorics: Probability and Mathematical Statistics*, **17**, Academic Press, New York, (1974).
- [73] EUROPEAN LOTTERY GUILD, *£1 Billion Pounds to be awarded soon*, pp. 76–77 in M JONES (ED): *High Life*, Cedar Communications Ltd., London, September (2002).
- [74] O FAVARON, *Least domination in a graph*, *Discrete Mathematics*, **150** (1996), pp. 115–122.
- [75] JA FELDMAN & DH BALLARD, *Connectionist models and their properties*, *Cognitive Science*, **6** (1982), pp. 205–254.
- [76] P FLACH & L VOLKMANN, *Estimations for the domination number of a graph*, *Discrete Mathematics*, **80** (1990), pp. 145–151.
- [77] FLORIDA STATE LOTTERY, *Florida Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.flalottery.com/>
- [78] A FOG, *Pseudo-random number generators*, [Online], [cited 2004, August 27], Available from: <http://www.agner.org/random/>
- [79] MK FORT, JR & GA HEDLUND, *Minimal coverings of pairs by triples*, *Pacific Journal of Mathematics*, **8** (1958), pp. 709–719.
- [80] C FOUGO, *SCML – Departamento de Jogos*, [Online], [cited 2002, March 7], Available from: <http://www.djogos.misericordiadelisboa.pt/>
- [81] FREE SOFTWARE FOUNDATION, *GLPK – GNU Project – Free Software Foundation (FSF)*, [Online], [cited 2004, September 21], Available from: <http://www.gnu.org/software/glpk/>

- 
- [82] J FULMAN, *A generalization of Vizing's theorem on domination*, Discrete Mathematics, **126** (1994), pp. 403–406.
- [83] Z FÜREDI, GJ SZÉKELY & Z ZUBOR, *On the lottery problem*, Journal of Combinatorial Designs, **4**(1) (1996), pp. 5–10.
- [84] FUTURE PROTOCOL, INC., *The Texas Lottery Commission*, [Online], [cited 2001, November 14], Available from: <http://www.txlottery.org/>
- [85] M GARDNER, *A New Kind of Cipher That Would Take Millions of Years to Break*, Scientific American, August (1977), pp. 120–124.
- [86] GEOCITIES, *Hong Kong Mark Six Lotto Statistics, Lottery Results, Wheels, Online Lotteries*, [Online], [cited 2002, March 30], Available from: [http://www.geocities.com/www\\_lottery\\_lotto/](http://www.geocities.com/www_lottery_lotto/)
- [87] GEORGIA LOTTERY CORPORATION, *Georgia Lottery Corporation*, [Online], [cited 2001, November 14], Available from: <http://www.galottery.com/>
- [88] J GLENN BROOKSHEAR, *Theory of Computation: Formal Languages, Automata, and Complexity*, The Benjamin/Cummings Publishing Company, Inc., New York, (1989), pp. 239–286.
- [89] F GLOVER, *Future paths for Integer Programming and links to Artificial Intelligence*, Computers & Operations Research, **13**(5) (1986), pp. 533–549.
- [90] F GLOVER, *Heuristics for Integer Programming using surrogate constraints*, Decision Sciences, **8** (1977), pp. 156–166.
- [91] AP GODBOLE, SE THOMPSON & E VIGODA, *General Upper Bounds for Covering Numbers*, Ars Combinatoria, **42** (1996), pp. 211–221.
- [92] C GODSIL & G ROYLE, *Algebraic Graph Theory*, Springer–Verlag, New York, (2001), pp. 1–18, 33–58, 103–134.
- [93] GOLDEN CASKET LOTTERY CORPORATION LIMITED, *Golden Casket*, [Online], [cited 2002, March 6], Available from: <http://www.goldencasket.com/>
- [94] DM GORDON, G KUPERBERG & O PATASHNIK, *New Constructions for Covering Designs*, Journal of Combinatorial Designs, **3**(4) (1995), pp. 269–284.
- [95] M GREIG, PC LI & GHJ VAN REES, *Covering Designs on 13 Blocks Revisited*, preprint (2004).
- [96] RP GRIMALDI, *Discrete and Combinatorial Mathematics: An Applied Introduction (3<sup>rd</sup> ed)*, Addison–Wesley Publishing Company, Reading, (1994), pp. 293–296, 529–606.
- [97] J GROSS & J YELLEN, *Graph Theory and its Applications*, CRC Press, London, (1999), pp. 441–447, 492.
- [98] WR GRÜNDLINGH & JH VAN VUUREN, *Lottery Repository*, [Online], [cited 2003, December 25], Available from: <http://dip.sun.ac.za/vuuren/lottery/>
- [99] H HANANI, D ORNSTEIN & VT SÓS, *On the lottery problem*, Magyar Tudományos Akademia Alkalmazott Matematikai Kutató Intézetének Közleményei, **9** (1964), pp. 155–158.
- [100] J HARANT, A PRUCHNEWSKI & M VOIGHT, *On Dominating Sets and Independent Sets of Graphs*, Combinatorics, Probability and Computing, **11** (1993), pp. 1–10.
- [101] F HARARY, *Graph Theory*, Addison–Wesley Publishing Company, London, (1969), pp. 160–165.
- [102] N HARTSFIELD & G RINGEL, *Pearls in Graph Theory — A Comprehensive Introduction*, Academic Press, Inc., Boston, (1990), pp. 1–22.
- [103] TW HAYNES, ST HEDETNIEMI & PJ SLATER, *Fundamentals of Domination in Graphs*, Marcel Dekker, Inc., New York, (1998).

- [104] P HELL & J NEŠETŘIL, *The core of a graph*, Discrete Mathematics, **109**(1–3) (1992), pp. 117–126.
- [105] MA HENNING, *Graphs with least domination number three-fifths their order*, Discrete Mathematics, **216** (2000), pp. 153–168.
- [106] HOOSIER LOTTERY, *Hoosier Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.in.gov/hoosierlottery/>
- [107] JD HORTON, RC MULLIN & RG STANTON, *Minimal coverings of pairs by quadruples*, Congressus Numerantium, **3** (1971), pp. 495–516.
- [108] HRVATSKA LUTRIJA D.O.O., *Hrvatska Lutrija d.o.o.*, [Online], [cited 2002, April 16], Available from: <http://www.lutrija.hr/>
- [109] HYPERTECH, ΚΑΛΩΣΟΡΙΣΤΑΤΕ ΣΤΗΝ ΟΠΑΠ ΑΕ, [Online], [cited 2002, March 8], Available from: <http://www.opap.gr/>
- [110] IDAHO STATE LOTTERY, *Idaho Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.idaholottery.com/>
- [111] ILLINOIS STATE LOTTERY, *Illinois Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.illinoislottery.com/>
- [112] INTEGER MIDWEST, *The Iowa Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.ialottery.com/>
- [113] INTERLOTTO WORLD, *Interlotto.com—Lotto and Lottery results online gambling*, [Online], [cited 2002, March 6], Available from: <http://www.interlotto.com/>
- [114] INTERNET INFIDELS, *On the Origin of Species*, [Online], [cited 2002, November 12], Available from: [http://www.infidels.org/library/historical/charles\\_darwin/origin\\_of\\_species/](http://www.infidels.org/library/historical/charles_darwin/origin_of_species/)
- [115] IPSS, *www.ipss.net*, [Online], [cited 2002, March 30], Available from: <http://www.ipss.net/>
- [116] ISLENSK GETSPÁ, *íslensk Getspá*, [Online], [cited 2002, March 6], Available from: <http://www.lotto.is/>
- [117] ISRAELI LOTTERY, *Mazal Israel – The Israeli Lottery*, [Online], [cited 2002, March 8], Available from: <http://www.mazalIsrael.com/>
- [118] JAMAICA LOTTERY COMPANY LIMITED, *Jamaica Lottery Company Limited*, [Online], [cited 2002, March 7], Available from: <http://www.jamaicalottery.com/>
- [119] R JANS, *Problem 326*, Statistica Neerlandica, Problem Section, **50**(2) (1996), p. 331 and **52**(2) (1998), pp. 249–251.
- [120] JAPAN LOTTERY ASSOCIATION, *TAKARA-KUJI — Japanese Lottery*, [Online], [cited 2002, March 6], Available from: <http://www.takarakuji.nippon-net.ne.jp/>
- [121] JEL PRODUCTIONS, *California Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.calottery.com/>
- [122] R JOHNSONBAUGH, *Discrete Mathematics*, Collier Macmillan Publishers, London, (1990), pp. 314–324.
- [123] JG KALBFLEISCH & RG STANTON, *Maximal and minimal coverings of  $(k-1)$ -tuples by  $k$ -tuples*, Pacific Journal of Mathematics, **26** (1968), pp. 131–140.
- [124] KANSAS STATE LOTTERY, *Kansas Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.kslottery.com/>
- [125] RM KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, Plenum, New York, (1972), pp. 85–103.

- 
- [126] KENTUCKY LOTTERY CORPORATION, *Kentucky Lottery Corporation*, [Online], [cited 2001, November 14], Available from: <http://www.kylottery.com/>
- [127] DE KNUTH, *Combinatorial Matrices*, pp. 177–186 in DE KNUTH: *Selected Papers on Discrete Mathematics*, CLSI Lecture Notes, **106**, CSLI, California, (2003), or [Online], [cited 2004, July 1], Available from: <http://www-cs-faculty.stanford.edu/~knuth/>
- [128] T KOHONEN, *Self-organised formation of topologically correct feature maps*, *Biological Cybernetics*, **43** (1982), pp. 59–69.
- [129] T KOHONEN, *Self-organisation and associative memory*, Springer-Verlag, Berlin, (1984).
- [130] B KRÖSE & P VAN DER SMAGT, *An introduction to Neural Networks (8<sup>th</sup> ed)*, University of Amsterdam, November (1996).
- [131] ER LAMKEN, WH MILLS, RC MULLIN & SA VANSTONE, *Covering of pairs by quintuples*, *Journal of Combinatorial Theory Series A*, **44**(1) (1987), pp. 49–68.
- [132] JJ LEWIS, *Wisdom Quotes*, [Online], [cited 2004, March 24], Available from: <http://www.wisdomquotes.com/>
- [133] PC LI, *Ben Li's Lotto Tables Page*, [Online], [cited 2002, April 23], Available from: <http://www.cs.umanitoba.ca/~lipakc/lottotables.html>
- [134] PC LI, *Some Results on Lotto Designs*, PhD dissertation, University of Manitoba, Canada, (1999), or [Online], [cited 2004, October 22], Available from <http://www.cs.umanitoba.ca/~lipakc/>
- [135] PC LI & GHJ VAN REES, *Determination of  $C(17, 10, 3)$* , Technical Report TR-02-03, Department of Computer Science, University of Manitoba, Canada, (2002).
- [136] PC LI & GHJ VAN REES, *Lotto Design Tables*, *Journal of Combinatorial Designs*, **10**(5) (2002), pp. 335–359.
- [137] PC LI & GHJ VAN REES, *Lower Bounds on Lotto Designs*, *Congressus Numerantium*, **141** (1999), pp. 5–30.
- [138] PC LI & GHJ VAN REES, *New Constructions of Lotto Designs*, *Utilitas Mathematica*, **58** (2000), pp. 45–64.
- [139] LINDO SYSTEMS, INC., *LINGO*, [Online], [cited 2002, March 4], Available from: <http://www.lindo.com/>
- [140] LOTERÍA ELECTRÓNICA DE PUERTO RICO, *Lotería Electrónica de Puerto Rico*, [Online], [cited 2002, March 25], Available from: <http://www.loteriaelectronicapr.com/>
- [141] LOTERIAS.COM, *LOTERIA, Spanish Gordo*, [Online], [cited 2002, March 22], Available from: <http://www.loterias.com/>
- [142] LOTO-QUÉBEC, *Portail de Loto-Québec*, [Online], [cited 2002, March 7], Available from: <http://www.loto-quebec.com/>
- [143] LOTTERIES COMMISSION SOUTH AUSTRALIA, *SA Lotteries*, [Online], [cited 2002, March 6], Available from: <http://www.salotteries.sa.gov.au/>
- [144] LOTTERIES COMMISSION WESTERN AUSTRALIA, *Lotteries Commission WA—Corporate and Community Funding*, [Online], [cited 2002, March 6], Available from: <http://www.lottery.wa.gov.au/>
- [145] LOTTOMATICA, *Lottomatica*, [Online], [cited 2002, March 7], Available from: <http://www.lottomatica.it/>
- [146] LOUISIANA STATE LOTTERY, *Louisiana Lottery Corporation*, [Online], [cited 2001, November 14], Available from: <http://www.louisianalottery.com/>



- [147] L LOVÁSZ, *On the ratio of optimal integral and fractional covers*, Discrete Mathematics, **13** (1975), pp. 383–390.
- [148] U MANBER, *Introduction to Algorithms: A Creative Approach*, Addison–Wesley Publishing Company, Reading, Massachusetts, (1989), pp. 341–374.
- [149] MANITOBA LOTTERIES CORPORATION, *Manitoba Lotteries Corporation*, [Online], [cited 2003, November 1], Available from: <http://www.mlc.mb.ca/>
- [150] MARYLAND STATE LOTTERY, *The Maryland State Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.msla.state.md.us/>
- [151] D MARCU, *A new upperbound for the domination number of a graph*, Quarterly Journal of Mathematics Oxford (2), **36** (1985), pp. 221–223.
- [152] D MARCU, *An upperbound on the domination number of a graph*, Mathematica Scandinavica, **59**(1) (1986), pp. 41–44.
- [153] GE MARTIN, *Counting: The Art of Enumerative Combinatorics*, Springer–Verlag, New York, (2001), pp. 27–31.
- [154] MASSACHUSETTS STATE LOTTERY COMMISSION, *Massachusetts State Lottery Commission*, [Online], [cited 2001, November 14], Available from: <http://www.masslottery.com/>
- [155] W MCCUAIG & B SHEPHERD, *Domination in Graphs with Minimum Degree Two*, Journal of Graph Theory, **13**(6) (1989), pp. 749–762.
- [156] WS MCCULLOCH & W PITTS, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, **5** (1943), pp. 115–133.
- [157] E MCI (ED), *The Concise Oxford Dictionary*, Oxford University Press, Oxford, (1950).
- [158] BD MCKAY, *Practical graph isomorphism*, Congressus Numerantium, **30** (1981), pp. 45–87 or [Online], [cited 2004, October 21], Available from <http://cs.anu.edu.au/~bdm/nauty/PGI/>
- [159] BD MCKAY, *The autoston page*, [Online], [cited 2004, June 29], Available from: <http://cs.anu.edu.au/~bdm/autoston/>
- [160] BD MCKAY, *The nauty page*, [Online], [cited 2004, August 27], Available from: <http://cs.anu.edu.au/~bdm/nauty/>
- [161] Z MICHALEWICZ, *Genetic Algorithms + Data Structures = Evolution Programs (3<sup>rd</sup> ed)*, Springer–Verlag, Berlin, (1999).
- [162] Z MICHALEWICZ & DB FOGEL, *How to Solve It: Modern Heuristics*, Springer–Verlag, Berlin, (2000), pp. 115–134, 139–155.
- [163] MICHIGAN STATE LOTTERY, *LOTTERY*, [Online], [cited 2001, November 14], Available from: <http://www.michigan.gov/lottery/>
- [164] MILLI PIYANGO, *Milli Piyango*, [Online], [cited 2002, March 8], Available from: <http://www.millipiyango.gov.tr/>
- [165] WH MILLS, *A covering of pairs by quintuples*, Ars Combinatoria, **18** (1984), pp. 21–31.
- [166] WH MILLS, *Covering designs I: Covering by a small number of subsets*, Ars Combinatoria, **8** (1979), pp. 199–315.
- [167] WH MILLS, *Covering Problems*, Congressus Numerantium, **8** (1983), pp. 23–52.
- [168] WH MILLS, *On the covering of pairs by quadruples I*, Journal of Combinatorial Theory Series A, **13** (1972), pp. 55–78.

- [169] WH MILLS, *On the covering of pairs by quadruples II*, Journal of Combinatorial Theory Series A, **15** (1973), pp. 138–166.
- [170] WH MILLS, *On the Covering of Triples by Quadruples*, Congressus Numerantium, **10** (1974), pp. 563–581.
- [171] WH MILLS, *The covering number  $C(11, 5, 3)$* , Utilitas Mathematica, **41** (1992), p. 63.
- [172] WH MILLS & RC MULLIN, *Covering pairs by quintuples: The case  $v$  congruent to 3 (mod 4)*, Journal of Combinatorial Theory Series A, **49**(2) (1988), pp. 308–322.
- [173] MINNESOTA STATE LOTTERY, *Minnesota State Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.lottery.state.mn.us/>
- [174] M MINSKY & S PAPERT, *Perceptrons: An Introduction to Computational Geometry*, The MIT Press, (1969).
- [175] MISSOURI STATE LOTTERY, *Missouri Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.molottery.state.mo.us/>
- [176] MONTANA LOTTERY AND SCIENTIFIC GAMES INTERNATIONAL, *Montana Lottery Official Web Site*, [Online], [cited 2001, November 14], Available from: <http://www.montanalottery.com/>
- [177] CA MORGAN & PJ SLATER, *A linear algorithm for a core of a tree*, Journal of Algorithms, **1**(3) (1980), pp. 247–258.
- [178] M MORLEY & GHJ VAN REES, *Lottery Schemes and Covers*, Utilitas Mathematica, **37** (1990), pp. 159–166.
- [179] NARODNA LUTRIJA JUGOSLAVIJA, *Narodna Lutrija*, [Online], [cited 2002, March 7], Available from: <http://www.yulottery.co.yu/>
- [180] NASPL, *North American Association of State & Provincial Lotteries*, [Online], [cited 2002, March 22], Available from: <http://www.naspl.org/>
- [181] NATIONAL LOTTERY, *National Lottery*, [Online], [cited 2002, February 22], Available from: <http://nationallottery.co.za/>
- [182] NATIONAL LOTTERY BOARD OF THE REPUBLIC OF KAZAKHSTAN, *www.kazlotto.com*, [Online], [cited 2002, March 11], Available from: <http://www.kazlotto.com/>
- [183] NEBRASK@ONLINE, *Nebraska Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.nelottery.com/>
- [184] NEW HAMPSHIRE SWEEPSTAKES COMMISSION, *New Hampshire Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.state.nh.us/lottery/nhlotto.htm>
- [185] NEW JERSEY STATE LOTTERY, *New Jersey Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.state.nj.us/lottery/>
- [186] NEW MEXICO LOTTERY, *Welcome to the New Mexico Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.nmlottery.com/>
- [187] NEW YORK LOTTERY, *New York Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.nylottery.org/>
- [188] NIKAT.COM, *LuckyIndia.com*, [Online], [cited 2002, March 8], Available from: <http://www.luckyindia.com/>
- [189] THE NOBEL FOUNDATION, *Nobel e-Museum*, [Online], [cited 2004, June 2], Available from: <http://www.nobel.se/>
- [190] K NONOBE & T IBARAKI, *A tabu search approach to the constraint satisfaction problem as a general problem solver*, European Journal of Operational Research, **106** (1998), pp. 599–623.

- [191] NORSK TIPPING AS, *Norsk Tipping*, [Online], [cited 2002, March 7], Available from: <http://www.norks-tipping.no/>
- [192] NSW LOTTERIES CORPORATION, *NSW-Lotteries*, [Online], [cited 2002, March 6], Available from: <http://www.nswlotteries.com.au/>
- [193] KJ NURMELA & PRJ ÖSTERGÅRD, *Constructing Covering Designs by Simulated Annealing*, Technical Reports, Series B, **10**, Digital Systems Laboratory, Helsinki University of Technology, Helsinki, (1993).
- [194] KJ NURMELA & PRJ ÖSTERGÅRD, *Upper Bounds for Covering Designs by Simulated Annealing*, *Congressus Numerantium*, **96** (1993), pp. 93–111.
- [195] NZPAGES LIMITED, The New Zealand Lotto Results, [Online], [cited 2002, April 23], Available from: <http://lotto.nzpages.net.nz/>
- [196] OHIO LOTTERY COMMISSION, *Ohio Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.ohiolottery.com/>
- [197] O ORE, *Theory of Graphs*, American Mathematical Society Colloquium Publications, **38** (1962), pp. 206–223, 259–261.
- [198] OREGON LOTTERY, *Oregon Lottery Web Center*, [Online], [cited 2001, November 14], Available from: <http://www.oregonlottery.org/>
- [199] PRJ ÖSTERGÅRD, *Constructing covering codes by tabu search*, *Journal of Combinatorial Designs*, **5**(1) (1997), pp. 71–80.
- [200] ÖSTERREICHISCHE LOTTERIEN GESELLSCHAFT M.B.H., *Österreichische Lotterien*, [Online], [cited 2002, March 6], Available from: <http://www.lotterien.at/>
- [201] JI PANALO, *Philippine Lotto Central*, [Online], [cited 2002, March 6], Available from: <http://www.lottocentral.net/>
- [202] CH PAPADIMITRIOU & K STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, Inc., New York, (1998), pp. 406–432.
- [203] C PAYAN, *Sur le nombre d'absorption d'un graphe simple* (French), *Cahiers du Centre d'Études de Recherche Opérationnelle*, **17** (1975), pp. 307–317.
- [204] M PIRLOT, *General local search methods*, *European Journal of Operational Research*, **92** (1996), pp. 493–511.
- [205] PUBLIC LOTTO DEPARTMENT, MALTA, *Malta Lotteries*, [Online], [cited 2002, March 25], Available from: <http://www.maltalotteries.com/>
- [206] QUOTEDB, *Quote DB – The Quotations Database*, [Online], [cited 2004, December 13], Available from: <http://www.quotedb.com/>
- [207] B RANDEARTH & L VOLKMANN, *Characterization of graphs with equal domination and covering number*, *Discrete Mathematics*, **191** (1998), pp. 159–169.
- [208] BA REED, *Paths, stars and the number three*, *Combinatorics, Probability and Computing*, **5**(3) (1996), pp. 277–295.
- [209] RHODE ISLAND LOTTERY, *RI Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.rilot.com/>
- [210] R RIVEST, A SHAMIR & L ADLEMAN, *On Digital Signatures and Public-Key Cryptosystems*, Technical Memo, **82**, Laboratory for Computer Science, Massachusetts Institute of Technology (MIT), April (1977).
- [211] DK ROBERTS, *Maine State Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.mianelottery.com/>

- [212] G ROBINS, *Good Quotations by Famous People*, [Online], [cited, March 11], Available from: <http://www.cs.virginia.edu/~robins/>
- [213] V RÖDL, *On a packing and covering problem*, *European Journal of Combinatorics*, **6**(1) (1985), pp. 69–78.
- [214] P ROWLINSON, *Dominating sets and eigenvalues of graphs*, *The Bulletin of the London Mathematical Society*, **26**(3) (1994), pp. 248–254.
- [215] RT&E INTERACTIVE, *The Delaware Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.lottery.state.de.us/>
- [216] DE RUMELHART, GE HINTON & RJ WILLIAMS, *Learning representations by back-propagating errors*, *Nature*, **323** (1986), pp. 533–536.
- [217] E SAMPATHKUMAR, *The least point covering and domination number of a graph*, *Discrete Mathematics*, **86** (1990), pp. 137–142.
- [218] E SAMPATHKUMAR & LP LATHA, *Strong weak domination and domination balance in a graph*, *Discrete Mathematics*, **161** (1996), pp. 235–242.
- [219] LA SANCHIS, *Bounds Related to Domination in Graphs with Minimum Degree Two*, *Journal of Graph Theory*, **25** (1997), pp. 139–152.
- [220] SAZKA, *SAZKA a.s.*, [Online], [cited 2002, March 6], Available from: <http://www.sazka.cz/>
- [221] J SCARNE, *Scarne's new complete guide to gambling*, Constable & Company, Ltd., London, (1975), pp. 141–185.
- [222] SCHAFFHAUSEN INTERACTIVE, *LOTTO.de, Deutscher Lotto- und Totoblock*, [Online], [cited 2002, March 6], Available from: <http://www.lotto-toto.de/>
- [223] J SCHÖNHEIM, *On Coverings*, *Pacific Journal of Mathematics*, **14** (1964), pp. 1 405–1 411.
- [224] J SCHÖNHEIM, *On the number of mutually disjoint triples in Steiner systems and related maximal packing and minimal covering systems*, pp. 311–318 in WT TUTTE (ED): *Recent Progress in Combinatorics*, Academic Press, New York, (1969).
- [225] A SCHRIJVER (ED), *Packing and Covering in Combinatorics*, *Mathematical Centre Tracts*, **106**, Mathematisch Centrum, Amsterdam, (1979).
- [226] M SCOTT, *Rocky Road: Charles Darwin*, [Online], [cited 2002, November 12], Available from: <http://www.strangescience.net/darwin.htm>
- [227] SIA LATTELEKOM, *APOLLO – Latloto*, [Online], [cited 2002, October 22], Available from: <http://www.apollo.lv/portal/latloto/>
- [228] SINGAPORE POOLS PTE LTD., *Singapore Pools*, [Online], [cited 2002, March 8], Available from: <http://www.singaporepools.com.sg/>
- [229] M SIPSER, *Introduction to the Theory of Computation*, PWS Publishing Company, New York, (1997), pp. 223–379.
- [230] SKANDINÁLOTTÓ, *Szerencsejatek Rt. fogadasi rendszer*, [Online], [cited 2002, March 8], Available from: <http://www.szerencsejatek.hu/Jatek/>
- [231] J SPENCER, *Asymptotically good coverings*, *Pacific Journal of Mathematics*, **118**(2) (1985), pp. 575–586.
- [232] ŠPORTNA LOTERIJA D.D., *Športna Loterija d.d.*, [Online], [cited 2002, March 7], Available from: <http://www.sportna-loterija.si/>
- [233] SOUTH DAKOTA STATE LOTTERY, *South Dakota Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.sdlottery.org/>

- [234] RG STANTON & JA BATE, *A computer search for B-coverings*, pp. 37–50 in RW ROBINSON, GW SOUTHERN & WD WALLIS (EDS): *Combinatorial Mathematics VII (Proceedings of the Seventh Australian Conference on Combinatorial Mathematics, University of Newcastle, Australia, 1979)*, Lecture Notes in Mathematics, **829**, Springer-Verlag, Berlin, (1980).
- [235] RG STANTON & RC MULLIN, *Some new results on the covering numbers  $N(t, k, v)$* , pp. 51–58 in RW ROBINSON, GW SOUTHERN & WD WALLIS (EDS): *Combinatorial Mathematics VII (Proceedings of the Seventh Australian Conference on Combinatorial Mathematics, University of Newcastle, Australia, 1979)*, Lecture Notes on Mathematics, **829**, Springer-Verlag, Berlin, (1980).
- [236] F STERBOUL, *Le problème du Loto* (French), *Cahiers du Centre d'Études de Recherche Opérationnelle*, **20** (1978), pp. 443–449.
- [237] D STOJILJKOVIC, *Dragon Stojiljkovic lotto wheels page*, [Online], [cited 2003, September 17], Available from: <http://www.geocities.com/dragons2000/>
- [238] B STROUSTRUP, *The C++ Programming Language (3<sup>rd</sup> ed)*, Addison-Wesley, England, (1997).
- [239] L SUN & J WANG, *An Upper Bound for the Independence Domination Number*, *Journal of Combinatorial Theory Series B*, **76**(2) (1999), pp. 240–246.
- [240] SWISSCOM AG, *Internet Swiss Lotto*, [Online], [cited 2002, March 7], Available from: <http://www.swisslotto.ch/>
- [241] TACHIRA SU LOTERIA, *Instituto de Beneficencia Pública y Bienestar Social del Estado Táchira - LOTERIA DEL TACHIRA*, [Online], [cited 2002, October 22], Available from: <http://www.loteriadeltachira.com/>
- [242] TAIWAN LOTTERY LTD., *Taiwan Lottery Ltd.*, [Online], [cited 2002, March 22], Available from: <http://www.taiwan-lottery.com/>
- [243] TEKTRON S.A. TECNOLOGÍA ELECTRÓNICA, *Tektron S.A.*, [Online], [cited 2002, March 22], Available from: <http://www.latinka.com/>
- [244] THE THOREAU INSTITUTE AT WALDEN WOODS, *Henry D. Thoreau Homepage*, [Online], [cited 2003, November 3], Available from: <http://www.walden.org/thoreau/>
- [245] TIPOS A.S., *TIPOS a.s.*, [Online], [cited 2002, March 7], Available from: <http://www.tipos.sk/>
- [246] DT TODOROV, *A method of constructing covering designs* (Russian), *Matematicheskie Zametki*, **35**(6) (1984), pp. 869–876.
- [247] DT TODOROV, *A table of coverings of pairs*, *Mathematics and Mathematical Education*, (1986), pp. 472–481.
- [248] DT TODOROV, *Coverings of finite sets* (Bulgarian), *Prilozhna Matematika*, **16**(3) (1980), pp. 179–190.
- [249] DT TODOROV, *On some covering designs*, *Comptes Rendus de l'Académie Bulgare des Sciences*, **37**(9) (1984), pp. 1185–1186.
- [250] DT TODOROV, *On some covering designs*, *Journal of Combinatorial Theory Series A*, **39**(1) (1985), pp. 83–101.
- [251] DT TODOROV, *On the covering of pairs by 13 blocks*, *Comptes Rendus de l'Académie Bulgare des Sciences*, **38**(6) (1985), pp. 691–694.
- [252] DT TODOROV, *On the covering of triples by eight blocks*, *Serdica*, **12**(1) (1986), pp. 20–29.
- [253] DT TODOROV, *Some coverings derived from finite planes*, *Finite and Infinite Sets*, **I, II** (1981), pp. 697–710.
- [254] DT TODOROV & VD TONCHEV, *On some coverings of triples*, *Comptes Rendus de l'Académie Bulgare des Sciences*, **35**(9) (1982), pp. 1209–1211.

- 
- [255] TOTALIZATOR SPORTOWY SPÓLKA, *LOTTO*, [Online], [cited 2002, March 7], Available from: <http://www.lotto.pl/>
- [256] P TURÁN, *Eine Extremalaufgabe aus der Graphentheorie*, *Matematicheskaya Fizika Lapok*, **48** (1941), pp. 436–452.
- [257] Z TUZA & PD VESTERGAARD, *Domination in partitioned graphs*, [Online], [cited 2002, March 2], NEC RESEARCH INSTITUTE, *ResearchIndex [NEC Research Institute, Citeseer, Computer Science]*, Available from: <http://www.citeseer.nj.nec.com/cs/>
- [258] C UHRIG & N BOGHOSSIAN, *Algorithmic Solutions Software GmbH*, [Online], [cited 2001, October 19], Available from: <http://www.algorithmic-solutions.com/>
- [259] N VAN NUFFELEN, *Rank and domination number*, pp. 209–211 in M FIEDLER (ED): *Graphs and other combinatorial topics*, Proceedings of the Third Czechoslovak Symposium on Graph Theory, Prague, Teubner Texts in Mathematics, **59** (1983).
- [260] VEIKKAUS, *Veikkaus*, [Online], [cited 2002, March 8], Available from: <http://www.veikkaus.fi/>
- [261] VERMONT LOTTERY COMMISSION, *Vermont Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.vtlottery.com/>
- [262] VIRGINIA LOTTERY, *Official Home of the Virginia Lottery*, [Online], [cited 2004, January 30], Available from: <http://www.valottery.com/>
- [263] VG VIZING, *Estimate of the number of external stability of a graph*, *Soviet Mathematics, Doklady*, **164**(4) (1965), pp. 1 275–1 278.
- [264] HB WALIKAR, BD ACHARYA & E SAMPATHKUMAR, *Recent developments in the theory of domination in graphs*, MRI Lecture Notes in Mathematics, Mahta Research Institute, Allahabad, **1** (1979).
- [265] WD WALLIS, *Combinatorial Designs*, Marcel Dekker, Inc., New York, (1988).
- [266] WEST VIRGINIA STATE LOTTERY, *West Virginia Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.state.wv.us/lottery/>
- [267] WESTERN CANADA LOTTERY CORPORATION, *Western Canada Lottery Corporation*, [Online], [cited 2002, February 22], Available from: <http://www.wclc.com/>
- [268] B WIDROW & ME HOFF, *Adaptive switching circuits*, 1960 IRE WESCON Convention Record IV, New York, pp. 96–104.
- [269] WISCONSIN STATE LOTTERY, *Wisconsin Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.wilottery.com/>
- [270] J YIN & AM ASSAF, *Constructions of Optimal Packing Designs*, *Journal of Combinatorial Designs*, **6**(4) (1998), pp. 245–260.
- [271] J ZUKOWSKI, *Washington State Lottery*, [Online], [cited 2001, November 14], Available from: <http://www.wa.gov/lot/>
- [272] IE ZVEROVICH, *Proof of a conjecture in domination theory*, *Discrete Mathematics*, **184** (1998), pp. 297–298.



# Appendix A

## Computer programs

“C makes it easy to shoot yourself in the foot;  
C++ makes it harder, but when you do,  
it blows away your whole leg.”  
*Bjarne Stroustrup* (1950–) [212]

“The purpose of computing is insight, not numbers.”  
*Richard W Hamming* (1915–1998) [30]

This appendix is devoted to giving the reader access to the source code that was used in programming some of the algorithms in Chapters 3 and 5. Most routines and algorithms (presented in §§A.1–A.2 and §§A.4–A.8) were programmed in C++, utilising the standard header files and functions available, although some set-functions from LEDA (short for *Library of Efficient Data Algorithms*, [258]) or the C++ STL (Standard Template Libraries) were used. However, the Minimal overlapping algorithm (Algorithm 4 presented in §A.3) was programmed in Microsoft Visual Basic for Applications (a macro/application extension to Microsoft Excel), due to built in spreadsheet functions used. More on how to program in C++ may be found in [62, 238].

### A.1 Classical random algorithm (Algorithm 2)

```
#include<iostream.h> // Header file used for input and output.
#include<iomanip.h> // Header file used to manipulate output.
#include<stdlib.h> // Header file containing random function.
#include<fstream.h> // Header file for using files.
#include<time.h> // Header file for calculating iteration times.
#include<stl.h> // Header file for STL (Standard Template Libraries) <set, multiset>.
#include<math.h> // Header file for mathematical functions.

long double fact (int);
long int round (const long double);
bool ValidTicket (const short int *, const short int &, const short int &);

// Compute the factorial of n (i.e. n!).
long double fact (int n) {
    if (n <= 1) return 1; // Stop recursion.
    else return n * fact(n - 1); // (n > 1)
}
// Rounds a number to the nearest integer to avoid possible numerical truncation errors.
long int round (const long double n) {
    long int flrnum = (long int)floor(n);
    if (n - flrnum < 0.5) return flrnum;
    else return (long int)ceil(n); // (n - flrnum >= 0.5)
}
// Determine whether a ticket (n-set) is valid (ticket contains no double numbers).
bool ValidTicket (short int *ticket, const short int &n, const short int &n) {
    multiset<short int, less<short int> > index;
    multiset<short int, less<short int> >::iterator i;
    for (short int counter = 0; counter < n; counter++) {
        index.insert(ticket[counter]); // Add number to index.
    }
}
```



```

    if (index.count(ticket[counter]) > 1)
        return false; // Ticket is invalid.
    }
    i = index.begin(); // Arrange ticket elements lexicographically.
    for (short int counter = 0; i != index.end(); ticket[counter++] = *i++);
    return true; // Ticket is valid.
}
int main () {
    short int m, n, k, L; // Variables containing the lottery parameters <m,n;k> and playing set cardinality L.
    cout << endl << "\t CLASSICAL RANDOM ALGORITHM." << endl << endl;
    cout << "Please specify the following parameters for the lottery (m,n;k):" << endl;
    cout << "m = "; cin >> m;
    cout << "n = "; cin >> n;
    cout << "k = "; cin >> k;
    // If the user specified invalid lottery parameters.
    if ((n > m-1) || (k > n-1) || (m < 3) || (n < 2) || (k < 1)) {
        cout << endl << "Invalid lotter parameters entered." << endl;
        return 0; // Exit program.
    }
    long int NumTickets, r = 0, CurrentTicketNumber, NumDominated, MaxDominated = 0, iter = 0;
    time_t StartTime; // Trace execution time.
    int CurrentTicket[n], CurrentNumber = n, size, Counter1, Counter2; // Reference to current ticket/vertex investigated.
    ofstream PlayingSetInfo("PlayingSet.Info", ios::out); // Playing set info file.
    if (!PlayingSetInfo) { // Check whether the file "PlayingSet.Info" could be opened.
        cout << "The file \"PlayingSet.Info\" could not be opened." << endl;
        return 0; // Exit program.
    }
    // Compute the number of vertex set cardinality and degree of regularity of the lottery graph G<m,n;k>.
    NumTickets = round((fact(m)/(fact(n)*fact(m-n)));
    for (int i = k; i < n; i++)
        r += round((fact(n)/(fact(n-i)*fact(i)))*(fact(m-n)/(fact(n-i)*fact(m-2*n+i))));
    cout << endl << "Order of the lottery graph G{" << m << " " << n << " " << k << " } is " << NumTickets << " and it is "
        << r << " regular." << endl;
    cout << endl << "Playing set cardinality = "; cin >> L;
    if (L < 1) // Invalid playing set cardinality specified.
        return 0; // Exit program.
    short int DomArray[L][n], Intersect, dcounter, tcounter; // Playing set, number of elements
        common to 2 tickets/vertex labels (intersection).
    srandom(time(NULL)); // Initialize the pseudo random number generator.
    StartTime = time(NULL); // Capture start of executed time.
    while (iter < 1000) { // This value may be changed, depending on the number of iterations preferred.
        for (short int i = 0; i < L; i++) // Generate pseudo random playing set.
            do
                for (short int j = 0; j < n; j++)
                    DomArray[i][j] = ((short int)(((random())/(float)RAND_MAX)*m)+1); // Generation random ticket.
                while (!ValidTicket(DomArray[i], m, n)); // Check whether generated ticket is valid.
            NumDominated = 0;
            // Generate the first lexicographic lottery Ticket [1,2,...,n].
            for (int i = 1; i <= n; i++)
                CurrentTicket[i-1] = i;

            // Determine resource utilisation  $\Psi_{\ell}(m, n; k)$  of playing set
            for (long int i = 1; i <= NumTickets; i++) {
                Intersect = 0;
                // Check whether current ticket is dominated.
                for (short int DomTicketNum = 0; (DomTicketNum < L) && (Intersect < k); DomTicketNum++) {
                    Intersect = dcounter = tcounter = 0;
                    while ((dcounter < n) && (tcounter < n) && (Intersect < k))
                        if (DomArray[DomTicketNum][dcounter] < CurrentTicket[tcounter]) dcounter++;
                        else if (DomArray[DomTicketNum][dcounter] > CurrentTicket[tcounter]) tcounter++;
                        else { // (DomArray[DomTicketNum][dcounter] == CurrentTicket[tcounter])
                            Intersect++; dcounter++; tcounter++;
                        }
                }
                if (Intersect == k) // Check whether current vertex/ticket is dominated by playing set.
                    NumDominated++;
            }
            CurrentTicket[CurrentNumber-1]++; // Generate next lexicographic ticket/vertex label.
            if (CurrentTicket[CurrentNumber-1] > m) {
                CurrentTicket[CurrentNumber-1]--;
                while ((CurrentNumber > 0) && (CurrentTicket[CurrentNumber-1] == m-(n-CurrentNumber)))
                    CurrentNumber--;
                CurrentTicket[CurrentNumber-1]++;
                for (int j = CurrentNumber; j < n; j++)
                    CurrentTicket[j] = CurrentTicket[j-1] + 1;
                CurrentNumber = n;
            }
        }
    }
    if (NumDominated > MaxDominated) { // New maximum resource utilisation found.
        // Write the (current best) playing set info to file.
        PlayingSetInfo.seekp(0, ios::beg); // Search to the beginning of the output file.
        PlayingSetInfo << endl << "Lottery {" << m << " " << n << " " << k << " }:" << endl << endl;
        PlayingSetInfo << "Playing set : " << endl;
        for (int i = 0; i < L; i++) {
            for (int j = 0; j < n; j++)

```

```

    PlayingSetInfo << setiosflags(ios::right) << setw(3) << DomArray[i][j];
    PlayingSetInfo << endl;
}
PlayingSetInfo << "Playing set dominates " << NumDominated << "/" << NumTickets << " vertices (" << setw(8)
    << setprecision(4) << setiosflags(ios::fixed|ios::showpoint) << (float)NumDominated/NumTickets*100
    << "%)." << endl;
    MaxDominated = NumDominated; // Store new maximum resource utilisation.
}
iter++; // Next iteration.
}
PlayingSetInfo.close(); // Close the playing set info file.
cout << "Time elapsed = " << time(NULL)-StartTime << endl;
return 0; // Exit program.
}

```

## A.2 Distributed random algorithm (Algorithm 3)

```

#include<iostream.h> // Header file used for input and output.
#include<iomanip.h> // Header file used to manipulate output.
#include<stdlib.h> // Header file containing random function.
#include<fstream.h> // Header file for using files.
#include<time.h> // Header file for calculating iteration times.
#include<stl.h> // Header file for STL (Standard Template Libraries) <set, multiset>.
#include<math.h> // Header file for mathematical functions.

long double fact (int);
long int round (long double);
bool ValidTicket (const short int *, const short int &, const short int &);

// Compute the factorial of n (i.e. n!).
long double fact (int n) {
    if (n <= 1) return 1; // Stop recursion.
    else return n * fact(n - 1); // (n > 1)
}
// Rounds a number off to the nearest integer to avoid possible numerical truncation errors.
long int round (long double n) {
    long int flrnum = (long int)floor(n);
    if (n - flrnum < 0.5) return flrnum;
    else return (long int)ceil(n); // (n - flrnum >= 0.5)
}
// Determine whether a ticket (n-set) is valid (ticket contains no double numbers).
bool ValidTicket (short int *ticket, const short int &m, const short int &n) {
    multiset<short int, less<short int> > index;
    multiset<short int, less<short int> >::iterator i;
    for (short int counter = 0; counter < n; counter++) {
        index.insert(ticket[counter]); // Add number to index.
        if (index.count(ticket[counter]) > 1)
            return false; // Ticket is invalid.
    }
    // Arrange ticket elements lexicographically.
    i = index.begin();
    for (short int counter = 0; i != index.end(); ticket[counter++] = *i++);
    return true; // Ticket is valid.
}
int main () {
    int m, n, k, L; // Variables containing the lottery parameters <m,n,k> and playing set cardinality L.
    cout << endl << "\t DISTRIBUTED RANDOM ALGORITHM" << endl << endl;
    cout << "Please specify the following parameters for the lottery {m,n,k}:" << endl;
    cout << "m = "; cin >> m;
    cout << "n = "; cin >> n;
    cout << "k = "; cin >> k;
    // If the user specified invalid lottery parameters.
    if ((n > m-1) || (k > n-1) || (m < 3) || (n < 2) || (k < 1)) {
        cout << endl << "Invalid lottery parameters entered." << endl;
        return 0; // Exit program.
    }
    long int NumTickets, r = 0, curiter = 1, CurrentTicketNumber, NumDominated, MaxDominated = 0;
    time_t StartTime; // Trace execution time.
    short int Intersect, dcounter, tcounter, average = (n*L)/m, retrycounter;
    int CurrentTicket[n], CurrentNumber = n, size, Counter1, Counter2;
    multiset <short int, less<short int> > distribution;
    fstream PlayingSetInfo("PlayingSet.Info", ios::out); // Playing set info file.
    if (!PlayingSetInfo) { // Check whether the file "PlayingSet.Info" could be opened.
        cout << "The file \"PlayingSet.Info\" could not be opened." << endl;
        return 0; // Exit program.
    }
    // Compute the order and degree of regularity of the lottery graph G<m,n,k>.
    NumTickets = round(fact(m)/(fact(n)*fact(m-n)));
    for (int i = k; i < n; i++)
        r += round((fact(n)/(fact(n-i)*fact(i)))*(fact(m-n)/(fact(n-i)*fact(m-2*n+i))));
}

```

```

cout << "The order of the lottery graph G(" << m << ", " << n << ", " << k << ") is " << NumTickets << " and it is "
<< r << " regular." << endl;
cout << endl << "Playing set cardinality = "; cin >> L;
if (L < 1) // Invalid playing set cardinality specified.
    return 0; // Exit program.
short int DomArray[L][n]; // Playing set.
srandom(time(NULL)); // Initialize the pseudo random number generator.
StartTime = time(NULL); // Capture start of execution time.
while (curiter <= 1000) { // This value may be changed, depending on the number of iterations preferred.
RestartLottoSet:
distribution.erase(distribution.begin(), distribution.end()); // Erase number distribution index.
for (short int i = 0; i < L; i++) { // Generate pseudo random playing set.
    retrycounter = 0;
    do
        for (short int j = 0; j < n; j++) {
            do // Determine correct number choice according to distribution index.
                dcounter = (short int)((float)random()/RAND_MAX)*m+1;
                while (distribution.count(dcounter) > average+1);
                DomArray[i][j] = dcounter; // Generation random ticket.
                retrycounter++;
            }
        while (!ValidTicket(DomArray[i], m, n) && retrycounter < 101); // Check whether generated ticket is valid.
        if (retrycounter > 100) goto RestartLottoSet; // Avoid faulty generated lottery sets.
        for (short int j = 0; j < n; j++)
            distribution.insert(DomArray[i][j]); // Add used lottery numbers to distribution index.
    }
    NumDominated = 0;
    // Generate the first lexicographic lottery ticket [1,2,...,n].
    for (int i = 1; i <= n; i++)
        CurrentTicket[i-1] = i;
    // Determine resource utilisation  $\Psi_\ell(m, n; k)$  of playing set.
    for (long int i = 1; i <= NumTickets; i++) {
        Intersect = 0;
        // Check whether current ticket is dominated.
        for (short int DomTicketNum = 0; DomTicketNum < L) && (Intersect < k; DomTicketNum++) {
            Intersect = dcounter = tcounter = 0;
            while ((dcounter < n) && (tcounter < n) && (Intersect < k))
                if (DomArray[DomTicketNum][dcounter] < CurrentTicket[tcounter]) dcounter++;
                else if (DomArray[DomTicketNum][dcounter] > CurrentTicket[tcounter]) tcounter++;
                else { // (DomArray[DomTicketNum][dcounter] == CurrentTicket[tcounter])
                    Intersect++; dcounter++; tcounter++;
                }
            }
            if (Intersect == k) // Check whether current vertex/ticket is dominated by playing set.
                NumDominated++;
        }
        CurrentTicket[CurrentNumber-1]++; // Generate next lexicographic ticket/vertex label.
        if (CurrentTicket[CurrentNumber-1] > m) {
            CurrentTicket[CurrentNumber-1]--;
            while ((CurrentNumber > 0) && (CurrentTicket[CurrentNumber-1] == m-(n-CurrentNumber)))
                CurrentNumber--;
            CurrentTicket[CurrentNumber-1]++;
            for (int j = CurrentNumber; j < n; j++)
                CurrentTicket[j] = CurrentTicket[j-1] + 1;
            CurrentNumber = n;
        }
    }
    if (NumDominated > MaxDominated) { // New maximum resource utilisation found.
        // Write the (current best) playing set info to file.
        PlayingSetInfo.seekp(0, ios::beg); // Search to the beginning of the output file.
        PlayingSetInfo << endl << "Lottery (" << m << ", " << n << ", " << k << ") : " << endl << endl;
        PlayingSetInfo << "Playing set : " << endl;
        for (int i = 0; i < L; i++) {
            for (int j = 0; j < n; j++)
                PlayingSetInfo << setiosflags(ios::right) << setw(3) << DomArray[i][j];
            PlayingSetInfo << endl;
        }
        PlayingSetInfo << "Playing set dominates " << NumDominated << "/" << NumTickets << " vertices (" << set(8)
        << setprecision(4) << setiosflags(ios::fixed|ios::showpoint) << (float)NumDominated/NumTickets*100
        << "%)." << endl;
        MaxDominated = NumDominated; // Store new maximum resource utilisation.
    }
    curiter++;
}
PlayingSetInfo.close(); // Close the playing set info file.
cout << "Time elapsed = " << time(NULL)-StartTime << endl;
return 0; // Exit program.
}

```

## A.3 Minimal overlapping algorithm (Algorithm 4)

```

Private Function AlphaCol(intcol As Integer) As String ' Determine alphabetic column presentation used by Excel.
  Dim counter1 As Integer, counter2 As Integer, counter3 As Integer
  counter1 = 0
  counter2 = 1
  counter3 = 1
  While (counter3 < intcol)
    counter3 = counter3 + 1
    counter2 = counter2 + 1
    If (counter2 > 26) Then
      counter2 = 1
      counter1 = counter1 + 1
    End If
  Wend
  If (counter1 > 0) Then
    AlphaCol = Chr(counter1 + 64) & Chr(counter2 + 64)
  Else ' (counter1 == 0)
    AlphaCol = Chr(counter2 + 64)
  End If
End Function

Private Function RU() As Long ' Determine the resource utilisation  $\Psi_\ell(m, n; k)$  of playing set.
  Dim m As Integer, n As Integer, k As Integer, L As Integer, CurrentTicket(50) As Integer, dcounter As Integer, _
  tcounter As Integer, Intersect As Integer, PlayingSet(200, 50) As Integer, CurrentNumber As Integer, _
  NumDominated As Long, NumTickets As Long, counter1 As Long, counter2 As Long, counter3 As Long
  m = Val(Sheet2.Cells(2, 8))
  n = Val(Sheet2.Cells(2, 9))
  k = Val(Sheet2.Cells(2, 10))
  L = Val(Sheet2.Cells(2, 11))
  Sheet2.Cells(2000, 1) = "=FACT(" & m & ")/(FACT(" & m - n & ")*FACT(" & n & ")"
  NumTickets = Val(Sheet2.Cells(2000, 1))
  Sheet2.Cells(2000, 1) = ""
  For dcounter = 1 To L ' Put  $m \times L$  ticket matrix in  $n \times L$  ticket matrix.
    Intersect = 1
    For tcounter = 1 To m
      If (Sheet2.Cells(dcounter + 3, tcounter + 1) <> "") Then
        PlayingSet(dcounter, Intersect) = tcounter
        Intersect = Intersect + 1
      End If
    Next tcounter
  Next dcounter
  For dcounter = 1 To n ' Generate first lexicographic lottery ticket [1,2,...,n].
    CurrentTicket(dcounter) = dcounter
  Next dcounter
  HuidigeGetal = n + 1
  For counter1 = 1 To NumTickets ' Determine playing set resource utilisation  $\Psi_\ell(m, n; k)$ 
    Intersect = 0
    counter2 = 1
    While ((counter2 <= L) And (Intersect < k))
      Intersect = 0
      dcounter = 1
      tcounter = 1
      While ((dcounter <= n) And (tcounter <= n) And (Intersect < k))
        If (PlayingSet(counter2, dcounter) < CurrentTicket(tcounter)) Then
          dcounter = dcounter + 1
        ElseIf (PlayingSet(counter2, dcounter) > CurrentTicket(tcounter)) Then
          tcounter = tcounter + 1
        Else ' (PlayingSet(counter2, dcounter) = CurrentTicket(tcounter))
          Intersect = Intersect + 1
          dcounter = dcounter + 1
          tcounter = tcounter + 1
        End If
      Wend
      If (Intersect = k) Then ' Check whether current ticket is dominated by playing set.
        NumDominated = NumDominated + 1
      End If
      counter2 = counter2 + 1
    Wend
    CurrentTicket(CurrentNumber - 1) = CurrentTicket(CurrentNumber - 1) + 1
    If (CurrentTicket(CurrentNumber - 1) > m) Then ' Generate next ticket in lexicographic sequence.
      CurrentTicket(CurrentNumber - 1) = CurrentTicket(CurrentNumber - 1) - 1
      While ((CurrentNumber > 1) And (CurrentTicket(CurrentNumber - 1) = m - (n - CurrentNumber) - 1))
        CurrentNumber = CurrentNumber - 1
      Wend
      CurrentTicket(CurrentNumber - 1) = CurrentTicket(CurrentNumber - 1) + 1
      For counter3 = CurrentNumber To n
        CurrentTicket(counter3) = CurrentTicket(counter3 - 1) + 1
      Next counter3
      CurrentNumber = n + 1
    End If
  Next counter1
End Function

```

```

Next counter1
RU = NumDominated
End Function

Public Sub GeneratePlayingSet() ' Generate playing set according to MINIMAL OVERLAP ALGORITHM.
Dim m As Integer, n As Integer, k As Integer, L As Integer, iter As Integer, counter1 As Integer, _
counter2 As Integer, counter3 As Integer, Counter4 As Integer, counter5 As Integer, counter6 As Integer, _
curmin As Integer, curmindist As Integer, curoverlap As Integer, elbegin As Integer, elend As Integer, _
elstep As Integer, maxoverlappings As Integer, overlapincrease As Boolean, simstart As Boolean
Sheet2.Cells(1, 1) = "MINIMAL OVERLAP ALGORITHM"
Sheet2.Cells(2, 1) = "Operation:"
Sheet2.Cells(2, 2) = "Requesting input..."
m = Val(InputBox("m =", "Lottery parameters"))
n = Val(InputBox("n =", "Lottery parameters"))
k = Val(InputBox("k =", "Lottery parameters"))
L = Val(InputBox("L =", "Lottery parameters"))
iter = Val(InputBox("Number of iterations to perform (0/empty for single iteration)?", "Iteration parameter"))
If ((m < 3) Or (n < 2) Or (k < 1) Or (n > m) Or (k > n) Or (L < 1)) Then
MsgBox "Invalid lottery parameters entered."
End ' Exit program.
End If
Randomize ' Initialise pseudo random number generator.
simstart = True
redoiteration:
Sheet2.Cells(2, 2) = "Initialising..."
Sheet2.Cells(3, 1) = "T\m"
' Initialise all variables
For counter1 = 4 To L + 4
Sheet2.Cells(counter1, 1) = counter1 - 3 ' Fill in ticket numbers 1,...,L.
For counter2 = 2 To m + 2
Sheet2.Cells(counter1, counter2) = "" ' Clear ticket matrix.
Next counter2
Next counter1
For counter1 = 1 To m ' Fill in numbers 1,...,m from Um
Sheet2.Cells(3, counter1 + 1) = counter1
Next counter1
Sheet2.Cells(L + 4, 1) = "Ovrtps"
For counter1 = 2 To m + 1 ' Clear overlapping vector
Sheet2.Cells(L + 4, counter1) = "=SUM(" & AlphaCol(counter1) & Trim(Str(4)) & ":" & _
AlphaCol(counter1) & Trim(Str(L + 3)) & "+)" & AlphaCol(counter2) & Trim(Str(L + 6))
Next counter1
Sheet2.Cells(L + 4, m + 2) = "=MIN(" & AlphaCol(2) & Trim(Str(L + 4)) & ":" & AlphaCol(m + 1) & Trim(Str(L + 4)) & ")"
Sheet2.Cells(L + 4, m + 3) = "=MAX(" & AlphaCol(2) & Trim(Str(L + 4)) & ":" & AlphaCol(m + 1) & Trim(Str(L + 4)) & ")"
Sheet2.Cells(L + 4, m + 4) = "=COUNTIF(" & AlphaCol(2) & Trim(Str(L + 4)) & ":" & _
AlphaCol(m + 1) & Trim(Str(L + 4)) & "," & AlphaCol(m + 3) & Trim(Str(L + 4)) & ")"
Sheet2.Cells(L + 5, 1) = "Ovrp Dstnce"
For counter1 = 2 To m + 1 ' Clear overlapping distance vector.
Sheet2.Cells(L + 5, counter1) = 0
Next counter1
Sheet2.Cells(L + 5, m + 2) = "=MIN(" & AlphaCol(2) & Trim(Str(L + 5)) & ":" & AlphaCol(m + 1) & Trim(Str(L + 5)) & ")"
For counter1 = 1 To L ' Clear overlapping ticket vector.
Sheet2.Cells(counter1 + 3, m + 2) = "0"
Next counter1
overlapincrease = False
maxoverlappings = 0
elbegin = m
elend = 1
elstep = -1
Sheet2.Cells(2, 2) = "Determining tickets..." & iter ' Setup interface.
Sheet2.Cells(1, 8) = "m" ' Lottery paramters <m,n;k>.
Sheet2.Cells(2, 8) = m
Sheet2.Cells(1, 9) = "n"
Sheet2.Cells(2, 9) = n
Sheet2.Cells(1, 10) = "k"
Sheet2.Cells(2, 10) = k
Sheet2.Cells(1, 11) = "L" ' Playing set cardinality L.
Sheet2.Cells(2, 11) = L
Sheet2.Cells(1, 13) = "T" ' Current ticket being generated.
Sheet2.Cells(1, 14) = "I" ' Current ticket index.
Sheet2.Cells(1, 15) = "O=" ' Overlapping cardinality.
Sheet2.Cells(2, 15) = "T=" ' 1st overlapping occurrence.
Sheet2.Cells(1, 16) = ""
Sheet2.Cells(2, 16) = ""
Sheet2.Cells(1, 17) = ""
Sheet2.Cells(2, 17) = ""
Sheet2.Cells(1, 18) = ""
Sheet2.Cells(2, 18) = ""
Sheet2.Cells(1, 19) = ""
Sheet2.Cells(2, 19) = ""
Sheet2.Cells(1, 20) = ""
Sheet2.Cells(2, 20) = ""
Sheet2.Cells(L + 6, 1) = "RU" ' Resource utilisation  $\Psi_{\ell}(m, n; k)$ 
For counter1 = 1 To L ' For all tickets 1,...,L.
Sheet2.Cells(L + 4, m + 5) = "" ' Clear 5 overlapping count.

```

```

Sheet2.Cells(L + 4, m + 6) = "" ' Clear 4 overlapping count.
Sheet2.Cells(L + 4, m + 7) = "" ' Clear 3 overlapping count.
Sheet2.Cells(L + 4, m + 8) = "" ' Clear 2 overlapping count.
Sheet2.Cells(L + 4, m + 9) = "" ' Clear 1 overlapping count.
Sheet2.Cells(L + 4, m + 10) = "" ' Clear 0 overlapping count.
Sheet2.Cells(2, 13) = counter1
curmin = Val(Sheet2.Cells(L + 4, m + 2))
curmindist = Val(Sheet2.Cells(L + 5, m + 2))
curoverlap = 0
For counter2 = 1 To L ' Clear overlapping ticket vector.
    Sheet2.Cells(counter2 + 3, m + 2) = 0
Next counter2
redonumber:
For counter2 = 1 To n ' For all elements of each ticket 1,...,n
    Sheet2.Cells(2, 14) = counter2
nextnumber:
    elbegin = Int(Rnd * m) + 1
    If (Rnd < 0.5) Then
        elstep = -1
    Else ' (Rnd >= 0.5)
        elstep = 1
    End If
    elend = elbegin - elstep
    If (elend > m) Then
        elend = 1
    ElseIf (elend < 1) Then
        elend = m
    End If
    counter3 = elbegin
    Do
        If (Val(Sheet2.Cells(L + 5, counter3 + 1)) = curmindist) And _
            (Val(Sheet2.Cells(L + 4, counter3 + 1)) = curmin) Then
            Counter4 = 4
            While (Counter4 < counter1 + 3)
                If (Sheet2.Cells(Counter4, counter3 + 1) <> "") And _
                    (Val(Sheet2.Cells(Counter4, m + 2)) > curoverlap) Then
                    GoTo nextelement
                End If
                Counter4 = Counter4 + 1
            Wend
            If (Counter4 = counter1 + 3) Then
                Sheet2.Cells(L + 4, m + 4) = Val(Sheet2.Cells(L + 4, m + 2))
                Sheet2.Cells(counter1 + 3, counter3 + 1) = 1
                Sheet2.Cells(L + 5, counter3 + 1) = counter1
                If overlapincrease Then
                    overlapincrease = False
                    curoverlap = 0
                End If
                Counter4 = counter1 + 2
                While (Counter4 > 3)
                    If (Sheet2.Cells(Counter4, counter3 + 1) <> "") Then
                        Sheet2.Cells(Counter4, m + 2) = Val(Sheet2.Cells(Counter4, m + 2)) + 1
                    End If
                    Counter4 = Counter4 - 1
                Wend
            End If
            GoTo nonextnumber
        End If
    nextelement:
        counter3 = counter3 + elstep
        If (counter3 < 1) Then
            counter3 = m
        ElseIf (counter3 > m) Then
            counter3 = 1
        End If
    Loop Until (counter3 = elbegin)
    curmindist = curmindist + 1
    If (curmindist = counter1) Then
        curmindist = Val(Sheet2.Cells(L + 5, m + 2))
        curmin = curmin + 1
        If (curmin > Val(Sheet2.Cells(L + 4, m + 4))) Then
            curmin = Val(Sheet2.Cells(L + 4, m + 2))
            elbegin = Int(Rnd * m) + 1
            If (Rnd < 0.5) Then
                elstep = -1
            Else ' (Rnd >= 0.5)
                elstep = 1
            End If
            elend = elbegin - elstep
            If (elend > m) Then
                elend = 1
            ElseIf (elend < 1) Then
                elend = m
            End If
        End If
    End If

```

```

counter3 = elbegin
Do
  Counter4 = 4
  While (Counter4 < counter1 + 3)
    If (Sheet2.Cells(Counter4, counter3 + 1) <> "") And _
      (Val(Sheet2.Cells(Counter4, m + 2)) > curoverlap) Then
      GoTo nextelement2
    End If
    Counter4 = Counter4 + 1
  Wend
  If (Counter4 = counter1 + 3) Then
    Sheet2.Cells(L + 4, m + 4) = Val(Sheet2.Cells(L + 4, m + 2))
    Sheet2.Cells(counter1 + 3, counter3 + 1) = 1
    Sheet2.Cells(L + 5, counter3 + 1) = counter1
    If overlapincrease Then
      overlapincrease = False
      curoverlap = 0
    End If
    Counter4 = counter1 + 2
    While (Counter4 > 3)
      If (Sheet2.Cells(Counter4, counter3 + 1) <> "") Then
        Sheet2.Cells(Counter4, m + 2) = Val(Sheet2.Cells(Counter4, m + 2)) + 1
      End If
      Counter4 = Counter4 - 1
    Wend
  End If
  GoTo nonextnumber
nextelement2:
  counter3 = counter3 + elstep
  If (counter3 < 1) Then
    counter3 = m
  ElseIf (counter3 > m) Then
    counter3 = 1
  End If
  Loop Until (counter3 = elbegin)
  curoverlap = curoverlap + 1
  If (curoverlap >= maxoverlappings) Then
    Sheet2.Cells(1, maxoverlappings + 16) = maxoverlappings + 1
    Sheet2.Cells(2, maxoverlappings + 16) = counter1
    maxoverlappings = maxoverlappings + 1
  End If
  overlapincrease = True
  GoTo nextnumber
End If
End If
GoTo nextnumber
nonextnumber:
  curoverlap = 0
  Next counter2
Next counter1
Sheet2.Cells(2, 2) = "Finished..." & iter
If (iter > 0) Then
  counter1 = 4 ' Insert all ticket overlappings for comparison with best thus far.
  While (counter1 - 3 < Val(Sheet2.Cells(2, 11))) ' Clear overlapping ticket vector.
    counter2 = 0
    While counter2 <= L - counter1 + 2
      Sheet2.Cells(counter1, Val(Sheet2.Cells(2, 8)) + 2 + counter2) = 0
      counter2 = counter2 + 1
    Wend
    counter1 = counter1 + 1
  Wend
  counter3 = 0
  Counter4 = L
  While (Counter4 > 0)
    For counter1 = 2 To Val(Sheet2.Cells(2, 8)) + 1 ' Update overlapping ticket vector.
      If (Sheet2.Cells(Counter4 + 3, counter1) <> "") Then
        counter2 = Counter4 + 2
        While (counter2 > 3)
          If (Sheet2.Cells(counter2, counter1) <> "") Then
            Sheet2.Cells(counter2, Val(Sheet2.Cells(2, 8)) + 2 + counter3) = _
              Val(Sheet2.Cells(counter2, Val(Sheet2.Cells(2, 8)) + 2 + counter3)) + 1
          End If
          counter2 = counter2 - 1
        Wend
      End If
    Next counter1
    Sheet2.Cells(Counter4 + 3, Val(Sheet2.Cells(2, 8)) + 2 + counter3) = "—"
    Counter4 = Counter4 - 1
    counter3 = counter3 + 1
  Wend
  Sheet2.Cells(L + 4, m + 5) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",5)"
  Sheet2.Cells(L + 5, m + 5) = 5
  Sheet2.Cells(L + 4, m + 6) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",4)"
  Sheet2.Cells(L + 5, m + 6) = 4

```

```

Sheet2.Cells(L + 4, m + 7) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",3)"
Sheet2.Cells(L + 5, m + 7) = 3
Sheet2.Cells(L + 4, m + 8) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",2)"
Sheet2.Cells(L + 5, m + 8) = 2
Sheet2.Cells(L + 4, m + 9) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",1)"
Sheet2.Cells(L + 5, m + 9) = 1
Sheet2.Cells(L + 4, m + 10) = "=COUNTIF(" & AlphaCol(m + 2) & "4:" & AlphaCol(m + 1 + L) & Trim(Str(L + 3)) & ",0)"
Sheet2.Cells(L + 5, m + 10) = 0
Sheet2.Cells(L + 6, 2) = RU() ' Determine resource utilisation  $\Psi_\ell(m,n;k)$  of constructed playing set.
If (Val(Sheet2.Cells(L + 6, 2)) > Val(Sheet21.Cells(L + 6, 2))) Then ' Save info if improved solution is found.
    Sheet2.Cells(2, 2) = "Saving data..." & iter
    For counter1 = 1 To L + 6
        For counter2 = 1 To m + 1 + L
            Sheet21.Cells(counter1, counter2) = Sheet2.Cells(counter1, counter2)
        Next counter2
    Next counter1
    Sheet2.Cells(2, 2) = "Done..."
    simstart = False
End If
End If
iter = iter - 1
If (iter > 0) Then GoTo redoiteration
End Sub

```

## A.4 Neighbourhood removal algorithm (Algorithm 5)

```

#include<iostream.h> // Header file for input and output.
#include<iomanip.h> // Header file used to manipulate output.
#include<fstream.h> // Header file for using files.
#include<time.h> // Header file for calculating iteration times.
#include<math.h> // Header file containing mathematical functions.
#include<bitset> // Header file containing the type bitset.

long double fact (int);
long int round (const long double);

// Compute the factorial of n (i.e. n!).
long double fact (int n) {
    if (n <= 1) return 1; // Stop recursion.
    else return n * fact(n - 1); // (n > 1)
}

// Rounds a number off to the nearest integer to avoid possible numerical truncation errors.
long int round (const long double n) {
    long int flrnum = (long int)floor(n);
    if (n - flrnum < 0.5) return flrnum;
    else return (long int)ceil(n); // (n - flrnum >= 0.5)
}

int main () {
    short int m, n, k, L; // Variables containing the lottery parameters <m,n;k> and playing set cardinality L.
    cout << endl << "\t NEIGHBOURHOOD REMOVAL ALGORITHM" << endl << endl;
    cout << "Please specify the following parameters for the lottery (m,n;k):" << endl;
    cout << "m = "; cin >> m;
    cout << "n = "; cin >> n;
    cout << "k = "; cin >> k;
    cout << "Playing set cardinality = "; cin >> L;
    // If the user specified invalid lottery parameters.
    if ((n > m-1) || (k > n-1) || (m < 3) || (n < 2) || (k < 1)) {
        cout << endl << "Invalid lottery parameters." << endl;
        return 0; // Exit program.
    }
    else if (L < 1) {
        cout << endl << "Playing set cardinality must be at least 1." << endl;
        return 0; // Exit program.
    }
    long int NumTickets, r = 0, CurPSTicketNumber, TicketIndexNumber, BestTicketNumber, DomElementNumber = 0, tmpsum,
        tmpsumback = 0; // Order of lottery graph, degree of regularity, Current playing set ticket number.
    short int CurTicket[n], TicketIndex[n], BestTicket[n];
    bitset<XX> CurTicketBITSET, TicketIndexBITSET, BestTicketBITSET; // The value XX should be fixed to take the value
        m at compile time.
    bitset<YY> G, CurTicketNeighbourhood, BestTicketNeighbourhood; // The value YY should be fixed to take the value
         $\binom{m}{n}$  at compile time.
    time_t StartTime; // Trace execution time.
    short int index1, index2, index3, index4, indexm, indexn, indexk, CurrentNumber = n, Intersect;
    // Compute the order and degree of regularity of the lottery graph G<m,n;k>.
    NumTickets = round(fact(m)/(fact(n)*fact(m-n)));
    for (short int i = k; i < n; i++)
        r += round((fact(n)/(fact(n-i)*fact(i)))*(fact(m-n)/(fact(n-i)*fact(m-2*n+i))));
    cout << endl << "The order of the lottery graph G(" << m << ", " << n << ", " << k << ") is " << NumTickets << " and it is"
        << r << " regular." << endl;
    G.set(); // Initialise G to have all (NumTickets) vertices (111...1111).

```



```

StartTime = time(NULL); // Capture start of execution time.
cout << endl << "No." << " Ticket (t) " << " —N[t]— " << endl
<< "—" << "—" << "—" << "—" << endl;
// Start the extracting vertices from G<m,n;k>.
while ((G.count() > 0) && (DomElementNumber < L)) { // G.count() returns the number of 1 bits in G.
  CurPSTicketNumber = -1;
nextCurPSTicketNumber:
  while ((CurPSTicketNumber < NumTickets) && (!G.test(++CurPSTicketNumber))); // G.test(x) returns TRUE if the
  bit x = 1, and FALSE otherwise.
  if (CurPSTicketNumber == NumTickets) goto Finished;
  CurTicketBITSET.reset(); // Clear current ticket (000...0000).
  // Determine actual ticket corresponding to lexicographic ticket number: CurPSTicketNumber.
  indexm = m; indexn = n; tmpsum = tmpsumback = 0; index4 = 1;
  for (index2 = 0; index2 < n-1; index2++) {
    index3 = 1; tmpsum = tmpsumback;
    while (index3 < m) {
      tmpsumback = tmpsum;
      tmpsum += round((long double)fact(indexm-index3)/(long double)(fact(indexn-1)*fact(indexm-index3-indexn+1)));
      if (tmpsum >= (CurPSTicketNumber + 1)) {
        CurTicket[index2] = index4++;
        goto nextCurTicketelement;
      }
      index3++; index4++;
    }
  }
nextCurTicketelement:
  indexm = m - CurTicket[index2]; // Rescale search range.
  indexn--;
}
CurTicket[n-1] = CurTicket[n-2] + (CurPSTicketNumber + 1) - tmpsumback; // Pad last element.
for (index1 = 0; index1 < n; CurTicketBITSET.flip(CurTicket[index1++]-1));
CurTicketNeighbourhood.reset(); // Current Ticket has no neighbours (000...0000).
// CurTicket contains first remaining lexicographic ticket in G.
// Start from first remaining lexicographic ticket in G to determine neighbourhood.
TicketIndexNumber = -1;
nextTicketIndexNumber:
  while ((TicketIndexNumber < NumTickets) && (!G.test(++TicketIndexNumber)));
  if (TicketIndexNumber == NumTickets) goto FinishedNeighbourhood;
  TicketIndexBITSET.reset(); // Clear Ticket index (000...0000).
  // Determine actual ticket corresponding to lexicographic ticket number: TicketIndexNumber.
  indexm = m; indexn = n; tmpsum = tmpsumback = 0; index4 = 1;
  for (index2 = 0; index2 < n-1; index2++) {
    index3 = 1; tmpsum = tmpsumback;
    while (index3 < m) {
      tmpsumback = tmpsum;
      tmpsum += round((long double)fact(indexm-index3)/(long double)(fact(indexn-1)*fact(indexm-index3-indexn+1)));
      if (tmpsum >= (TicketIndexNumber + 1)) {
        TicketIndex[index2] = index4++;
        goto nextTicketIndexelement;
      }
      index3++; index4++;
    }
  }
nextTicketIndexelement:
  indexm = m - TicketIndex[index2]; // Rescale search range.
  indexn--;
}
TicketIndex[n-1] = TicketIndex[n-2] + (TicketIndexNumber + 1) - tmpsumback; // Pad last element.
for (index1 = 0; index1 < n; TicketIndexBITSET.flip(TicketIndex[index1++]-1));
TicketIndexBITSET &= CurTicketBITSET; // Determine set intersection.
// Check whether current ticket is dominated by candidate playing set.
if (TicketIndexBITSET.count() >= k)
  CurTicketNeighbourhood.flip(TicketIndexNumber); // CurTicketNeighbourhood[TicketIndexNumber] = 1;
goto nextTicketIndexNumber;
FinishedNeighbourhood:
  if (BestTicketNeighbourhood.count() < CurTicketNeighbourhood.count()) {
    BestTicketNeighbourhood = CurTicketNeighbourhood; // Store ticket with largest neighbourhood set.
    BestTicketNumber = CurPSTicketNumber;
    for (index1 = 0; index1 < n; BestTicket[index1++] = CurTicket[index1]);
  }
  if (DomElementNumber > 0)
    goto nextCurPSTicketNumber; // Consider next vertex to be removed from G<m,n;k>.
Finished:
  PlayingSet[DomElementNumber].Number = BestTicketNumber; // Record best ticket candidate information.
  PlayingSet[DomElementNumber].NumDominate = BestTicketNeighbourhood.count(); // Determine |N_G[v_t]|.
  PlayingSet[DomElementNumber].set = new(short int [n]);
  for (index1 = 0; index1 < n; PlayingSet[DomElementNumber].set[index1++] = BestTicket[index1]);
  G ^= BestTicketNeighbourhood; // Remove closed neighbourhood of best candidate.
  BestTicketNeighbourhood.reset(); // Clear best candidate neighbourhood.
  CurTicketNeighbourhood.reset(); // Clear current ticket neighbourhood.
  // Output best ticket candidate information per iteration.
  cerr << setw(4) << (DomElementNumber + 1) << " ";
  for (index1 = 0; index1 < n; index1++)
    cerr << setw(3) << PlayingSet[DomElementNumber].set[index1];
  cerr << setw((10-n)*3) << " " << " " << setw(7) << PlayingSet[DomElementNumber++].NumDominate << "/" << setw(10)
  << NumTickets << " " << setw(10) << G.count() << " vertices remaining." << endl;

```

```

}
cerr << endl << "Finished in " << (time(NULL) - StartTime) << " seconds." << endl << endl;
return 0; // Exit program.
}

```

## A.5 Tabu search algorithm (Algorithm 6)

```

#include<iostream.h> // Header file for receiving input and generating output.
#include<iomanip.h> // Header file for manipulating output.
#include<time.h> // Header file for calculating iteration times.
#include<math.h> // Header file for mathematical function.
#include<stl.h> // Header file containing Standard Template Libraries (STLs) <set>.
using namespace std;
long double fact (const int);
long int round (const long double);
long int fitness (const short int, const short int, const short int, const short int, const long int, const long int []);

// Compute the factorial of n (i.e. n!).
long double fact (const int n) {
    if (n <= 1) return 1; // Stop recursion.
    else return n * fact(n-1); // (n > 1)
}

// Rounds a number off to the nearest integer to avoid possible numerical truncation errors.
long int round (const long double n) {
    long int flrnum = (long int)floor(n);
    if (n - flrnum < 0.5) return flrnum;
    else return (long int)ceil(n); // (n - flrnum >= 0.5)
}

long int fitness (const short int m, const short int n, const short int k, const short int L,
    const long int NumTickets, const long int DomArrayNumber[]) {
    short int CurTicket[n], CurrentNumber = n; // Current ticket used in fitness measurement (CurTicket), index for
    generating tickets lexicographically (CurrentNumber).
    short int index1, index2, index3, index4, indexm, indexn, indexk, DomArray[L][n];
    long int tmpsum, tmpsumback = 0, CandidateFitness;
    for (index1 = 0; index1 < L; index1++) { // Determine actual tickets corresponding to lexicographic ticket numbers.
        indexm = m; indexn = n; tmpsum = tmpsumback = 0; index4 = 1;
        for (index2 = 0; index2 < n-1; index2++) {
            index3 = 1; tmpsum = tmpsumback;
            while (index3 < m) {
                tmpsumback = tmpsum;
                tmpsum += round(((long double)fact(indexm-index3)/((long double)(fact(indexn-1)*fact(indexm-index3-indexn+1)))));
                if (tmpsum >= DomArrayNumber[index1]) {
                    DomArray[index1][index2] = index4++; goto nextelement1;
                }
                index3++; index4++;
            }
        }
        nextelement1:
        indexm = m - DomArray[index1][index2]; // Rescale search range.
        indexn--;
        DomArray[index1][n-1] = DomArray[index1][n-2] + DomArrayNumber[index1] - tmpsumback; // Pad last element.
    }
    // Determine candidate resource utilisation  $\Psi_t(m, n; k)$ . Initialise Ticket to [1, 2, ..., n].
    for (index1 = 0; index1 < n; index1++) CurTicket[index1] = index1 + 1;
    for (tmpsum = 0; tmpsum < NumTickets; tmpsum++) {
        indexk = 0;
        for (index1 = 0; (index1 < L) && (indexk < k); index1++) { // Check whether current ticket is dominated.
            indexk = index2 = index3 = 0;
            while ((index2 < n) && (index3 < n) && (indexk < k))
                if (DomArray[index1][index2] < CurTicket[index3])
                    index2++;
            else if (DomArray[index1][index2] > CurTicket[index3])
                index3++;
            else { // (DomArray[index1][index2] == CurTicket[tcounter])
                indexk++; index2++; index3++; // Tickets intersect in 1 element.
            }
            if (indexk == k) // Check whether current ticket is dominated by playing set.
                CandidateFitness++;
        }
        // Generate next ticket in lexicographic sequence.
        CurTicket[CurrentNumber-1]++;
        if (CurTicket[CurrentNumber-1] > m) {
            CurTicket[CurrentNumber-1]--;
            while ((CurrentNumber > 0) && (CurTicket[CurrentNumber-1] == m-(n-CurrentNumber))) CurrentNumber--;
            CurTicket[CurrentNumber-1]++;
            for (index4 = CurrentNumber; index4 < n; index4++) CurTicket[index4] = CurTicket[index4-1] + 1;
            CurrentNumber = n;
        }
    }
}

return CandidateFitness;

```

```

}
int main () {
short int m, n, k, L; // Variables containing the lottery parameters <m,n,k> and playing set cardinality L.
const long int iter = 1000; // This value may be changed, depending on the number of iterations preferred.
cerr << endl << "\t TABU SEARCH ALGORITHM" << endl << endl;
cerr << "Please specify the following parameters for the lottery (m,n,k):" << endl;
cerr << "m = "; cin >> m;
cerr << "n = "; cin >> n;
cerr << "k = "; cin >> k;
// If the user specified invalid lottery parameters.
if ((n > m-1) || (k > n-1) || (m < 3) || (n < 2) || (k < 1)) {
cerr << endl << "Invalid lottery parameters entered." << endl;
return 0; // Exit program.
}
cerr << "Playing set cardinality L = "; cin >> L;
if (L < 1) {
cerr << endl << "Playing set cardinality must be at least 1." << endl;
return 0; // Exit program.
}
const long int NumTickets = round(fact(m)/(fact(n)*fact(m-n)));
cerr << endl << "The order of the lottery graph G(" << m << ", " << n << ", " << k << ") is " << NumTickets << "."
<< endl << endl;
const long int tabutenure = NumTickets/m, NumMoves = 3;
long int SectorSize, tempvar, curiter = 1, numelements;
struct HistoryMove {
short int move, wrapdirection, bitchange; // Move corresponds to Tabu move 1 (wrap), Tabu move 2 (flip), Tabu move 3
(sector), Move 1 data (wrap direction, 0 = ←, 1 = →), Move 2 data (bit changed).
long int wraplength, bitpos, sec1start, sec2start, seclength; // Move 1 data (wrap length), Move 2 data (original
bit position), Move 3 data (sector 1 start pos., sector 2 start pos., sector length).
HistoryMove *prev, *next; // Pointer to previous & next element in Tabu search's recency-based memory history list.
};
struct NewCandidate {
short int move, wrapdirection, bitchange; // Move corresponds to Tabu move 1 (wrap), Tabu move 2 (flip), Tabu move 3
(sector), Move 1 data (wrap direction, 0 = ←, 1 = →), Move 2 data (bit to be changed).
long int wraplength, bitpos, sec1start, sec2start, seclength, *DomNumbers, Fitness; // Move 1 data (wrap length),
Move 2 data (original bit position), Move 3 data (sector 1 start pos., sector 2 start pos., sector length),
Domination element lexicographic numbers, Fitness of new candidate solution.
NewCandidate *prev, *next; // Pointers to previous & next new candidate solution.
};
NewCandidate *Candidates = NULL, *newcandidateindex, *bestcandidateindex; // Generate new candidate list.
HistoryMove *History = NULL, *historyindex; // Generate Tabu search recency-based memory history list.
long int DomArrayNumber[L], BestSolution[L], counter1, counter2, counter3, counter4, *tmpArray = new(long int [L]),
Fitness, BestFitness = 0, historycount = 0;
short int DomArray[L][n], dcounter = 0, curmove; // Playing set (of cardinality L), Playing set counter, current move
evaluated in the Tabu search.
set<long int, less<long int> > DAN;
set<long int, less<long int> >::iterator i;
bool movetabu; // Classifies a move as tabu or non-tabu.
srand(time(NULL)); // Initialise pseudo random number generator.
while (DAN.size() < L) // Generate random initial candidates.
DAN.insert((long int)((random()/(float)RAND_MAX)*NumTickets)+1);
i = DAN.begin();
for (counter1 = 0; i != DAN.end(); DomArrayNumber[counter1++] = *i++); // Order domination elements lexicographically.
nextiteration:
curmove = (short int)((random()/(float)RAND_MAX)*200); // Determine next neighbouring move.
if (curmove < 1) curmove = 1;
else if (curmove < 199) curmove = 2;
else curmove = 3; // (curmove == 199)
if (curmove == 1) { // MOVE 1
newcandidateindex = new(NewCandidate); newcandidateindex->move = curmove; // Store move.
newcandidateindex->wrapdirection = 0; // Direction ← (left).
newcandidateindex->wraplength = (long int)((random()/(float)RAND_MAX)*NumTickets)+1; // Determine wrap length.
for (counter1 = 0; counter1 < L; counter1++) { // Perform ← (left) wrap-around (modulo NumTickets).
tmpArray[counter1] = (DomArrayNumber[counter1] - newcandidateindex->wraplength + NumTickets) % NumTickets;
if (tmpArray[counter1] == 0) tmpArray[counter1] = NumTickets;
}
newcandidateindex->DomNumbers = tmpArray;
newcandidateindex->next = newcandidateindex->prev = newcandidateindex; // First element linked list.
Candidates = newcandidateindex; // Insert candidate solution in new candidate solution list (Candidates == NULL).
newcandidateindex = new(NewCandidate); newcandidateindex->move = curmove;
newcandidateindex->wrapdirection = 1; // Direction → (right).
newcandidateindex->wraplength = Candidates->wraplength; // Copy wraplength.
tmpArray = new(long int [L]);
for (counter1 = 0; counter1 < L; counter1++) { // Perform → (right) wrap--around (modulo NumTickets).
tmpArray[counter1] = (DomArrayNumber[counter1] + newcandidateindex->wraplength) % NumTickets;
if (tmpArray[counter1] == 0) tmpArray[counter1] = NumTickets;
}
newcandidateindex->DomNumbers = tmpArray;
newcandidateindex->next = newcandidateindex->prev = Candidates; // Add new candidate solution to current solution list.
Candidates->prev = Candidates->next = newcandidateindex;
}
else if (curmove == 2) { // MOVE 2
newelementchange:
dcounter = (short int)((random()/(float)RAND_MAX)*L); // Choose a bit to change.

```

```

counter1 = DomArrayNumber[dcounter]; // Store original bit value.
// Determine range of values new bit may assume/take.
(dcounter == 0 ? counter2 = 1 : counter2 = DomArrayNumber[dcounter-1]+1);
(dcounter == L-1 ? counter3 = NumTickets+1 : counter3 = DomArrayNumber[dcounter+1]-1);
if (counter2 == counter3) goto newelementchange; // Redo choice of element to be changed.
while (counter2 < counter3) {
  if (Candidates == NULL) { // No candidate solution exist.
    Candidates = new(NewCandidate); Candidates->prev = Candidates->next = Candidates;
  }
  else { // (Candidates != NULL)
    newcandidateindex = Candidates->prev; //Record last entered candidate solution position.
    Candidates->prev = new(NewCandidate); // Generate new candidate solution.
    newcandidateindex->next = Candidates->prev; // Add new candidate solution to list.
    Candidates->prev->next = Candidates; Candidates->prev->prev = newcandidateindex;
  }
  Candidates->prev->DomNumbers = new(long int [L]);
  for (counter4 = 0; counter4 < L; counter4++)
    Candidates->prev->DomNumbers[counter4] = DomArrayNumber[counter4]; // Copy original information.
  Candidates->prev->DomNumbers[dcounter] = counter2;
  Candidates->prev->bitchange = dcounter; // Store changed bit.
  Candidates->prev->bitpos = counter2;
  Candidates->move = curmove; // Store move.
  counter2++; // Move to next possible solution.
  if (counter2 == counter1) counter2++; // Jump over "current solution."
}
}
else { // (curmove == 3), MOVE 3
  numelements = ((int)(random()/(float)RAND_MAX)*50+1)*100; // SectorSize determining sector size of move 3.
  SectorSize = (long int)((-2*fact(m)*fact(m-n)*fact(n)+3*pow(fact(m-n),2)*pow(fact(n),2)+4*NumTickets*pow(fact(m-n),2)
    *pow(fact(n),2)-fact(m-n)*fact(n)*sqrt(8*pow(fact(m),2)-16*NumTickets*fact(m)*fact(m-n)*fact(n)
    +(numelements/100)*801*pow(fact(m-n),2)*pow(fact(n),2)+8*pow(NumTickets,2)*pow(fact(m-n),2)
    *pow(fact(n),2)))/(4*pow(fact(m-n),2)*pow(fact(n),2)));
  for (counter1 = 1; counter1 <= (NumTickets-(2*SectorSize)+1); counter1++) {
    for (counter2 = counter1+SectorSize; counter2 <= (NumTickets-SectorSize+1); counter2++) {
      counter3 = 0;
      if (Candidates == NULL) { // No candidate solution exist.
        Candidates = new(NewCandidate); Candidates->prev = Candidates->next = Candidates;
      }
      else { // (Candidates != NULL)
        newcandidateindex = Candidates->prev; // Record last entered candidate solution position.
        Candidates->prev = new(NewCandidate); // Generate new candidate solution.
        newcandidateindex->next = Candidates->prev; // Add new candidate solution to list.
        Candidates->prev->next = Candidates; Candidates->prev->prev = newcandidateindex;
      }
      Candidates->prev->DomNumbers = new(long int [L]);
      do {
        if ((DomArrayNumber[counter3] >= counter1) && (DomArrayNumber[counter3] < (counter1+SectorSize)))
          Candidates->prev->DomNumbers[counter3++] = DomArrayNumber[counter3]+(counter2-counter1);
        else if ((DomArrayNumber[counter3] >= counter2) && (DomArrayNumber[counter3] < (counter2+SectorSize)))
          Candidates->prev->DomNumbers[counter3++] = DomArrayNumber[counter3]-(counter2-counter1);
        else // ((DomArrayNumber[counter3] < counter1) || ((DomArrayNumber[counter3] >= counter1+SectorSize)
          && (DomArrayNumber[counter3] < counter2)) || (DomArrayNumber[counter3] >= counter2+SectorSize))
          Candidates->prev->DomNumbers[counter3++] = DomArrayNumber[counter3];
      } while (counter3 < L);
      Candidates->prev->move = curmove; // Store move.
      Candidates->prev->sec1start = counter1; // Store move information.
      Candidates->prev->sec2start = counter2; Candidates->prev->seclength = SectorSize;
    }
  }
}
}
newcandidateindex = Candidates;
do {
  if (curmove != 2) {
    DAN.erase(DAN.begin(), DAN.end()); // Empty set.
    for (counter1 = 0; counter1 < L; DAN.insert(newcandidateindex->DomNumbers[counter1++]));
    i = DAN.begin();
    for (counter1 = 0; i != DAN.end(); newcandidateindex->DomNumbers[counter1++] = *i++); // Lexicographic ordering.
  }
  newcandidateindex->Fitness = fitness(m, n, k, L, NumTickets, newcandidateindex->DomNumbers); // Determine  $\Psi_{\ell}(m, n; k)$ .
  newcandidateindex = newcandidateindex->next; // Move to next element in candidate solution list.
} while (newcandidateindex != Candidates);
movetabu = false; // Initialise move as non-tabu.
bestcandidateindex = newcandidateindex = Candidates; // Initialise best solution.
do { // Choose neighbouring solution from candidate solutions.
  if (History != NULL) { // Check recency-based memory for tabu moves.
    historyindex = History;
    do {
      if (historyindex->move == newcandidateindex->move) // Move may possibly be tabu.
        if (newcandidateindex->move == 1) // MOVE 1
          if ((newcandidateindex->wrapdirection == historyindex->wrapdirection) &&
            (newcandidateindex->wraplength == historyindex->wraplength))
            movetabu = true; // Move is clasified as tabu.
          else if (newcandidateindex->move == 2) // MOVE 2
            if ((newcandidateindex->bitchange == historyindex->bitchange) &&

```

```

        ((newcandidateindex->bitpos > historyindex->bitpos-(long int)(NumTickets/(float)(L*tabutenure))) &&
         (newcandidateindex->bitpos < historyindex->bitpos+(long int)(NumTickets/(float)(L*tabutenure))))))
        movetabu = true; // Move is classified as tabu.
    else // (newcandidateindex->move == 3), MOVE 3
        if ((newcandidateindex->sec1start == historyindex->sec1start) &&
            (newcandidateindex->sec2start == historyindex->sec2start))
            movetabu = true; // Move is classified as tabu.
        historyindex = historyindex->next; // Move to next recency-based history/memory element (in Tabu list).
    } while (historyindex != History);
}
if (!movetabu)
    if (bestcandidateindex->Fitness < newcandidateindex->Fitness) bestcandidateindex = newcandidateindex;
else // (movetabu)
    if (BestFitness < newcandidateindex->Fitness) // Aspiration criterion (move tabu but yields improvement).
        bestcandidateindex = newcandidateindex;
    movetabu = false; // Clear move as non-tabu.
    newcandidateindex = newcandidateindex->next; // Select next candidate solution.
} while (newcandidateindex != Candidates);
// Add selected move to new solution to recency-based history/memory (Tabu list).
if (History == NULL) {
    History = new(HistoryMove); History->next = History->prev = History; historyindex = History;
}
else historyindex = new(HistoryMove); // (History != NULL)
historyindex->move = bestcandidateindex->move;
if (bestcandidateindex->move == 1) { // MOVE 1
    historyindex->wrapdirection = 1 - bestcandidateindex->wrapdirection;
    historyindex->wraplength = bestcandidateindex->wraplength;
}
else if (bestcandidateindex->move == 2) { // MOVE 2
    historyindex->bitchange = bestcandidateindex->bitchange; historyindex->bitpos = bestcandidateindex->bitpos;
}
else { // (bestcandidateindex->move == 3), MOVE 3
    historyindex->sec1start = bestcandidateindex->sec2start; historyindex->sec2start = bestcandidateindex->sec1start;
    historyindex->seclength = bestcandidateindex->seclength;
}
History->prev->next = historyindex; // Insert best candidate solution move to recency-based memory tabu list.
historyindex->prev = History->prev; historyindex->next = History; History->prev = historyindex;
if (historycount > tabutenure) {
    historyindex = History; History->prev->next = History->next; History->next->prev = History->prev;
    History = History->next; historyindex->next = historyindex->prev = NULL;
    delete historyindex; // Remove oldest tabu move from recency-based history/memory tabu list.
}
else historycount++; // (historyindex <= tabutenure)
Fitness = bestcandidateindex->Fitness; // Store current new candidate solution.
for (counter1 = 0; counter1 < L; counter1++) DomArrayNumber[counter1] = bestcandidateindex->DomNumbers[counter1];
cerr << setw(6) << curiter++ << setw(10) << Fitness << "/" << NumTickets << " (Move " << curmove << " : " << endl;
if (curmove == 1) // MOVE 1
    cerr << "direction = " << bestcandidateindex->wrapdirection << ", length = " << bestcandidateindex->wraplength << " ";
else if (curmove == 2) // MOVE 2
    cerr << "bitchange = " << bestcandidateindex->bitchange << ", bitpos = " << bestcandidateindex->bitpos << " ";
else // (curmove == 3), MOVE 3
    cerr << "sec1start = " << bestcandidateindex->sec1start << ", sec2start = " << bestcandidateindex->sec2start
        << ", seclength = " << bestcandidateindex->seclength << ", numelements = " << numelements << " ";
if (BestFitness < bestcandidateindex->Fitness) { // Store best solution found thus far.
    BestFitness = bestcandidateindex->Fitness;
    for (counter1 = 0; counter1 < L; counter1++) BestSolution[counter1] = bestcandidateindex->DomNumbers[counter1];
    cerr << setw(3) << "*" << endl;
}
else cerr << endl;
while (Candidates != Candidates->next) { // Clear all generated candidate solutions.
    newcandidateindex = Candidates; Candidates->prev->next = Candidates->next; Candidates->next->prev = Candidates->prev;
    Candidates = Candidates->next; newcandidateindex->next = newcandidateindex->prev = NULL;
    delete [] newcandidateindex->DomNumbers;
    delete newcandidateindex; // Remove generated candidate from solution list.
}
Candidates->next = Candidates->prev = NULL; delete Candidates; Candidates = NULL;
if (curiter <= iter) goto nextiteration; // Iterations not finished.
short int index1, index2, index3, index4, indexm, indexn, indexk;
long int tmpsum, tmpsumback = 0;
cerr << endl << "Best solution found : (fitness " << BestFitness << "/" << NumTickets << " )" << endl;
// Determine actual tickets corresponding to lexicographic ticket numbers of the Best solution found.
for (index1 = 0; index1 < L; index1++) {
    indexm = m; indexn = n; tmpsum = tmpsumback = 0; index4 = 1;
    for (index2 = 0; index2 < n-1; index2++) {
        index3 = 1; tmpsum = tmpsumback;
        while (index3 < m) {
            tmpsumback = tmpsum;
            tmpsum += round((long double)fact(indexm-index3)/(long double)(fact(indexn-1)*fact(indexm-index3-indexn+1)));
            if (tmpsum >= DomArrayNumber[index1]) {
                DomArray[index1][index2] = index4++; goto nextelement2;
            }
            index3++; index4++;
        }
    }
}
nextelement2:

```

```

        indexm = m - DomArray[index1][index2]; // Rescale search range.
        indexn--;
    }
    DomArray[index1][n-1] = DomArray[index1][n-2] + DomArrayNumber[index1] - tmpsumback; // Pad last element.
    for (index2 = 0; index2 < n; index2++) cerr << setw(3) << DomArray[index1][index2];
    cerr << endl;
}
return 0; // Exit program.
}

```

## A.6 Classical genetic algorithm

```

#include<iostream> // Header file for receiving input and generating output.
#include<iomanip> // Header file for manipulating output strings.
#include<stdlib.h> // Header file for using atoi (int char *) function.
#include<time.h> // Header file for calculating iteration times.
#include<math.h> // Header file for mathematical functions.
#include<fstream> // Header file for file input/output.
#include<set> // STL header file for handling sets.
#include<iterator> // STL header file for using iterators.
#include "randoma.h" // Header file for using Mersenne Twister pseudo-random number generator [78].

using namespace std;

long double combination (const int, const int);

long double combination (const int m, const int n) {
    if ((n == 1) || (n == m-1))
        return (long) m; //  $\binom{m}{1} = \binom{m}{m-1} = m$ .
    if ((m == n) || (n == 0))
        return 1; //  $\binom{m}{m} = 1$ .
    if (n > m)
        return 0; // If  $n > m$   $\binom{m}{n} = 0$ .

    long double fraction = m;

    // Generate the sequence  $\frac{(m-i+1)!}{i!}$  where  $i = 2, \dots, n$ .
    for (short int i = 2; i <= n; fraction = fraction * (m - i + 1) / i++);

    return (long double) fraction;
}

int main(int argc, char* argv[]) {
    if ((argc > 1) && (argc < 7)) { // If insufficient command line parameters were given by the user.
        cerr << endl << "\tGenetic algorithm for generating lottery sets for (m,n;k)" << endl;
        cerr << "\tUsage: ganormal m n k (where you supply the values for the lottery (m,n;k))" << endl << endl;
        return 0;
    }
    short int m, n, k, L, population, generations;
    time_t StartTime;

    if (argc == 1) { // If no command line parameters were given by the user, read from standard input.
        cerr << endl << "\tGenetic algorithm for generating lottery sets for (m,n;k)" << endl << endl;
        cerr << "\tPlease enter the following lottery parameters for (m,n;k):" << endl << endl;
        cerr << "m = "; cin >> m;
        cerr << "n = "; cin >> n;
        cerr << "k = "; cin >> k;
        cerr << "L = "; cin >> L;
        cerr << endl << endl << "\tPlease enter the following genetic algorithm parameters:" << endl << endl;
        cerr << "Population size = "; cin >> population;
        cerr << "# of generations = "; cin >> generations;
    }
    else { // (argc != 1) Sufficient command line parameters were given by the user.
        m = (short int) atoi (argv[1]); n = (short int) atoi (argv[2]);
        k = (short int) atoi (argv[3]); L = (short int) atoi (argv[4]);
        population = (short int) atoi (argv[5]); generations = (short int) atoi (argv[6]);
        cerr << endl << "\tGenetic algorithm for generating lottery sets for (" << m << ", " << n << ", " << k << ")" << endl << endl;
    }

    // Check whether the user input was correct/valid.
    if ((n >= m) || (k >= n) || (k <= 0) || (population < 2) || (generations < 2)) {
        cerr << endl << "Incorrect/invalid lottery parameter input..." << endl;
        return 0;
    }
    if ((population % 2) != 0) {
        cerr << endl << "Population size has to be even and > 2..." << endl;
        return 0;
    }
}

```



```

for (CurrentTicketNumber = 0; CurrentTicketNumber < NumTickets; CurrentTicketNumber++) {
    // Generate the next lexicographic lottery ticket.
    CurrentTicket[index - 1]++;
    if (CurrentTicket[index - 1] == m) {
        CurrentTicket[index - 1]--;
        while ((index > 0) && (CurrentTicket[index - 1] == m - (n - index + 1)))
            index--;
        if (index > 0)
            CurrentTicket[index - 1]++;
        for (i = index; i < n; i++)
            CurrentTicket[i] = CurrentTicket[i - 1] + 1;
        index = n;
    }
    for (CurrentSet.clear (), i = 0; i < n; CurrentSet.insert (CurrentSet.end (), CurrentTicket[i++])); // Initialise
    ticket in set form.
    for (i = 0; i < population; i++) // Check for common k-subsets with current ticket in whole of population.
        for (j = 0; j < L; j++) {
            TempSet.clear (); // Clear temporary set to determine intersection.
            set_intersection (GAPopulation[i][j].begin (), GAPopulation[i][j].end (), CurrentSet.begin (),
                CurrentSet.end (), inserter (TempSet, TempSet.end ()));
            if (!(TempSet.size () < k)) { // Determine whether sets share at least a common k-subset.
                GAFitness[i]++;
                break;
            }
        }
}

for (GAFitnessSum = i = 0; i < population; GAFitnessSum += GAFitness[i++]); // Tally fitness of population candidates
(used in distribution of pairing candidates for crossover).
for (GAMinFitness = NumTickets, GAMeAnFitness = GAMAXFitness = i = 0; i < population; i++) {
    if (GAFitness[i] < GAMinFitness)
        GAMinFitness = GAFitness[i]; // Keep track of minimum fitness of population of candidates.
    else if (GAFitness[i] > GAMAXFitness)
        GAMAXFitness = GAFitness[i]; // Keep track of MAXIMUM fitness of population of candidates.
}
GAMeAnFitness = (GAFitnessSum / population); // Determine the mEAn fitness of the population of candidates.
// Pair candidates for crossover relative to individual fitness (fitter/less fit candidates are more likely to be paired
with fitter/less fit candidates).

while (GAFitnessSum > 0) {
    FitnessIndex = TIRandom (1, GAFitnessSum); // Select 1st candidate, relative to fitness of whole population.
    for (TempFitnessSum = i = 0; TempFitnessSum < FitnessIndex; TempFitnessSum += GAFitness[i++]); // Find 1st candidate.
    GAFitnessSum -= GAFitness[--i];
    GAFitness[i] = 0;
    FitnessIndex = TIRandom (1, GAFitnessSum); // Select 2nd candidate, relative to fitness of left over population.
    for (TempFitnessSum = j = 0; TempFitnessSum < FitnessIndex; TempFitnessSum += GAFitness[j++]); // Find 2nd candidate.
    GAFitnessSum -= GAFitness[--j];
    GAFitness[j] = 0;

    GACrossoverPairs[i] = j;
    GACrossoverPairs[j] = i;
}

// Output minimum, mEAn and MAXIMUM fitness of population.
if (CurrentGeneration++ <= generations) {
    cerr << setw (5) << (CurrentGeneration - 1) << " " << setw (9) << setprecision (4) << setiosflags (ios::showpoint)
        << (100 * (double) GAMinFitness / NumTickets) << "% " << setw (10) << setprecision (4) <<
        setiosflags (ios::showpoint) << (100 * GAMeAnFitness / NumTickets) << "% " << setw (9) <<
        setprecision (4) << setiosflags (ios::showpoint) << (100 * (double) GAMAXFitness / NumTickets) << "% " << endl;
    if (TRandom () > 0.5)
        goto mutationoperator;
    else // !(TRandom() ≤ 0.5)
        goto crossoveroperator;
}
else { // (CurrentGeneration++ >= generations)
}
return 0; // Exit program.
}

```

## A.7 Intelligent genetic algorithm (Algorithm 7)

```

#include<iostream.h> // Header file for receiving input and generating output.
#include<stdlib.h> // Header file containing C++ standard libraries.
#include<iomanip.h> // Header file for manipulating output.
#include<fstream.h> // Header file for using files.
#include<time.h> // Header file for calculating iteration times.
#include<math.h> // Header file for mathematical functions.
#include<stl.h> // Header file for Standard Template Library (STLs) <set, multiset>.
// Fitness of playing set is given by:  $fitness = \frac{\# \text{ vertices dominated by candidate}}{\# \text{ vertices in graph}} = \Psi_{\ell}(m, n; k)$ 

```



```

long double fact (const int &);
long int round (const long double);
bool ValidTicket (const short int *, const short int &, const short int &);

// Compute the factorial of n (i.e. n!).
long double fact (const int &n) {
    if (n < 2) return 1; // Stop recursion.
    else return n * fact(n - 1); // (n > 1).
}
// Rounds a number off to the nearest integer to avoid possible numerical truncation errors.
long int round (const long double n) {
    long int flrnum = (long int)floor(n);
    if (n - flrnum < 0.5) return flrnum;
    else return (long int)ceil(n); // (n - flrnum >= 0.5)
}
// Determine whether a ticket (n-set) is valid (ticket contains no double numbers).
bool ValidTicket(short int *ticket, const short int &m, const short int &n) {
    multiset<short int, less<short int> > index;
    multiset<short int, less<short int> >::iterator i;
    for (short int counter = 0; counter < n; counter++) {
        index.insert(ticket[counter]); // Add number to index.
        if (index.count(ticket[counter]) > 1) return false; // Ticket is invalid.
    }
    i = index.begin(); // Arrange ticket elements lexicographically.
    for (short int counter = 0; i != index.end(); ticket[counter++] = *i++);
    return true; // Ticket is valid.
}
int main () {
    short int m, n, k, L, p = 1, cMutate = 1, gMutate = 1; // Parameters for the lottery <m,n;k>, playing set size (L),
    // population size (p), candidate mutate size (cMutate < p), mutate set size (gMutate < L).
    long int t, gen, iter=1; // Time limit (t), generation limit (gen), number of iterations on the same parameters (iter).
    cout << endl << "\t GENETIC ALGORITHM (GA)" << endl;
    cout << "Please specify the following parameters for the lottery (m,n;k):" << endl;
    cout << "m = "; cin >> m;
    cout << "n = "; cin >> n;
    cout << "k = "; cin >> k;
    // If the user specified invalid lottery parameters.
    if ((n > m-1) || (k > n-1) || (m < 3) || (n < 2) || (k < 1)) {
        cout << endl << "Invalid lottery parameters entered." << endl;
        return 0; // Exit program.
    }
    const long int NumTickets = round(fact(m)/(fact(n)*(fact(m-n))));
    cout << endl << "The order of the lottery graph G(" << m << ", " << n << ", " << k << ") is " << NumTickets << "." << endl;
    cout << endl << "Please specify the following GA parameters:" << endl;
    cout << "Playing set size = "; cin >> L;
    do {
        if (gMutate >= L)
            cout << "Mutation set size must be strictly less than the playing set size (" << L << ")." << endl;
        cout << "gMutation set size = "; cin >> gMutate;
    } while (gMutate >= L);
    while (floor(p/2) < ((double)p/2)) {
        cout << "Population size = "; cin >> p;
        if (floor(p/2) < ((double)p/2)) cout << "Population size needs to be a multiple of 2." << endl;
    }
    do {
        if (cMutate >= p)
            cout << "# of candidates to mutate per generation must be strictly less than the population size (" << p << ")." << endl;
        cout << "cMutation set size = "; cin >> cMutate;
    } while (cMutate >= p);
    cout << "Generation limit = "; cin >> gen; cout << "Time limit (in sec) = "; cin >> t;
    // If the user specified invalid GA parameters.
    if ((L < 2) || (L > (NumTickets-1)) || (p < 2) || (gen < 2) || (t < 2) || (cMutate < 1) ||
        (gMutate < 1) || (iter < 1)) {
        cout << endl << "Invalid GA parameters." << endl; return 0; // Exit program.
    }
    short int Population[p][L][n], TempInt, CurrentNumber = n, CurTicket[n], dcounter, tcounter, Intersect = 0,
    candidate1, candidate2; // A GA population (of size p) consisting of playing sets (of size L).
    float GAPopFit[p], selection, ccandidate, fitnessmin, fitnessavg, fitnessmax; // Vector for storing fitness of candidate
    // playing sets (GAPopFit), values used for determining crossover partners (fitnesssum, selection, ccandidate), generation
    // minimum fitness (fitnessmin), generation average fitness (fitnessavg), generation maximum fitness (fitnessmax).
    time_t StartTime; // Trace execution time.
    long int NumDominated, fitnesssum, UniquelyDominated[p+1][L], curgen = 0, NewFitness[L];
    set<short int> CrossoverIndex, CrossoverGenes; // (Integrity) Check for crossover procedure (CrossoverIndex), check genes
    // used during crossover procedure (CrossoverGenes).
    struct Fitness {
        long int fitness; // Fitness of generation candidate (given by the number of dominated vertices.
        bool used, changed; // Candidate has been used and/or changed for crossover during a previous crossover procedure.
        short int number; // Generation candidate identification number.
        Fitness *next; // Pointer to next fitness information of nextcandidate in linked-list.
    };
    Fitness *FitnessList = new Fitness, *TempFitness, *TempFitness2;
    fstream GAFitnessFile("GAFitness.txt", ios::out); // Open "GAFitness.txt" for output of GA fitness information.
    if (!GAFitnessFile) { // Check whether the file "GAFitness.txt" could be opened.

```

```

cerr << "The file \"GAFitness.txt\" could not be opened." << endl;
exit(1); // Exit program.
}
fstream GAPPopulationFile("GAPPopulation.txt", ios::out); // Open "GAPPopulation.txt" for output of GA population info.
if (!GAPPopulationFile) { // Check whether the file "GAPPopulation.txt" could be opened.
    cerr << "The file \"GAPPopulation.txt\" could not be opened." << endl;
    exit(1); // Exit program.
}
GAPPopulationFile << "GA for (" << setw(2) << m << ',' << setw(2) << n << ',' << setw(2) << k << "):" << endl;
GAFitnessFile << "GA for (" << setw(2) << m << ',' << setw(2) << n << ',' << setw(2) << k << "):" << endl << endl;
cerr << endl << "Initialising GA with " << p << " random playing set candidates, please wait...";
GAFitnessFile << "GA Parameters:" << endl
    << "Playing set size = " << setw(5) << L << endl
    << "gMutation set size = " << setw(5) << gMutate << endl
    << "Population size = " << setw(5) << p << endl
    << "cMutation set size = " << setw(5) << cMutate << endl
    << "Generation limit = " << setw(5) << gen << " generations" << endl
    << "Time limit = " << setw(5) << t << " seconds" << endl << endl;
GAFitnessFile << "Initialising GA population...";
random(StartTime = time(NULL)); // Initialise execution time & pseudo random number generator.
for (short int i = 0; i < p; i++) // Initialise GA with random population consisting of (p) playing set candidates.
    for (short int j = 0; j < L; j++)
        do
            for (short int l = 0; l < n; l++)
                Population[i][j][l] = ((short int)((random())/(float)RAND_MAX)*m)+1; // Generate random ticket.
            while (!ValidTicket(Population[i][j],m,n)); // Check whether generated ticket is valid.
GAFitnessFile << "OK! (" << time(NULL)-StartTime << "s)" << endl << "Initialising GA fitness...";
StartTime = time(NULL); // Re-initialise execution time.
fitnessmax = fitnesssum = 0; fitnessmin = 1; TempFitness = FitnessList;
for (short int PopNum = 0; PopNum < p; PopNum++) {
    NumDominated = 0;
    for (short int i = 0; i < n; i++) CurTicket[i] = i+1; // Initialise ticket to [1,2,...,n]
    for (long int counter = 1; counter <= NumTickets; counter++) {
        Intersect = 0; // Check whether current ticket is dominated.
        for (short int DomTicketNum = 0; (DomTicketNum < L) && (Intersect < k); DomTicketNum++) {
            Intersect = dcounter = tcounter = 0;
            while ((dcounter < n) && (tcounter < n) && (Intersect < k))
                if (Population[PopNum][DomTicketNum][dcounter] < CurTicket[tcounter])
                    dcounter++;
                else if (Population[PopNum][DomTicketNum][dcounter] > CurTicket[tcounter])
                    tcounter++;
                else { // (Population[PopNum][DomTicketNum][dcounter] == CurTicket[tcounter])
                    Intersect++; dcounter++; tcounter++; // Tickets intersect in 1 element.
                }
            }
            if (Intersect == k) NumDominated++; // Check whether current ticket is dominated by candidate playing set.
        }
        CurTicket[CurrentNumber-1]++; // Generate next ticket in lexicographic sequence.
        if (CurTicket[CurrentNumber-1] > m) {
            CurTicket[CurrentNumber-1]--;
            while ((CurrentNumber > 0) && (CurTicket[CurrentNumber-1] == m-(n-CurrentNumber))) CurrentNumber--;
            CurTicket[CurrentNumber-1]++;
            for (short int i = CurrentNumber; i < n; i++) CurTicket[i] = CurTicket[i-1] + 1;
            CurrentNumber = n;
        }
    }
    GAPPopFit[PopNum] = (float)NumDominated/(float)NumTickets; // Store playing set candidate fitness.
    TempFitness->next = new Fitness; TempFitness = TempFitness->next;
    TempFitness->fitness = NumDominated; // Store playing set candidate fitness.
    TempFitness->number = PopNum; TempFitness->next = NULL; TempFitness->used = false;
    if (GAPPopFit[PopNum] > fitnessmax) fitnessmax = GAPPopFit[PopNum]; // Store maximum fitness of generation.
    if (GAPPopFit[PopNum] < fitnessmin) fitnessmin = GAPPopFit[PopNum]; // Store minimum fitness of generation.
    fitnesssum += NumDominated;
}
fitnessavg = fitnesssum/p; // Store average fitness of generation.
cerr << "finished!" << endl;
GAFitnessFile << "OK! (" << time(NULL)-StartTime << "seconds)" << endl << endl;
cerr << "Genetic algorithm initialised, running...(generation limit " << gen << " generations, time limit " << t << "s)" << endl;
StartTime = time(NULL); // Re-initialise execution time.
GAFitnessFile << "0"; // Output initial GA population fitness.
for (TempFitness = FitnessList; TempFitness->next != NULL; TempFitness = TempFitness->next)
    GAFitnessFile << setw(9) << setprecision(4) << setiosflags(ios::fixed|ios::showpoint) << TempFitness->next->fitness;
GAFitnessFile << endl;
short int inputgene1, outputgene1, inputgene2, outputgene2, curcandidate, TempArray[L];
long int newcandidatefitness, newcandidate2fitness;
while ((time(NULL)-StartTime < t) && (curgen++ < gen)) { // Start GA.
    GAPPopulationFile << "Generation " << setw(4) << curgen << ":" << endl; // Write GA Population to file.
    for (short int i = 0; i < L; i++) {
        for (short int j = 0; j < p; j++) {
            GAPPopulationFile << '{' << Population[j][i][0];
            for (short int l = 1; l < n; l++) GAPPopulationFile << ',' << Population[j][i][l];
            GAPPopulationFile << "}" << endl;
        }
        GAPPopulationFile << endl;
    }
}
}

```

```

GAPopulationFile << endl;
for (TempFitness = FitnessList, fitnesssum = 0; TempFitness->next != NULL; TempFitness = TempFitness->next) {
    TempFitness->next->used = TempFitness->next->changed = false; // Candidate has not been used nor changed during
    crossover operation/procedure.
    fitnesssum += TempFitness->next->fitness; // Determine fitness range.
}
// CROSSOVER procedure (mating) relative to fitness of chromosomes (candidate playing sets).
for (short int cocounter = 0; cocounter < p/2; cocounter++) { // Find crossover partners for 1/2 of population.
    // Determine crossover partners for crossover procedure.
    ccandidate = ((random())/(float)RAND_MAX)*(float)fitnesssum/(float)NumTickets; // First mating candidate.
    selection = 0; TempFitness = FitnessList;
    while (selection < ccandidate) { // Search for chosen first candidate.
        while (TempFitness->next->used) TempFitness = TempFitness->next;
        selection += (float)TempFitness->next->fitness/(float)NumTickets; TempFitness = TempFitness->next;
    }
    candidate1 = TempFitness->number; // Store first crossover candidate.
    TempFitness->used = true; // Candidate has been used for crossover during this crossover procedure.
    fitnesssum -= TempFitness->fitness; // Remove chosen first candidate from fitness list by rescaling fitness range
    for next candidate.
    ccandidate = ((random())/(float)RAND_MAX)*(float)fitnesssum/(float)NumTickets; // Second mating candidate.
    selection = 0; TempFitness2 = FitnessList;
    while (selection < ccandidate) { // Search for chosen second candidate.
        while (TempFitness2->next->used)
            TempFitness2 = TempFitness2->next;
        selection += (float)TempFitness2->next->fitness/(float)NumTickets; TempFitness2 = TempFitness2->next;
    }
    candidate2 = TempFitness2->number; // Store second crossover candidate.
    TempFitness2->used = true; // Candidate has been used for crossover during this crossover procedure.
    fitnesssum -= TempFitness2->fitness; // Remove chosen second candidate from fitness list by rescaling fitness range
    for next candidate.
    // Check which single gene exchange from different crossover candidates would yield fitness increase.
    curcandidate = candidate1;
docrossover:
    if (curcandidate == candidate2) { // Store evolving maximum fitness.
        fitnessmax = (float)(TempFitness2->fitness)/(float)NumTickets;
        candidate2 = candidate1; candidate1 = curcandidate; inputgene2 = -1; // Exchange between crossover candidates.
    }
    else { // (curcandidate != candidate2) || (curcandidate == candidate1)
        fitnessmax = (float)(TempFitness->fitness)/(float)NumTickets; inputgene1 = -1;
    }
    for (short int i = 0; i < L; i++) { // First check if input gene is equivalent to any gene in crossover candidate.
        Intersect = 0;
        for (short int j = 0; j < L; j++) { // (Intersect < k); j++) {
            dcounter = tcounter = 0; Intersect = k;
            while ((dcounter < n) && (tcounter < n) && (Intersect == k))
                if (Population[candidate1][j][dcounter] == Population[candidate2][i][tcounter]) {
                    dcounter++; tcounter++;
                }
            else Intersect = 0; // (Population[candidate1][j][dcounter] != Population[candidate2][i][tcounter])
        }
    }
    if (Intersect < k) { // Input gene is distinct from all genes in first candidate.
        for (short int j = 0; j < L; j++) {
            for (short int l = 0; l < n; l++) {
                TempArray[l] = Population[candidate1][j][l]; Population[candidate1][j][l] = Population[candidate2][i][l];
            }
        }
        NumDominated = 0; // Recalculate chromosome fitness with new gene.
        for (short int l = 0; l < n; l++) CurTicket[l] = l+1; // Initialise ticket to [1,2,...,n]
        for (long int counter = 1; counter <= NumTickets; counter++) {
            Intersect = 0; // Check whether current ticket is dominated.
            for (short int DomTicketNum = 0; (DomTicketNum < L) && (Intersect < k); DomTicketNum++) {
                Intersect = dcounter = tcounter = 0;
                while ((dcounter < n) && (tcounter < n) && (Intersect < k))
                    if (Population[candidate1][DomTicketNum][dcounter] < CurTicket[tcounter]) dcounter++;
                    else if (Population[candidate1][DomTicketNum][dcounter] > CurTicket[tcounter]) tcounter++;
                    else { // (Population[candidate1][DomTicketNum][dcounter] == CurTicket[tcounter])
                        Intersect++; dcounter++; tcounter++; // Tickets intersect in 1 element.
                    }
            }
            if (Intersect == k) // Check whether current ticket is dominated by candidate playing set.
                NumDominated++;
        }
        CurTicket[CurrentNumber-1]++; // Generate next ticket in lexicographic sequence.
        if (CurTicket[CurrentNumber-1] > m) {
            CurTicket[CurrentNumber-1]--;
            while ((CurrentNumber > 0) && (CurTicket[CurrentNumber-1] == m-(n-CurrentNumber))) CurrentNumber--;
            CurTicket[CurrentNumber-1]++;
            for (short int l = CurrentNumber; l < n; l++) CurTicket[l] = CurTicket[l-1] + 1;
            CurrentNumber = n;
        }
    }
    if (((float)NumDominated/(float)NumTickets) > fitnessmax) { // Store specific gene exchange information.
        fitnessmax = (float)NumDominated/(float)NumTickets;
        if (curcandidate == TempFitness->number) {
            inputgene1 = j; outputgene1 = i; newcandidatefitness = NumDominated;
        }
    }
}

```

```

        else { // (curcandidate != TempFitness->number) || (curcandidate == TempFitness2->number)
            inputgene2 = j; outputgene2 = i; newcandidate2fitness = NumDominated;
        }
    }
    // Copy gene information back to candidate.
    for (short int l = 0; l < n; l++) Population[candidate1][j][l] = TempArray[l];
}
}
}
if (curcandidate != TempFitness2->number) { // Repeat crossover procedure for other crossover candidate.
    curcandidate = candidate2; goto docrossover;
}
// If an exchange in genes is considered reasonable (causes fitness increase).
if (inputgene1 > -1) { // Change first crossover candidate.
    TempFitness->fitness = newcandidate1fitness; // Change candidate 1 fitness.
    TempFitness->changed = true; // Candidate was changed during crossover operation/procedure.
    for (short int j = 0; j < n; j++) {
        TempArray[j] = Population[candidate2][inputgene1][j];
        Population[TempFitness->number][inputgene1][j] = Population[TempFitness2->number][outputgene1][j];
    }
}
if (inputgene2 > -1) { // Change second crossover candidate.
    TempFitness2->fitness = newcandidate2fitness; // Change candidate 2 fitness.
    TempFitness2->changed = true; // Candidate was changed during crossover operation/procedure.
    if (inputgene1 == outputgene2)
        for (short int j = 0; j < n; j++) Population[TempFitness->number][inputgene2][j] = TempArray[j];
    else // (inputgene1 != outputgene2)
        for (short int j = 0; j < n; j++)
            Population[TempFitness2->number][inputgene2][j] = Population[TempFitness->number][outputgene2][j];
}
}
for (short int i = 0; i < cMutate; i++) { // MUTATE (gMutate) elements of cMutate candidates of the population (p).
    candidate1 = (short int)((random()/(float)RAND_MAX)*p); TempFitness = FitnessList;
    while (TempFitness->next->number < candidate1) // Search for selected mutation candidate.
        TempFitness = TempFitness->next;
    for (short int j = 0; j < gMutate; j++) {
        candidate2 = (short int)((random()/(float)RAND_MAX)*L);
        do // Move current domination ticket to nearest neighbour (maybe further).
            Population[candidate1][candidate2][((short int)((random()/(float)RAND_MAX)*n)) =
                ((short int)((random()/(float)RAND_MAX)*m)+1);
        while (!ValidTicket(Population[candidate1][candidate2], m, n)); // Check whether generated ticket is valid.
    }
    NumDominated = 0; // Recalculate mutated chromosome fitness.
    for (short int j = 0; j < n; j++) CurTicket[j] = j+1; // Initialise ticket to [1,2,...,n].
    for (long int counter = 0; counter < NumTickets; counter++) {
        Intersect = 0;
        // Check whether current ticket is dominated.
        for (short int DomTicketNum = 0; (DomTicketNum < L) && (Intersect < k); DomTicketNum++) {
            Intersect = dcounter = tcounter = 0;
            while ((dcounter < n) && (tcounter < n) && (Intersect < k))
                if (Population[candidate1][DomTicketNum][dcounter] < CurTicket[tcounter]) dcounter++;
                else if (Population[candidate1][DomTicketNum][dcounter] > CurTicket[tcounter]) tcounter++;
                else { // (Population[candidate1][DomTicketNum][dcounter] == CurTicket[tcounter])
                    Intersect++; dcounter++; tcounter++; // Tickets intersect in 1 element.
                }
            if (Intersect == k) NumDominated++; // Check whether current ticket is dominated by candidate playing set.
        }
        CurTicket[CurrentNumber-1]++; // Generate next ticket in lexicographic sequence.
        if (CurTicket[CurrentNumber-1] > m) {
            CurTicket[CurrentNumber-1]--;
            while ((CurrentNumber > 0) && (CurTicket[CurrentNumber-1] == m-(n-CurrentNumber))) CurrentNumber--;
            CurTicket[CurrentNumber-1]++;
            for (short int j = CurrentNumber; j < n; j++) CurTicket[j] = CurTicket[j-1] + 1;
            CurrentNumber = n;
        }
    }
    TempFitness->next->fitness = NumDominated; // New fitness of mutated candidate.
}
}
GAFitnessFile << setw(3) << curgen; // Write population fitness to file.
for (TempFitness = FitnessList; TempFitness->next != NULL; TempFitness = TempFitness->next)
    GAFitnessFile << setw(9) << TempFitness->next->fitness;
GAFitnessFile << endl; cerr << '\n';
if ((long int)(curgen/20) == (float)curgen/20) cerr << "(" << setw(4) << curgen << "/" << gen << ")" << endl;
}
GAFitnessFile << "GA simulation time: " << time(NULL)-StartTime << "s." << endl;
GAFitnessFile.close(); // Close the Genetic Algorithm (GA) Fitness file.
GAPopulationFile.close(); // Close the Genetic Algorithm (GA) Population file.
cout << endl;
return 0; // Exit program.
}

```

## A.8 $L_\psi(m, n; k)$ -set characterisation algorithm (Algorithm 8)

```

#include<iostream> // Header file for receiving input and generating output.
#include<iomanip> // Header file for manipulating output.
#include<fstream> // Header file for using files.
#include<stdio.h> // Header file for standard input/output.
#include<signal.h> // Header file for catching signals.
#include<time.h> // Header file for calculating iteration times.
#include<math.h> // Header file for mathematical functions.
#include<stdlib.h> // Header file used for catching signals.
#include<unistd.h> // Header file used for deleting files (int unlink(const char *pathname)).
#include<sstream> // Header file for manipulating strings.

using namespace std;

long double combination (const int, const int);
void savedata (const short int);

struct PsiElement {
    char *Construction; // Pointer to ticket construction.
    long double Value; // Number of elements dominated by ticket construction.
    PsiElement *next, *down;
};

volatile sig_atomic_t keepgoing = 1;
PsiElement *PList = new (PsiElement), *TempPList, *TempPList2;
const short int maxbranches = 20;
ostringstream filename[maxbranches];
char Ticket[1024]; // Pointer to ticket construction.
short int m, L, SubTree = 1, CurrentLevel;
long int NumTickets[10], NumTicketsRemoved[10], SubTreeTimes[maxbranches], SubTreeConstructions[maxbranches];
time_t StartBranch;

long double combination (const int m, const int n) {
    if ((n == 1) || (n == m-1))
        return (long) m; //  $\binom{m}{1} = \binom{m}{m-1} = m$ .
    if ((m == n) || (n == 0))
        return 1; //  $\binom{m}{m} = 1$ .
    if (n > m)
        return 0; // If  $n > m$   $\binom{m}{n} = 0$ .

    long double fraction = m;

    // Generate the sequence  $\frac{(m-i+1)!}{i!}$  where  $i = 2, \dots, n$ .
    for (short int i = 2; i <= n; fraction = fraction * (m - i + 1) / i++);

    return (long double) fraction;
}
void savedata (const short int cont) {
    SubTreeTimes[SubTree] += time (NULL) - StartBranch; // Add processing time to current subtree time.

    // Open file "LottoTreeSave.txt" to store last ticket (in case of a restart).
    ofstream SaveOutput ("LottoTreeSave.txt", ios::out);

    // Check whether the file "LottoTreeSave.txt" could be opened.
    if (!SaveOutput) {
        cerr << "The file \"LottoTreeSave.txt\" could not be opened." << endl;
        exit (0);
    }
    else { // (SaveOutput)
        for (short int i = 0; i < pow (2, L + 1); i++)
            SaveOutput << setw (3) << (short int) Ticket[i]; // Store the current ticket construction being evaluated.
            SaveOutput << endl;
        for (short int i = 1; i < CurrentLevel; i++)
            SaveOutput << NumTickets[i] << " " << NumTicketsRemoved[i] << endl; // Store the number of tickets/constructions
            inserted [removed] from the lottery tree.
        for (short int i = 0; i < SubTree; i++)
            SaveOutput << SubTreeConstructions[i + 1] << " " << SubTreeTimes[i + 1] << endl; // Store the calculation times
            involved in traversing the completed subtrees/branches of the lottery tree.
        TempPList = PList->next->next;
        while (TempPList != NULL) { // Write saved/valid construction of Branch Subtree to save file.
            for (TempPList2 = TempPList, TempPList = TempPList->next; TempPList2 != NULL; TempPList2 = TempPList2->down) {
                SaveOutput << setw (10) << resetiosflags (ios::showpoint) << setiosflags (ios::fixed) << setprecision (0) <<
                    TempPList2->Value << " ";
                for (short int i = 0; i < pow (2, L); i++)
                    SaveOutput << setw (3) << (short int) TempPList2->Construction[i];
                SaveOutput << endl;
            }
        }
        SaveOutput << endl; SaveOutput.close ();
    }
}
if (!cont) exit (0);
}

```

```

/*
 * Algorithm: Permutations of  $\{1, \dots, n\}$  [41]
 *
 * FIRST PERMUTATION is given by  $x(i) = i$  for  $i = 1, \dots, n$ .
 * NEXT PERMUTATION after  $(x(1), \dots, x(n))$ :
 *   Find the largest  $j$  for which  $x(j) < x(j + 1)$  (working back from the end).
 *   If no such  $j$  exists, then the current permutation is the last.
 *   Interchange the value of  $x(j)$  with the least  $x(k)$  greater than  $x(j)$  with  $k > j$ ;
 *   then reverse the sequence of values of  $x(j + 1), \dots, x(n)$ ;
 *   return this permutation.
 */

int main () {
    short int n, t, k, TimeLimit, Aborted;
    long int LimitSeconds, ConstructionNumber;
    long double minPsi;

    // Read Lotto Tree input from file.
    ifstream LottoTreeInput ("LottoTreeInput.txt", ios::in);
    // Check whether the file "LottoTreeInput.txt" could be opened.
    if (!LottoTreeInput) {
        cerr << "The file \"LottoTreeInput.txt\" could not be opened." << endl;
        return 0;
    }
    else // (LottoTreeInput)
        LottoTreeInput >> m >> n >> t >> k >> L >> TimeLimit >> LimitSeconds >> minPsi >> Aborted; // Read the lottery tree
        parameters from the file "LottoTreeInput.txt".

    // If the user specified invalid Lottery parameters.
    if ((n > m - 1) || (t > m - 1) || (k > n - 1) || (k > t - 1) || (m < 3) || (n < 2) || (t < 2) || (k < 1)) {
        cerr << endl << "Invalid Lottery parameters." << endl;
        return 0;
    }

    char Parent[(long int) pow (2, L + 1)]; // Pointer to ticket construction Parent.
    char TicketDup[(long int) pow (2, L + 1)]; // Pointer to ticket duplicate.
    char TicketBranch[(long int) pow (2, L + 1)]; // Pointer to Ticket with whole branch (predecessor) structure.

    PList->next = new (PsiElement); // Create dummy element in front of list in order to manage list effectively.
    PList->down = PList->next->down = PList->next->next = NULL;
    PList->Value = PList->next->Value = combination(m,t) + 1;

    ifstream Constructions ("Constructions.txt", ios::out);
    // Check whether the file "Constructions.txt" could be opened.
    if (!Constructions) {
        cerr << "The file \"Constructions.txt\" could not be opened." << endl;
        return 0;
    }
    else // (Constructions)
        Constructions << "(" << m << ", " << n << ", " << t << ", " << k << ") : " << L << " tickets" << endl << endl;

    time_t StartTime, TempTime;
    short int Counter, TicketCover, LastTicketSum, i, TempVar;
    bool LastTicketFound = false, SimilarFound = false, Dominates;
    short int *Permutation = new short int [L + 1], *NewPermutation = new short int [L + 1], *TempPermutation;
    long int DominatePercentage = 0, r = 0, TicketSize; // (Best) Number of elements that Domination Element
    Characterisation dominates [does not dominate], Characterisation number that dominates the most for  $\langle m, n; k \rangle$ .
    long double Dominate = 0, TotalDominate = 0;

    for (Counter = 2; Counter < pow (2, L+1); Ticket[Counter] = 0, Counter++); // Clear Ticket construction.
    for (Counter = 1; Counter <= L; NumTickets[Counter] = NumTicketsRemoved[Counter] = 0, Counter++); // Clear Ticket
    Level Info.
    for (Counter = 1; Counter <= n; SubTreeConstructions[Counter++] = 0) // Clear number of constructions/subtree memory.
        filename[Counter - 1] << "Branch" << Counter;

    // Initialise domination construction tree.
    if (!Aborted) {
        Parent[0] = m - n;
        Parent[1] = n;
        CurrentLevel = 2;
        NumTickets[1] = 1; // Lottery tree has 1 root.
        TicketCover = n;
        LastTicketSum = 0;
    }
    else { // (Aborted)
        ifstream SavedOutput ("LottoTreeSave.txt", ios::in); // Open "LottoTreeSave.txt" containing all the saved information.
        for (short int l = 0, j; l < pow (2, L + 1); l++) {
            SavedOutput >> j;
            Parent[l] = Ticket[l] = j; // Read last saved child information/construction.
        }
        // Determine the current subtree in LottoTree.
        for (short int l = L + 1; l > 2; l--)
            for (short int j = 0; j < pow (2, l - 1); Parent[j++] += Parent[j + (short int) pow (2, l - 1)]);
        SubTree = (short int) Parent[1];
    }
}

```

```

for (short int l = 0; l < pow (2, L); l++) // Generate Parent of last child.
    Parent[l] = Ticket[l] + Ticket[l + (short int) pow (2, L)];
LastTicketSum = Ticket[(short int) pow (2, L)]; Counter = 0;
while (Ticket[Counter + (short int) pow (2, L)] == Parent[Counter++])
    LastTicketSum += Ticket[Counter + (short int) pow (2, L)];
CurrentLevel = L + 1;
for (long int l = 0; l < L; l++) // Read the number of tickets [removed] per level.
    SavedOutput >> NumTickets[l + 1] >> NumTicketsRemoved[l + 1];
for (short int s = 0; s < SubTree; s++)
    SavedOutput >> SubTreeConstructions[s + 1] >> SubTreeTimes[s + 1]; // Read times from "LottoTreeSave.txt" file.
for (short int s = 1; s < SubTree; s++)
    Constructions << "Subtree " << s << " traversed in " << SubTreeTimes[s] << " seconds..." << endl;

// Read valid constructions found in last traversed subtree (necessarily sorted).
TempPList2 = PList;
SavedOutput >> Dominate; i = 0;
while (i++ < SubTreeConstructions[SubTree]) {
    TempPList = new (PsiElement);
    TempPList->Construction = new char [(long int) pow (2, L)];
    TempPList->Value = Dominate;
    for (short int j = 0; j < pow (2, L); j++) {
        SavedOutput >> Dominate;
        TempPList->Construction[j] = (char) Dominate;
    }

    if (TempPList2->next->Value == TempPList->Value) {
        TempPList->down = TempPList2->next;
        TempPList->next = NULL;
        TempPList2->next = TempPList;
    }
    else { // (TempPList2->next->Value > TempPList->Value)
        TempPList2->next->next = TempPList;
        TempPList->down = TempPList->next = NULL;
        TempPList2 = TempPList2->next;
    }
    SavedOutput >> Dominate; // Read in the multiplicities of the specific construction.
}
SavedOutput.close ();
TicketCover = 0;
}

{
    StartTime = StartBranch = TempTime = time (NULL); // Capture start time of subtree traversal.
    TicketSize = (short int) pow (2, CurrentLevel);

GenerateNewTicket:
    if (TimeLimit && ((time (NULL) - StartTime) > LimitSeconds))
        savedata (0);
    else if (time (NULL) - TempTime > 900) { // Save data every 15 minutes.
        savedata (1);
        StartBranch = TempTime = time (NULL);
    }
    if (!keepgoing) savedata (0);
    Counter = TicketSize - 1;
    Ticket[Counter]++;
    TicketCover--;
    while (((Ticket[Counter] > Parent[Counter - (TicketSize / 2)]) && (Counter >= TicketSize / 2))
        || (TicketCover < 0)) {
        TicketCover += Ticket[Counter] - 1;
        Ticket[Counter--] = 0;
        Ticket[Counter]++;
    }
    if (TicketCover == 0) { // If valid ticket is generated (i.e., n numbers covered).
        for (Counter = 0; Counter < TicketSize / 2; Ticket[Counter] = Parent[Counter] - Ticket[Counter + TicketSize / 2],
            Counter++); // Update ticket construction.
        Counter = 0;
        LastTicketSum = Ticket[TicketSize / 2];
        while (Ticket[Counter + TicketSize / 2] == Parent[Counter++])
            LastTicketSum += Ticket[Counter+TicketSize / 2];

        // Check for possible pathological equivalent ticket constructions.
        for (i = 0, SimilarFound = false; i < CurrentLevel - 1 && (!SimilarFound) && (CurrentLevel < L + 1); i++) {
            for (Counter = (TicketSize / 2) + 1, TempVar = 0; Counter < TicketSize && (!SimilarFound); Counter++)
                if ((Counter & (short int) pow (2, i)) > 0)
                    TempVar += Ticket[Counter];
            SimilarFound = (TempVar == n); // Invalid ticket construction (i.e., duplicate tickets).
        }
        if (SimilarFound) { // Match was found or not.
            NumTicketsRemoved[CurrentLevel]++;
            if (LastTicketSum == (CurrentLevel == L + 1 ? t : n))
                goto SimilarFoundLastTicket;
            else // (LastTicketSum != n or t)
                goto GenerateNewTicket;
        }
}

```

```

else { // (!SimilarFound)
  if ((CurrentLevel > 2) && (CurrentLevel < L + 1)) {
    // Generate Ticket Branch.
    for (i = TicketSize - 1; i > 0; TicketBranch[i] = Ticket[i], i--); // Make copy of latest ticket addition.
    for (Counter = TicketSize; Counter > 4; Counter /= 2)
      for (i = Counter - 1; i > Counter / 2; TicketBranch[i - Counter / 2] = TicketBranch[i] +
          TicketBranch[i - Counter / 2], i--); // Generate Branch (parents).
    // Check for possible duplicate ticket construction generated (permutation of current ticket construction).
    // Keep track of current Permutation/Positioning.
    for (i = 0; i < CurrentLevel - 2; Permutation[i] = i++); // Initialise Permutation.
    Permutation[CurrentLevel - 2] = CurrentLevel - 1;
    Permutation[CurrentLevel - 1] = CurrentLevel - 2; // Skip 1st (identity) Permutation.
    Permutation[CurrentLevel] = NewPermutation[CurrentLevel] = CurrentLevel; // Avoid errors.

    while (!SimilarFound) {
      // Check for match according to some Permutation.
      for (Counter = 1; Counter < TicketSize - 1; Counter++) {
        for (i = r = 0; i < CurrentLevel; i++) // Construct Permutation indexing.
          r += ((short int) pow(2, Permutation[i]) & Counter) >> Permutation[i] << i;
        TicketDup[r] = Ticket[Counter]; // Generate Ticket Permutation.
      }

      TicketDup[TicketSize - 1] = Ticket[TicketSize - 1];
      // Ticket permutation generated; sort entries to check for duplicates.
      for (Counter = TicketSize; Counter > 4; Counter /= 2)
        for (i = Counter - 1; i > Counter / 2; TicketDup[i - Counter / 2] += TicketDup[i], i--); // Generate
          Branch (parents) of Duplicate Ticket.

      // Check if ticket is already generated in tree. Traverse 2 tickets for equivalence.
      for (i = 1; (i < TicketSize - 1) && (TicketDup[i] == TicketBranch[i]); i++);
      if ((i < TicketSize - 1) && (SimilarFound = (TicketDup[i] < TicketBranch[i]))) { // Match was found.
        NumTicketsRemoved[CurrentLevel]++; // Ticket Construction has been removed from current level.
        if (LastTicketSum == (CurrentLevel == L + 1 ? t : n))
          goto SimilarFoundLastTicket;
        else // (LastTicketSum != n or t)
          goto GenerateNewTicket;
      }
    }
    else { // (!SimilarFound)
      for (i = CurrentLevel - 2; (i >= 0) && (Permutation[i] > Permutation[i + 1]); i--);
      if (i >= 0) { // Not all Permutations have been sought through.
        TempVar = CurrentLevel;
        for (Counter = CurrentLevel - 1; Counter > i; Counter--)
          if ((Permutation[Counter] > Permutation[i]) && (Permutation[Counter] < Permutation[TempVar]))
            TempVar = Counter;
        Counter = Permutation[i]; // Interchange entries x(j) & x(k) according to Permutation algorithm.
        Permutation[i] = Permutation[TempVar];
        Permutation[TempVar] = Counter;
        for (TempVar = 0; TempVar <= i; NewPermutation[TempVar] = Permutation[TempVar++]);
        for (TempVar = CurrentLevel - 1; TempVar > i; NewPermutation[i + (CurrentLevel - TempVar)] =
            Permutation[TempVar], TempVar--);
        TempPermutation = NewPermutation;
        NewPermutation = Permutation;
        Permutation = TempPermutation;
      }
      else // (i < 0) All permutations have been sought through.
        goto NextLevel;
    }
  }
}
NextLevel:
NumTickets[CurrentLevel]++; // Ticket Construction has been added to current level.
for (i = 0; i < TicketSize; Parent[i] = Ticket[i], Ticket[i + TicketSize] = 0, i++);
TicketSize = (short int) pow(2, ++CurrentLevel); // Child becomes new Parent.
TicketCover = (CurrentLevel == L + 1 ? t : n);
LastTicketSum = 0;
goto GenerateNewTicket;
}
else if (CurrentLevel == L + 1) {
  // Check if current ticket is dominated by parent (on level L).
  for (i = 0, Dominates = false; (i < CurrentLevel - 1) && (!Dominates); i++) {
    for (r = (TicketSize / 2) + 1, TempVar = 0; (r < TicketSize) && (TempVar < k); r++)
      if (r & (short int) pow(2, i))
        TempVar += Ticket[r]; // Count overlap with all other tickets.
    Dominates = (TempVar >= k); // New ticket is dominated by some other ticket.
  }
  if (Dominates) {
    Dominate = 1;
    for (i = 0; i < TicketSize / 2; i++)
      if ((Parent[i] > 1) && (Ticket[i] > 0))
        Dominate *= combination(Parent[i], Ticket[i]);
    TotalDominate += Dominate;
  }
  if (LastTicketSum == (CurrentLevel == L + 1 ? t : n))
    goto SimilarFoundLastTicket;
  else // (LastTicketSum != n or t)

```



```

        goto GenerateNewTicket;
    }
    else // (CurrentLevel == 2)
        goto NextLevel;
    }
}
else // Invalid ticket was generated (i.e., < n numbers covered).
    goto GenerateNewTicket;
}
}
SimilarFoundLastTicket:
if (CurrentLevel == L + 1) {
    if (TotalDominate > minPsi) {
        SubTreeConstructions[SubTree]++;
        TempPList = new (PsiElement);
        TempPList->Construction = new char [(long int) pow (2, L)]; // Pointer to Ticket construction.
        for (i = 0; i < pow (2, L); TempPList->Construction[i++] = Parent[i]); // Copy Parent construction in PsiList.
        TempPList->Value = TotalDominate; // Number of elements dominated.
        TempPList2 = PList; // lem Start finding correct insertion position from start of list.
        while ((TempPList2->next->Value > TotalDominate) && (TempPList2->next->next != NULL))
            TempPList2 = TempPList2->next; // Search for appropriate position of insertion.
        if (TempPList2->next->Value == TotalDominate) { // Element with equivalent psi found.
            TempPList->next = TempPList2->next->next;
            TempPList->down = TempPList2->next;
            TempPList2->next = TempPList;
        }
        else if (TempPList2->next->Value < TotalDominate) { // Element with larger psi found.
            TempPList->next = TempPList2->next;
            TempPList->down = NULL;
            TempPList2->next = TempPList;
        }
        else { // (TempPList2->next->next == NULL)
            TempPList->next = TempPList->down = NULL; // Element with lowest psi found.
            TempPList2->next->next = TempPList;
        }
    }
    TotalDominate = 0;
}
// Move up one level in tree.
for (i = 0; i < TicketSize / 4; i++) {
    Ticket[i] = Parent[i];
    Parent[i] += Parent[i + TicketSize / 4];
}
for (i = TicketSize / 4; i < TicketSize / 2; Ticket[i] = Parent[i], i++);
TicketSize = (short int) pow (2, --CurrentLevel);
if (CurrentLevel == 1) {
    Constructions << endl << "\ tLottery tree statistics:" << endl << endl;
    for (i = 1; i <= L; i++) {
        Constructions << "Level" << i << ": Inserted" << setw (8) << NumTickets[i] << " Removed" << setw (8) <<
            NumTicketsRemoved[i] << endl;
    }
    Constructions << endl;
    for (SubTree = 1; SubTree < n + 1; SubTree++) {
        // Load the specific branch of the lottery tree.
        ifstream Branchfile (filename[SubTree - 1].str ().c_str (), ios::in);
        // lem Check whether the file "Branch?" could be opened.
        if (!Branchfile) {
            cerr << "The file \"Branch\" << SubTree << \" could not be opened." << endl;
            return 0;
        }
        else { // (Branchfile)
            // Read information of Branch Subtree from file.
            while (SubTreeConstructions[SubTree]-- > 0) {
                TempPList = new (PsiElement);
                TempPList->Construction = new char [(long int) pow (2, L)];
                Branchfile >> TempPList->Value;
                for (i = 0; i < pow (2, L); i++) {
                    Branchfile >> Dominate;
                    TempPList->Construction[i] = (char) Dominate;
                }
                TempPList2 = PList; // lem Start finding correct insertion position from start of list.
                while ((TempPList2->next->Value > TempPList->Value) && (TempPList2->next->next != NULL))
                    TempPList2 = TempPList2->next; // Search for appropriate position of insertion.
                if (TempPList2->next->Value == TempPList->Value) { // Element with equivalent psi found.
                    TempPList->next = TempPList2->next->next;
                    TempPList->down = TempPList2->next;
                    TempPList2->next = TempPList;
                }
                else if (TempPList2->next->Value < TempPList->Value) { // Element with larger psi found.
                    TempPList->next = TempPList2->next;
                    TempPList->down = NULL;
                    TempPList2->next = TempPList;
                }
            }
        }
        else { // lem (TempPList2->next->next == NULL)

```

```

    TempPList->next = TempPList->down = NULL; // Element with lowest psi found.
    TempPList2->next->next = TempPList;
  }
}
Branchfile.close ();
i = unlink (filename[SubTree - 1].str ().c_str ()); // Delete the "Branch?" filename.
}
}
TempPList = PList->next->next;
while (TempPList != NULL) {
  if (TempPList->down == NULL) {
    Constructions << "*" << setw (4) << " " << endl;
    for (i = 0; i < pow (2, L); i++)
      Constructions << setw (3) << (short int) TempPList->Construction[i];
    delete [] TempPList->Construction; // Clear/release memory.
    Constructions << " " << setw (10) << setiosflags (ios::fixed) << resetiosflags (ios::showpoint) <<
      setprecision (0) << TempPList->Value << "/" << combination (m, t) << " = " << setw (8) <<
      setiosflags (ios::fixed | ios::showpoint) << setprecision (4) << ((float) 100 * TempPList->Value) /
      combination (m, t) << "%" << endl;
    TempPList2 = TempPList;
    TempPList = TempPList->next;
    TempPList2->next = NULL;
    delete TempPList2; // Clear/release memory.
  }
  else { // (TempPList->down != NULL)
    ConstructionNumber = 1;
    for (PList = TempPList, TempPList = TempPList->next; PList != NULL; TempPList2 = PList, PList = PList->down,
        TempPList2->down = NULL, delete TempPList2) {
      if (ConstructionNumber == 1)
        Constructions << "*" << setw (4) << ConstructionNumber++;
      else // (ConstructionNumber > 1)
        Constructions << setw (5) << ConstructionNumber++;
      for (i = 0; i < pow (2, L); i++)
        Constructions << setw (3) << (short int) PList->Construction[i];
      delete [] PList->Construction;
      Constructions << " " << setw (10) << resetiosflags (ios::showpoint) << setprecision (0) << PList->Value
        << "/" << combination (m, t);
      if (ConstructionNumber == 2)
        Constructions << " = " << setw (8) << setiosflags (ios::fixed | ios::showpoint) << setprecision (4) <<
          ((float) 100 * PList->Value) / combination (m, t) << "%" << endl;
      else // (ConstructionNumber != 2)
        Constructions << endl;
    }
  }
}
Constructions.close ();
return 0;
}
else if (CurrentLevel == 2) {
  SubTreeTimes[SubTree] += time (NULL) - StartBranch; // Save subtree traversal time.
  Constructions << "Subtree " << SubTree << " traversed in " << SubTreeTimes[SubTree++] << " seconds..." << endl;
  {
    // Save the specific branch of the lottery tree (for backup purposes and to possibly save memory).
    ofstream Branchfile (filename[SubTree-2].str ().c_str (), ios::out);
    // Check whether the file "Brach?" could be opened.
    if (!Branchfile) {
      cerr << "The file \"Branch\" << SubTree - 1 << \" could not be opened." << endl;
      return 0;
    }
  }
  else { // (Branchfile)
    // Write information of Branch Subtree to file.
    TempPList = PList->next->next;
    while (TempPList != NULL) {
      for (PList = TempPList, TempPList = TempPList->next; PList != NULL; TempPList2 = PList, PList = PList->down,
          TempPList2->down = NULL, delete TempPList2) {
        Branchfile << setw (10) << resetiosflags (ios::showpoint) << setiosflags (ios::fixed) << setprecision (0)
          << PList->Value << " ";
        for (i = 0; i < pow (2, L); i++)
          Branchfile << setw (3) << (short int) PList->Construction[i];
        delete [] PList->Construction;
        Branchfile << endl;
      }
    }
    Branchfile.close ();
    PList = new (PsiElement);
    PList->next = new (PsiElement); // Create dummy element in front of list in order to manage list effectively.
    PList->down = PList->next->down = PList->next->next = NULL;
    PList->Value = PList->next->Value = combination (m, t) + 1;
  }
}
StartBranch = time (NULL); // Capture start time of subtree traversal.
}
// Generate new LastTicketSum.
for (Counter = 0; Counter < TicketSize / 2; Ticket[Counter] = Parent[Counter] - Ticket[Counter + TicketSize/2],

```

```
    Counter++); // pdate Ticket construction.
Counter = 0;
LastTicketSum = Ticket[TicketSize / 2];
while (Ticket[Counter + TicketSize / 2] == Parent[Counter++])
    LastTicketSum += Ticket[Counter + TicketSize / 2];
if (LastTicketSum == (CurrentLevel == L + 1 ? t : n))
    goto SimilarFoundLastTicket;
else { // (LastTickerSum != n) Generate new TicketCover.
    TicketCover = (CurrentLevel == L + 1 ? t : n);
    for (i = TicketSize/2; i < TicketSize); TicketCover -= Ticket[i], i++);
    goto GenerateNewTicket;
}
}
```

# Appendix B

## Best Internet bounds on $L_1(m, n; k)$

In this appendix, the playing set and best known lottery set of cardinality  $\ell = 163$  constructed by Algorithm 4 and obtained via the Internet [19, 44, 133] respectively for  $\langle 49, 6; 3 \rangle$  is given in Table B.1. Tables B.2 and B.3 present the best Internet bounds together with the best analytic bounds obtained in Chapter 4.

01 02 13 19 34 40	02 04 13 27 30 48	03 12 29 37 46 49	05 11 26 36 44 45	07 16 25 34 42 49	10 17 27 33 40 47	15 17 30 38 40 43
01 02 15 28 30 45	02 05 06 25 28 49	03 13 21 31 35 46	05 12 17 36 37 47	07 17 26 30 42 46	10 18 32 44 45 46	15 19 40 46 48 49
01 02 18 31 42 44	02 05 13 20 29 31	03 16 23 24 40 45	05 14 16 23 38 41	07 19 26 35 44 48	10 21 22 36 38 42	15 23 29 33 41 43
01 03 04 10 28 29	02 06 10 18 23 29	03 17 18 37 39 43	05 15 16 22 27 42	07 20 23 28 37 38	11 12 19 21 27 31	16 17 24 29 35 38
01 03 06 11 24 49	02 06 11 21 32 36	03 22 32 35 44 49	05 15 37 39 40 41	07 21 24 32 33 34	11 13 27 29 33 35	16 33 34 37 45 46
01 03 14 18 27 45	02 07 08 17 27 35	04 05 10 30 36 39	05 16 25 26 35 39	07 29 33 36 37 39	11 14 15 20 22 38	18 21 24 26 29 45
01 04 07 25 32 37	02 08 14 21 39 46	04 06 08 28 30 46	05 17 18 22 25 38	07 31 36 38 40 47	11 19 38 41 46 47	18 23 25 36 41 42
01 05 10 33 35 42	02 09 12 22 35 36	04 06 10 17 26 38	05 19 31 36 42 46	08 10 16 23 25 31	11 20 27 34 39 42	18 29 32 39 42 47
01 06 16 28 32 43	02 11 22 40 42 47	04 06 23 32 42 48	05 24 27 43 46 48	08 12 13 24 26 30	11 21 22 29 30 44	19 21 25 30 34 36
01 08 10 19 36 41	02 12 15 21 24 48	04 06 27 29 39 44	06 07 22 38 39 49	08 15 18 27 47 49	11 28 31 33 46 49	19 23 27 35 40 43
01 08 14 29 30 37	02 15 17 31 34 49	04 09 15 26 33 49	06 08 36 37 42 43	08 18 20 22 30 33	12 19 20 22 26 39	19 29 32 33 45 48
01 08 20 23 42 45	02 16 17 28 44 48	04 11 16 18 29 41	06 09 10 28 34 48	09 12 13 14 33 37	12 25 28 38 42 45	20 26 27 31 35 37
01 09 11 25 39 43	02 30 33 37 38 44	04 11 22 23 35 46	06 09 11 33 40 41	09 13 16 19 26 36	12 27 36 38 39 41	21 25 27 41 43 44
01 12 31 45 46 47	03 05 08 11 25 30	04 12 16 20 35 48	06 12 14 17 18 41	09 14 19 28 35 39	13 14 28 43 44 47	22 26 29 34 41 47
01 13 15 36 38 48	03 07 09 20 30 49	04 13 39 41 42 49	06 13 22 25 40 46	09 14 20 32 40 42	13 16 31 33 40 48	24 25 29 30 40 47
01 14 17 26 32 39	03 08 34 42 44 48	04 14 25 31 43 48	06 15 20 21 35 49	09 17 19 42 43 49	13 18 22 24 41 48	24 27 28 30 32 39
01 15 17 24 27 44	03 08 35 37 40 47	04 15 17 19 28 36	06 21 41 45 47 48	09 18 19 23 24 37	13 23 30 37 42 47	24 31 34 37 47 48
01 16 25 30 38 46	03 09 17 28 31 45	04 18 20 21 34 38	07 08 09 15 38 46	09 20 24 33 44 45	13 23 32 34 36 49	30 31 35 41 43 49
01 17 21 22 23 43	03 10 14 19 22 30	05 06 07 16 19 47	07 08 11 20 41 48	09 21 32 35 43 47	14 15 27 34 36 40	34 40 43 45 47 49
01 23 27 36 37 46	03 10 15 18 25 33	05 07 09 11 27 32	07 08 14 23 24 44	09 23 26 29 39 48	14 17 20 25 44 49	
02 03 04 06 22 33	03 12 13 41 44 45	05 08 10 20 26 32	07 10 14 29 34 38	09 24 26 31 32 46	14 20 24 34 35 46	
02 03 15 16 26 32	03 12 18 28 30 40	05 08 12 21 28 43	07 12 32 38 43 48	10 11 13 17 37 44	14 21 22 28 37 45	
02 04 08 32 40 44	03 12 20 23 40 44	05 09 19 22 34 45	07 13 35 39 45	10 15 19 31 39 43	14 23 26 28 34 42	
02 04 10 24 41 45	03 12 25 26 31 34	05 10 12 18 24 31	07 16 20 21 26 43	10 16 21 29 40 49	15 16 25 28 41 47	

(a) Playing set of cardinality 163 generated by Algorithm 4

01 02 03 04 05 49	02 03 16 17 18 19	03 09 12 15 18 49	08 09 15 17 19 20	23 24 29 36 40 47	24 29 35 39 40 46	27 29 33 34 44 48
01 02 06 11 17 20	02 04 06 09 15 16	04 05 06 07 10 11	10 11 14 17 19 21	23 24 32 34 38 39	24 30 34 35 38 47	27 29 34 35 38 41
01 02 07 10 16 21	02 04 07 08 14 17	04 05 08 09 12 13	10 11 15 16 18 20	23 25 27 28 33 37	24 32 35 36 38 41	27 30 31 37 43 44
01 02 08 13 15 18	02 04 10 13 19 20	04 05 14 15 18 19	12 13 14 17 18 20	23 25 30 37 46 48	24 34 36 38 42 46	27 30 32 38 42 45
01 02 09 12 14 19	02 04 11 12 18 21	04 05 16 17 20 21	12 13 15 16 19 21	23 26 29 32 39 45	24 37 41 43 44 45	27 30 35 40 41 47
01 03 06 10 15 19	02 05 06 08 19 21	04 06 12 17 19 49	22 23 25 31 34 48	23 26 32 35 42 45	25 26 28 33 38 40	27 32 34 35 36 40
01 03 07 11 14 18	02 05 07 09 18 20	04 07 13 16 18 49	22 23 25 35 42 43	23 26 34 36 41 47	25 26 31 40 41 44	27 35 38 39 45 46
01 03 08 12 17 21	02 05 10 12 15 17	04 08 10 15 21 49	22 23 28 30 44 46	23 27 32 39 40 41	25 28 29 33 44 46	27 36 40 41 42 46
01 03 09 13 16 20	02 05 11 13 14 16	04 09 11 14 20 49	22 24 26 27 28 44	23 27 36 38 45 47	25 28 30 33 34 43	28 30 31 36 37 39
01 04 06 13 14 21	02 06 10 14 18 49	05 06 13 15 20 49	22 24 31 41 45 48	23 28 31 35 37 42	25 29 37 38 41 48	28 31 32 37 46 47
01 04 07 12 15 20	02 07 11 15 19 49	05 07 12 14 21 49	22 24 33 37 40 41	23 29 30 36 39 42	25 31 32 35 36 44	28 32 35 39 43 48
01 04 08 11 16 19	02 08 12 16 20 49	05 08 11 17 18 49	22 25 27 29 34 43	23 29 37 43 44 48	25 31 32 39 42 44	28 32 36 42 43 48
01 04 09 10 17 18	02 09 13 17 21 49	05 09 10 16 19 49	22 25 30 36 39 43	23 30 31 33 43 46	25 31 35 39 44 47	28 35 36 43 47 48
01 05 06 12 16 18	03 04 06 08 18 20	06 07 14 16 19 20	22 25 32 43 46 47	23 30 34 38 40 46	25 31 36 42 44 47	28 39 42 43 47 48
01 05 07 13 17 19	03 04 07 09 19 21	06 07 15 17 18 21	22 26 31 38 40 48	23 30 35 36 46 47	25 34 37 40 45 48	29 31 33 38 41 43
01 05 08 10 14 20	03 04 10 12 14 16	06 08 10 13 16 17	22 26 33 37 38 45	23 33 35 42 44 48	26 28 41 43 45 48	30 33 36 39 44 48
01 05 09 11 15 21	03 04 11 13 15 17	06 08 11 12 14 15	22 27 28 31 46 48	24 25 26 27 37 48	26 29 30 35 45 47	31 33 34 40 43 45
01 06 07 08 09 49	03 05 06 09 14 17	06 09 10 12 20 21	22 28 29 38 41 44	24 25 28 33 41 45	26 29 32 45 46 47	32 33 44 46 47 48
01 10 11 12 13 49	03 05 07 08 15 16	06 09 11 13 18 19	22 28 34 40 44 45	24 25 31 38 44 45	26 29 36 42 45 46	38 39 40 41 42 47
01 14 15 16 17 49	03 05 10 13 18 21	07 08 10 12 18 19	22 29 30 31 33 48	24 26 27 31 33 43	26 30 32 34 41 42	
01 18 19 20 21 49	03 05 11 12 19 20	07 08 11 13 20 21	22 32 33 35 36 37	24 26 29 34 40 45	26 30 32 36 39 45	
02 03 06 07 12 13	03 06 11 16 21 49	07 09 10 13 14 15	22 33 37 39 42 47	24 27 34 39 42 47	26 34 35 39 41 46	
02 03 08 09 10 11	03 07 10 17 20 49	07 09 11 12 16 17	22 34 37 43 44 46	24 28 38 40 43 48	26 37 38 40 43 44	
02 03 14 15 20 21	03 08 13 14 19 49	08 09 14 16 18 21	23 24 27 30 41 46	24 29 30 32 40 42	27 28 29 31 34 37	

(b) Lottery set of cardinality 163 obtained from Internet repository tables

Table B.1: The (a) playing set generated by Algorithm 4 and (b) lottery set obtained from Internet repository tables [19, 44, 133] yielding the lower bounds  $\Psi_{163}(49, 6; 3) \geq \frac{13752983}{13983816} \approx 98.3493\%$  and  $\Psi_{163}(49, 6; 3) \geq 1$  (or equivalently  $L_1(49, 6; 3) \leq 163$ ), respectively.

$m$	$L_1(m, 3; 2)$	$L_1(m, 4; 2)$	$L_1(m, 4; 3)$	$L_1(m, 5; 2)$	$L_1(m, 5; 3)$	$L_1(m, 5; 4)$
3	$1^{a,u}$	—	—	—	—	—
4	$1^{a,u}$	$1^{a,u}$	$1^{o,r,u}$	—	—	—
5	$2^{a,u}$	$1^{a,u}$	$1^{o,r,u}$	$1^{o,r,u,v}$	$1^{o,r,u}$	$1^{o,r,u}$
6	$2^{a,u}$	$1^{a,u}$	$3^{s,u}$	$1^{o,r,u,v}$	$1^{o,r,u}$	$1^{o,r,u}$
7	$4^{a,u}$	$2^{a,u}$	$4^u$	$1^{o,r,u,v}$	$1^{o,r,u}$	$3^u$
8	$5^{a,u}$	$2^{a,u}$	$6^u$	$1^{o,r,u,v}$	$2^{o,j,k,l,u}$	$5^u$
9	$7^{a,u}$	$2^{a,u}$	$9^{o,k,l,u}$	$2^{o,p,u,v}$	$2^{o,j,k,l,u}$	$9^u$
10	$8^{a,u}$	$3^{a,u}$	$12^u : 14^u$	$2^{o,u,v}$	$2^{g,o,j,k,l,u}$	$10^{k,l,o,u} : 14^u$
11	$10^{c,u}$	$3^{a,u}$	$16^u : 19^u$	$2^{o,u,v}$	$5^u$	$17^u : 22^u$
12	$11^{a,u}$	$3^{a,u}$	$21^u : 26^u$	$2^{o,u,v}$	$6^u$	$25^u : 35^u$
13	$13^{a,u}$	$5^{a,u}$	$28^u : 33^u$	$3^{j,k,u,v}$	$8^u$	$36^u : 48^u$
14	$14^{a,u}$	$5^{a,u}$	$35^u : 43^u$	$3^{j,k,u,v}$	$6^u : 10^u$	$50^u : 72^u$
15	$18^{a,u}$	$7^{a,u}$	$44^u : 52^u$	$3^{j,g,u,v}$	$9^u : 13^u$	$68^u : 99^u$
16	$19^{a,u}$	$7^{a,u}$	$54^u : 66^u$	$3^{j,g,u,v}$	$11^u : 16^u$	$91^u : 134^u$
17	$23^{a,u}$	$9^{a,u}$	$66^u : 80^u$	$4^{j,k,u,v}$	$13^u : 20^u$	$119^u : 193^u$
18	$24^{a,u}$	$9^{a,u}$	$79^u : 103^u$	$4^{j,g,u,v}$	$16^u : 24^u$	$153^u : 256^u$
19	$29^{a,u}$	$11^{a,u}$	$94^u : 124^u$	$4^{j,g,u,v}$	$19^u : 28^u$	$194^u : 319^u$
20	$31^{a,u}$	$12^{a,u}$	$111^u : 147^u$	$4^{j,g,u,v}$	$22^u : 32^u$	$242^u : 400^u$
21	$36^{a,u}$	$12^a : 14^a$	$111^{i,u} : 147^u$	$6^{f,v}$	$22^{i,u} : 37^u$	$283^g : 461^u$
22	$38^{a,u}$	$13^{a,f} : 14^a$	$122^g : 173^u$	$6^{f,v}$	$24^g : 40^u$	$347^g : 575^u$
23	$43^{c,f,u}$	$14^{a,f} : 17^a$	$141^g : 196^u$	$7^f : 8^{i,u,v}$	$27^g : 49^u$	$421^g : 702^u$
24	$45^{a,u}$	$16^a : 18^a$	$161^g : 231^u$	$7^f : 8^u$	$31^g : 54^u$	$506^g : 835^u$
25	$50^{a,u}$	$17^{a,f} : 18^a$	$184^g : 255^u$	$8^f : 9^u$	$35^g : 63^u$	$604^g : 980^u$
26	$52^{a,u}$	$18^{a,f} : 21^a$	$208^g : 293^u$	$9^f : 10^u$	$40^g : 68^u$	$715^g : 1158^u$
27	$59^{a,u}$	$20^a : 22^a$	$234^g : 324^u$	$9^f : 11^u$	$45^g : 77^u$	$841^g : 1355^u$
28	$61^{a,u}$	$21^{a,f} : 22^a$	$263^g : 375^u$	$10^f : 12^u$	$51^g : 86^u$	$983^g : 1608^u$
29	$68^{a,u}$	$23^a : 25^a$	$294^g : 411^u$	$11^f : 13^u$	$57^g : 97^u$	$1142^g : 1854^u$
30	$70^{a,u}$	$24^a : 27^a$	$327^g : 469^u$	$11^f : 14^u$	$63^g : 102^u$	$1320^g : 2134^u$
31	$78^{a,u}$	$26^{a,f} : 27^a$	$362^g : 512^u$	$12^f : 15^u$	$70^g : 111^u$	$1518^g : 2497^u$
32	$81^{a,u}$	$28^a : 29^a$	$400^g : 580^u$	$12^{f,g,h} : 16^u$	$78^g : 124^u$	$1736^g : 2918^u$
33	$89^{a,u}$	$30^a : 31^a$	$440^g : 613^u$	$14^f : 17^u$	$86^g : 133^u$	$1978^g : 3330^u$
34	$92^{a,u}$	$32^{a,f}$	$484^g : 686^u$	$14^f : 18^u$	$94^g : 136^{d,u}$	$2244^g : 3868^u$
35	$100^{c,f,u}$	$33^{a,f} : 34^a$	$529^g : 742^u$	$14^{f,g,h} : 19^u$	$103^g : 162^u$	$2537^g : 4464^u$
36	$103^{a,u}$	$35^a$	$578^g : 821^u$	$15^{f,g,h} : 20^u$	$112^g : 176^u$	$2856^g : 4918^u$
37	$111^{a,u}$	$37^{a,f}$	$629^g : 871^u$	$17^f : 21^u$	$123^g : 201^u$	$3206^g : 5520^u$
38	$114^{a,u}$	$38^{f,u}$	$684^g : 956^u$	$18^f : 22^u$	$133^g : 216^u$	$3586^g : 5898^u$
39	$124^{a,u}$	$39^{f,g,h,u}$	$741^g : 1014^u$	$18^{f,g,h} : 23^u$	$145^g : 240^u$	$3999^g : 6555^u$
40	$127^{a,u}$	$44^{a,f}$	$802^g : 1126^u$	$19^f : 24^u$	$156^g : 255^u$	$4446^g : 7514^u$
41	$137^{a,u}$	$45^{f,u}$	$866^g : 1183^u$	$21^f : 25^u$	$169^g : 280^u$	$4931^g : 8557^u$
42	$140^{a,u}$	$46^{f,g,h,u}$	$933^g : 1306^u$	$22^f : 26^u$	$182^g : 295^u$	$5453^g : 9609^u$
43	$151^{a,u}$	$51^{a,f}$	$1004^g : 1307^u$	$23^f : 27^u$	$196^g : 320^u$	$6017^g : 10643^u$
44	$155^{a,u}$	$52^{f,u}$	$1078^g : 1500^u$	$23^f : 27^u$	$211^g : 338^u$	$6622^g : 11292^u$
45	$166^{a,u}$	$53^{f,g,h,u}$	$1155^g : 1575^u$	$26^f : 29^u$	$226^g : 359^u$	$7273^g : 11910^u$
46	$170^{a,u}$	$58^{a,f}$	$1237^g : 1726^u$	$26^f : 30^u$	$242^g : 374^u$	$7970^g : 13485^u$
47	$181^{c,f,u}$	$59^{f,u}$	$1322^g : 1815^u$	$27^f : 32^u$	$259^g : 411^u$	$8716^g : 15205^u$
48	$185^{a,u}$	$60^{f,g,h,u}$	$1410^g : 1971^u$	$27^{f,g,h} : 33^u$	$276^g : 432^u$	$9513^g : 16940^u$
49	$196^{a,u}$	$66^{a,f}$	$1503^g : 2104^u$	$30^f : 35^u$	$294^g : 447^u$	$10364^g : 18911^u$
50	$200^{a,u}$	$67^{f,u}$	$1600^g : 2250^u$	$30^f : 36^u$	$314^g : 491^u$	$11270^g : 20961^u$

Table B.2: Known lottery numbers and best bounds (using the notation lower bound : upper bound) on  $L_1(m, n; k)$ ,  $1 \leq k \leq n$  and  $3 \leq m \leq 50$  for  $n = 3, 4$  and  $5$ . Motivation for table entries are as follows: <sup>a</sup>Class of lottery numbers  $L(m, 3; 2)$ , (1.2), and  $L_1(m, 4; 2)$  due to Bate & Stanton [15]. <sup>b</sup>Class of lottery numbers  $L(m, 6; 2)$  due to Bate & Van Rees [17]. <sup>c</sup>Actual bound on  $L_1(m, 3; 2)$  for the case  $m \equiv 11 \pmod{12}$  that differs from that derived by Bate & Stanton [15]. <sup>d</sup>Due to Colbourn [50]. <sup>e</sup>The Payan asymptotic upper bound, (4.9). <sup>f</sup>The Füredi lower bound, (4.25) [83]. <sup>g</sup>The Túrán lower bound, (4.20) [59]. <sup>h</sup>The Hanani lower bound, (4.24) [99]. <sup>i</sup>The recursive lower bound, given in Theorem 2.2(a). <sup>j</sup>The generalised Schönheim lower bound, (4.19). <sup>k</sup>The Sterboul lower bound, (4.21) [236]. <sup>l</sup>The Sterboul lower bound, (4.22) [236]. <sup>m</sup>The Arnautov upper bound, (4.8). <sup>n</sup>The Clark, et al. upper bound, (4.15). <sup>o</sup>The graph theoretic lower bound, (4.1). <sup>p</sup>The Payan and Marcu upper bound, (4.10). <sup>q</sup>The Caro & Roditty upper bound, (4.11). <sup>r</sup>The Vizing upper bound, (4.6). <sup>s</sup>The Fulman upper bound, (4.13). <sup>t</sup>The Reed upper bound, (4.14). <sup>u</sup>Lottery and covering numbers available in repository tables on the Internet [19, 44, 133, 237]. <sup>v</sup>Upper bound obtained via Algorithms 2–7.

$m$	$L_1(m, 6; 2)$	$L_1(m, 6; 4)$	$L_1(m, 6; 5)$
6	$1^{b,o,r,u}$	$1^{o,r,u}$	$1^{o,r,u}$
7	$1^{b,o,r,u}$	$1^{o,r,u}$	$1^{o,r,u}$
8	$1^{b,o,r,u}$	$1^{o,r,u}$	$4^u$
9	$1^{b,o,r,u}$	$3^u$	$7^u$
10	$1^{b,o,r,u}$	$3^u$	$12^u : 14^u$
11	$2^{b,o,p,s,u}$	$5^u$	$17^u : 22^u$
12	$2^{b,f,j,k,l,o,u}$	$6^u$	$27^u : 38^u$
13	$2^{b,f,j,k,l,o,u}$	$8^u : 10^u$	$43^u : 61^u$
14	$2^{b,f,j,k,l,o,u}$	$8^u : 14^u$	$67^u : 100^u$
15	$2^{b,f,j,k,l,o,u}$	$10^u : 19^u$	$100^u : 152^u$
16	$3^{b,f,j,k,l,u}$	$13^u : 26^u$	$146^u : 251^u$
17	$3^{b,f,k,l,u}$	$17^u : 36^u$	$207^u : 355^u$
18	$3^{b,f,j,k,l,u}$	$22^u : 42^{d,u}$	$286^u : 475^u$
19	$3^{b,f,j,k,l,u}$	$27^u : 54^u$	$388^u : 652^u$
20	$3^{b,f,j,k,l,u}$	$34^u : 66^u$	$517^u : 929^u$
21	$4^{b,f,j,k,l,u}$	$36^g : 80^u$	$639^g : 1\,126^u$
22	$4^{b,f,j,k,l,u}$	$44^g : 105^u$	$830^g : 1\,467^u$
23	$4^{b,f,j,k,l,u}$	$54^g : 127^u$	$1\,063^g : 1\,914^u$
24	$4^{b,f,g,j,k,l,u}$	$65^g : 152^u$	$1\,346^g : 2\,432^u$
25	$4^{b,f,g,j,k,l,u}$	$77^g : 175^u$	$1\,687^g : 3\,080^u$
26	$5^{b,f,g,k,l,u}$	$91^g : 220^u$	$2\,093^g : 3\,880^u$
27	$5^{b,f,g,k,l,u}$	$108^g : 243^u$	$2\,574^g : 4\,966^u$
28	$5^{b,f,g,h,j,k,l,u}$	$126^g : 301^u$	$3\,140^g : 6\,196^u$
29	$5^{b,f,g,h,j,k,l,u}$	$147^g : 354^u$	$3\,801^g : 7\,701^u$
30	$5^{b,f,g,h,j,k,l,u}$	$170^g : 411^u$	$4\,568^g : 9\,424^u$
31	$7^{b,f,u}$	$195^g : 477^u$	$5\,454^g : 11\,461^u$
32	$7^{b,f,u}$	$224^g : 577^u$	$6\,473^g : 13\,805^u$
33	$7^{b,f,g,h,u}$	$255^g : 681^u$	$7\,639^g : 16\,650^u$
34	$8^{b,f,u}$	$290^g : 792^u$	$8\,967^g : 19\,778^u$
35	$9^{b,u}$	$328^g : 922^u$	$10\,472^g : 23\,508^u$
36	$9^{b,f,u}$	$369^g : 1\,008^u$	$12\,174^g : 27\,725^u$
37	$10^{b,f,u}$	$415^g : 1\,180^u$	$14\,090^g : 32\,470^u$
38	$11^{b,u}$	$464^g : 1\,364^u$	$16\,240^g : 37\,640^u$
39	$11^{b,f,u}$	$518^g : 1\,578^u$	$18\,644^g : 43\,040^u$
40	$12^{b,u}$	$577^g : 1\,727^u$	$21\,325^g : 49\,510^u$
41	$12^{b,f,u}$	$640^g : 1\,974^u$	$24\,305^g : 56\,972^u$
42	$13^{b,f,u}$	$708^g : 2\,100^u$	$27\,610^g : 65\,249^u$
43	$13^{b,f,u}$	$782^g : 2\,392^u$	$31\,264^g : 74\,491^u$
44	$14^{b,u}$	$861^g : 2\,579^u$	$35\,296^g : 84\,791^u$
45	$15^{b,u}$	$946^g : 2\,822^u$	$39\,732^g : 94\,714^u$
46	$15^{b,f,u}$	$1\,038^g : 3\,074^u$	$44\,604^g : 106\,384^u$
47	$17^{b,u}$	$1\,136^g : 3\,399^u$	$49\,943^g : 119\,777^u$
48	$18^{b,u}$	$1\,240^g : 3\,593^u$	$55\,780^g : 134\,839^u$
49	$19^{b,u}$	$1\,352^g : 3\,977^u$	$62\,151^g : 151\,771^u$
50	$19^{b,u}$	$1\,470^g : 4\,278^u$	$69\,090^g : 365\,795^u$

Table B.2 (continued): Known lottery numbers and best bounds (using the notation lower bound : upper bound) on  $L_1(m, n; k)$ ,  $k = 2, 4$  and  $6$ ,  $n = 6$  and  $3 \leq m \leq 50$ . Motivation for table entries are as follows: <sup>a</sup>Class of lottery numbers  $L(m, 3; 2)$ , (1.2), and  $L_1(m, 4; 2)$  due to Bate & Stanton [15]. <sup>b</sup>Class of lottery numbers  $L_1(m, 6; 2)$  due to Bate & Van Rees [17]. <sup>c</sup>Actual bound on  $L(m, 3; 2)$  for the case  $m \equiv 11 \pmod{12}$  that differs from that derived by Bate & Stanton [15]. <sup>d</sup>Due to Colbourn [50]. <sup>e</sup>The Payan asymptotic upper bound, (4.9). <sup>f</sup>The Füredi lower bound, (4.25) [83]. <sup>g</sup>The Turán lower bound, (4.20) [59]. <sup>h</sup>The Hanani lower bound, (4.24) [99]. <sup>i</sup>The recursive lower bound, given in Theorem 2.2(a). <sup>j</sup>The generalised Schönheim lower bound, (4.19). <sup>k</sup>The Sterboul lower bound, (4.21) [236]. <sup>l</sup>The Sterboul lower bound, (4.22) [236]. <sup>m</sup>The Arnautov upper bound, (4.8). <sup>n</sup>The Clark, et al. upper bound, (4.15). <sup>o</sup>The graph theoretic lower bound, (4.1). <sup>p</sup>The Payan and Marcu upper bound, (4.10). <sup>q</sup>The Caro & Roditty upper bound, (4.11). <sup>r</sup>The Vizing upper bound, (4.6). <sup>s</sup>The Fulman upper bound, (4.13). <sup>t</sup>The Reed upper bound, (4.14). <sup>u</sup>Lottery and covering numbers available in repository tables on the Internet [19, 44, 133, 237]. <sup>v</sup>Upper bound obtained via Algorithms 2–7.

$m$	$\binom{m}{6}$	$r$	$\underline{L}_1$	$\underline{L}_2$	$\underline{L}_3$	$\underline{L}_4$	$\underline{L}_5$	$\underline{L}_6$	$L$	$\overline{L}_1$	$\overline{L}_2$	$\overline{L}_3$	$\overline{L}_4$	$\overline{L}_5$	$\overline{L}_6$	$\overline{L}_7$
6	1	0	1	1	1	1	–	1	1	–	–	1	1	–	1	1
7	7	6	1	1	1	1	–	1	1	–	2	2	2	–	2	1
8	28	27	1	1	1	1	–	1	1	–	2	3	3	4	4	1
9	84	83	1	1	1	1	–	1	1	–	2	4	5	5	5	1
10	210	194	2	2	2	2	2	1	2	–	3	6	6	6	6	–
11	462	380	2	2	2	2	2	1	2	–	4	7	7	8	8	–
12	924	661	2	2	2	2	2	1	2	–	4	9	9	10	10	2
13	1 716	1 057	2	2	2	2	2	2	2	–	4	12	12	12	12	4
14	3 003	1 588	2	2	3	3	3	2	4	–	5	14	15	15	15	6
15	5 005	2 274	3	3	3	3	3	2	4	–	7	18	18	19	19	9
16	8 008	3 135	3	3	3	3	3	3	5	–	7	22	22	23	23	13
17	12 376	4 191	3	3	3	4	3	3	6	–	8	26	26	27	27	18
18	18 564	5 462	4	4	4	4	5	4	7	–	10	31	31	32	32	24
19	27 132	6 968	4	4	4	5	5	4	?	9	11	36	36	38	38	31
20	38 760	8 729	5	5	5	5	6	5	?	10	13	42	42	44	44	39
21	54 264	10 765	6	6	6	6	7	6	?	13	14	49	49	51	51	49
22	74 613	13 096	6	6	6	7	6	7	?	15	19	57	57	59	59	60
23	100 947	15 742	7	7	7	8	6	8	?	17	21	65	65	68	68	72
24	134 596	18 723	8	8	8	8	7	9	?	20	24	74	74	77	77	86
25	177 100	22 059	9	9	9	9	8	10	?	22	28	84	84	88	88	102
26	230 230	25 770	9	9	9	10	8	12	?	25	31	95	95	99	99	120
27	296 010	29 876	10	10	10	11	9	13	?	27	35	107	107	111	112	139
28	376 740	34 397	11	11	11	13	10	15	?	31	38	120	120	125	125	161
29	475 020	39 353	13	13	13	14	11	17	?	35	43	134	134	139	139	184
30	593 775	44 764	14	14	14	15	13	19	?	39	48	149	149	155	155	210
31	736 281	50 650	15	15	15	17	14	21	?	45	53	165	165	171	172	238
32	906 192	57 031	16	16	16	18	16	23	?	50	60	183	183	189	189	269
33	1 107 568	63 927	18	18	18	20	18	25	?	55	66	201	201	209	209	302
34	1 344 904	71 358	19	19	19	21	20	28	?	60	72	221	221	229	229	337
35	1 623 160	79 344	21	21	21	23	22	30	?	66	78	242	242	251	251	376
36	1 947 792	87 905	23	23	23	25	24	33	?	72	85	265	265	274	274	417
37	2 324 784	97 061	24	24	24	27	26	36	?	78	92	288	288	298	298	461
38	2 760 681	106 832	26	26	26	29	29	39	?	83	99	314	314	325	325	507
39	3 262 623	117 238	28	28	28	32	31	42	?	89	105	340	340	352	352	557
40	3 838 380	128 299	30	30	30	34	34	46	?	96	112	369	369	381	381	611
41	4 496 388	140 035	33	33	33	36	37	50	?	102	120	398	399	412	412	667
42	5 245 786	152 466	35	35	35	39	40	54	?	109	127	430	430	445	445	727
43	6 096 454	165 612	37	37	37	42	43	58	?	117	135	463	463	479	479	790
44	7 059 052	179 493	40	40	40	44	47	62	?	124	144	498	498	515	515	857
45	8 145 060	194 129	42	42	42	47	51	66	?	131	151	535	535	552	552	927
46	9 366 819	209 540	45	45	45	51	55	71	?	138	160	573	573	592	592	1 001
47	10 737 573	225 746	48	48	48	54	59	76	?	146	168	613	613	633	633	1 080
48	12 271 512	242 767	51	51	51	57	63	81	?	153	178	655	655	677	677	1 162
49	13 983 816	260 623	54	54	54	61	67	87	?	163	186	700	700	722	722	1 248
50	15 890 700	279 334	57	57	57	64	72	92	?	175	198	746	746	770	770	1 338

Table B.3: Best bounds on the lottery number  $L_1(m, 6; 3)$  for all  $6 \leq m \leq 50$  in a comparative fashion. The table shows the order of the lottery graph,  $\binom{m}{6}$ ; the degree of regularity of the lottery graph,  $r$  given in (2.1); the lower bound,  $\underline{L}_1$ , from classical graph domination theory according to (4.1) (also given in (4.4) and (4.23)); the combinatorial lower bound,  $\underline{L}_2$ , (4.23); the Sterboul lower bound,  $\underline{L}_3$ , (4.22); the Sterboul lower bound,  $\underline{L}_4$ , (4.21); the generalised Schönheim lower bound,  $\underline{L}_5$ , (4.19); the Turán lower bound,  $\underline{L}_6$  (4.20); the known complete lottery number  $L_1(m, 6; 3)$ ,  $L$ ; the best upper bounds,  $\overline{L}_1$ , available in covering and lottery number repository tables on the Internet [19, 44, 133, 237]; the recursive Li & Van Rees upper bound,  $\overline{L}_2$ , (4.31); the Clark, et al. upper bound,  $\overline{L}_3$ , (4.15); the Arnautov upper bound,  $\overline{L}_4$ , (4.8); the Caro & Roditty upper bound,  $\overline{L}_5$ , (4.11); the asymptotic upper bound,  $\overline{L}_6$ , (4.9); and the upper bound,  $\overline{L}_7$ , according to the generalised domination result 4.2, truncated after the second term, (4.17). A question mark (?) denotes that the lottery number  $L_1(m, 6; 3)$  is not known.





$\vec{X}^{(\ell)}$ -vector structure encodings associated with Table 6.5, by table superscript:

1.	(00032110)	2.	(01122110)	3.	(0002011001102000)
4.	(00000001000101100110100010000000)				(00000002000110000100001020000000)
5.	(02212110)	6.	(0002111011101000)		
7.	(00010011010010000100100010100000)				(00010001010010100110100010000000)
	(00000002011010000110100010000000)				(00010110000110000100001020000000)
8.	(03213010)	(03303001)	9.	(0111111011101000)	
10.	(00010110011010000110100010000000)				11. (03314000) (13213010) (13303001)
	(22212110)	12. (01145000)			13. (0211111020101000)
	(0220111020011000)	(1111111011101000)			14. (0004011020003000)
15.	(01010110011010001010100010000000)				(01100110011010001001100010000000)
	(10010110011010000110100010000000)				16. (04404000) (13314000) (23213010)
	(23303001) (32212110)	17. (03313110)			(03224010)
18.	(0221200130001000)	(0221111030001000)			(0221210030000010)
	(1211111020101000)	(1220111020011000)			(2111111011101000)
19.	(0003022040001000)	20.	(01110110101010001100100010000000)		
	(01101110100110001110000010000000)				(01110110011010002000100010000000)
	(01111000110100020000010100000000)				(01111100101010002000001001000000)
	(11010110011010001010100010000000)				(11100110011010001001100010000000)
	(20010110011010000110100010000000)				21. (00010111011010000110100020000000)
22.	(14404000) (23314000) (33213010) (33303001)				(42212110) 23. (04314010)
24.	(0311301030100000)	(0321201030001000)			(1221201021001000)
	(0330200130001000)	(0330300030000001)			(0221211040000000)
	(1221111030001000)	(1221210030000010)			(2211111020101000)
	(2220111020011000)	(3111111011101000)			25. (0112211021101000)
26.	(01111110101010002100000010000000)				(02010110201010002010100000000000)
	(02110110101010002000100010000000)				(02111010101010002000010010000000)
	(11110110101010001100100010000000)				(02101110100110002010000010000000)
	(02200110100110002000100010000000)				(02200110200010002000100000010000)
	(02201100100110002000001010000000)				(11101110100110001110000010000000)
	(01111110011010003000000010000000)				(01111110111000003000000000001000)
	(11110110011010002000100010000000)				(11111000110100020000010100000000)
	(11111001010100020000010010000000)				(21010110011010001010100010000000)
	(21100110011010001001100010000000)				(30010110011010000110100010000000)
27.	(00020110011020001110100010000000)				28. (24404000) (33314000) (43213010)
	(43303001) (52212110)	29. (04415000)			30. (02246000)
31.	(0331300030001000)	(0321301040000000)			(1311301030100000)
	(1321201030001000)	(2221201021001000)			(0330300140000000)
	(1330200130001000)	(1330300030000001)			(1221211040000000)
	(2221111030001000)	(2221210030000010)			(3211111020101000)
	(3220111020011000)	(4111111011101000)			32. (0221211021101000)
33.	(02112010110010002010000010000000)				(02111110200010002010000010000000)
	(02210110200010002000100010000000)				(01212100200000102100000010000000)
	(02211100200000102000100010000000)				(02212000200000102000010010000000)
	(02110110201010003000100000000000)				(02111100201010003000001000000000)
	(02111110101010003000000010000000)				(02111110201000003000000000001000)
	(11111110101010002100000010000000)				(12010110201010002010100000000000)
	(12110110101010002000100010000000)				(12111010101010002000010010000000)
	(21110110101010001100100010000000)				(02200110200110003000100000000000)
	(02201110100110003000000010000000)				(02201110200010003000000000010000)
	(02201110200100003000000000001000)				(12101110100110002010000010000000)
	(12200110100110002000100010000000)				(12200110200010002000100000010000)
	(12201100100110002000001010000000)				(21101110100110001110000010000000)
	(01111110111010004000000000000000)				(11111110011010003000000010000000)
	(11111110111000003000000000001000)				(21110110011010002000100010000000)
	(21111100011010002000001010000000)				(21111100101010002000001001000000)
	(31010110011010001010100010000000)				(31100110011010001001100010000000)
	(40010110011010000110100010000000)				34. (00021110111010001110100010000000)
35.	(34404000) (43314000) (53213010) (53303001)				(62212110) 36. (05505000)
	(03336000)	38. (0331400040000000)			(1331300030001000)
	(1321301040000000)	(2311301030100000)			(2321201030001000)
	(3221201021001000)	(1330300140000000)			(2330200130001000)
	(2330300030000001)	(2221211040000000)			(3221111030001000)
	(3221210030000010)	(4211111020101000)			(4220111020011000)
	(5111111011101000)	39. (0321211030101000)			

40. (02212010200010002100000010000000) (02212010110010003000000010000000)  
(12112010110010002010000010000000) (03202001200010002010000010000000)  
(02111110300010003010000000000000) (02210110300010003000100000000000)  
(022111102000100030000000010000000) (0221111030000003000000000001000)  
(12111110200010002010000010000000) (12210110200010002000100010000000)  
(01212100300000103100000000000000) (02211100300000103000100000000000)  
(02212000300000103000010000000000) (02212100200000103000000010000000)  
(02212100300000030000000000000010) (11212100200000102100000010000000)  
(12211100200000102000100010000000) (12212000200000102000010010000000)  
(02111110201010004000000000000000) (12110110201010003000100000000000)  
(12111100201010003000001000000000) (1211110101010003000000010000000)  
(1211111020100003000000000001000) (21111110101010002100000010000000)  
(22010110201010002010100000000000) (22110110101010002000100010000000)  
(22111010101010002000010010000000) (31110110101010001100100010000000)  
(02201102001100040000000000000000) (12200110200110003000100000000000)  
(12201101001100030000000010000000) (122011020001000300000000001000)  
(122011020010003000000000001000) (22101110100110002010000010000000)  
(22200110100110002000100010000000) (22200110200010002000100000010000)  
(22201100100110002000001010000000) (31101110100110001110000010000000)  
(11111110111010004000000000000000) (21111110011010003000000010000000)  
(2111111011100003000000000001000) (31110110011010002000100010000000)  
(31111100011010002000001010000000) (31111100101010002000001001000000)  
(41010110011010001010100010000000) (41100110011010001001100010000000)  
(50010110011010000110100010000000) 41. (01111110111010001110100010000000)  
42. (44404000) (53314000) (63213010) (63303001) (72212110) 43. (15505000)  
44. (04426000) 45. (0440400040000000) (1331400040000000)  
(2331300030001000) (2321301040000000) (3311301030100000)  
(3321201030001000) (4221201021001000) (2330300140000000)  
(3330200130001000) (3330300030000001) (3221211040000000)  
(4221111030001000) (4221210030000010) (5211111020101000)  
(5220111020011000) (6111111011101000) 46. (0331301031001000)  
(0331211040001000) 47. (0004222060000000)  
48. (03013010301000003010000000000000)\* 49. (02111110111010002010100010000000)  
50. (54404000) (63314000) (73213010) (73303001) (82212110) 51. (25505000)  
52. (05516000) 53. (1440400040000000) (2331400040000000)  
(3331300030001000) (3321301040000000) (4311301030100000)  
(4321201030001000) (5221201021001000) (3330300140000000)  
(4330200130001000) (4330300030000001) (4221211040000000)  
(5221111030001000) (5221210030000010) (6211111020101000)  
(6220111020011000) (7111111011101000) 54. (0421401040100000)  
(0431301040001000) (0331311050000000) 55. (0113222060000000)  
56. (03213010300000003000000010000000)\* 57. (02211110201010002100100010000000)  
(02211110111010003000100010000000) 58. (00010112022020006000000000000000)  
59. (64404000) (73314000) (83213010) (83303001) (92212110) 60. (35505000)  
61. (06606000) 62. (2440400040000000) (3331400040000000)  
(4331300030001000) (4321301040000000) (5311301030100000)  
(5321201030001000) (6221201021001000) (4330300140000000)  
(5330200130001000) (5330300030000001) (5221211040000000)  
(6221111030001000) (6221210030000010) (7211111020101000)  
(7220111020011000) (8111111011101000) 63. (0441400040001000)  
(0431401050000000) 64. (0222222060000000)  
65. (03313000300010004000000000000000)\* 66. (02212110201010003100000010000000)  
(03111110301010003010100000000000) (03211110201010003000100010000000)  
(022121101110100040000000010000000) 67. (00020220022020006000000000000000)  
68. (74404000) (83314000) (93213010) (93303001) (A2212110) 69. (45505000)  
70. (16606000) 71. (3440400040000000) (4331400040000000)  
(5331300030001000) (5321301040000000) (6311301030100000)  
(6321201030001000) (7221201021001000) (5330300140000000)  
(6330200130001000) (6330300030000001) (6221211040000000)  
(7221111030001000) (7221210030000010) (8211111020101000)  
(8220111020011000) (9111111011101000) 72. (0441500050000000)  
73. (0322312060000000) 74. (03314000400000004000000000000000)\*  
75. (03213010210010003010000010000000) (032121103000100030100000010000000)  
(03311110300010003000100010000000) (03211110301010004000100000000000)  
(03212110201010004000000010000000) (02212110211010005000000000000000)  
76. (01020220112020006000000000000000) 77. (84404000) (93314000) (A3213010)  
(A3303001) (B2212110) 78. (55505000) 79. (26606000)

80.	(4440400040000000)	(5331400040000000)	(6331300030001000)
	(6321301040000000)	(7311301030100000)	(7321201030001000)
	(8221201021001000)	(6330300140000000)	(7330200130001000)
	(7330300030000001)	(7221211040000000)	(8221111030001000)
	(82212100300000010)	(9211111020101000)	(9220111020011000)
	(A111111011101000)	81.	(0550500050000000)
82.	(0441311050001000)	83.	(04404000400000004000000000000000)*
84.	(03313010300010003100000010000000)		(03313010210010004000000010000000)
	(03212110400010004010000000000000)		(03311110400010004000100000000000)
	(03312110300010004000000010000000)		(03212110301010005000000000000000)
85.	(01120220211020006000000000000000)		

$\vec{X}^{(\ell)}$ -vector structure encodings associated with Table 6.6, by table superscript:

1.	(01020230)	2.	(0001010102000030)	3.	(01121220)	
4.	(0001011102101010)	5.	(00000001000101100100011010100000)			
	(00010001000001100000011011100000)		(00010000000001200001011011000000)			
	(00000001000101100200001000101000)	6.	(02211220)	(11121220)	(02212111)	
7.	(0101012001201100)	(0001022002201000)	(0101012010111100)			
8.	(00010010000111000100011010100000)	9.	(03301220)	(12211220)	(03302111)	
	(03212120)	(21121220)	(12212111)	(03303002)		
11.	(0210012001201100)	(0110022002201000)	(1101012001201100)			
	(1001022002201000)	(0210012010111100)	(0201102010200200)			
	(0201012010201100)	(0111111011100110)	(0111021011101010)			
	(0110022011111000)	(0101022011201000)	(0201012001202000)			
	(1101012010111100)	(0111111020010110)	(0111111011101001)			
12.	(0001013004000020)	13.	(01000110002001000110001001001000)			
	(00000120011001000210000000101000)	(01010010001011001000011001100000)				
	(01000110002001001001001001001000)	(00010110011010001100001000100100)				
	(00010110011001001100001000101000)	(00010110002001001100001001001000)				
	(00010110001011001100001001100000)	(00010100002011001100002001000000)				
	(00000120011001001101000000101000)	(00010200002010001000002002000000)				
	(00010110011010001000011001100000)	(00000210012000001001001001001000)				
	(01100010000111000100011010100000)	(01000110001110000110001010000100)				
	(01000110001101000110001010001000)	(00100110001101000200001010001000)				
	(00000120001101000210000010001000)	(00000110001111000210001010000000)				
	(01010010001002000100101010100000)	(00010020001002000200100010100000)				
	(00010110011001000110100010000010)	(00010110010001100110100010100000)				
	(00000110011001100111100010000000)	(00010200011000100110001010001000)				
	(10010010000111000100011010100000)	(01010010001011001000011010010000)				
	(01000020100101001001010000101000)	(00010110011010000110100010000001)				
14.	(13301220)	(04302120)	(22211220)	(13302111)	(13212120)	(04303011)
	(31121220)	(22212111)	(13303002)	15.	(02222220)	(03313110)
16.	(1210012001201100)	(1110022002201000)	(0310102010200200)			
	(0310012010201100)	(0220111011100110)	(0220021011101010)			
	(0210112011200100)	(0210022011201000)	(0110122012200000)			
	(0310012001202000)	(0220012002102000)	(2101012001201100)			
	(2001022002201000)	(1210012010111100)	(1201102010200200)			
	(1201012010201100)	(1111111011100110)	(1111021011101010)			
	(1110022011111000)	(1101022011201000)	(0220111020010110)			
	(0220021020011010)	(0211111020100110)	(0211102020100200)			
	(0211012020101100)	(0210112020110100)	(0210022020111000)			
	(0111121021100010)	(0111022021101000)	(0310101110201100)			
	(0220111011101001)	(1201012001202000)	(0310012010112000)			
	(0211111011101010)	(0111112012101000)	(2101012010111100)			
	(111111020010110)	(1111111011101001)	(0220200120010110)			
	(0220111020011001)	17.	(0002022002202000)			
18.	(11000110002001000110001001001000)*	19.	(00010000002003000200003010000000)			
	(00000001003002000300002000001000)	20.	(23301220)	(14302120)	(05303020)	
	(04403110)	(32211220)	(23302111)	(23212120)	(14303011)	(13313110)
	(41121220)	(32212111)	(23303002)	21.	(03312220)	(04404001)
22.	(2210012001201100)*	23.	(0111022002202000)			
24.	(01111100101010002000001001000001)*	25.	(00010100012011000200003010000000)			
	(00010101010101001101010110000000)	26.	(33301220)	(24302120)	(15303020)	
	(14403110)	(05404010)	(42211220)	(33302111)	(33212120)	(24303011)
	(14404001)	(51121220)	(42212111)	(33303002)	27.	(04313120)
28.	(3210012001201100)*	29.	(0211022011202000)			

30. (02200110200010002000100000010001)\* 31. (00000220011110000220000010001000)  
 32. (43301220) (34302120) (25303020) (24403110) (15404010) (05505000) (52211220)  
 (43302111) (43212120) (34303011) (33313110) (24404001) (61121220) (52212111)  
 (43303002) 33. (04414110) 34. (07101060)  
 35. (4210012001201100)\* 36. (0311112011202000)  
 37. (0200006005101000) 38. (12200110200010002000100000010001)\*  
 39. (00010220012010000210100020000000) 40. (53301220) (44302120) (35303020)  
 (34403110) (25404010) (15505000) (62211220) (53302111) (53212120) (44303011)  
 (43313110) (34404001) (71121220) (62212111) (53303002)  
 41. (05504110) (14414110) (05505001) 42. (04413220)  
 43. (5210012001201100)\* 44. (0221122031101000) (0221212022101000)  
 45. (0401013001303000) 46. (22200110200010002000100000010001)\*  
 47. (01010120012020001210100010000000) 48. (00010220022000000220000000003000)  
 49. (74100140) (73200230) (64201130) (63301220) (54302120) (45303020) (44403110)  
 (35404010) (25505000) (72211220) (63302111) (63212120) (54303011) (53313110)  
 (44404001) (81121220) (72212111) (63303002) 50. (15504110) (06505010)  
 (24414110) (15505001) 51. (05414120) 52. (3330300030000002)\*  
 53. (0321212031101000) 54. (0411013011203000)  
 55. (0330300030000000300000000000000002)\* 56. (02010120012020002110100010000000)  
 57. (00010220022010000220100010001000) 58. (84100140) (83200230) (74201130)  
 (73301220) (64302120) (55303020) (54403110) (45404010) (35505000) (82211220)  
 (73302111) (73212120) (64303011) (63313110) (54404001) (91121220) (82212111)  
 (73303002) 59. (25504110) (16505010) (06606000) (34414110) (25505001)  
 60. (05515110) 61. (08202060) 62. (4330300030000002)\*  
 63. (0331311031101000) 64. (0321122021202000)  
 65. (1330300030000000300000000000000002)\* 66. (01110220211010002110100010000000)  
 (01111120121010002110100010000000) 67. (00010220022020001220100010000000)  
 68. (94100140) (93200230) (84201130) (83301220) (74302120) (65303020) (64403110)  
 (55404010) (45505000) (92211220) (83302111) (83212120) (74303011) (73313110)  
 (64404001) (A1121220) (92212111) (83303002)  
 69. (35504110) (26505010) (16606000) (44414110) (35505001) 70. (06605110)  
 71. (08303050) 72. (5330300030000002)\* 73. (0440311031101000)  
 (1331311031101000) (0440311040011000) 74. (0421212021202000)  
 75. (2330300030000000300000000000000002)\* 76. (02111120211010002110100010000000)  
 77. (03010120012020000120200020000000) 78. (94100140) (93200230) (84201130)  
 (83301220) (74302120) (65303020) (64403110) (55404010) (45505000) (92211220)  
 (83302111) (83212120) (74303011) (73313110) (64404001) (A1121220) (92212111)  
 (83303002) 79. (45504110) (36505010) (26606000) (54414110) (45505001)  
 80. (07606010) 81. (08404040) 82. (6330300030000002)  
 83. (1440311031101000) (0540401040100100) (0540311040101000)  
 (0440411041100000) (2331311031101000) (1440311040011000)  
 84. (0331222041101000) 85. (3330300030000000300000000000000002)  
 86. (02212110211010002110100010000000) 87. (01110220022020003110100010000000)



