On two combinatorial optimisation problems involving lotteries

André du Plessis

Thesis presented in partial fulfilment of the requirements for the degree
MComm (Operations Research)
Department of Logistics, Stellenbosch University

Supervisor: Prof JH van Vuuren
Co-supervisor: Dr AP Burger

March 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

Suppose a lottery draw consists of forming a *winning ticket* by randomly choosing $t \leq m$ distinct numbers from a universal set $\mathcal{U}_m = \{1, \ldots, m\}$. Each lottery participant forms a set of *tickets* prior to the draw, each ticket consisting of $n \leq m$ distinct numbers from $\mathcal{U}_m$, and is awarded a prize if $k \leq \min\{n, t\}$ or more numbers in at least one of his/her tickets matches those of the winning ticket. A lottery of this form is denoted by the quadruple $\langle m, n, t, k \rangle$, and the prize is known as a *k-prize*. The participant's set of tickets is also known as a *playing set*.

The participant may wish to form a playing set in such a way that the probability of winning a k-prize is at least $0 < \psi \leq 1$. Naturally, the participant will want to minimise the cost of forming such a playing set, which means that the cardinality of the playing set should be as small as possible. This combinatorial minimisation problem is known as the *incomplete lottery problem* and was introduced by Gründlingh [16], who also formulated a related problem called the *resource utilisation problem*. In this problem one attempts to select a playing set of pre-specified cardinality ℓ in such a way that the probability of winning a k-prize is maximised.

Gründlingh [16] studied the incomplete lottery problem and the resource utilisation problem in the special case where $n = t$. In this thesis both problems are considered in the general case where $n \neq t$. Exact and approximate solution methods are presented and compared to each other in terms of solution quality achieved, execution time and practical feasibility. The first solution method involves a mathematical programming formulation of both problems. Using this solution method, both problems are solved for small lottery instances. An exhaustive enumeration solution method, which uses the concept of overlapping playing set structures [5, 16], is reviewed and used to solve both combinatorial optimisation problems for the same small lottery instances. The concept of an overlapping playing set structure is further explored and incorporated in an attempt to solve both combinatorial optimisation problems approximately by means of various metaheuristic solution approaches, including a simulated annealing algorithm, a tabu search and a genetic algorithm.

The focus of the thesis finally shifts to a different problem involving lotteries. An investigation is conducted into the probability, $P(N, \lambda)$, of λ participants sharing a k-prize if a total of N tickets are purchased by participants of the lottery $\langle m, n, t, k \rangle$. Special attention is afforded in this problem to the jackpot prize of the South African national lottery, *Lotto*, represented by the quadruple $\langle 49, 6, 6, 6 \rangle$ and how the value of $P(N, \lambda)$ is affected by the way that participants select their playing sets.

Uittreksel

Gestel 'n lotery-trekking bestaan uit die ewekansige seleksie van 'n *wenkaartjie* bestaande uit $t \leq m$ verskillende getalle uit 'n universele versameling $\mathcal{U}_m = \{1, \ldots, m\}$. Elke lotery-deelnemer vorm 'n versameling *kaartjies* voor die trekking, wat elk uit $n \leq m$ verskillende getalle in $\mathcal{U}_m$ bestaan, en wen 'n prys indien $k \leq \min\{n, t\}$ of meer getalle in minstens een van sy/haar kaartjies ooreenstem met dié in die wenkaartjie. 'n Lotery van hierdie vorm word deur die viertal $\langle m, n, t, k\rangle$ aangedui, en die prys staan as 'n *k-prys* bekend. 'n Deelnemer se kaartjies staan ook as a *spelversameling* bekend.

'n Lotery-deelnemer mag poog om sy spelversameling só te selekteer dat die waarskynlikheid om 'n k-prys te wen, minstens $0 < \psi \leq 1$ is. Die deelnemer sal natuurlik die koste wat met so 'n spelversameling gepaard gaan, wil minimeer, wat beteken dat die kardinaliteit van sy spelversameling so klein as moontlik moet wees. Hierdie kombinatoriese minimeringsprobleem staan as die *onvolledige lottery-probleem* bekend en is vir die eerste keer deur Gründlingh [16] bestudeer, wat ook die verwante *hulpbronbenuttingsprobleem* geformuleer het. In laasgenoemde probleem word daar gesoek na 'n spelversameling van vooraf-gespesifiseerde kardinaliteit wat die waarskynlikheid om 'n k-prys te wen, maksimeer.

Gründlingh [16] het die onvolledige lottery-probleem en die hulpbronbenuttingsprobleem in die spesiale geval oorweeg waar $n = t$. In hierdie tesis word beide probleme in die algemeen oorweeg waar $n \neq t$. Eksakte en heuristiese oplossingstegnieke word vir beide probleme daargestel en met mekaar in terme van oplossingskwaliteit, oplossingstyd en praktiese haalbaarheid vergelyk. Die eerste oplossingstegniek behels 'n wiskundige programmeringsformulering van beide probleme. Die probleme word deur middel van hierdie benadering vir klein loterye opgelos. 'n Uitputtende enumerasietegniek, wat gebruik maak van die konsep van spelversameling oorvleuelingstrukture [5, 16], word daarna in oënskou geneem en beide kombinatoriese optimeringsprobleme word vir dieselfde klein loterye met behulp van hierdie tegniek opgelos. Die konsep van 'n spelversameling oorvleuelingstruktuur word verder ondersoek en in 'n benaderde oplossingstegniek vir beide kombinatoriese optimeringsprobleme geïnkorporeer deur gebruik te maak van verskeie metaheuristiese oplossingsbenaderings, insluitende 'n gesimuleerde afkoelingsalgoritme, 'n tabu-soektog en 'n genetiese algoritme.

Die fokus in die tesis verskuif laastens na 'n ander probleem oor loterye. 'n Ondersoek word geloots na die waarskynlikheid, $P(N, \lambda)$, dat λ lottery-deelnemers 'n k-prys sal deel indien 'n totaal van N kaartjies in die lotery $\langle m, n, t, k\rangle$ gekoop word. Spesiale aandag word aan hierdie probleem geskenk in die geval van die boerpot-prys in die Suid-Afrikaanse nasionale lotery, *Lotto*, wat deur die viertal $\langle 49, 6, 6, 6\rangle$ voorgestel word, en hoe die waarde van $P(N, \lambda)$ beïnvloed word deur die manier waarop deelnmers hul spelversamelings selekteer.

Acknowledgements

I would like to thank the following people:

- My parents, for their love, moral, and financial support throughout my studies.
- My friends and family for their patience and support when I ended up spending more time with my studies instead of with them.
- My supervisor, Prof Jan van Vuuren, for sharing his broad knowledge with me, for his patience, and for his belief in me — all factors which inspired me to grow more academically than I ever expected.
- Prof Stephan Visagie for introducing me to the wonderful field of Operations Research.
- Dr Alewyn Burger for his friendly and invaluable advice.
- Taryn Johnston for her caring love and support throughout the time that this thesis was written.

The Department of Logistics is hereby thanked for the use of their computing facilities and office space. The financial support of the South African National Research Foundation (NRF) under grant number GUN 2072999 is hereby acknowledged. Any opinions or findings in this thesis are those of the authour and do not necessarily reflect the views of Stellenbosch University or the NRF.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Reserved Symbols

$\mathbf{A}$	A $\binom{m}{t} \times \binom{m}{n}$ binary adjacency matrix of the lottery graph $\mathbf{G}\langle m, n, t, k\rangle$.
a_{ij}	A binary variable indicating the presence of an edge between government ticket i and participant ticket j. The variables a_{ij} form the matrix $\mathbf{A}$.
$B(\kappa)$	A type of lottery ticket in which the difference between any two consecutive numbers is no less than κ.
$\mathcal{C}$	A covering set $\mathcal{C} \subseteq \Phi(\mathcal{U}_m, n)$ 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 $\{\phi_k\} \cap \Phi(c, k) \neq \emptyset$.
$C(\kappa)$	A type of lottery ticket in which exactly κ of the numbers in it are calendar numbers (*i.e.* in the range $1, 2, \ldots, 31$).
$C(m, n, t)$	The optimal solution to the covering problem, known as the covering number. It represents the minimum cardinality of a covering set $\mathcal{C}$. The covering number is also the answer to the complete lottery problem for lotteries of the form $\langle m, n, t, t\rangle$.
$\mathcal{D}_F$	A set comprising overlapping playing set structures which fail a so-called domination test.
$\mathcal{D}_T$	A set comprising overlapping playing set structures which pass a so-called domination test.
$D(\kappa, c)$	A type of lottery ticket in which the difference between any two consecutive numbers is no less than κ, and exactly c of the numbers are calendar numbers.
$E(N, \lambda)$	The expected number of government tickets covered λ times by N participant tickets.
$\gamma(\mathbf{G})$	The cardinality of a minimum dominating set of graph $\mathbf{G}$.
$\mathbf{G}\langle m, n, n, k\rangle$	A regular graph of order $\binom{m}{n}$. This graph may represent a lottery of the form $\langle m, n, n, k\rangle$. The vertices in this graph represent lottery tickets of cardinality n, and two vertices are adjacent if the tickets which they represent share a common k-subset.
$\mathbf{G}\langle m, n, t, k\rangle$	A bipartite lottery graph of order $\binom{m}{n} + \binom{m}{t}$. There are $\binom{m}{n}$ vertices in this graph representing the possible participant tickets in a lottery, and the remaining $\binom{m}{t}$ vertices in this graph represent the possible government tickets. A vertex representing a participant ticket is adjacent to a vertex representing a government ticket if those two tickets share a common k-subset.
k	The minimum number of lottery ticket numbers that a participant ticket is required to have in common with the winning government ticket in order to win a k-prize in a lottery.
ℓ	The cardinality of a participant's playing set.
λ	The number of concurrent winners of a k-prize in a lottery draw.
$\mathcal{L}_\ell(m, n, t, k)$	A participant's playing set of cardinality ℓ in the lottery $\langle m, n, t, k\rangle$.

$L_\psi(m,n,t,k)$	The incomplete lottery number, representing the smallest possible playing set cardinality which guarantees a participant a probability, ψ, of winning a k-prize in the lottery $\langle m,n,t,k\rangle$.
m	The cardinality of the universal set $\mathcal{U}_m$ in the lottery $\langle m,n,t,k\rangle$ from which n numbers may be chosen to be included in a participant's ticket, and from which t numbers are chosen in order to form the winning government ticket.
$M(\vec{\mathbf{X}}^{(\ell)})$	The multiplicity of an overlapping playing set structure represented by the vector $\vec{\mathbf{X}}^{(\ell)}$.
n	The cardinality of a participant ticket in the lottery $\langle m,n,t,k\rangle$.
$\eta_\psi(m,n,t,k)$	The lottery characterisation number of the lottery $\langle m,n,t,k\rangle$ denoting the number of different overlapping playing set structures associated with the optimal answer to the incomplete lottery problem and the resource utilisation problem.
N	The number of tickets purchased in total by the participants in a lottery draw.
$\mathcal{N}[v]$	The set of government (or participant) tickets which have k or more numbers in common with participant (or government) ticket v. This set is known as the neighbourhood of ticket v.
$\mathcal{O}(g(n))$	A function $f(n)$ grows no faster than $g(n)$ as $n \to \infty$ (denoted by $f(n) = \mathcal{O}(g(n))$), if there exists constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.
θ	The proportion of the total participant tickets purchased in a lottery that are special tickets (*i.e.* of type $A, B(\kappa), C(\kappa)$ or $D(\kappa,c)$).
$P(N,\lambda)$	The probability that λ of the tickets selected by the participants in a lottery draw are winning tickets, under the assumption that N tickets are selected in total by the participants.
$P(x)$	The objective function in an ILP formulation of either the incomplete lottery problem or the resource utilisation problem.
$\Phi(\mathcal{U}_m,n)$	The set of participant tickets consisting of n numbers chosen from $\mathcal{U}_m$.
$\Phi(\mathcal{U}_m,t)$	The set of government tickets consisting of t numbers chosen from $\mathcal{U}_m$.
ψ	The desired probability-of-win value in the incomplete lottery problem.
$\Psi_\ell(m,n,t,k)$	The maximum resource utilisation value associated with a playing set of cardinality ℓ in the lottery $\langle m,n,t,k\rangle$.
$\mathcal{R}_j$	Given the partite sets $\mathcal{V}_1$ and $\mathcal{V}_2$ of the bipartite lottery graph $\mathbf{G}\langle m,n,t,k\rangle$, $\mathcal{R}_j$ is the number of vertices remaining in $\mathbf{G}\langle m,n,t,k\rangle$ after the j^{th} iteration of the greedy covering algorithm (Algorithm 3.1).
t	The number of lottery numbers chosen from $\mathcal{U}_m$ in a government ticket in the lottery $\langle m,n,t,k\rangle$.
$T(m,n,t)$	The Turán number, denoting the smallest number of n-subsets of $\mathcal{U}_m$ such that any t-subset of $\mathcal{U}_m$ contains at least one of these n-subsets.
T_i	The i^{th} ticket in a participant's playing set.
$\mathcal{U}_m$	The universal set of numbers $\{1,\ldots,m\}$ in the lottery $\langle m,n,t,k\rangle$.
u	The number of special participant tickets $u \in \{u_1,u_2,\ldots,u_z\}$ covering a winning government ticket.
$\mathcal{V}_1$	The vertices representing participant tickets in the bipartite lottery graph $\mathbf{G}\langle m,n,t,k\rangle$.
$\mathcal{V}_1'$	A bipartite covering of minimum cardinality in the bipartite minimum covering problem.
$\mathcal{V}_1''$	The bipartite covering set obtained via the greedy covering algorithm (Algorithm 3.1).

$\mathcal{V}_2$	The vertices representing government tickets in the bipartite lottery graph $\mathbf{G}\langle m, n, t, k\rangle$.
w_j	The number of government tickets covered by u_j special tickets ($1 \leq j \leq z$).
x_i	A binary variable indicating the presence or absence of participant ticket i in the participant's playing set.
$\vec{\mathbf{X}}^{(\ell)}$	A vector representing an overlapping playing set structure of cardinality ℓ.
$x^{(\ell)}_{(t_\ell t_{\ell-1} \ldots t_2 t_1)_2}$	The number of elements in a given compartment in a overlapping playing set structure.
$\vec{\mathbf{Y}}$	An ancestor vector of a vector $\vec{\mathbf{X}}^{\ell}$ in the exhaustive enumeration lottery tree.
y_j	A binary variable indicating whether government ticket j is covered by the participant's playing set.

CHAPTER 1

Introduction

Contents

1.1 Background

A lottery is defined in the Oxford English Dictionary [40] as *"An arrangement for the distribution of prizes by chance among persons purchasing tickets. Slips or lots, numbered in correspondence with the tickets, and representing either prizes or blanks, are drawn from a wheel. Usually intended as a means of raising money for the benefit of the promoters, of the State, or of some charitable institution."*

The history of lotteries dates back to biblical times where in the book of Numbers, Moses is instructed by God to divide land amongst different families by drawing lots. Many other references to drawing lots appear in the Bible. One of the oldest non-biblical records of lotteries taking place dates back to 200 BC—it is documented that Emperor Cheung Leung invented the Chinese Lottery, presently known as Keno [35]. The original purpose of the Chinese lottery was to raise funds for taxes. Funds were also used to support the building of the Great Wall of China. In Europe, one of the oldest records of a lottery was a raffle held by the painter Jan van Eyck, in 1446. As from the year of 1465, lotteries gained popularity in Belgium to such an extent that they were held on a regular basis with the purpose of raising money for the building of houses, caring for the underprivileged, construction of religious buildings, and the construction of much needed water canal systems. This method of raising funds, and awarding of prizes to participants by chance was possibly given its name in Italy. Results from an Italian election were considered highly controversial, and therefore, in a re-election all the candidates names were replaced my numbers. This caused the election of a winning candidate to be completely up to chance. In the Italian language, the word lottery means "unchangeable fate." In 1539

King Francis I of France found his kingdom to be in financial debt, and resorted to running a lottery in order to raise funds. In the year 1567, Queen Elizabeth I established the English lottery in which 400 000 tickets were for sale to the public. Prizes did not only include cash, but also included china and tapestries. Originally, the aim of the English lottery was to raise money for the repairing of harbours. Following the success of the English lottery, the first London lottery was started by King James I in 1612. The funds raised from this lottery were used for the building of the colony of Jamestown, the first English colony in America [12]. Lotteries were also used in many countries to fund cultural activities. In the year 1753, a special lottery was held to raise funds used to build the British Museum. At the same time, it is documented that Casanova, a venetian adventurer and author, urged Louis XV to found the *Loterie Royale* later to be known as the *Loterie National*. This lottery was based on the game Keno, where players could choose one to five numbers between 1 and 90. Lotteries started to gain popularity in America in the 1700s. Benjamin Franklin was able to finance the construction of cannons for the Revolutionary War by using some of the money earned from running various lotteries. It is also documented that George Washington founded a lottery in Virginia in order to finance the construction of roads to the west of that state. The popularity of lotteries grew even more in the United States after the constitution was adopted. The funds raised in those lotteries were used to fund over 300 schools and 200 churches. Universities such as Columbia, Harvard, Princeton and Yale were also built using funds from popular lotteries in America. The game was also used to improve civilian life and this included the building of orphanages, libraries, hospitals, jails and courthouses.

Unfortunately, corruption began to plague lotteries, and big jackpots were often advertised, but no prizes were awarded to participants [35]. This caused the civilians of North American regions to start campaigning for the closure and banning of lotteries. Religious organizations played a significant role in these fights. Together with the prohibition of alcohol, the abolition of slavery and workers' rights, the running of lotteries was one of the most controversial topics of those times. Throughout the remainder of the 19th century and early 20th century, many groups and organisations fought to ban lotteries which had become associated with criminals and moral decay in society. In the year 1819, the Province of Quebec in Canada made lotteries illegal, starting a drastic trend throughout North America and the world. The state of New York quickly followed suit in 1820. By the year 1856 any form of a lottery was a banned practice in Canada. The US Supreme Court eventually prohibited all forms of gambling in 1905.

It took many years for lotteries to be reinstated. The Queensland state lottery of Australia was among the first to be reinstated in 1917. During the next nine decades, lotteries would grow ever more popular throughout the world. During the 1960s, and 1970s, the United States of America and Canada reinstated lotteries, respectively. In 1973, the Olympic Lottery Corporation of Canada was formed with the aim of raising funds for the hosting of the 1976 Olympic Games which were to be held in Montreal. From those years to the present day, strict laws have been put in to place throughout the world in order to make lotteries more reliable and trustworthy to the general public.

Currently, lotteries are present in most major countries throughout the Americas, Europe, Asia, and Australasia. In Africa, only South Africa and Kenya run legal lotteries. In many countries, lotteries are controlled by the governments themselves. The existence of on-line lotteries makes the purchase of tickets more convenient for participants [17]. Some on-line lotteries are operated privately, while some are linked with major well-known lotteries. In the United States, each state has it's own lottery which is operated according to the specific laws of that state. It is common for the funds raised by lotteries in the United States to be donated to the public education system. In Canada, there are two nationwide lotteries in operation. In New Zealand,

the national lottery is controlled by the government and prizes are often small in comparison with other countries' national lotteries. In the United States, participants can choose between an annuity payment or a much smaller than advertised lump sum. In many countries, lottery winnings are not taxed. Many lottery draws are overseen by independent auditors in order to assure participants that no fraudulent activities are involved in the draw. It has been widely reported that people (even non-participants) are tricked into believing they have won a lottery jackpot after receiving a hoax e-mail.

In the South African National Lottery, *Lotto*, a participant forms a playing set of tickets, in which each ticket consists of 6 numbers chosen from a universal set containing 49 numbers. The lottery board then forms the winning ticket by also choosing 6 numbers from the same universal set. A bonus ball is also drawn, which gives participants and additional chance of winning a prize. The more numbers that a participant's ticket has in common with the winning ticket, the higher the financial reward for the participant. A draw takes place every Wednesday and Saturday. The South African National Lottery is a major source of charity to many beneficiaries, such as women in technology, rural women, women entrepreneurs, the youth, non-governmental organizations, and community based organizations [13]. The structure of this lottery implies that there are $\binom{49}{6} = 13\,983\,816$ possible lottery tickets from which the participant may choose. It may therefore be deduced that if a participant purchases one ticket, he/she will stand a $\frac{1}{13\,983\,816} = 0.000\,007\,15\%$ chance of winning the jackpot. In the South African National Lottery, the largest prize from a single winning ticket was R30 352 465 [31], while the largest amount won from a single winning ticket, worldwide, was \$365 000 000 in the United States [43]. In South Africa, a record 33 participants won the jackpot on the 15^{th} of March 2003 [31], and on the 7^{th} of February 2009, 18 players won the jackpot. This led some people to believe that, due to the extremely small probability of such occurences, there were fraudulent activities taking place during or after the lottery draw and some called for an independant investigation to be carried out regarding this issue [14]. Despite criticism from certain sectors of the South African public, the South African National Lottery remains popular according to 2006 research, in which it was found that 82 percent of the population played the lottery once a week [44].

1.2 Problem descriptions

Suppose a lottery consists of forming a *winning ticket* by randomly choosing t distinct numbers from the universal set $\mathcal{U}_m = \{1, \ldots, m\}$. A participant also forms a set of *tickets*, each consisting of n distinct numbers from $\mathcal{U}_m$, and is awarded a prize if k or more numbers in at least one of his/her tickets matches those of the winning ticket. A lottery of this form is denoted by the quadruple $\langle m, n, t, k \rangle$, and the prize is known as a *k-prize*. The participant's set of tickets is also known as a *playing set*.

The participant may wish to form a playing set which yields a probability of at least $0 < \psi \leq 1$ of winning a *k-prize*. Naturally, the participant will want to minimise the cost of forming such a playing set, which means that the cardinality of the playing set should be as small as possible. This combinatorial minimisation problem is known as the *incomplete lottery problem*, introduced in [16]. A related problem is the so-called *resource utilisation problem*, also introduced in [16]. In this problem one attempts to select a (fixed) playing set of cardinality ℓ in such a way that the probability of winning a k-prize is maximised.

These problems may be defined more formally. To this end, suppose $\mathcal{A}$ is a finite set of integers and let $\Phi(\mathcal{A}, a)$ denote the set of all (unordered) subsets of cardinality $a < |\mathcal{A}|$ from the set $\mathcal{A}$. Each ticket $v \in \Phi(\mathcal{U}_m, n)$ in a playing set has a neighbourhood $\mathcal{N}[v] \subseteq \Phi(\mathcal{U}_m, t)$ associated with

it. This neighbourhood is the set of all tickets from $\Phi(\mathcal{U}_m, t)$ which have at least k numbers in common with the ticket v, that is $\mathcal{N}[v] = \{u \in \Phi(\mathcal{U}_m, t) : \Phi(v, k) \cap \Phi(u, k) \neq \emptyset\}$.

Definition 1.1 (Incomplete lottery problem) *A* $(1 - \psi)$-incomplete lottery *set for* $\langle m, n, t, k\rangle$ *is a subset* $\mathcal{L}_\psi(\mathcal{U}_m, n, t, k) \subseteq \Phi(\mathcal{U}_m, n)$ *with the property that*

$$\bigcup_{v \in \mathcal{L}_\psi(\mathcal{U}_m, n, t, k)} \mathcal{N}[v]$$

has cardinality at least $\left\lceil \psi \binom{m}{t} \right\rceil$. *The* incomplete lottery problem *is: What is the smallest possible cardinality of a* $(1 - \psi)$*-incomplete lottery set* $\mathcal{L}_\psi(\mathcal{U}_m, n, t, k)$*? Denote the answer to this problem by the* incomplete lottery number $L_\psi(m, n, t, k)$. *An incomplete lottery set* $\mathcal{L}_\psi(U_m, n, t, k)$ *of minimum cardinality,* $L_\psi(m, n, t, k)$*, is referred to as an* $L_\psi(m, n, t, k)$*-set for the lottery* $\langle m, n, t, k\rangle$. □

Let $\Phi(\mathcal{U}_m, n)$ be known as the set of *participant tickets*, and let $\Phi(\mathcal{U}_m, t)$ be known as the set of *government tickets*. A small instance of the incomplete lottery problem is illustrated in following simple example.

Example 1.1 *Consider the lottery* $\langle 7, 5, 4, 3\rangle$. *The set of all unordered sets from which the winning lottery ticket may be chosen is*

$$\begin{aligned}\Phi(\mathcal{U}_7, 4) = \{&\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\},\{1,2,3,7\},\{1,2,4,5\},\{1,2,4,6\},\{1,2,4,7\},\\ &\{1,2,5,6\},\{1,2,5,7\},\{1,2,6,7\},\{1,3,4,5\},\{1,3,4,6\},\{1,3,4,7\},\{1,3,5,6\},\\ &\{1,3,5,7\},\{1,3,6,7\},\{1,4,5,6\},\{1,4,5,7\},\{1,4,6,7\},\{1,5,6,7\},\{2,3,4,5\},\\ &\{2,3,4,6\},\{2,3,4,7\},\{2,3,5,6\},\{2,3,5,7\},\{2,3,6,7\},\{2,4,5,6\},\{2,4,5,7\},\\ &\{2,4,6,7\},\{2,5,6,7\},\{3,4,5,6\},\{3,4,5,7\},\{3,4,6,7\},\{3,5,6,7\},\{4,5,6,7\}\},\end{aligned}$$

while the set of all unordered sets from which the participant can chose tickets is

$$\begin{aligned}\Phi(U_7, 5) = \{&\{1,2,3,4,5\},\{1,2,3,4,6\},\{1,2,3,4,7\},\{1,2,3,5,6\},\{1,2,3,5,7\},\\ &\{1,2,3,6,7\},\{1,2,4,5,6\},\{1,2,4,5,7\},\{1,2,4,6,7\},\{1,2,5,6,7\},\\ &\{1,3,4,5,6\},\{1,3,4,5,7\},\{1,3,4,6,7\},\{1,3,5,6,7\},\{1,4,5,6,7\},\\ &\{2,3,4,5,6\},\{2,3,4,5,7\},\{2,3,4,6,7\},\{2,3,5,6,7\},\{2,4,5,6,7\},\\ &\{3,4,5,6,7\}\}.\end{aligned}$$

The different participant tickets and the government tickets which have at least 3 *numbers in common with them are shown graphically in Figure 1.1. For different values of* ψ*, the cardinality of an optimal playing set changes. For this lottery, the incomplete lottery numbers* $L_\psi(7, 5, 4, 3) = 1$ *for all* $0 < \psi \leq \frac{25}{35}$, $L_\psi(7, 5, 4, 3) = 2$ *for all* $\frac{25}{35} < \psi \leq \frac{34}{35}$, *and* $L_\psi(7, 5, 4, 3) = 3$ *for all* $\frac{34}{35} < \psi \leq 1$ *may be deduced via a brute force approach.* ■

The resource utilisation problem, mentioned above, is defined formally below.

Definition 1.2 (Resource utilisation problem) *The* resource utilisation *of a playing set* $\mathcal{L}_\ell = \{v_1, \ldots, v_\ell\} \subseteq \Phi(U_m, n)$ *in the lottery* $\langle m, n, t, k\rangle$ *is defined as the proportion*

$$0 < \frac{\left|\bigcup_{i=1}^{\ell} \mathcal{N}[v_i]\right|}{\binom{m}{t}} \leq 1,$$

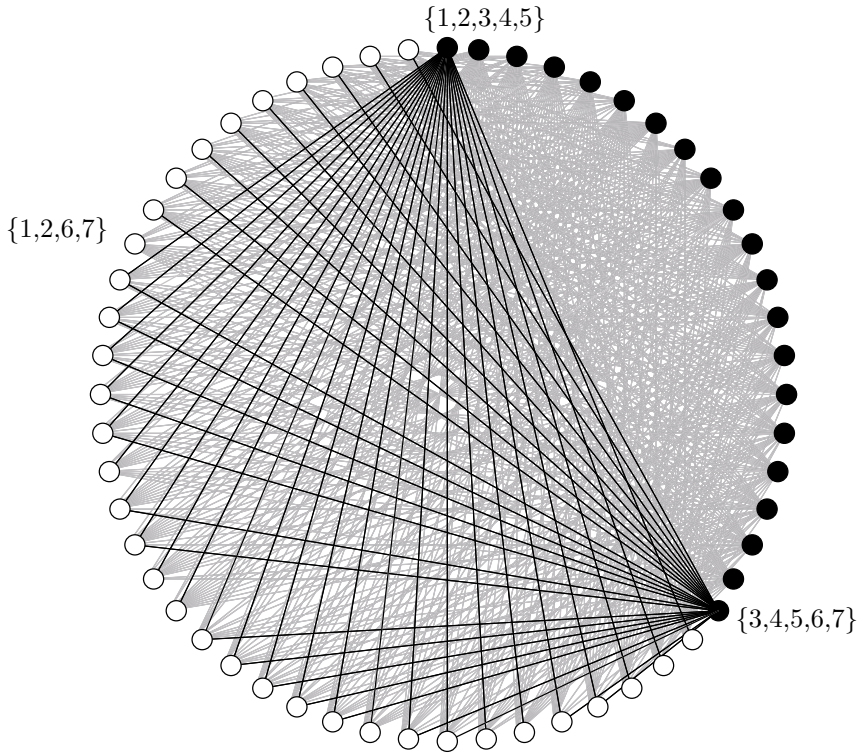


FIGURE 1.1: *Lottery graph representing the lottery* $\langle 7,5,4,3\rangle$. *All the black vertices represent participant tickets, and all the white vertices represent government tickets. A vertex which represents a government ticket is joined to a vertex which represents a participant ticket if the two tickets have at least three numbers in common. It can be seen that if the participant were to select the playing set* $\{\{1,2,3,4,5\},\{3,4,5,6,7\}\}$, 34 *out of the* 35 *government tickets are joined by an edge to at least one of the playing set tickets. The only government ticket which is not joined by an edge to any of the two tickets in the participant's playing set is ticket* $\{1,2,6,7\}$. *If this playing set is chosen by the participant, he/she will have a* $\frac{34}{35} \approx 0.9714$ *probability of winning a 3-prize. Note that in this lottery instance,* $5 = n \neq t = 4$, *which implies that the vertices which represent participant tickets have a higher degree than the vertices which represent government tickets, and that explains why the graph seems more populated with edges nearer to the area where the participant tickets are.*

and therefore the resource utilisation problem *is: Given a fixed playing set cardinality of* $1 \leq \ell \leq L_1(m,n,t,k)$, *what is the maximum resource utilisation that may be achieved by some playing set* $\mathcal{L}_\ell$ *of fixed cardinality* ℓ*? Denote the answer to this problem by the* resource utilisation number,

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

A playing set of cardinality ℓ *that realises this maximum resource utilisation of* $\Psi_\ell(m,n,t,k)$ *is referred to as a* $\Psi_\ell(m,n,t,k)$*-set for* $\langle m,n,t,k\rangle$. □

The notion of the resource utilisation problem is illustrated next.

Example 1.2 *Suppose a participant in the lottery* $\langle 7,5,4,3\rangle$ *wants to buy two tickets and would like to maximise the associated resource utilisation. If all the playing sets of cardinality* 2 *are examined, various resource utilisation values are obtained from which a playing set must be chosen that achieves the maximum value,* $\Psi_2\langle 7,5,4,3\rangle$. *For example, the playing set* $\{\{1,2,3,4,5\},\{1,2,3,4,6\}\}$ *yields a resource utilisation of* $\frac{31}{34}$, *but the playing set*

$\{\{1,2,3,4,5\},\{3,4,5,6,7\}\}$ *yields a resource utilisation of* $\frac{34}{35}$. *By analysing all the possible playing sets it is found that no playing set of cardinality* 2 *yields a resource utilisation larger than* $\frac{34}{35}$, *and hence* $\Psi_2\langle 7,5,4,3\rangle = \frac{34}{35}$. ■

TABLE 1.1: *Solution to the incomplete lottery problem for the lottery* $\langle 7,5,4,3\rangle$ *presented in Example 1.2. The first column contains all the possible values of* ψ. *The second column contains a playing set meeting the desired probability-of-win value in the first column. The third column contains all the government tickets which have* $k = 3$ *or more numbers in commmon with at least one ticket in the playing set in the second column.*

ψ	Smallest playing set	Matching government tickets
$0 < \psi \leq \frac{25}{35}$	$\{\{1,2,3,4,5\}\}$	$\{\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\},\{1,2,3,7\},$ $\{1,2,4,5\},\{1,2,4,6\},\{1,2,4,7\},\{1,2,5,6\},$ $\{1,2,5,7\},\{1,3,4,5\},\{1,3,4,6\},\{1,3,4,7\},$ $\{1,3,5,6\},\{1,3,5,7\},\{1,4,5,6\},\{1,4,5,7\},$ $\{2,3,4,5\},\{2,3,4,6\},\{2,3,4,7\},\{2,3,5,6\},$ $\{2,3,5,7\},\{2,4,5,6\},\{2,4,5,7\},\{3,4,5,6\},$ $\{3,4,5,7\}\}$
$\frac{25}{35} < \psi \leq \frac{34}{35}$	$\{\{1,2,3,4,5\},$ $\{3,4,5,6,7\}\}$	$\{\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\},\{1,2,3,7\},$ $\{1,2,4,5\},\{1,2,4,6\},\{1,2,4,7\},\{1,2,5,6\},$ $\{1,2,5,7\},\{1,3,4,5\},\{1,3,4,6\},\{1,3,4,7\},$ $\{1,3,5,6\},\{1,3,5,7\},\{1,3,6,7\},\{1,4,5,6\},$ $\{1,4,5,7\},\{1,4,6,7\},\{1,5,6,7\},\{2,3,4,5\},$ $\{2,3,4,6\},\{2,3,4,7\},\{2,3,5,6\},\{2,3,5,7\},$ $\{2,3,6,7\},\{2,4,5,6\},\{2,4,5,7\},\{2,4,6,7\},$ $\{2,5,6,7\},\{3,4,5,6\},\{3,4,5,7\},\{3,4,6,7\},$ $\{3,5,6,7\},\{4,5,6,7\}\}$
$\frac{34}{35} < \psi \leq 1$	$\{\{1,2,3,4,5\},$ $\{1,2,3,4,7\},$ $\{3,4,5,6,7\}\}$	$\{\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\},\{1,2,3,7\},$ $\{1,2,4,5\},\{1,2,4,6\},\{1,2,4,7\},\{1,2,5,6\},$ $\{1,2,5,7\},\{1,2,6,7\},\{1,3,4,5\},\{1,3,4,6\},$ $\{1,3,4,7\},\{1,3,5,6\},\{1,3,5,7\},\{1,3,6,7\},$ $\{1,4,5,6\},\{1,4,5,7\},\{1,4,6,7\},\{1,5,6,7\},$ $\{2,3,4,5\},\{2,3,4,6\},\{2,3,4,7\},\{2,3,5,6\},$ $\{2,3,5,7\},\{2,3,6,7\},\{2,4,5,6\},\{2,4,5,7\},$ $\{2,4,6,7\},\{2,5,6,7\},\{3,4,5,6\},\{3,4,5,7\},$ $\{3,4,6,7\},\{3,5,6,7\},\{4,5,6,7\}\}$

Gründlingh [16] established the following result which shows that the two problems in Definitions 1.1 and 1.2 are essentially inverses of each other. Although this proof in [16] is for the three-parameter case where $n = t$, the proof easily generalises to the four-parameter case.

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

Both the incomplete lottery problem and the resource utilisation problem have only recently been introduced into the combinatorial optimisation literature. However, prior to this introduction, a considerable amount of work had been done on the so-called *complete lottery problem* which is the special case of the incomplete lottery problem that arises by taking the win probability $\psi = 1$. See, for example, [27] in this regard. The well-known *covering problem*, and *Turán problem* are, in turn, special instances of the complete lottery problem.

Definition 1.3 (The covering problem) *A* t-$(m, n, 1)$ *covering set* $\mathcal{C} \subseteq \Phi(\mathcal{U}_m, n)$ *has the property that, for any element* $\phi_t \in \Phi(\mathcal{U}_m, t)$*, there exists an element* $c \in \mathcal{C}$ *such that* $\{\phi_t\} \cap \{c\} \neq \emptyset$*. The optimal solution to the covering problem is the smallest possible cardinality of such a covering set* $\mathcal{C}$*. The solution to the problem is denoted by the covering number,* $C(m, n, t)$*.* □

Definition 1.4 (The Turán problem) *The Turán number,* $T(m, n, t)$ *is defined as the smallest number of* n*-subsets of* $\mathcal{U}_m$ *such that any* t*-subset of* $\mathcal{U}_m$ *contains at least one of these* n*-subsets.* □

The lottery problem, with $t = k$, coincides with the covering problem, and similarly, if $n = k$, the lottery problem coincides with the Turán problem. Therefore, $L_1(m, n, t, t) = C(m, n, t)$ and $L_1(m, n, t, n) = T(m, n, t)$.

Consider the following example of the 2-$(5, 3, 1)$ covering problem, which is analogous to solving the $\langle 5, 3, 2, 2\rangle$-lottery problem.

Example 1.3 (The 2-$(5, 3, 1)$ covering problem) *Let* $\mathcal{U}_5 = \{1, 2, 3, 4, 5\}$*, and consider the cover* $\mathcal{C} = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 2, 4\}, \{1, 2, 5\}\}$*. The set of all unordered pairs from* $\mathcal{U}_5$ *is*

$$\mathcal{A} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}.$$

The set $\mathcal{C}$ *is clearly a* 2-$(5, 3, 1)$ *cover, since it contains each element of* $\mathcal{A}$ *as a subset of one of its elements. It may be verified that* $\mathcal{C}$ *is, in fact, a minimum cover, and hence* $C(5, 3, 2) = 4$*.* ■

Consider the following example of the $(5, 2, 3)$ Turán problem, which is analogous to solving the $\langle 5, 2, 3, 2\rangle$-lottery problem.

Example 1.4 (The $(5, 2, 3)$ Turán problem) *Let* $\mathcal{U}_5 = \{1, 2, 3, 4, 5\}$*. The set of all* 2*-subsets of* $\mathcal{U}_5$ *is*

$$\mathcal{A} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}.$$

The set of all 3*-subsets of* $\mathcal{U}_5$ *are*

$$\begin{aligned}\mathcal{B} \;=\; & \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\},\\ & \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}\}.\end{aligned}$$

Let $\mathcal{A}' \in \mathcal{A}$ *be the smallest set of* 2*-subsets of* $\mathcal{U}_5$ *such that any* 3*-subset of* $\mathcal{U}_5$ *contained in* $\mathcal{B}$ *contains at least one of these* 2*-subsets. Then* $|\mathcal{A}'| = 4$ *and such a minimal set of* 2*-subsets of* $\mathcal{U}_5$ *is* $\mathcal{A}' = \{\{1, 2\}, \{4, 5\}, \{2, 3\}, \{1, 3\}\}$*, which yields the Turán number* $T(5, 2, 3) = 4$*.* ■

Bounds on the complete lottery number are established in the following theorem in terms of the covering and Turán numbers.

Theorem 1.1 (Covering and Turán numbers as bounds [39])

$$\frac{T(m, k, t)}{\binom{n}{k}} \leq L_1(m, n, t, k) \leq C(m, n, t)$$

for all $m \geq \{n, t\} \geq k \geq 1$*.* □

Consider the following small example illustrating Theorem 1.1 for a small instance of the lottery problem.

Example 1.5 *Consider the lottery* $\langle 9,4,5,3\rangle$. *The covering number for* $C(9,4,3) = 25$ *is an upper bound on* $L_1(9,3,5,3)$. *The Turán number* $T(9,3,5) = 12$ *divided by* $\binom{4}{3} = 4$ *is a lower bound on* $L_1(9,3,5,2)$. *Hence* $3 \leq L_1(9,3,5,3) \leq 25$ *is a lower bound on* $L_1(9,3,5,3)$. *It is interesting to note how much closer the bound based on the Turán number is to the complete lottery number of* $L_1(9,3,5,3) = 5$ *than the bound based on the covering number.* ■

The following is a list of growth properties for the complete lottery problem [22].

Theorem 1.2 (Growth properties of $L_1(m,n,t,k)$)
(a) $L_1(m,n,t,k) \leq L_1(m+1,n,t,k)$.
(b) $L_1(m,n,t,k) \geq L_1(m,n,t+1,k)$.
(c) $L_1(m,n,t,k) \geq L_1(m+1,n+1,t,k)$.
(d) $L_1(m,n,t,k) \geq L_1(m+1,n+1,t+1,k)$.
(e) $L_1(m,n,t,k) \geq L_1(m,n+1,t,k)$.
(f) $L_1(m,n,t,k) \leq L_1(m,n,t,k+1)$.
(g) $L_1(m,n,t,k) \leq L_1(m+1,n+1,t+1,k+1)$.
(h) $L_1(m,n,t,k) \leq L_1(m+1,n+1,t,k+1)$.
(i) $L_1(m,n,t,k) \leq L_1(m+1,n,t,k+1)$.
(j) $L_1(m,n,t,k) \leq L_1(m,n+1,t,k+1)$.
(k) $L_1(m,n,t,k) \leq L_1(m,n,t+1,k+1)$.
(l) $L_1(m,n,t,k) \leq L_1(m+1,n,t+1,k+1)$.
(m) $L_1(m,n,t,k) \geq L_1(m+1,n,t+1,k)$.
(n) $L_1(m,n,t,k) \geq L_1(m,n+1,t+1,k)$. □

For the incomplete lottery problem and the resource utilisation problem, the following bounds were established in [6].

Theorem 1.3 (Growth properties of $L_\psi(m,n,t,k)$ and $\Psi_\ell(m,n,t,k)$)
(a) $L_\psi(m',n,t,k) \leq L_\psi(m,n,t,k)$ *for all* $1 \leq k \leq \{n,t\} \leq m' < m$ *and* $0 < \psi \leq 1$.
(b) $L_\psi(m,n',t,k) \geq L_\psi(m,n,t,k)$ *for all* $1 \leq k \leq n' < \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(c) $L_\psi(m,n,t',k) \geq L_\psi(m,n,t,k)$ *for all* $1 \leq k \leq t' < \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(d) $L_\psi(m,n,t,k') \leq L_\psi(m,n,t,k)$ *for all* $1 \leq k' < k \leq \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(e) $L_{\psi'}(m,n,t,k) \leq L_\psi(m,n,t,k)$ *for all* $1 \leq k \leq \{n,t\} \leq m$ *and* $0 < \psi' < \psi \leq 1$.
(f) $\Psi_\ell(m',n,t,k) \geq \Psi_\ell(m,n,t,k)$ *for all* $1 \leq k \leq \{n,t\} \leq m' < m$ *and* $0 < \psi \leq 1$.
(g) $\Psi_\ell(m,n',t,k) \leq \Psi_\ell(m,n,t,k)$ *for all* $1 \leq k \leq n' < \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(h) $\Psi_\ell(m,n,t',k) \leq \Psi_\ell(m,n,t,k)$ *for all* $1 \leq k \leq t' < \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(i) $\Psi_\ell(m,n,t,k') \geq \Psi_\ell(m,n,t,k)$ *for all* $1 \leq k' < k \leq \{n,t\} \leq m$ *and* $0 < \psi \leq 1$.
(j) $\Psi_{\ell'}(m,n,t,k) \leq \Psi_\ell(m,n,t,k)$ *for all* $1 \leq k \leq \{n,t\} \leq m$ *and* $0 < \ell' < \ell \leq 1$. □

The example which follows illustrates the use of the above-mentioned theorems for the case where the value of n changes.

Example 1.6 (Incomplete lottery bounds — A change in the value of n) *Consider the lottery* $\langle 10,n,3,2\rangle$, *for the values of* $n = 4$, *and* $n = 5$. *If* $n = 4$, *an incomplete lottery number is* $L_{0.5}(10,4,3,2) = 2$, *while the complete lottery number is* $L_1(10,4,3,2) = 4$. *If*

$n = 5$, an incomplete lottery number is $L_{0.5}(10, 5, 3, 2) = 1$, while the complete lottery number is $L_1(10, 5, 3, 2) = 2$. This result complies with Theorem 1.2(e) and Theorem 1.3(b). It may also be seen that $L_{0.5}(10, 5, 3, 2) < L_1(10, 5, 3, 2)$, which complies with Theorem 1.3(e). ■

1.3 Two kinds of lottery isomorphisms

In this section, the well-known notions of an isomorphism between lotteries and of an isomorphism between playing sets are reviewed. These notions have a significant impact on analyses of both the incomplete lottery problem and the resource utilisation problem. The notion of an isomorphism between lotteries implies that there are some lotteries which, although they have different parameters, are structurally the same. This means that if one of the isomorphic lotteries has been studied in terms of its combinatorial properties, those properties will also hold for lotteries which are isomorphic to the lottery concerned. Similarly, the concept of isomorphic playing sets implies that it is possible for two solutions to the incomplete lottery and resource utilisation problems to be distinct, yet structurally equivalent in terms of elements of the universal set $\mathcal{U}_m$.

1.3.1 Isomorphic lotteries

A lottery ticket may be identified uniquely by either specifying which elements of $\mathcal{U}_m$ appear in the ticket, or by specifying which elements of $\mathcal{U}_m$ do not appear in the ticket. In the lottery $\langle m, n, t, k\rangle$ a playing set ticket may therefore be identified uniquely by specifying either n numbers or by specifying $m - n$ numbers. Similarly the winning ticket is identified uniquely by specifying either t numbers or by specifying $m - t$ numbers. This dual view of lottery ticket identification gives rise to a pair of equivalent lotteries, called *isomorphic lotteries* and denoted by writing $\langle m, n, t, k\rangle \cong \langle m, m - n, m - t; m + k - n - t\rangle$, as formalised in the following result.

Theorem 1.4 (Isomorphic lotteries [16]) *If $m + k \geq n + t$, then*
(i) $L_\psi(m, n, t, k) = L_\psi(m, m - n, m - t, m + k - n - t)$ for all $0 < \psi \leq 1$, and
(ii) $\Psi_\ell(m, n, t, k) = \Psi_\ell(m, m - n, m - t, m + k - n - t)$ for all $1 \leq \ell \leq L_1(m, n, t, k)$. □

In the following example it is illustrated how $\langle 5, 3, 2, 2\rangle \cong \langle 5, 2, 3, 2\rangle$.

Example 1.7 *Figure 1.2 is a graphical representation of the lotteries $\langle 5, 3, 2, 2\rangle$ and $\langle 5, 2, 3, 2\rangle$ in which tickets are represented by vertices, and in which two vertices are adjacent if the corresponding tickets have two or more numbers in common. In the lottery $\langle 5, 3, 2, 2\rangle$, the black vertices represent the participant tickets, while the white vertices represent the government tickets and similarly, in the lottery $\langle 5, 2, 3, 2\rangle$, the white vertices in Figure 1.2 represent the participant tickets, while the black vertices represent the government tickets. It may be seen that each black vertex has the same degree as each white vertex. This property enables the black and white vertex sets to swap participant and government roles in the lottery problem. Hence $\langle 5, 3, 2, 2\rangle \cong \langle 5, 2, 3, 2\rangle$ from which it follows that $L_\psi(5, 3, 2, 2) = L_\psi(5, 2, 3, 2)$ and $\Psi_\ell(5, 3, 2, 2) = \Psi_\ell(5, 2, 3, 2)$ for all $0 < \psi \leq 1$ and all $\ell = 1, \ldots, 4 = L_1(5, 3, 2, 2) = L_1(5, 2, 3, 2)$.* ■

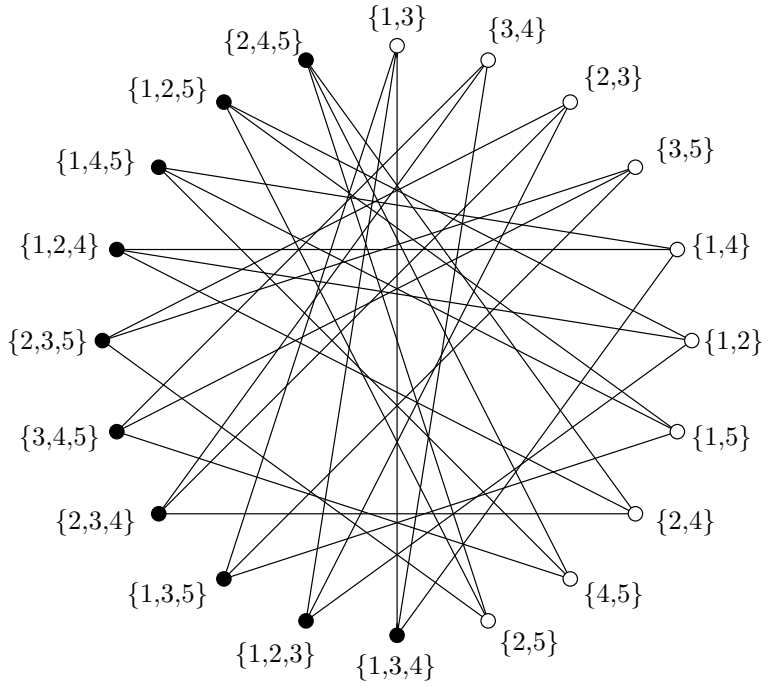


FIGURE 1.2: *Graphic representations of the lotteries* $\langle 5,3,2,2\rangle$ *and* $\langle 5,2,3,2\rangle$*. For the lottery* $\langle 5,3,2,2\rangle$*, the black vertex set represents the participant tickets, and the white vertex set represents the government tickets. For the lottery* $\langle 5,2,3,2\rangle$*, the white vertex set represents the participant tickets, and the black vertex set represents the government tickets.*

1.3.2 Isomorphic playing sets

As mentioned previously, the existence of isomorphic playing sets imply that it is possible for two solutions to either the incomplete lottery problem or the resource utilisation problem to be visually distinct, yet structurally similar as a result of the ubiquity of the symbols attached to members of the universal set $\mathcal{U}_m$. This raises the question of how many structurally different optimal solutions to the lottery and resource utilisation problems there might be.

Definition 1.5 (The overlapping structure of a playing set) *A playing set of cardinality* ℓ *partitions the universal set* $\mathcal{U}_m$ *into* 2^ℓ *subsets or compartments, known as the inclusion-exclusion compartments. All the elements of* $\mathcal{U}_m$ *that are unique to ticket* 1 *appear in one compartment, all elements of* $\mathcal{U}_m$ *that are unique to ticket* 2 *appear in another compartment, all elements of* $\mathcal{U}_m$ *that are members of both tickets* 1 *and* 2 *appear in a third compartment, and so on.* □

An overlapping playing set structure allows one to identify which tickets share numbers with each other, and which tickets contain numbers which are unique to those tickets, irrespective of the symbols attached to elements of these tickets (numbers in the traditional sense). The notion of an overlapping playing set structure is illustrated in Figure 1.3. The following definition is taken from [16].

Definition 1.6 (The lottery characterisation number) *Let the set* $\mathcal{L}_\psi = \{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \ldots, \mathcal{L}^{(\eta_\psi(m,n,t,k))}\}$ *contain all structurally different playing sets for the lottery* $\langle m,n,t,k\rangle$ *(where* $1 \leq k \leq \{n,t\} \leq m$*,* $0 < \psi \leq 1$ *and* $1 \leq \ell \leq L_1(m,n,t,k)$*). Then, we refer to* $\eta_\psi(m,n,t,k) = |\mathcal{L}_\psi|$ *as the lottery characterisation number for the lottery* $\langle m,n,t,k\rangle$*.* □

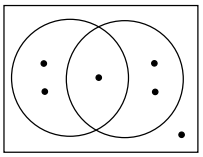


FIGURE 1.3: *An example of an overlapping playing set structure for a lottery in which* $m = 6$ *and* $n = 3$*. There is one number in the compartment shared by neither ticket* 1 *nor ticket* 2*, two numbers in the compartment exclusive to ticket* 1*, two numbers in the compartment exclusive to ticket* 2*, and one number in the compartment shared by ticket 1 and ticket 2. Two distinct playing sets conforming to this structure are* $\{\{1,2,3\},\{3,4,5\}\}$ *and* $\{\{1,3,6\},\{2,4,6\}\}$*.*

The following example illustrates the notion of a lottery characterisation number for the lottery $\langle 10,5,4,3\rangle$.

Example 1.8 (The lottery characterisation number of the lottery $\langle 10,5,4,3\rangle$**)** *The only two overlapping playing set structures for the lottery* $\langle 10,5,4,3\rangle$ *are shown in Figure 1.4. An example of a playing set conforming to the overlapping playing set structure on the left is* $\{\{1,2,3,4,5\},\{1,6,7,8,9\}\}$*. It may be seen that both tickets share the number* 1*, that the number* 10 *does not appear in either ticket and that the other numbers* $(2,3,4,5,6,7,8,9)$ *are each unique to a specific ticket. An example of a playing set conforming to the overlapping playing set structure on the right is* $\{\{1,2,3,4,5\},\{6,7,8,9,10\}\}$*. For either of these overlapping playing set structures,* $\Psi_2(10,5,4,3) = \frac{110}{210}$*. Consequently,* $\eta_{\frac{110}{210}}(10,5,4,3) = 2$*.* ■

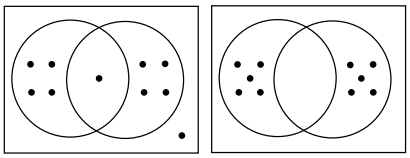


FIGURE 1.4: *The two different optimal overlapping playing set structures of the lottery* $\langle 10,5,4,3\rangle$ *where* $\ell = 2$*, resulting in* $\eta_{\frac{110}{210}}(10,5,4,3) = 2$*.*

1.4 Scope and objectives

The objectives pursued in this thesis are:

I (a) To *model* lotteries as bipartite graphs in order to graphically represent lottery instances in which $n \neq t$.

(b) To *establish* new bounds on the incomplete lottery number from the bipartite graph representation of lotteries in Objective I(a) above, where $n \neq t$.

(c) To *compare* the new bounds to known bounds associated with small and large lottery problem instances documented in [16].

II (a) To *formulate* the incomplete lottery problem and the resource utilisation problem as integer programming problems.

(b) To *implement* the formulations in II(a) above in a suitable, commercially available mathematical programming solver and to verify the correctness of these implementations by comparing results thus obtained with results documented in [16].

(c) To *investigate* the limits of applicability of the implementations in II(a) above, by exploring the execution time of solving small problem instances in parallel.

(d) To *use* the implementation in II(b) above to *analyse* those small lotteries identified in II(c) above by *solving* the incomplete lottery problem (for the decimal values $\psi = 0.1, 0.2, 0.3, \ldots, 1$) and the resource utilisation problem exactly (taking into account the lottery isomorphisms described in §1.3.1), and to *document* these solutions in an accessible manner.

III (a) To *re-implement* the brute force (backtracking) enumeration method in [16], as described in §1.3.

(b) To *verify* the correctness of the implementation in III(a) above by comparing the results obtained for small lotteries via the enumeration method with the results documented in [16].

(c) To *validate* the results in Objective II(d) above by comparing those results with the results obtained via the enumeration method in III(a)–(b) above.

(d) To *use* the implementation in III(b) above to find all structurally non-isomorphic optimal solutions to the largest possible subset of problem instances identified in Objective II(c)–(d).

IV (a) To *formulate* a combinatorial optimisation problem involving the distribution of the elements of the overlapping playing set structures investigated in Objectives III(a)–(d), amongst the inclusion-exclusion principle compartments into which it is separated, as an alternative to the problem of constructing different playing sets from all the possible tickets in the lottery.

(b) To *implement* heuristic methods which may be used in an attempt to solve the problem formulated in Objective IV(a) above and to find good approximate solutions to the incomplete lottery problem and the resource utilisation problem.

(c) To *apply* the heuristic methods to the same small lottery instances investigated in Objectives II(d) and III(c).

(d) To *compare* the results obtained in Objective IV(c) above to the results obtained via the method in Objective III(a) in terms of execution time and practical feasibility.

V (a) To *compute* (analytically) the probability of multiple winners in a given lottery, in which the numbers in the winning government ticket are drawn randomly according to a uniform distribution and the numbers chosen by participants to be included in their playing sets are also randomly chosen according to a uniform distribution.

(b) To *compute* (by means of simulation) the probability of multiple winners in a given lottery, in which the numbers in the winning government ticket are drawn randomly according to a uniform distribution and the numbers chosen by participants to be included in their playing set tickets are randomly chosen according to a uniform distribution.

(c) To *compute* (by means of simulation) the probability of multiple winners in a given lottery, in which the numbers in the winning government ticket are drawn randomly according to a uniform distribution and the numbers chosen by some proportion of participants to be in their playing set tickets are *not* randomly chosen, but conform to some pre-defined structure of player preference.

1.5 Thesis organisation

Chapter 2 of this thesis contains a concise literature review on the complete lottery problem, the incomplete lottery problem, and the resource utilisation problem. Various bounds on the complete lottery number, the incomplete lottery number, and the resource utilisation number are presented. Exact formulas of the complete lottery number, the incomplete lottery number, and the resource utilisation number for certain special lottery instances are also presented. The chapter also contains a concise summary of solution methods adopted in the literature for the above-mentioned problems.

Chapter 3 contains the formulation of a new bound on the incomplete lottery number for lotteries in which $n \neq t$. Known bounds from graph domination are presented and computed for graphs in which $n = t$; a process analogous to computing bounds on the complete lottery number. An upper bound on the cardinality of a so-called covering of a bipartite graph is adapted and used to establish a new bound on the incomplete lottery number. The newly established bound is then compared to known bounds on the complete lottery number. Finally, the new bound on the incomplete lottery number is computed for small lottery instances and the results are analysed.

Mathematical programming approaches towards solving the incomplete lottery problem and resource utilisation problem are presented in Chapter 4. The results obtained by solving these mathematical programming problems are analysed in terms of execution time and solutions obtained as a function of the possible values of the parameter ψ in the incomplete lottery problem, and the parameter ℓ in the resource utilisation problem. An investigation is then conducted with respect to the boundaries of feasibility of these mathematical programming approaches when solving the incomplete lottery problem and the resource utilisation problem in parallel.

In Chapter 5, a previously established exhaustive enumeration lottery tree solution method [5, 16] is reviewed. The working of this solution method is described in detail. An implementation is presented and applied to the same small problem instances explored in Chapter 4 in order to compare the mathematical programming solutions to those of the exhaustive enumeration lottery tree solution method in terms of execution time and practical feasibility.

The overlapping playing set structure of a solution to the incomplete lottery problem and the resource utilisation problem is further explored in Chapter 6. Methods of distributing the elements of an overlapping playing set structure amongst its different compartments are explored in order to find good approximate solutions to the incomplete lottery problem and the resource utilisation problem in shorter execution time frames than required by the previous methods.

In Chapter 7, the expected number of winners in a lottery draw is investigated. Special focus is afforded to the probability of 18 or more participants winning concurrently in the lottery $\langle 49, 6, 6, 6 \rangle$ due to occurrences of such extreme events in the recent history of the South African National lottery, *Lotto*. This probability is computed under the assumption that participants select the numbers in their playing set tickets according to a non-uniform random distribution. The results of the investigation are documented and analysed. A similar investigation is conducted for the same small lottery instances investigated in Chapters 4 and 5.

Chapter 8 is the conclusion of the thesis. It contains a brief summary of work contained within as well as a number of ideas with respect to possible future work.

CHAPTER 2

Literature Review

Contents

In this chapter, a concise summary is presented of known work on the complete lottery problem, the incomplete lottery problem, and the resource utilisation problem. The earliest work, based only on the complete lottery problem is presented in the first section, and in the section that follows, more recent work (on the incomplete lottery problem and the resource utilisation problem) is presented.

2.1 The complete lottery problem

In this section, a summary of past work on the complete lottery problem is presented. In general, $L_1(m, n, t, k)$ is a nondecreasing function under any one of the following conditions: if m increases, n decreases, t decreases, or k increases (see Theorem 1.2). Furthermore, $L_1(m, n, t, k)$ is a nondecreasing function under any one of the following conditions: if m and n both increase, m and t both increase, n and k both increase, or t and k both increase, and $L_1(m, n, t, k)$ is also a non-decreasing function if: m, n and t decrease, or m, n and k increase or m, t and k increase. Finally, $L_1(m, n, t, k)$ is a nondecreasing function if m, n, t, k all increase [27].

2.1.1 Results from graph theory

A lottery may be modelled as a graph, known as a *lottery graph* [16]. The vertices of the graph represent the tickets in the lottery, and two vertices are adjacent if the tickets which they represent have a subset of cardinality at least k in common. The degree of each vertex in the graph is

$$r = \sum_{i=k}^{n-1} \binom{n}{i}\binom{m-n}{n-i} \tag{2.1}$$

for all $1 \leq k \leq n \leq m$. Gründlingh [16] used this lottery graph representation to formulate and prove a theorem characterising small values of $L_1(m, n, n, k)$ and designed an algorithm for representing a lottery graph graphically, exposing its symmetric nature. Graphs of small lottery instances are plotted in [16], and values of $L_\psi(m, n, t, k)$ are determined through an analysis of these graphs. By using known properties of lottery graphs, the previously unknown value $L_1(10, 6, 6, 5) = 14$ was also established by Gründlingh.

2.1.2 Bounds on $L_1(m, n, t, k)$

As mentioned in Chapter 1, the well-known *covering problem* and *Turán problem* are analogous to the complete lottery problem for the lotteries $\langle m, n, t, t\rangle$ and $\langle m, n, t, n\rangle$, respectively. Therefore, $L_1(m, n, t, t) = C(m, n, t)$, the covering number, and $L_1(m, n, t, n) = T(m, n, t)$, the Turán number. Li & van Rees [24] derived the bound

$$L_1(m, n, t, k) \geq \frac{\binom{n}{p}}{\sum_{i=k}^{\min(n,t)} \binom{n}{i}\binom{m-n}{t-i}} \tag{2.2}$$

from work on covering designs by Nurmela & Ostergard [33]. Also in [24], further lower bounds are stated which are derived from Turán numbers. De Caen [7] showed that

$$T(m, t, k) \geq \frac{\binom{m}{k}}{\binom{t-1}{k-1}} \frac{m-t+1}{m-k+1} \tag{2.3}$$

while it is known [3, 4, 39] that

$$L_1(m, n, t, k) \geq \frac{T(m, t, k)}{\binom{n}{k}}. \tag{2.4}$$

From (2.3) and (2.4), it follows that

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

Schönheim's [38] lower bound for lottery designs states that

$$L_1(m, n, t, t) \geq \left\lceil \frac{m}{n} L_1(m-1, n-1, t-1, t-1) \right\rceil, \tag{2.5}$$

while Li & van Rees [24] also established the bounds

$$L_1(m, n, k+1, k) \geq \min\{L_1(m, n, k-1, k-1), L_1(m-k+1, n-k+2, 2, 2)\}, \tag{2.6}$$

$$L_1(m,n,k+2,k) \geq \min\{L_1(m,n,k,k-1), L_1(m-k,n-k+2,2,2)\}, \tag{2.7}$$

$$L_1(m,n,6,3) \geq \min\{L_1(m,n,4,2), L_1(m-4,n-1,2,2)\}. \tag{2.8}$$

They also state that if $n \geq 3$, then $L_1(2n+1,n,5,3) \geq 5$, $L_1(2n+2,n,6,3) \geq 4$, $L_1(3n+1,n,7,3) \geq 6$, $L_1(n+2,n,4,3) \geq 3$, $L_1(3n+2,n,8,3) \geq 5$, and if $n \geq 4$, then $L_1(2n+3,n,9,4) \geq 4$, $L_1(3n+2,n,11,4) \geq 5$, $L_1(3n+3,n,12,4) \geq 5$, and if $n \geq 5$, then $L_1(2n+4,n,12,5) \geq 4$ [24].

The bounds presented in Theorem 1.2 were established by Li & van Rees [25] and by Gründlingh [16].

The bounds

$$\frac{\binom{m}{k}}{\binom{t-1}{k-1}\binom{n}{k}} \cdot \frac{m-t+1}{m-k+1} \leq L_1(m,n,t,k) \leq \left\lceil \frac{\min\{\binom{m}{t},\binom{m}{k}\}}{\lfloor \frac{n}{k} \rfloor} \right\rceil \tag{2.9}$$

are well known [27], while the generalised Schönheim bound,

$$L_1(m,n,t,k) \geq \left\lceil \frac{\binom{m}{n-k+1} L_1(m-n+k-1,n,t,k)}{\binom{m}{n-k+1} - \binom{n}{n-k+1}} \right\rceil, \tag{2.10}$$

was presented by Li & van Rees [24] as well as in [27].

It is also mentioned in *The handbook of combinatorial designs* [27] that if $m = m_1 + m_2$ and $t = t_1 + t_2 - 1$, then $L_1(m,n,t,k) \leq L_1(m_1,n,t_1,k) + L_1(m_2,n,t_2,k)$ [27]. Another bound by Li & van Rees [25, 27] is

$$L_1(m,n,t,k) \leq L_1(m_1,k,t-r,k) + L_1(m_1,n_1,t-r-1,t-r-1), \tag{2.11}$$

if it is supposed that m, n, t, k, m_1, n_1 and r are integers satisfying $m_1 < m$, $t - r \geq k$, $n_1 \geq k - r - 1$ and $n_1 = n - m + m_1$.

In 1999, Li & van Rees [26] derived a recursive upper bound on the complete lottery number for the lottery $\langle m,n,t,k \rangle$, namely

$$L_1(m,n,t,k) \leq L_1(m-1,n-1,t-1,k-1) + L_1(m-1,n,t,k). \tag{2.12}$$

2.1.3 Known values of $L_1(m,n,t,k)$

In 1978, Bate [3] presented the formula

$$L_1(m,n,t,1) = \left\lceil \frac{m-t+1}{n} \right\rceil, \tag{2.13}$$

which may be used to compute the complete lottery number for any lottery instance in which $k = 1$. He also showed that

$$L_1(m,2,t,2) = \left\lceil \frac{m^2 - (t-1)m}{2(t-1)} \right\rceil \tag{2.14}$$

and that

$$L_1(m,3,3,2) = \begin{cases} \left\lceil \frac{m^2-2m}{12} \right\rceil & \text{if } m \equiv 2,4,6 \pmod{12}, \\ \left\lceil \frac{m^2-2m}{12} + 1 \right\rceil & \text{if } m \equiv 0,8,10 \pmod{12}, \\ \left\lceil \frac{m^2-m}{12} \right\rceil & \text{if } m \equiv 1,3,5,7 \pmod{12}, \\ \left\lceil \frac{m^2-m}{12} + 1 \right\rceil & \text{if } m \equiv 9,11 \pmod{12}. \end{cases} \tag{2.15}$$

In addition, Li & van Rees [24] showed that

$$L_1(m,n,t,k) = r+1 \tag{2.16}$$

if $m-n \geq t-k+1$, where $r = \left\lfloor \frac{t}{t-k+1} \right\rfloor$.

The following theorem was established by Burger, *et al.* [5, 27].

Theorem 2.1 *For all* $1 \leq k \leq \{n,t\} \leq m$,
(a) $L_1(m,n,t,k) = 1$ *if and only if* $n+t \geq m+k$.
(b) $L_1(m,n,t,k) = 2$ *if and only if* $2k-1+\max\{m-2n,\ 0\} \leq t \leq m+k-n-1$.
(c) $L_1(m,n,t,k) = 3$ *if and only if* $t \leq \min\{2k-2+\max\{m-2n,\ 0\}, n-k+t-1\}$ *and*

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

Table 2.1 contains known values of $L_1(m,n,n,k)$ where $2 \leq m \leq 10$ [16]. The values $L_1(10,4,4,3)$, $L_1(10,5,5,4)$ and $L_1(10,6,6,4)$ were all established by Gründlingh [16]. Many more known values of $L_1(m,n,n,k)$ may be found in [5, 6, 20, 22, 25, 27] and also in the online lottery repository [23].

2.1.4 An ILP approach to the complete lottery problem

Jans & Degraeve [20] proposed an Interger Linear Programming (ILP) formulation of the complete lottery problem for lotteries of the form $\langle m,n,n,k \rangle$. They considered lottery instances in which $5 \leq m \leq 16$, $n \in \{4,5,6\}$, and $k < n < m$. Let

$$x_j = \begin{cases} 1 & \text{if participant ticket } j \text{ is in the playing set} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

be a binary decision variable, and $\vec{x} = [x_1, \ldots, x_{\binom{m}{n}}]$. Jans & Degraeve constructed a playing set $\mathcal{L}_\psi(m,n,t,k)$ from the set $\Phi(\mathcal{U}_m, n)$ subject to the constraint that at least one participant ticket j should have at least k numbers in common with the winning ticket drawn randomly from the set $\Phi(\mathcal{U}_m, t)$. Let the neighbourhood, $\mathcal{N}[i]$, of a participant ticket i be the set of government tickets from $\Phi(\mathcal{U}_m, t)$ which have at least k numbers in common with the participant ticket. Suppose $\mathbf{A}$ is a $\binom{m}{t} \times \binom{m}{n}$ matrix whose (i,j)-th entry is

$$a_{ij} = \begin{cases} 1 & \text{if government ticket } i \text{ is in the neighbourhood of participant ticket } j \\ 0 & \text{otherwise.} \end{cases}$$

In the complete lottery problem, the union of the neighbourhoods of the tickets in an optimal playing set must equal the entire set of t-subsets of $\mathcal{U}_m$, *i.e.*, $\bigcup_{i \in \mathcal{L}_\phi(m,n,t,k)} \mathcal{N}[i] = \Phi(\mathcal{U}_m, t)$. Jans

k \ n	1	2
1	2	1
2	–	1

(a) $L_1(2, n, n, k)$

k \ n	1	2	3
1	3	1	1
2	–	1	1
3	–	–	1

(b) $L_1(3, n, n, k)$

k \ n	1	2	3	4
1	4	2	1	1
2	–	6	1	1
3	–	–	1	1
4	–	–	–	1

(c) $L_1(4, n, n, k)$

k \ n	1	2	3	4	5
1	5	2	1	1	1
2	–	10	2	1	1
3	–	–	10	1	1
4	–	–	–	5	1
5	–	–	–	–	1

(d) $L_1(5, n, n, k)$

k \ n	1	2	3	4	5	6
1	6	3	2	1	1	1
2	–	15	2	1	1	1
3	–	–	20	3	1	1
4	–	–	–	15	1	1
5	–	–	–	–	1	1
6	–	–	–	–	–	1

(e) $L_1(6, n, n, k)$

k \ n	1	2	3	4	5	6	7
1	7	3	2	1	1	1	1
2	–	21	4	2	1	1	1
3	–	–	35	4	1	1	1
4	–	–	–	35	3	1	1
5	–	–	–	–	21	1	1
6	–	–	–	–	–	7	1
7	–	–	–	–	–	–	1

(f) $L_1(7, n, n, k)$

k \ n	1	2	3	4	5	6	7	8
1	8	4	2	2	1	1	1	1
2	–	28	5	2	1	1	1	1
3	–	–	56	6	2	1	1	1
4	–	–	–	70	5	1	1	1
5	–	–	–	–	56	4	1	1
6	–	–	–	–	–	28	1	1
7	–	–	–	–	–	–	8	1
8	–	–	–	–	–	–	–	1

(g) $L_1(8, n, n, k)$

k \ n	1	2	3	4	5	6	7	8	9
1	9	4	3	2	1	1	1	1	1
2	–	36	7	2	2	1	1	1	1
3	–	–	84	9	2	1	1	1	1
4	–	–	–	126	9	3	1	1	1
5	–	–	–	–	126	7	1	1	1
6	–	–	–	–	–	84	4	1	1
7	–	–	–	–	–	–	36	1	1
8	–	–	–	–	–	–	–	9	1
9	–	–	–	–	–	–	–	–	1

(h) $L_1(9, n, n, k)$

k \ n	1	2	3	4	5	6	7	8	9	10
1	10	5	3	2	2	1	1	1	1	1
2	–	45	8	3	2	1	1	1	1	1
3	–	–	120	14	2	2	1	1	1	1
4	–	–	–	210	14	3	1	1	1	1
5	–	–	–	–	252	14	3	1	1	1
6	–	–	–	–	–	210	8	1	1	1
7	–	–	–	–	–	–	120	5	1	1
8	–	–	–	–	–	–	–	45	1	1
9	–	–	–	–	–	–	–	–	10	1
10	–	–	–	–	–	–	–	–	–	1

(i) $L_1(10, n, n, k)$

TABLE 2.1: *Known values of the complete lottery number $L_1(m, n, n, k)$ for small lottery instances, where $2 \leq m \leq 10$.*

& Degraeve found an optimal playing set by adopting a *set covering problem* ILP formulation approach in which the objective is to

$$\text{minimise} \quad P(\boldsymbol{x}) = \sum_{1 \leq j \leq \binom{m}{n}} x_j \tag{2.18}$$

subject to

$$\sum_{1 \leq j \leq \binom{m}{n}} a_{ij} x_j \geq 1, \qquad 1 \leq i \leq \binom{m}{t} \tag{2.19}$$

$$x_j \in \{0, 1\}, \qquad 1 \leq j \leq \binom{m}{n}. \tag{2.20}$$

The objective in (2.18) is to minimise the number of tickets in the participant's playing set. Constraint set (2.19) ensures that each government ticket is covered by at least one participant ticket. Constraint set (2.20) enforces the binary nature of the decision variables in (2.17).

Jans & Degraeve noted that (somewhat superfluously) the ILP approach is valuable because it allows a participant in the a lottery draw to know exactly how many tickets to select *and* it provides participants with an indication of the specific numbers to select in their playing set tickets. They also claimed that their ILP approach may be extended to lotteries of the form $\langle m, n, t, k \rangle$.

Li & van Rees [25] noted that although it is easy to formulate the complete lottery problem into an ILP, their attempt to solve for $L_1(m, n, t, k)$ using an ILP approach was met with little success, and suggest that a parallel approach may offer an improvement.

2.2 The incomplete lottery and resource utilisation problems

This section contains a summary of known work on the incomplete lottery problem and the resource utilisation problem. In general, it is known that $L_\psi(m, n, t, k)$ is a non-decreasing function if m increases, n decreases, t decreases, k increases, or ψ increases, and $\Psi_\ell(m, n, t, k)$ is a non-decreasing function if m decreases, n increases, t increases, k decreases or ℓ increases (See theorem 1.3).

2.2.1 Known values of $L_\psi(m, n, t, k)$ and $\Psi_\ell(m, n, t, k)$

The following formulas for $L_\psi(m, n, t, k)$ and $\Psi_\ell(m, n, t, k)$ exist [16].

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

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

where r is given in (2.1).
(g) For all $1 \leq n \leq m$ and $0 < \psi \leq 1$, $L_\psi(m, n, n, 1) = \ell$ is the smallest integer solution to the inequality

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

(h) $\Psi_1(m, m, m, k) = 1$, for all $1 \leq k \leq m$.
(i) $\Psi_1(m, n, n, k) = (r+1)/\binom{m}{n}$, for all $1 \leq k \leq n \leq m$, where r is given in (2.1).
(j) $\Psi_1(m, n, n, k) = 1$, for all $1 \leq k \leq n \leq m$ such that $2n \geq m + k$.
(k) $L_1(m, n, n, 1) = \left\lfloor \frac{m}{n} \right\rfloor$, for all $1 \leq n \leq m$.
(l) For all $0 < \psi \leq 1$,

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

2.2.2 An ILP approach to the incomplete lottery problem

Gründlingh [16] proposed binary programming formulations for the incomplete lottery problem and the resource utilisation problem for lottery problem instances of the form $\langle m, n, n, k \rangle$. Both formulations contained non-linear elements which led Gründlingh to conclude that the formulations were practically infeasible. He also deduced that a more efficient solution method than binary programming is desirable. Gründlingh's attempt to find the value of $L_\psi(m, n, n, k)$ was to adopt an ILP formulation approach in which the objective is to

$$\text{minimise} \quad P(\boldsymbol{x}) = \sum_{1 \leq j \leq \binom{m}{n}} x_j \tag{2.21}$$

subject to

$$\sum_{1 \leq j \leq \binom{m}{n}} a_{ij} x_j \geq 1, \qquad 1 \leq i \leq \left\lceil \psi \binom{m}{n} \right\rceil \tag{2.22}$$

$$x_j \in \{0, 1\}, \qquad 1 \leq j \leq \binom{m}{n}. \tag{2.23}$$

In (2.22), only the lexicographic first $\left\lceil \psi \binom{m}{n} \right\rceil$ sets are considered. Therefore an optimal solution to the formulation (2.21)–(2.23) only yields an upper bound on $L_\psi(m, n, n, k)$. An additional attempt by Gründlingh to find the value of $L_\psi(m, n, n, k)$ was to adopt an ILP formulation approach in which the objective was to

$$\text{minimise} \quad P(\boldsymbol{x}) = \sum_{1 \leq i \leq \binom{m}{n}} x_i \tag{2.24}$$

subject to

$$\sum_{p=1}^{\binom{m}{n}}(-1)^{p+1}\left(\sum_{1\leq i_1<\ldots<i_p}^{\binom{m}{n}}\left(\sum_{j=1}^{\binom{m}{n}}\prod_{q=1}^{p}a_{i_qj}x_{i_q}\right)\right) \geq \left\lceil\psi\binom{m}{t}\right\rceil \tag{2.25}$$

$$x_j \in \{0,1\}, \quad 1\leq i\leq\binom{m}{n}. \tag{2.26}$$

An optimal solution to the formulation (2.24)–(2.26) indeed yields the value $L_\psi(m,n,n,k)$; however, the non-linear nature of (2.25) renders the formulation practically infeasible. Finally, Gründlingh also adopted an ILP formulation approach to find the value of $\Psi_\ell(m,n,t,k)$ in which the objective was to

$$\text{maximise} \quad P(\boldsymbol{x})=\sum_{p=1}^{\ell}(-1)^{p+1}\left(\sum_{1\leq i_1<\ldots<i_p}^{\binom{m}{n}}\left(\sum_{j=1}^{\binom{m}{n}}\prod_{q=1}^{p}a_{i_qj}x_{i_q}\right)\right) \tag{2.27}$$

subject to

$$\sum_{j=1}^{\binom{m}{n}}a_{ij}x_j \geq 1, \qquad 1\leq i\leq\left\lceil\psi\binom{m}{t}\right\rceil \tag{2.28}$$

$$x_j \in \{0,1\}, \quad 1\leq j\leq\binom{m}{n}. \tag{2.29}$$

An optimal solution to formulation (2.27)–(2.29) yields the value $\Psi_\ell(m,n,n,k)$; however, the non-linear nature of (2.27) again renders the formulation practically infeasible.

2.2.3 Approximate methods

Burger *et al.* [6] and Gründlingh [16] proposed algorithmic approaches to finding approximate solutions to the incomplete lottery problem and the resource utilisation problem in the case where $n=t$. The solutions are in the form of playing sets constructed from $\Phi(\mathcal{U}_m,n)$. Heuristics were used to explore different combinations of these sets in search of lower bounds on the number $\Psi_\ell(m,n,n,k)$ and upper bounds on $L_\psi(m,n,n,k)$. Random search methods, a minimal overlapping algorithm (a method involving the construction of interdependent playing sets), a neighbourhood removal method (in which vertices with the largest neighbourhood are removed), a tabu search algorithm, and a genetic algorithm were all employed by Gründlingh in the search for good approximate solutions. The genetic algorithm was found to be the most effective algorithm; however, it was also found to be the most computationally expensive.

2.2.4 Overlapping playing set structures

Burger, *et al.* [5] and Gründlingh [16] proposed that a vector representation of an overlapping playing set structure may be formed by defining the function

$$x^{(\ell)}_{(t_\ell t_{\ell-1}\ldots t_2t_1)_2}=\left|\bigcap_{i=1}^{\ell}\begin{cases}T_i & \text{if } t_i=1\\ T_i' & \text{if } t_i=0\end{cases}\right|$$

where $(t_\ell t_{\ell-1} \dots t_2 t_1)_2$ denotes the binary representation of an integer in the range $\{0, 1, \dots, 2^\ell - 1\}$, T_i denotes ticket i, and T_i' denotes the complement $\mathcal{U}_m \backslash T_i$. This function induces the vector

$$\vec{X}^{(\ell)} = (x^{(\ell)}_{(000\dots0)}, x^{(\ell)}_{(000\dots1)}, \dots, x^{(\ell)}_{(111\dots1)}), \tag{2.30}$$

which represents all the information required to describe the n-set overlapping structure of a playing set. The entries in each vector $X^{(\ell)}$ add up to m, and imply that
(a) there are $x^{(\ell)}_{(000\dots00)}$ elements in the compartment excluded from all T_i,
(b) there are $x^{(\ell)}_{(000\dots01)}$ elements in the compartment exclusive to T_1,
(c) there are $x^{(\ell)}_{(000\dots10)}$ elements in the compartment exclusive to T_2,
(d) there are $x^{(\ell)}_{(000\dots11)}$ elements in the compartment shared by T_1 and T_2, and so on.

A graphical example of a playing set structure for a lottery instance in which $m = 6$ and $n = 3$ is presented in Figure 1.3. Various playing sets may belong to this structure, such as $\{\{1, 2, 3\}, \{1, 4, 5\}\}$ or $\{\{2, 3, 4\}, \{1, 2, 5\}\}$. These two playing sets which exhibit the same overlapping structure are known as *isomorphic playing sets*, as mentioned in Chapter 1.

The exhaustive enumeration lottery tree method, presented by Burger, *et al.* [5] and Gründlingh [16], involves the iterative construction of a rooted tree data structure with the aim of solving the incomplete lottery problem and the resource utilisation problem simultaneously. Level i in the tree contains nodes representing overlapping playing set structures of cardinality i. The nodes on level i of the tree are constructed from the nodes on level $i - 1$. Each overlapping ticket structure is represented by a vector $\mathbf{X}^{(\ell)}$, of size 2^ℓ such as the vector described in (2.30) above. In order to compute a probability of win value associated with a node, its children are constructed. A so-called domination test is conducted on these children: if the newly added ticket has at least k numbers in common with at least one of the existing tickets, the domination test is passed (and the vector is placed into set D_T), otherwise the domination test is failed (and the vector is placed into set D_F). The value,

$$M(X^{(i+1)}) = \frac{m!}{\Pi_{j=0}^{2^\ell-1} x_j^{(i+1)}!},$$

is associated with each child at level $i + 1$ and used to compute the probability of win value,

$$\Psi_{X^{(i)}} = \frac{\sum_{X^{(i+1)} \in D_T} M(X^{(i+1)})}{\sum_{X^{(i+1)} \in D_T \cup D_F} M(X^{(i+1)})},$$

associated with the parent node.

From the exhaustive enumeration lottery tree associated with the lottery $\langle 5, 3, 3, 2\rangle$ in Figure 2.1, it may be seen that $L_{\frac{2}{3}}(5, 3, 3, 2) = 1$, $L_{0.9}(5, 3, 3, 2) = 2$, $L_1(5, 3, 3, 2) = 2$, $\Psi_1(5, 3, 3, 2) = \frac{2}{3}$ and $\Psi_2(5, 3, 3, 2) = 1$. If a playing set of cardinality 2 conforms to the overlapping playing set structure $(1, 1, 1, 2)$, it will possess a resource utilisation value of 0.9. The node representing the overlapping playing set structure $(0, 2, 2, 1)$ possesses a resource utilisation value of 1, because all its children pass the domination test. The node representing the overlapping playing set structure $(1, 1, 1, 2)$ possesses a resource utilisation value of *less than* 1, because one of its children (the node representing the overlapping playing set structure $(0, 0, 0, 2, 1, 1, 1, 0)$) does not pass the domination test.

This exhaustive enumeration solution method is covered more extensively in Chapter 5.

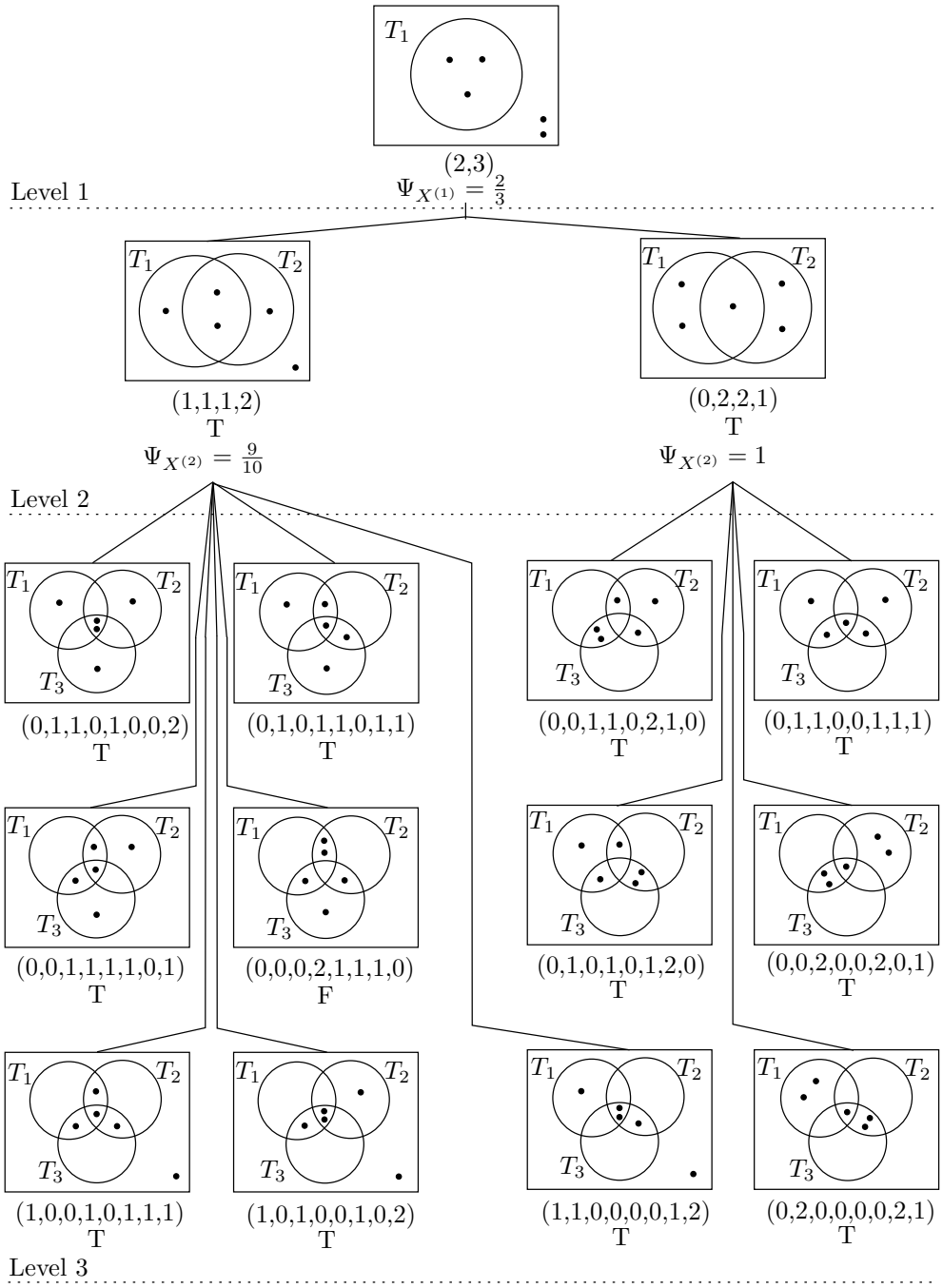


FIGURE 2.1: *The exhaustive enumeration lottery tree for the lottery* $\langle 5, 3, 3, 2 \rangle$. *From this tree, it may be seen that* $L_{\frac{2}{3}}(5, 3, 3, 2) = 1$, $L_{0.9}(5, 3, 3, 2) = 2$, $L_1(5, 3, 3, 2) = 2$, $\Psi_1(5, 3, 3, 2) = \frac{2}{3}$ *and* $\Psi_2(5, 3, 3, 2) = 1$. *If a playing set of cardinality* 2 *conforms to the overlapping playing set structure* $(1, 1, 1, 2)$, *it will possess a resource utilisation value of* 0.9. *The node representing the overlapping playing set structure* $(0, 2, 2, 1)$ *possesses a resource utilisation value of* 1, *because all its children pass the domination test. The node representing the overlapping playing set structure* $(1, 1, 1, 2)$ *possesses a resource utilisation value of less than* 1, *because one of its children (the node representing the overlapping playing set structure* $(0, 0, 0, 2, 1, 1, 1, 0)$*) does not pass the domination test.*

2.3 The number of winners in lottery draws

The average number of winners per draw in the South African national lottery, *Lotto* (a lottery of the form $\langle 49, 6, 6, 6 \rangle$), is approximately 1.002 [32]. However, on the dates of October 31$^{\text{st}}$, 2001, March 15$^{\text{th}}$, 2003, and February 7$^{\text{th}}$, 2009, there were 19, 33, and 18 winners, respectively. After the February 7$^{\text{th}}$ draw, members of the public raised questions regarding the legitimacy of the results [14, 42].

Haigh concluded that the numbers drawn for the winning ticket in the UK Lottery are drawn randomly with a uniform distribution [18]. Haigh also concluded that the choice of numbers by participants is not random.

It is widely speculated that participants tend to select their numbers in such a way as to include calendar numbers, an arithmetic sequence, or spread their n selected numbers out amongst the m numbers [19, 45].

2.4 Chapter summary

In this chapter, known work on the complete lottery problem, incomplete lottery problem and the resource utilisation problem was reviewed. The purpose of this chapter was to introduce the reader to the work which served as inspiration for the work done in the remainder of this thesis.

This chapter opened with a section on work done with regards to the complete lottery problem. Known bounds on the complete lottery number for special lottery instances were presented, followed by formulas for the exact values of the complete lottery number for certain special lottery instances. The section closed with the presentation of an ILP approach, by Jans & Degraeve [20], towards solving the complete lottery problem.

The chapter continued with a section on known work done on the incomplete lottery problem and the resource utilisation problem. Formulas for known values of the incomplete lottery number and the resource utilisation number were presented for special lottery instances. ILP approaches, by Gründlingh [16], towards solving the incomplete lottery problem and the resource utilisation problem were reviewed and discussed. The section on the incomplete lottery problem and resource utilisation problem closed with a discussion of approximate solution methods employed by Burger, *et al.* [6] and Gründlingh, and a brief introduction to the exhaustive enumeration lottery tree solution method, introduced by Gründlingh.

Finally, a mention was made of the public interest in the problem of determining the number of winners in a draw of the South African national lottery.

CHAPTER 3

A new upper bound from graph theory

Contents

In this chapter, the incomplete lottery problem is modelled as a special kind of domination problem over bipartite graphs in fulfilment of Thesis Objective I(a). Using this model, a new upper bound is established on the complete lottery number. The newly established bound is then generalised to be applicable to the incomplete lottery problem according to Thesis Objective I(b). The upper bounds on the complete lottery number and the incomplete lottery number are then finally computed for small and large lottery instances, and the results are compared to known bounds according to Thesis Objective I(c).

3.1 Bounds from graph domination theory: The case $n = t$

Given a graph $\mathbf{G}$ with vertex set $\mathcal{V}(\mathbf{G})$, a dominating set is a subset $\mathcal{D} \in \mathcal{V}(\mathbf{G})$ such that every vertex in $\mathbf{G}$ is either an element of $\mathcal{D}$ or adjacent to at least one vertex in $\mathcal{D}$, and a dominating set of minimum cardinality is called a *minimum dominating set*. Denote the cardinality of a minimum dominating set of $\mathbf{G}$ by $\gamma(\mathbf{G})$. Much work has been done with respect to finding upper bounds on the cardinality of such a dominating set.

Upper bounds on the cardinality of a minimum dominating set of graph $\mathbf{G}$ may be used to derive upper bounds on the complete lottery number $L_1(m, n, n, k)$ — note here that $n = t$ — since a lottery of the form $\langle m, n, n, k\rangle$ may be represented by a regular graph $\mathbf{G}$ denoted by $\mathbf{G}\langle m, n, n, k\rangle$ on $\binom{m}{n}$ vertices. Each vertex represents a ticket in $\langle m, n, n, k\rangle$ and two vertices are joined by means of an edge if the corresponding tickets share at least k numbers. The degrees of the vertices in this graph are

$$r = \sum_{p=k}^{n-1} \binom{n}{p}\binom{m-n}{n-p}. \tag{3.1}$$

Playing a dominating set of the graph will guarantee the player a k-prize, since every possible winning ticket is adjacent to at least one of the tickets in the participant's playing set. The lottery graph $\mathbf{G}\langle 6,3,3,2\rangle$ of order $\binom{6}{3} = 20$ and degree

$$r = \sum_{p=2}^{2} \binom{3}{p}\binom{3}{3-p} = \binom{3}{2}\binom{3}{1} = 3 \times 3 = 9$$

is shown as an example in Figure 3.1.

In 1974, Arnautov [1] obtained an upper bound on the cardinality of a minimum dominating set for a graph $\mathbf{G}$ with p vertices where each vertex has a degree of d. Arnautov showed that

$$\gamma(\mathbf{G}) \leq \frac{p}{d+1}\left(1 + \frac{1}{2} + \frac{1}{2} + \ldots + \frac{1}{d+1}\right). \tag{3.2}$$

In the same year, he also established the asymptotic result of a well-known theorem stating that for any graph containing p non-isolated vertices,

$$\gamma(\mathbf{G}) \leq p\left(\frac{1+\ln(\delta(\mathbf{G})+1)}{\delta(\mathbf{G})+1}\right), \tag{3.3}$$

where $\delta(\mathbf{G})$ denotes the minimum degree of $\mathbf{G}$. Thereafter, Payan [34] (in 1975) and Marcu [29] (in 1986) both proved independently that

$$\gamma(\mathbf{G}) \leq \frac{(p-\delta(\mathbf{G})-1)(p-\delta(\mathbf{G})-2)}{p-1} + 2. \tag{3.4}$$

In 1985, Caro & Roditty [9] showed that

$$\gamma(\mathbf{G}) \leq p\left[1 - \delta(\mathbf{G})\left(\frac{1}{\delta(\mathbf{G})+1}\right)^{1+\frac{1}{\delta(\mathbf{G})}}\right]. \tag{3.5}$$

Over ten years later, in 1996, Reed [36] proved a long-standing conjecture that

$$\gamma(\mathbf{G}) \leq \frac{3}{8}p \tag{3.6}$$

for graphs of minimum degree 2. Reed's result was followed two years later by work of Clark, *et al.* [11] in which it was shown that

$$\gamma(\mathbf{G}) \leq p\left(1 - \frac{\delta(\mathbf{G})^3+\delta(\mathbf{G})}{\delta(\mathbf{G})^3+1}\prod_{j=1}^{\delta(\mathbf{G})}\left(1+\frac{1}{j\delta(\mathbf{G})}\right)^{-1}\right). \tag{3.7}$$

As mentioned, the bounds (3.2)–(3.7) may be used to compute upper bounds on the complete lottery number for lotteries with parameters $\langle m,n,n,k\rangle$. In 2004, Gründlingh [16] additionally deduced the upper bound

$$\gamma(\mathbf{G}) \leq \frac{2(r+1)+x+\sqrt{(2(r+1)+x)^2-8xp}}{2x} \tag{3.8}$$

on the complete lottery number for the lottery $\langle m,n,n,k\rangle$ without using the theory of graphs, where

$$x = \sum_{i=k}^{n-1}\sum_{j=k}^{n-1}\binom{n}{i}\binom{n}{j}\binom{m-2n}{n-i-j}. \tag{3.9}$$

Note that (3.8) yields the bound ∞ if $m-2n<0$ or $n-i-j<0$ or $m-2n<n-i-j$. The above discussion was for lotteries in which $n=t$. However, if $n \neq t$, an alternative, so-called bipartite graph representation of the lottery is required.

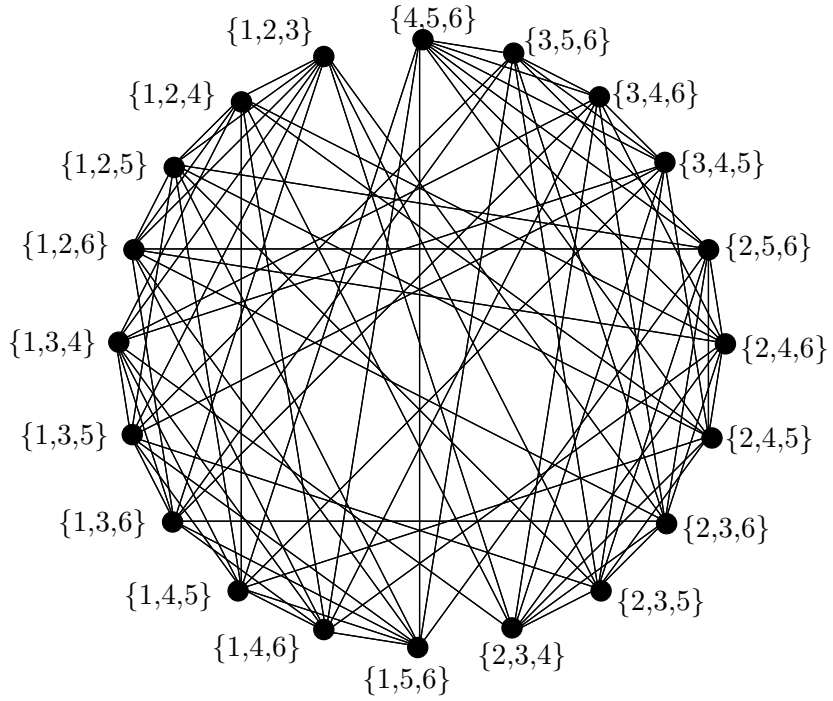


FIGURE 3.1: *A regular graph with* $\binom{6}{3} = 20$ *vertices, and degree* $r = 9$*, representing the lottery* $\langle 6, 3, 3, 2\rangle$*. An example of a dominating set for this graph consists for the vertices labelled* $\{1, 2, 3\}$ *and* $\{4, 5, 6\}$*.*

3.2 Bipartite graph representation of the lottery problem

A graph $\mathbf{G}$ is bipartite if its vertex set $\mathcal{V}(\mathbf{G})$ may be partitioned into two sets $\mathcal{V}_1$ and $\mathcal{V}_2$, called *partite sets*, in such a way that no two vertices from the same partite set are adjacent. In [2], various properties of bipartite graphs are presented. A lottery may $\langle m, n, t, k\rangle$ may be represented by a bipartite graph $\mathbf{G}$, containing partite sets $\mathcal{V}_1 = \Psi(\mathcal{U}_m, n)$ and $\mathcal{V}_2 = \Psi(\mathcal{U}_m, t)$. A bipartite graph representation of the lottery $\langle 7, 5, 4, 3\rangle$ is shown in Figure 1.1. In the figure, the black vertices represent the partite set $\mathcal{V}_1 = \Phi(\mathcal{U}_7, 5)$, modelling tickets which may be played by a lottery participant, while the white vertices represent the partite set $\mathcal{V}_2 = \Phi(\mathcal{U}_7, 4)$ in $\langle 7, 5, 4, 3\rangle$, modelling the tickets from which the government selects the winning ticket of the lottery $\langle m, n, t, k\rangle$.

Suppose a connected bipartite graph has partite sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and that the neighbourhood of a vertex $v \in \mathcal{V}_1$ is denoted by $\mathcal{N}[v] \subseteq \mathcal{V}_2$. Then the elements in $\mathcal{N}[v]$ are said to be *covered* by v. It may easily be shown that $\bigcup_{v\in\mathcal{V}_1} \mathcal{N}[v] = \mathcal{V}_2$. In the *bipartite minimum covering problem* a subset $\mathcal{V}_1' \in \mathcal{V}_1$ of minimum cardinality is sought such that

$$\left| \bigcup_{v\in\mathcal{V}_1'} \mathcal{N}[x] \right| = |\mathcal{V}_2| \,.$$

In this problem the set $\mathcal{V}_1'$ is called the *covering set* (or *covering* for short), and the partite set $\mathcal{V}_2$ is called the *covered set*. The aim, therefore, is to find a covering consisting of as few elements as possible, known as a *minimum covering* of $\mathcal{V}_2$ by $\mathcal{V}_1'$. If the problem is large there may be a large number of combinations of vertices from $\mathcal{V}_1$ from which the set $\mathcal{V}_1'$ may be constructed, implying that it may be computationally expensive to find a minimum covering. In such cases

an approximation algorithm may be used instead. One of the most common approximation algorithms is the Greedy Covering Algorithm, a pseudocode listing of which is presented in Algorithm 3.1.

Algorithm 3.1: Greedy Covering Algorithm

Data: A bipartite graph $\mathbf{G}$ of order p with partite sets $\mathcal{V}_1$ and $\mathcal{V}_2$
Result: A covering $\mathcal{V}_1'' \subseteq \mathcal{V}_1$

```
1 V_1'' ← ∅;
2 while V_2 ≠ ∅ do
3     Choose a vertex v of maximum degree in V_1;
4     V(G) ← V(G)\({v} ∪ N[v]);
5     V_1'' ← V_1'' ∪ {v};
6 end
```

The output of Algorithm 3.1 is a covering of vertex set $\mathcal{V}_2$ by a subset $\mathcal{V}_1''$ of $\mathcal{V}_1$. It may be possible, through the use of more advanced algorithms, to compute a covering $\mathcal{V}_1'$ of $\mathcal{V}_2$ of minimum cardinality for small graphs. However, the cardinality of the subset $\mathcal{V}_1''$ produced by Algorithm 3.1 is not necessarily of minimum cardinality, implying that $|\mathcal{V}_1''|$ is an upper bound on $|\mathcal{V}_1'|$. Therefore, an upper bound on the complete lottery number $L_1(m, n, t, k)$ may be computed by means of the output of Algorithm 3.1 when applied to the bipartite lottery graph $\mathbf{G}\langle m, n, t, k\rangle$.

In order to compute the degree, s, of each vertex in partite set $\mathcal{V}_2$, all the tickets from $\Phi(\mathcal{U}_m, n)$, or $\mathcal{V}_1$, are counted which have k or more numbers in common with a given ticket in $\Phi(\mathcal{U}_m, t)$, or $\mathcal{V}_2$. First, the tickets from $\mathcal{V}_1$ are counted which have *exactly* k numbers in common with a given ticket in $\mathcal{V}_2$. From the t numbers in a government ticket, k numbers may be chosen in $\binom{t}{k}$ ways and from the remaining $m - t$ numbers, there are $\binom{m-t}{n-k}$ ways in which to choose the remaining $n - k$ numbers in the participant ticket. Therefore, there are $\binom{t}{k}\binom{m-t}{n-k}$ tickets from $\mathcal{V}_1$ which have *exactly* k numbers in common with a given ticket in $\Phi(\mathcal{U}_m, t)$. If this procedure is repeated for $k+1, \ldots, \min\{n, t\}$ instead of k, all the tickets from $\mathcal{V}_1$ are counted which have k or more numbers in common with a given ticket in $\mathcal{V}_2$. Therefore,

$$s = \sum_{p=k}^{\min\{n,t\}} \binom{t}{p}\binom{m-t}{n-p}. \tag{3.10}$$

3.3 A greedy bound on the complete lottery number

In this section, an analytic upper bound on the minimum cardinality of a bipartite covering, is reviewed from [2]. This bound is then used to compute upper bounds on the complete lottery number for small and large lottery instances. The upper bounds obtained for those lottery instances are then compared to the upper bounds presented in §3.1 in the special case where $n = t$. The following lemma is used to establish the upper bound on the complete lottery number.

Lemma 3.1 *Let $\mathbf{G}$ be a bipartite graph with partite sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and assume s is the degree of the vertices in $\mathcal{V}_2$. Let $\mathcal{R}_j$ be the subset of vertices remaining in $\mathcal{V}_2$ after the j^{th} iteration of*

Algorithm 3.1 with the convention that $\mathcal{R}_0 = \mathcal{V}_2$. *Then*

$$|\mathcal{R}_j| \leq |\mathcal{V}_2| \prod_{i=0}^{j-1} \left(1 - \frac{s}{|\mathcal{V}_1| - i}\right). \tag{3.11}$$

Proof: *In order to prove the inequality in (3.11), it may be seen that after* j *iterations of Algorithm 3.1,* $|\mathcal{R}_j|s$ *edges remain in the bipartite graph, while* $|\mathcal{V}_1| - j$ *vertices remain in* $\mathcal{V}_1$. *If all the edges in the graph are distributed amongst the* $|\mathcal{V}_1| - j$ *vertices in* $\mathcal{V}_1$, *then there is a vertex in* $\mathcal{V}_1$ *of degree at least*

$$\frac{|\mathcal{R}_j|s}{|\mathcal{V}_1| - j}. \tag{3.12}$$

Therefore,

$$|\mathcal{R}_{j+1}| \leq |\mathcal{R}_j| - \frac{|\mathcal{R}_j|s}{|\mathcal{V}_1| - j} = |\mathcal{R}_j| \left(1 - \frac{s}{|\mathcal{V}_1| - j}\right), \tag{3.13}$$

from which (3.11) follows inductively for any $0 \leq j \leq |\mathcal{V}_1|$. □

In the following theorem, an upper bound on the complete lottery number (obtained from [2]) is established.

Theorem 3.1 *Let* $\boldsymbol{G}$ *be the bipartite graph representation of the lottery* $\langle m, n, t, k\rangle$ *with partite sets* $\mathcal{V}_1 = \Phi(\mathcal{U}_m, n)$ *and* $\mathcal{V}_2 = \Phi(\mathcal{U}_m, t)$. *Then*

$$L_1(m, n, t, k) \leq 1 + \frac{\binom{m}{n}}{s} \left(1 + \log \frac{s\binom{m}{t}}{\binom{m}{n}}\right), \tag{3.14}$$

where s *in (3.10) is the degree of vertices in* $\mathcal{V}_2$.

Proof: *For any value of* $j \geq 0$, *the number of remaining iterations of the while-loop spanning lines* 2–6 *in Algorithm 3.1 required to complete the algorithm is bounded from above by* $|\mathcal{R}_j|$. *It follows by Lemma 3.1 that*

$$L_1(m, n, t, k) \leq j + |\mathcal{R}_j| \leq j + \binom{m}{t} \left(1 - \frac{s}{\binom{m}{n}}\right)^j \leq j + \binom{m}{t} \mathrm{e}^{\frac{-s \times j}{\binom{m}{n}}} \tag{3.15}$$

at termination of the algorithm. In order to simplify inequality (3.15), a substitute value for j *is selected which is less than* $\binom{m}{n}$, *and a function of* $\binom{m}{n}$, $\binom{m}{t}$ *and* s. *It is known that* $\mathrm{e}^{\log(z)} = z$, *so by substituting*

$$j = \left\lceil \frac{\binom{m}{n}}{s} \log \frac{s\binom{m}{t}}{\binom{m}{n}} \right\rceil$$

into (3.15), it follows that

$$L_1(m, n, t, k) \leq 1 + \tfrac{\binom{m}{n}}{s} \left(1 + \log \tfrac{s\binom{m}{t}}{\binom{m}{n}}\right). \qquad \square$$

The following two examples illustrate how Theorem 3.1 may be applied to the lottery $\langle 8, 4, 3, 2\rangle$ and to the well-known lottery $\langle 49, 6, 6, 3\rangle$.

Example 3.1 (Greedy bound on $L_1(8,4,3,2)$ and $L_1(49,6,6,3)$)
For the lottery $\langle 8,4,3,2\rangle$,

$$s=\sum_{p=2}^{3}\binom{3}{p}\binom{5}{4-p}=\binom{3}{2}\binom{5}{2}+\binom{3}{3}\binom{5}{1}=35.$$

The greedy bound (3.14) yields

$$L_1(8,4,3,2)\leq 1+\frac{70}{35}\left(1+\log\frac{35\times 56}{70}\right)\approx 5.894.$$

It is, in fact, known that $L_1(8,4,3,2)=2$. For the lottery $\langle 49,6,6,3\rangle$,

$$s=\sum_{p=3}^{6}\binom{6}{p}\binom{43}{6-p}=\binom{6}{3}\binom{43}{3}+\binom{6}{4}\binom{43}{2}+\binom{6}{5}\binom{43}{1}+\binom{6}{6}\binom{43}{0}=260\,624.$$

The greedy bound (3.14) yields

$$L_1(49,6,6,3)\leq 1+\frac{13\,983\,816}{260\,624}\left(1+\log\frac{260\,624\times 13\,983\,816}{13\,983\,816}\right)\approx 345.252.$$

It is, in fact, known that $L_1(49,6,6,3)\leq 163$. ■

Tables 3.1 and 3.2 contain comparisons of the bounds presented in this chapter. The bounds (3.4) and (3.6) are weak upper bounds and they are therefore omitted from the tables. In Table 3.1 the bounds in §3.1, together with the bound in (2.12) and best known bounds available in the online lottery repository [22], are compared to the new greedy bound in (3.14) for small lottery instances (where $6\leq m\leq 10$, $m>\{n,t\}>k\geq 2$ and $L_1(m,n,t,k)>1$). In Table 3.2, a similar comparison is presented for larger lotteries which are of the form $\langle m,6,6,3\rangle$, for $6\leq m\leq 50$. Table 3.1 also contains $|\mathcal{V}_1''|$, the cardinality of the covering set obtained via Algorithm 3.1. The value $|\mathcal{V}_1''|$ is omitted from Table 3.2 due to the computational complexity involved in applying Algorithm 3.1 to the larger lottery instances.

The values of s, $\binom{m}{n}$ and $\binom{m}{t}$ affect the value of the greedy bound (3.14) — as the values of $\binom{m}{n}$ and $\binom{m}{t}$ increase, so does the value of the greedy bound. However, as the value of s increases, the value of the greedy bound (3.14) decreases. This makes sense in the context of the greedy algorithm, because if a participant ticket x is removed from the corresponding graph model, the neighbourhood set $\mathcal{N}(x)$ is also removed from the graph. If the degree of the government tickets is large, removal of the set $\mathcal{N}(x)$ from the graph results in the degrees of a large number of the remaining participant tickets being decreased, leaving a small number of participant tickets, each with a relatively larger degree to choose from during the next iteration of Algorithm 3.1.

3.4 A greedy bound on the incomplete lottery number

The bounds presented in §3.1–3.3 are only applicable to the complete lottery problem. However, the result of Theorem 3.1 may be generalised to hold for the incomplete lottery problem. In the incomplete lottery problem, it is required that at least a proportion ψ of the government tickets be covered. If x tickets are removed from the $\binom{m}{t}$ government tickets, a proportion ψ

TABLE 3.1: *The greedy bound (3.14), compared to previously known upper bounds (2.12), (3.2), (3.3), (3.5), (3.7), (3.8) and (3.14), on the complete lottery number for small lottery instances where* $6 \leq m \leq 10$, $m > \{n, t\} > k \geq 2$ *and* $L_1(m, n, t, k) > 1$. *The entries in the column labelled* $|\mathcal{V}_1''|$ *represent the cardinality of the resulting playing set after applying Algorithm 3.1. A "–" indicates that it is not possible to compute that bound for the specific lottery.*

Lottery	$\binom{m}{n}$	$\binom{m}{t}$	s	L_1	$\lvert\mathcal{V}_1''\rvert$	(2.12)	(3.14)	(3.7)	(3.2)	(3.5)	(3.3)	(3.8)
$\langle 6,3,3,2\rangle$	20	20	10	2	2	4	5	5	6	6	7	∞
$\langle 7,3,3,2\rangle$	35	35	13	4	4	5	7	8	9	9	10	∞
$\langle 7,3,4,2\rangle$	35	35	22	2	2	4	5	–	–	–	–	–
$\langle 8,3,3,2\rangle$	56	56	16	5	5	7	9	11	12	12	13	∞
$\langle 8,3,4,2\rangle$	56	70	28	3	3	4	7	–	–	–	–	–
$\langle 8,3,5,2\rangle$	56	56	40	2	2	3	5	–	–	–	–	–
$\langle 8,4,3,2\rangle$	70	56	35	2	2	5	6	–	–	–	–	–
$\langle 8,4,4,2\rangle$	70	70	53	2	2	4	5	6	6	7	7	3
$\langle 8,4,4,3\rangle$	70	70	17	6	8	8	11	13	14	15	16	∞
$\langle 9,3,3,2\rangle$	84	84	19	7	9	9	12	14	16	16	17	∞
$\langle 9,3,4,2\rangle$	84	126	34	3	3	5	8	–	–	–	–	–
$\langle 9,3,5,2\rangle$	84	126	50	3	3	4	6	–	–	–	–	–
$\langle 9,3,6,2\rangle$	84	84	65	2	2	3	5	–	–	–	–	–
$\langle 9,4,3,2\rangle$	126	84	51	4	4	6	8	–	–	–	–	–
$\langle 9,4,4,2\rangle$	126	126	81	2	4	4	6	8	8	9	9	4
$\langle 9,4,4,3\rangle$	126	126	21	9	11	11	15	20	22	23	24	∞
$\langle 9,4,5,2\rangle$	126	126	105	2	2	4	5	–	–	–	–	–
$\langle 9,4,5,3\rangle$	126	126	45	5	5	5	9	–	–	–	–	–
$\langle 9,4,6,3\rangle$	126	84	75	2	2	4	6	–	–	–	–	–
$\langle 10,3,3,2\rangle$	120	120	22	8	10	11	14	19	20	21	22	∞
$\langle 10,3,4,2\rangle$	120	210	40	5	5	6	10	–	–	–	–	–
$\langle 10,3,5,2\rangle$	120	252	60	3	3	5	8	–	–	–	–	–
$\langle 10,3,6,2\rangle$	120	210	80	3	3	3	6	–	–	–	–	–
$\langle 10,3,7,2\rangle$	120	120	98	2	2	3	5	–	–	–	–	–
$\langle 10,4,3,2\rangle$	210	120	70	4	4	8	9	–	–	–	–	–
$\langle 10,4,4,2\rangle$	210	210	115	3	3	5	7	10	10	11	11	6
$\langle 10,4,4,3\rangle$	210	210	25	14	17	16	22	30	32	33	35	∞
$\langle 10,4,5,2\rangle$	210	252	155	2	2	4	6	–	–	–	–	–
$\langle 10,4,5,3\rangle$	210	252	55	7	7	8	12	–	–	–	–	–
$\langle 10,4,6,2\rangle$	210	210	185	2	2	3	5	–	–	–	–	–
$\langle 10,4,6,3\rangle$	210	210	95	4	4	5	8	–	–	–	–	–
$\langle 10,4,7,3\rangle$	210	120	140	2	2	4	6	–	–	–	–	–
$\langle 10,5,3,2\rangle$	252	120	126	2	2	6	7	–	–	–	–	–
$\langle 10,5,4,2\rangle$	252	210	186	2	2	5	6	–	–	–	–	–
$\langle 10,5,4,3\rangle$	252	210	66	7	8	9	12	–	–	–	–	–
$\langle 10,5,5,2\rangle$	252	252	226	2	2	4	5	7	7	8	8	3
$\langle 10,5,5,3\rangle$	252	252	126	2	2	4	8	11	11	12	12	∞
$\langle 10,5,5,4\rangle$	252	252	26	14	21	18	25	35	37	39	41	∞

TABLE 3.2: *The greedy bound in (3.14), compared to previously known bounds (2.12), (3.2), (3.3), (3.5), (3.7), (3.8) and (3.14) on the complete lottery number for instances of the lottery* $\langle m, 6, 6, 3\rangle$, *where* $6 \leq m \leq 50$. *A question mark indicates that the lottery number does not appear in any online lottery tables, and is hence not known. The column labelled '[22]' contains best known upper bounds available in the online lottery repository [22].*

Lottery	L_1	[22]	(2.12)	(3.14)	(3.7)	(3.2)	(3.5)	(3.3)	(3.8)
$\langle 6,6,6,3\rangle$	1	1	–	2	1	1	1	1	∞
$\langle 7,6,6,3\rangle$	1	1	2	2	3	3	3	3	∞
$\langle 8,6,6,3\rangle$	1	1	2	4	4	4	5	5	∞
$\langle 9,6,6,3\rangle$	1	1	2	4	5	5	6	6	∞
$\langle 10,6,6,3\rangle$	2	2	3	5	7	7	7	7	∞
$\langle 11,6,6,3\rangle$	2	2	4	6	8	8	9	9	∞
$\langle 12,6,6,3\rangle$	2	2	4	7	10	10	11	11	3
$\langle 13,6,6,3\rangle$	2	2	4	8	13	13	13	13	5
$\langle 14,6,6,3\rangle$	4	4	5	9	15	16	16	16	7
$\langle 15,6,6,3\rangle$	4	4	7	11	19	19	20	20	10
$\langle 16,6,6,3\rangle$	5	5	7	13	23	23	24	24	14
$\langle 17,6,6,3\rangle$	6	6	8	15	27	27	28	28	19
$\langle 18,6,6,3\rangle$	7	7	10	18	32	32	33	33	25
$\langle 19,6,6,3\rangle$	?	9	11	20	37	37	39	39	32
$\langle 20,6,6,3\rangle$	?	10	13	23	43	43	45	45	40
$\langle 21,6,6,3\rangle$	?	13	14	27	50	50	52	52	50
$\langle 22,6,6,3\rangle$	?	15	19	31	58	58	60	60	61
$\langle 23,6,6,3\rangle$	?	17	21	35	66	66	69	69	73
$\langle 24,6,6,3\rangle$	?	20	24	39	75	75	78	78	87
$\langle 25,6,6,3\rangle$	?	22	28	44	85	85	89	89	103
$\langle 26,6,6,3\rangle$	?	25	31	50	96	96	100	100	121
$\langle 27,6,6,3\rangle$	?	27	35	56	108	108	112	113	140
$\langle 28,6,6,3\rangle$	?	31	38	62	121	121	126	126	162
$\langle 29,6,6,3\rangle$	?	35	43	69	135	135	140	140	185
$\langle 30,6,6,3\rangle$	?	39	48	76	150	150	156	156	211
$\langle 31,6,6,3\rangle$	?	45	53	84	166	166	172	173	239
$\langle 32,6,6,3\rangle$	?	50	60	93	184	184	190	190	270
$\langle 33,6,6,3\rangle$	?	55	66	102	202	202	210	210	303
$\langle 34,6,6,3\rangle$	?	60	72	112	222	222	230	230	338
$\langle 35,6,6,3\rangle$	?	66	78	122	243	243	252	252	377
$\langle 36,6,6,3\rangle$	?	72	85	133	266	266	275	275	418
$\langle 37,6,6,3\rangle$	?	78	92	145	289	289	299	299	462
$\langle 38,6,6,3\rangle$	?	83	99	157	315	315	326	326	508
$\langle 39,6,6,3\rangle$	?	89	105	170	341	341	353	353	558
$\langle 40,6,6,3\rangle$	?	96	112	184	370	370	382	382	612
$\langle 41,6,6,3\rangle$	?	102	120	199	399	400	413	413	668
$\langle 42,6,6,3\rangle$	?	109	127	214	431	431	446	446	728
$\langle 43,6,6,3\rangle$	?	117	135	230	464	464	480	480	791
$\langle 44,6,6,3\rangle$	?	124	144	247	499	499	516	516	858
$\langle 45,6,6,3\rangle$	?	131	151	265	536	536	553	553	928
$\langle 46,6,6,3\rangle$	?	138	160	284	574	574	593	593	1 002
$\langle 47,6,6,3\rangle$	?	146	168	304	614	614	634	634	1 081
$\langle 48,6,6,3\rangle$	?	153	178	324	656	656	678	678	1 163
$\langle 49,6,6,3\rangle$	?	163	186	346	701	701	723	723	1 249
$\langle 50,6,6,3\rangle$	?	175	198	368	747	747	771	771	1 339

of the remaining $\mathcal{R}_j$ government tickets together with the x tickets already covered will be at least as large as a proportion ψ of the $\binom{m}{t}$ government tickets, which implies that

$$\begin{aligned} x + \psi\left|\mathcal{R}_j\right| &= \binom{m}{t} - \left|\mathcal{R}_j\right| + \psi\left|\mathcal{R}_j\right| \\ &= \psi\binom{m}{t} + \binom{m}{t} - \left|\mathcal{R}_j\right|(1-\psi) - \psi\binom{m}{t} \\ &= \psi\binom{m}{t} + \binom{m}{t}(1-\psi) - \left|\mathcal{R}_j\right|(1-\psi) \\ &= \psi\binom{m}{t} + (1-\psi)\left(\binom{m}{t} - \left|\mathcal{R}_j\right|\right) \\ &\geq \psi\binom{m}{t}, \end{aligned} \tag{3.16}$$

because $\binom{m}{t} \geq \left|\mathcal{R}_j\right|$ and $1 \geq \psi$.

This, in turn, implies that if $\left|\mathcal{R}_j\right|$ government tickets remain after iteration j of Algorithm 3.1, at least a proportion ψ of all the government tickets may be covered by covering a proportion ψ of the $\left|\mathcal{R}_j\right|$ remaining government tickets, together with the remaining $\binom{m}{t} - \left|\mathcal{R}_j\right|$ tickets. Therefore, from (3.15),

$$\begin{aligned} L_\psi(m,n,t,k) &\leq j + \psi\left|\mathcal{R}_j\right| \\ &\leq 1 + \left(\frac{\binom{m}{n}}{s}\right)\log\left(\frac{s\binom{m}{t}}{\binom{m}{n}}\right) + \psi\left(\frac{\binom{m}{n}}{s}\right) \\ &= 1 + \frac{\binom{m}{n}}{s}\left(\log\left(\frac{s\binom{m}{t}}{\binom{m}{n}}\right) + \psi\right). \end{aligned} \tag{3.17}$$

The following example illustrates how the greedy bound in (3.17) performs.

Example 3.2 (Greedy bound on $L_{0.6}(8,4,3,2)$ and $L_{0.6}(49,6,6,3)$) *Consider the lottery $\langle 8,4,3,2\rangle$ and let $\psi = 0.6$. It is known from Example 3.1 that $s = 35$ in this case. The greedy bound in (3.17) yields*

$$L_{0.6}(8,4,3,2) \leq 1 + \frac{70}{35}\left(0.6 + \log\frac{35\times 56}{70}\right) \approx 5.094.$$

Now consider the lottery $\langle 49,6,6,3\rangle$, still assuming that $\psi = 0.6$. It is known from Example 3.1 that $s = 260\,624$ in this case. The greedy bound in (3.17) therefore yields

$$L_{0.6}(49,6,6,3) \leq 1 + \frac{13\,983\,816}{260\,624}\left(0.6 + \log\frac{260\,624\times 13\,983\,816}{13\,983\,816}\right) \approx 323.790,$$

as an upper bound for the incomplete lottery number $L_{0.6}(49,6,6,3)$. ■

Tables 3.3–3.5 contain the greedy bound in (3.17) for small instances of the incomplete lottery problem where $6 \leq m \leq 10$, $m > \{n,t\} > k \geq 2$ and $L_1(m,n,t,k) > 1$, along with known incomplete lottery numbers for break point values[1] of ψ as computed in Chapter 4.

[1] A break point value is the maximum resource utilisation achieved by constructing a playing set of cardinality $L_\psi(m,n,t,k)$.

TABLE 3.3: *The upper bound (3.17) for small instances of the incomplete lottery problem in which* $6 \leq m \leq 8$ *at break-point values of* ψ.

Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)
$\langle 6,3,3,2\rangle$	0.5	1	4	$\langle 8,3,3,2\rangle$	0.786	3	8	$\langle 8,4,4,2\rangle$	0.757	1	5
$\langle 6,3,3,2\rangle$	1	2	5	$\langle 8,3,3,2\rangle$	0.893	4	9	$\langle 8,4,4,2\rangle$	1	2	5
$\langle 7,3,3,2\rangle$	0.371	1	5	$\langle 8,3,3,2\rangle$	1	5	9	$\langle 8,4,4,3\rangle$	0.243	1	8
$\langle 7,3,3,2\rangle$	0.743	2	6	$\langle 8,3,4,2\rangle$	0.5	1	6	$\langle 8,4,4,3\rangle$	0.486	2	9
$\langle 7,3,3,2\rangle$	0.914	3	7	$\langle 8,3,4,2\rangle$	0.871	2	6	$\langle 8,4,4,3\rangle$	0.671	3	9
$\langle 7,3,3,2\rangle$	1	4	7	$\langle 8,3,4,2\rangle$	1	3	7	$\langle 8,4,4,3\rangle$	0.857	4	10
$\langle 7,3,4,2\rangle$	0.629	1	5	$\langle 8,3,5,2\rangle$	0.714	1	5	$\langle 8,4,4,3\rangle$	0.914	5	10
$\langle 7,3,4,2\rangle$	1	2	5	$\langle 8,3,5,2\rangle$	1	2	5	$\langle 8,4,4,3\rangle$	1	6	11
$\langle 8,3,3,2\rangle$	0.286	1	7	$\langle 8,4,3,2\rangle$	0.5	1	5				
$\langle 8,3,3,2\rangle$	0.571	2	8	$\langle 8,4,3,2\rangle$	1	2	6				

TABLE 3.4: *The upper bound (3.17) for small instances of the incomplete lottery problem in which* $m = 9$ *at break-point values of* ψ.

Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)
$\langle 9,3,3,2\rangle$	0.226	1	8	$\langle 9,3,6,2\rangle$	0.774	1	5	$\langle 9,4,4,3\rangle$	0.833	6	14
$\langle 9,3,3,2\rangle$	0.452	2	9	$\langle 9,3,6,2\rangle$	1	2	5	$\langle 9,4,4,3\rangle$	0.913	7	15
$\langle 9,3,3,2\rangle$	0.679	3	10	$\langle 9,4,3,2\rangle$	0.405	1	6	$\langle 9,4,4,3\rangle$	0.96	8	15
$\langle 9,3,3,2\rangle$	0.774	4	11	$\langle 9,4,3,2\rangle$	0.81	2	7	$\langle 9,4,4,3\rangle$	1	9	15
$\langle 9,3,3,2\rangle$	0.893	5	11	$\langle 9,4,3,2\rangle$	0.952	3	8	$\langle 9,4,5,2\rangle$	0.833	1	5
$\langle 9,3,3,2\rangle$	0.952	6	11	$\langle 9,4,3,2\rangle$	1	4	8	$\langle 9,4,5,2\rangle$	1	2	5
$\langle 9,3,3,2\rangle$	1	7	12	$\langle 9,4,4,2\rangle$	0.643	1	5	$\langle 9,4,5,3\rangle$	0.357	1	7
$\langle 9,3,4,2\rangle$	0.397	1	7	$\langle 9,4,4,2\rangle$	1	2	6	$\langle 9,4,5,3\rangle$	0.706	2	8
$\langle 9,3,4,2\rangle$	0.738	2	8	$\langle 9,4,4,3\rangle$	0.167	1	10	$\langle 9,4,5,3\rangle$	0.857	3	9
$\langle 9,3,4,2\rangle$	1	3	8	$\langle 9,4,4,3\rangle$	0.333	2	11	$\langle 9,4,5,3\rangle$	0.976	4	9
$\langle 9,3,5,2\rangle$	0.603	1	6	$\langle 9,4,4,3\rangle$	0.5	3	12	$\langle 9,4,5,3\rangle$	1	5	9
$\langle 9,3,5,2\rangle$	0.929	2	6	$\langle 9,4,4,3\rangle$	0.627	4	13	$\langle 9,4,6,3\rangle$	0.595	1	5
$\langle 9,3,5,2\rangle$	1	3	6	$\langle 9,4,4,3\rangle$	0.738	5	14	$\langle 9,4,6,3\rangle$	1	2	6

TABLE 3.5: *The upper bound (3.17) for small instances of the incomplete lottery problem in which $m = 10$ at break-point values of ψ.*

Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)	Lottery	ψ	L_ψ	(3.17)
$\langle 10,3,3,2\rangle$	0.183	1	10	$\langle 10,4,4,3\rangle$	0.362	3	16	$\langle 10,5,3,2\rangle$	1	2	7
$\langle 10,3,3,2\rangle$	0.367	2	11	$\langle 10,4,4,3\rangle$	0.481	4	17	$\langle 10,5,4,2\rangle$	0.738	1	5
$\langle 10,3,3,2\rangle$	0.55	3	12	$\langle 10,4,4,3\rangle$	0.6	5	18	$\langle 10,5,4,2\rangle$	1	2	6
$\langle 10,3,3,2\rangle$	0.667	4	12	$\langle 10,4,4,3\rangle$	0.662	6	19	$\langle 10,5,4,3\rangle$	0.262	1	9
$\langle 10,3,3,2\rangle$	0.767	5	13	$\langle 10,4,4,3\rangle$	0.738	7	19	$\langle 10,5,4,3\rangle$	0.519	2	10
$\langle 10,3,3,2\rangle$	0.85	6	13	$\langle 10,4,4,3\rangle$	0.81	8	20	$\langle 10,5,4,3\rangle$	0.7	3	11
$\langle 10,3,3,2\rangle$	0.917	7	14	$\langle 10,4,4,3\rangle$	0.862	9	20	$\langle 10,5,4,3\rangle$	0.881	4	12
$\langle 10,3,3,2\rangle$	1	8	14	$\langle 10,4,4,3\rangle$	0.919	10	21	$\langle 10,5,4,3\rangle$	0.919	5	12
$\langle 10,3,4,2\rangle$	0.329	1	8	$\langle 10,4,4,3\rangle$	0.952	11	21	$\langle 10,5,4,3\rangle$	0.99	6	12
$\langle 10,3,4,2\rangle$	0.619	2	9	$\langle 10,4,4,3\rangle$	0.971	12	21	$\langle 10,5,4,3\rangle$	1	7	12
$\langle 10,3,4,2\rangle$	0.871	3	10	$\langle 10,4,4,3\rangle$	0.99	13	22	$\langle 10,5,5,2\rangle$	0.901	1	5
$\langle 10,3,4,2\rangle$	0.962	4	10	$\langle 10,4,4,3\rangle$	1	14	22	$\langle 10,5,5,2\rangle$	1	2	5
$\langle 10,3,4,2\rangle$	1	5	10	$\langle 10,4,5,2\rangle$	0.738	1	6	$\langle 10,5,5,3\rangle$	0.5	1	7
$\langle 10,3,5,2\rangle$	0.5	1	7	$\langle 10,4,5,2\rangle$	1	2	6	$\langle 10,5,5,3\rangle$	1	2	8
$\langle 10,3,5,2\rangle$	0.829	2	7	$\langle 10,4,5,3\rangle$	0.262	1	9	$\langle 10,5,5,4\rangle$	0.099	1	16
$\langle 10,3,5,2\rangle$	1	3	8	$\langle 10,4,5,3\rangle$	0.52	2	10	$\langle 10,5,5,4\rangle$	0.21	2	17
$\langle 10,3,6,2\rangle$	0.671	1	6	$\langle 10,4,5,3\rangle$	0.71	3	11	$\langle 10,5,5,4\rangle$	0.31	3	18
$\langle 10,3,6,2\rangle$	0.962	2	6	$\langle 10,4,5,3\rangle$	0.829	4	12	$\langle 10,5,5,4\rangle$	0.409	4	19
$\langle 10,3,6,2\rangle$	1	3	6	$\langle 10,4,5,3\rangle$	0.948	5	12	$\langle 10,5,5,4\rangle$	0.52	5	20
$\langle 10,3,7,2\rangle$	0.817	1	5	$\langle 10,4,5,3\rangle$	0.98	6	12	$\langle 10,5,5,4\rangle$	0.619	6	21
$\langle 10,3,7,2\rangle$	1	2	5	$\langle 10,4,5,3\rangle$	1	7	12	$\langle 10,5,5,4\rangle$	0.671	7	22
$\langle 10,4,3,2\rangle$	0.333	1	7	$\langle 10,4,6,2\rangle$	0.881	1	5	$\langle 10,5,5,4\rangle$	0.762	8	23
$\langle 10,4,3,2\rangle$	0.667	2	8	$\langle 10,4,6,2\rangle$	1	2	5	$\langle 10,5,5,4\rangle$	0.81	9	23
$\langle 10,4,3,2\rangle$	0.867	3	9	$\langle 10,4,6,3\rangle$	0.452	1	7	$\langle 10,5,5,4\rangle$	0.869	10	24
$\langle 10,4,3,2\rangle$	1	4	9	$\langle 10,4,6,3\rangle$	0.829	2	8	$\langle 10,5,5,4\rangle$	0.901	11	24
$\langle 10,4,4,2\rangle$	0.552	1	6	$\langle 10,4,6,3\rangle$	0.971	3	8	$\langle 10,5,5,4\rangle$	0.921	12	24
$\langle 10,4,4,2\rangle$	0.919	2	7	$\langle 10,4,6,3\rangle$	1	4	8	$\langle 10,5,5,4\rangle$	0.94	13	24
$\langle 10,4,4,2\rangle$	1	3	7	$\langle 10,4,7,3\rangle$	0.667	1	5	$\langle 10,5,5,4\rangle$	1	14	25
$\langle 10,4,4,3\rangle$	0.119	1	14	$\langle 10,4,7,3\rangle$	1	2	6				
$\langle 10,4,4,3\rangle$	0.238	2	15	$\langle 10,5,3,2\rangle$	0.5	1	6				

3.5 Chapter overview

In this chapter, analytic upper bounds were established on the incomplete lottery number $L_\psi(m, n, t, k)$, where $0 < \psi \leq 1$ (see (3.14) and (3.17)). In Tables 3.1 and 3.2, it may be seen that the greedy bound (3.14) performs well compared to bounds (3.2)–(3.8) for the complete lottery problem, but it is worse than the bound (2.12). However, it must be noted that the greedy bound may be computed for any $L_1(m, n, t, k)$, while the bound (2.12) by Li & Van Rees [27] can only be computed if the values of the complete lottery numbers $L_1(m-1, n-1, t-1, k-1)$ and $L_1(m-1, n, t, k)$ are known or if good bounds on these numbers are known. It may also be seen from Tables 3.1 and 3.2 that the bound (3.8) by Gründlingh [16] is very good for small lottery instances, but it is comparatively weaker for larger lottery instances.

TABLE 3.6: *An analysis of the goodness of a bound for different lottery problem instances. The second column contains the degree, s, of each government ticket, while the third column contains the known complete lottery number, denoted by L_1. Finally, the last two columns contain the value of the greedy bound (3.14) and the difference between the greedy bound (3.14) and the known complete lottery number, respectively.*

Lottery	s	L_1	(3.14)	(3.14)$-L_1$
$\langle 10, 4, 6, 3\rangle$	95	4	8	4
$\langle 10, 4, 4, 3\rangle$	25	14	22	8
$\langle 10, 5, 4, 3\rangle$	66	7	12	5

As the degree of government tickets in the bipartite lottery graph model increases, the bound (3.14) appears to improve. This is illustrated in Table 3.6 in which it may be seen that as s increases, the difference between the bound and the known value of the complete lottery number decreases. In Tables 3.3, 3.4 and 3.5 it may be seen that as the value of ψ increases, the difference between the value of the greedy bound (3.17) and the value of $L_\psi(m, n, t, k)$ is non-increasing, which implies an improvement in the greedy bound as the value of ψ increases.

CHAPTER 4

A mathematical programming approach

Contents

The aim in this chapter is to present, implement and analyse integer programming formulations of the complete lottery problem, the incomplete lottery problem, and the resource utilisation problem, in fulfilment of Thesis Objectives II(a) and II(b) in §1.4.

The chapter is organised as follows. After casting the complete lottery problem in an Integer Linear Programming (ILP) problem setting in §4.1, small lotteries are analysed by solving the relevant ILP formulation instances. The incomplete lottery problem is similarly considered in §4.2, analysing the same small lottery instances. Finally the resource utilisation problem is considered for these same lottery instances in §4.3.

Apart from documenting solutions to the incomplete lottery and resource utilisation problems for small lottery instances in §4.4, another aim of this chapter is to investigate the boundaries of feasibility of an ILP approach towards solving these problems in terms of execution times in §4.5, in fulfilment of Thesis Objective II(c) in §1.4.

ILP formulations are constructed for both the incomplete lottery and resource utilisation problems associated with small lottery instances, where $6 \leq m \leq 10$, $m > \{n, t\} > k \geq 2$ and $L_1(m, n, t, k) > 1$, the problems are solved, and the results are documented in fulfilment of Thesis Objective II(d). The complete lottery number for each lottery considered is presented in Table 4.5 and the lotteries implicitly considered through isomorphism, are listed in Table 4.6.

4.1 An ILP formulation of the complete lottery problem

As mentioned in §2.1.4, Jans & Degraeve [20] suggested an ILP formulation for the lottery problem similar to the general set covering problem [8]. According to the authors, the main advantage of their method is that it provides a specific solution. This means that a participant in a lottery may use the solution to know exactly which ticket combinations to select.

Assume that a number may be assigned to each participant ticket (for example, by arranging the numbers of each participant ticket in increasing order, and subsequently ordering all participant tickets in lexicographically increasing order). Let

$$x_j = \begin{cases} 1 & \text{if participant ticket } j \text{ is in the playing set} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

be a binary decision variable. A playing set $\mathcal{L}_\psi(m,n,t,k)$ is constructed from the set $\Phi(\mathcal{U}_m,n)$ subject to the constraint that at least one participant ticket j should have at least k numbers in common with the winning ticket drawn randomly from the set $\Phi(\mathcal{U}_m,t)$. Let the neighbourhood, $\mathcal{N}[j]$, of a participant ticket j be the set of government tickets from $\Phi(\mathcal{U}_m,t)$ which have at least k numbers in common with the participant ticket. Suppose $\mathbf{A}$ is an $\binom{m}{t} \times \binom{m}{n}$ matrix whose (i,j)-th entry is

$$a_{ij} = \begin{cases} 1 & \text{if government ticket } i \text{ is in the neighbourhood} \\ & \text{of participant ticket } j \\ 0 & \text{otherwise.} \end{cases}$$

In the complete lottery problem, the union of the neighbourhoods of the tickets in a desired playing set must equal the entire set of t-subsets of U_m, *i.e.* $\bigcup_{i\in\mathcal{L}_\phi(m,n,t,k)} \mathcal{N}[i] = \Phi(\mathcal{U}_m,t)$. An optimal playing set may be found by adopting a *set covering problem* ILP formulation approach in which the objective is to

$$\text{minimise} \quad P(\boldsymbol{x}) = \sum_{1\le j\le\binom{m}{n}} x_j \tag{4.2}$$

subject to

$$\sum_{1\le j\le\binom{m}{n}} a_{ij}x_j \;\geq\; 1, \qquad 1 \le i \le \binom{m}{t} \tag{4.3}$$

$$x_j \;\in\; \{0,1\}, \qquad 1 \le j \le \binom{m}{n}. \tag{4.4}$$

The objective in (4.2) is to minimise the number of tickets in the participant's playing set. Constraint set (4.3) ensures that each government ticket is covered by at least one participant ticket. Constraint set (4.4) enforces the binary nature of the decision variables in (4.1).

One computational aspect that may be of interest is the number of decision variables included in each constraint in (4.3). This number is

$$\sum_{p=k}^{\min\{n,t\}} \binom{t}{p}\binom{m-t}{n-p}, \tag{4.5}$$

which may be found by counting the participant tickets that have at least k numbers in common with any given government ticket. For each of the small lottery instances investigated in this

chapter, the numbers of decision variables, non-trivial constraints and terms per non-trivial constraint for the complete lottery problem formulation are presented in Table 4.1. To some extent this identifies the size or scope of the respective ILP problems.

The following example contains the instance of the formulation (4.2)–(4.4) for the lottery $\langle 6, 4, 3, 2\rangle$, together with a solution.

TABLE 4.1: *Dimensions of the ILP formulation of the complete lottery problem. The column labelled DV represents the number of decision variables as described in (4.1), the column labelled NC represents the number of nontrivial constraints in (4.3), and the column labelled TC represents the number of terms per non-trivial constraint in (4.3), calculated from (4.5).*

Lottery	DV	NC	TC	Lottery	DV	NC	TC
$\langle 6,3,3,2\rangle$	20	20	10	$\langle 9,4,7,3\rangle$	126	36	105
$\langle 7,3,3,2\rangle$	35	35	13	$\langle 10,3,3,2\rangle$	120	120	22
$\langle 7,3,4,2\rangle$	35	35	22	$\langle 10,3,4,2\rangle$	120	210	40
$\langle 7,3,5,2\rangle$	35	21	30	$\langle 10,3,5,2\rangle$	120	252	60
$\langle 8,3,3,2\rangle$	56	56	16	$\langle 10,3,6,2\rangle$	120	210	80
$\langle 8,3,4,2\rangle$	56	70	28	$\langle 10,3,7,2\rangle$	120	120	98
$\langle 8,3,5,2\rangle$	56	56	40	$\langle 10,3,8,2\rangle$	120	45	112
$\langle 8,3,6,2\rangle$	56	28	50	$\langle 10,4,3,2\rangle$	210	120	70
$\langle 8,4,3,2\rangle$	70	56	35	$\langle 10,4,4,2\rangle$	210	210	115
$\langle 8,4,4,2\rangle$	70	70	53	$\langle 10,4,4,3\rangle$	210	210	25
$\langle 8,4,4,3\rangle$	70	70	17	$\langle 10,4,5,2\rangle$	210	252	155
$\langle 9,3,3,2\rangle$	84	84	19	$\langle 10,4,5,3\rangle$	210	252	55
$\langle 9,3,4,2\rangle$	84	126	34	$\langle 10,4,6,2\rangle$	210	210	185
$\langle 9,3,5,2\rangle$	84	126	50	$\langle 10,4,6,3\rangle$	210	210	95
$\langle 9,3,6,2\rangle$	84	84	65	$\langle 10,4,7,2\rangle$	210	120	203
$\langle 9,3,7,2\rangle$	84	36	77	$\langle 10,4,7,3\rangle$	210	120	140
$\langle 9,4,3,2\rangle$	126	84	51	$\langle 10,4,8,3\rangle$	210	45	182
$\langle 9,4,4,2\rangle$	126	126	81	$\langle 10,5,3,2\rangle$	252	120	126
$\langle 9,4,4,3\rangle$	126	126	21	$\langle 10,5,4,2\rangle$	252	210	186
$\langle 9,4,5,2\rangle$	126	126	105	$\langle 10,5,4,3\rangle$	252	210	66
$\langle 9,4,5,3\rangle$	126	126	45	$\langle 10,5,5,2\rangle$	252	252	226
$\langle 9,4,6,2\rangle$	126	84	120	$\langle 10,5,5,3\rangle$	252	252	126
$\langle 9,4,6,3\rangle$	126	84	75	$\langle 10,5,5,4\rangle$	252	252	26

Example 4.1 (The lottery $\langle 6, 4, 3, 2\rangle$) *Consider the lottery $\langle 6, 4, 3, 2\rangle$, in which the set*

$$\begin{aligned}\Phi(\mathcal{U}_6, 4) \;=\; & \{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\},\{1,2,4,5\},\{1,2,4,6\},\{1,2,5,6\},\{1,3,4,5\},\\ & \{1,3,4,6\},\{1,3,5,6\},\{1,4,5,6\},\{2,3,4,5\},\{2,3,4,6\},\{2,3,5,6\},\{2,4,5,6\},\\ & \{3,4,5,6\}.\end{aligned}$$

Let x_1 represent ticket $\{1,2,3,4\}$, x_2 represent ticket $\{1,2,3,5\}$, ..., and x_{15} represent ticket $\{3,4,5,6\}$. The objective in the complete lottery problem for the lottery $\langle 6, 4, 3, 2\rangle$ is to

$$\textit{minimise} \sum_{1\le j\le 15} x_j \tag{4.6}$$

subject to the constraints

$$
\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{11} + x_{12} + x_{13} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} + x_{11} + x_{12} + x_{14} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_9 + x_{10} + x_{11} + x_{13} + x_{14} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{14} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{15} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{13} + x_{15} &\geq 1 \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{15} &\geq 1 \\
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{14} + x_{15} &\geq 1 \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_8 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_1 + x_3 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_2 + x_3 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq 1 \\
x_i &\in \{0,1\}, \quad 1 \leq i \leq 15.
\end{aligned}
$$

If this formulation is solved by means of the commercial software suite LINGO 11 [28], the optimal solution variables which take the value 1 *are* x_1 *and* x_6*. If ticket* i *is the element listed in the* i^{th} *position in the set* $\Phi(\mathcal{U}_6, 4)$ *above, an optimal playing set for the participant is* $\{\{1,2,3,4\},\{1,2,5,6\}\}$. ■

4.2 An ILP formulation of the incomplete lottery problem

In this section, an ILP formulation of the incomplete lottery problem is presented. The formulation of the incomplete lottery problem is an extension of the ILP formulation of the complete lottery problem in §4.1. An additional constraint set, and an additional array of decision variables are introduced into the formulation.

The formulation for the complete lottery problem states the participant's playing set (which is a subset of $\Phi(\mathcal{U}_m, n)$) must cover all the possible government tickets (which are chosen from $\Phi(\mathcal{U}_m, t)$). However, in the incomplete lottery problem, not all of the $\binom{m}{t}$ possible government tickets need to be covered by the participant's playing set. The participant merely requires a certain proportion ψ of the $\binom{m}{t}$ government tickets to be covered. Define the decision variables

$$
y_i = \begin{cases} 1 & \text{if government ticket } i \text{ is in the neighbourhood} \\ & \text{of one of the tickets in the participant}'\text{s playing set} \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}
$$

The objective is to minimise the number of tickets in the participant's playing set, subject to the constraint that at least $\psi\binom{m}{t}$ government tickets must be covered by the playing set, that is to

$$\text{minimise} \quad P(\boldsymbol{x}) = \sum_{1 \leq j \leq \binom{m}{n}} x_j \tag{4.8}$$

subject to

$$\sum_{1 \leq j \leq \binom{m}{n}} a_{ij} x_j \geq y_i, \qquad 1 \leq i \leq \binom{m}{t} \tag{4.9}$$

$$\sum_{1 \leq i \leq \binom{m}{t}} y_i \geq \psi\binom{m}{t} \tag{4.10}$$

$$x_j \in \{0,1\}, \qquad 1 \leq j \leq \binom{m}{n} \tag{4.11}$$

$$y_i \in \{0,1\}, \qquad 1 \leq i \leq \binom{m}{t}, \tag{4.12}$$

where the decision variable x_j has the same meaning as in (4.1). Constraint set (4.9) ensures that at least one of the playing set tickets covering government ticket i is in the playing set. Constraint set (4.10) ensures that at least $\psi\binom{m}{t}$ government tickets are covered by the playing set. Constraint sets (4.11) and (4.12) enforce the binary nature of the decision variables in (4.1) and (4.7), respectively. Note that if $\psi = 1$, then the incomplete lottery problem formulation (4.8)–(4.12) reduces to the complete lottery problem formulation (4.2)–(4.4).

The following example contains the instance of the formulation (4.8)–(4.12) for the lottery $\langle 6,4,3,2\rangle$, together with a solution.

Example 4.2 (The lottery $\langle 6,4,3,2\rangle$) *The objective in the incomplete lottery problem formulation for the lottery $\langle 6,4,3,2\rangle$ is to*

$$\textit{minimise} \sum_{1 \leq j \leq 15} x_j \tag{4.13}$$

subject to the constraints

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{11} + x_{12} + x_{13} &\geq y_1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} + x_{11} + x_{12} + x_{14} &\geq y_2 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_9 + x_{10} + x_{11} + x_{13} + x_{14} &\geq y_3 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{14} &\geq y_4 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{15} &\geq y_5 \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{13} + x_{15} &\geq y_6 \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{15} &\geq y_7 \\
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{14} + x_{15} &\geq y_8 \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{14} + x_{15} &\geq y_9 \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{14} + x_{15} &\geq y_{10} \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{11} \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{12} \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_8 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{13}
\end{aligned}$$

$$
\begin{aligned}
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{14} \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{15} \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{16} \\
x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{17} \\
x_1 + x_3 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{18} \\
x_2 + x_3 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{19} \\
x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\geq y_{20} \\
\sum_{1 \leq j \leq 20} y_j &\geq 20\psi \\
x_i, y_j &\in \{0,1\}, \quad 1 \leq i \leq 15, \\
& \qquad\qquad 1 \leq j \leq 20.
\end{aligned}
$$

Solving the formulation above using LINGO 11 [28] yields the results in Table 4.2. Ticket 1 *covers exactly* $0.8 \times 20 = 16$ *government tickets, because it occurs in 16 of the 20 constraints associated with the binary covering matrix. Hence* $L_\psi(6,4,3,2) = 1$ *for all* $0 < \psi \leq 0.8$. *However,* $L_\psi(6,4,3,2) = 2$ *for all* $0.8 < \psi \leq 1$. ■

TABLE 4.2: *Results obtained from solving of the incomplete lottery problem for the lottery* $\langle 6,4,3,2 \rangle$ *after being modelled as an ILP formulation.*

ψ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Solution variables with a value of 1	x_1	x_1	x_1	x_1	x_1	x_1	x_1	x_1	x_6, x_7	x_1, x_6

4.3 An ILP formulation of the resource utilisation problem

In this section, an ILP formulation of the resource utilisation problem is presented. This ILP formulation is an adaptation of the ILP formulation of the incomplete lottery problem in the previous section.

The objective in the resource utilisation problem is to maximise the proportion of government tickets which have at least k numbers in common with at least one ticket in a participant's playing set of fixed cardinality ℓ. Or, more formally, to

$$\text{maximise} \quad P(\boldsymbol{x}) = \frac{\sum_{1 \leq i \leq \binom{m}{t}} y_i}{\binom{m}{t}} \tag{4.14}$$

subject to

$$\sum_{1 \leq j \leq \binom{m}{n}} a_{ij} x_j \geq y_i, \qquad 1 \leq i \leq \binom{m}{t} \tag{4.15}$$

$$\sum_{1 \leq j \leq \binom{m}{n}} x_j = \ell \tag{4.16}$$

$$x_j \in \{0,1\}, \qquad 1 \leq j \leq \binom{m}{n} \tag{4.17}$$

$$y_i \in \{0,1\}, \qquad 1 \leq i \leq \binom{m}{t}, \tag{4.18}$$

TABLE 4.3: *Dimensions of the ILP formulations of the incomplete lottery problem, and the resource utilisation problem. In the columns labelled DV, the number of decision variables in the formulation for the problem instance shown in the first column are displayed. In the columns labelled NC, the number of constraints in the formulation is displayed. In the last column labelled TC, the number of terms per constraint in (4.9) or (4.15) are shown. The value in the final column is calculated via (4.5).*

	DV			NC		TC
$\langle m, n, t, k \rangle$	in (4.1)	in (4.7)	Total	in (4.9)	Total	
$\langle 6, 3, 3, 2 \rangle$	20	20	40	20	21	10
$\langle 7, 3, 3, 2 \rangle$	35	35	70	35	36	13
$\langle 7, 3, 4, 2 \rangle$	35	35	70	35	36	22
$\langle 7, 3, 5, 2 \rangle$	35	21	56	21	22	30
$\langle 8, 3, 3, 2 \rangle$	56	56	112	56	57	16
$\langle 8, 3, 4, 2 \rangle$	56	70	126	70	71	28
$\langle 8, 3, 5, 2 \rangle$	56	56	112	56	57	40
$\langle 8, 3, 6, 2 \rangle$	56	28	84	28	29	50
$\langle 8, 4, 3, 2 \rangle$	70	56	126	56	57	35
$\langle 8, 4, 4, 2 \rangle$	70	70	140	70	71	53
$\langle 8, 4, 4, 3 \rangle$	70	70	140	70	71	17
$\langle 9, 3, 3, 2 \rangle$	84	84	168	84	85	19
$\langle 9, 3, 4, 2 \rangle$	84	126	210	126	127	34
$\langle 9, 3, 5, 2 \rangle$	84	126	210	126	127	50
$\langle 9, 3, 6, 2 \rangle$	84	84	168	84	85	65
$\langle 9, 3, 7, 2 \rangle$	84	36	120	36	37	77
$\langle 9, 4, 3, 2 \rangle$	126	84	210	84	85	51
$\langle 9, 4, 4, 2 \rangle$	126	126	252	126	127	81
$\langle 9, 4, 4, 3 \rangle$	126	126	252	126	127	21
$\langle 9, 4, 5, 2 \rangle$	126	126	252	126	127	105
$\langle 9, 4, 5, 3 \rangle$	126	126	252	126	127	45
$\langle 9, 4, 6, 2 \rangle$	126	84	210	84	85	120
$\langle 9, 4, 6, 3 \rangle$	126	84	210	84	85	75
$\langle 9, 4, 7, 3 \rangle$	126	36	162	36	37	105
$\langle 10, 3, 3, 2 \rangle$	120	120	240	120	121	22
$\langle 10, 3, 4, 2 \rangle$	120	210	330	210	211	40
$\langle 10, 3, 5, 2 \rangle$	120	252	372	252	253	60
$\langle 10, 3, 6, 2 \rangle$	120	210	330	210	211	80
$\langle 10, 3, 7, 2 \rangle$	120	120	240	120	121	98
$\langle 10, 3, 8, 2 \rangle$	120	45	165	45	46	112
$\langle 10, 4, 3, 2 \rangle$	210	120	330	120	121	70
$\langle 10, 4, 4, 2 \rangle$	210	210	420	210	211	115
$\langle 10, 4, 4, 3 \rangle$	210	210	420	210	211	25
$\langle 10, 4, 5, 2 \rangle$	210	252	462	252	253	155
$\langle 10, 4, 5, 3 \rangle$	210	252	462	252	253	55
$\langle 10, 4, 6, 2 \rangle$	210	210	420	210	211	185
$\langle 10, 4, 6, 3 \rangle$	210	210	420	210	211	95
$\langle 10, 4, 7, 2 \rangle$	210	120	330	120	121	203
$\langle 10, 4, 7, 3 \rangle$	210	120	330	120	121	140
$\langle 10, 4, 8, 3 \rangle$	210	45	255	45	46	182
$\langle 10, 5, 3, 2 \rangle$	252	120	372	120	121	126
$\langle 10, 5, 4, 2 \rangle$	252	210	462	210	211	186
$\langle 10, 5, 4, 3 \rangle$	252	210	462	210	211	66
$\langle 10, 5, 5, 2 \rangle$	252	252	504	252	253	226
$\langle 10, 5, 5, 3 \rangle$	252	252	504	252	253	126
$\langle 10, 5, 5, 4 \rangle$	252	252	504	252	253	26

where the decision variables x_j and y_i have the same meanings as in (4.1) and (4.7), respectively. Constraint set (4.15) is exactly the same as constraint (4.9) in the ILP formulation of the incomplete lottery problem, while constraint set (4.16) ensures that the playing set has fixed cardinality ℓ. Finally, the constraint sets (4.17) and (4.18) enforce the binary nature of the decision variables in (4.1) and (4.7), respectively.

The numbers of decision variables, non-trivial constraints and terms per non-trivial constraint in the resource utilisation problem are exactly the same as those for the incomplete lottery problem, and may be found in Table 4.3 for small values of m, n, t and k.

The following example contains the instance of the formulation (4.8)–(4.12) for the lottery $\langle 6, 4, 3, 2\rangle$, together with a solution to the problem instance.

Example 4.3 (The lottery $\langle 6, 4, 3, 2\rangle$) *The objective in the resource utilisation problem for the lottery $\langle 6, 4, 3, 2\rangle$ is to*

$$\textit{maximise } \frac{1}{20} \sum_{1 \le j \le 20} y_j \tag{4.19}$$

subject to the constraints

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{11} + x_{12} + x_{13} &\ge y_1 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} + x_{11} + x_{12} + x_{14} &\ge y_2 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_9 + x_{10} + x_{11} + x_{13} + x_{14} &\ge y_3 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{14} &\ge y_4 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{15} &\ge y_5 \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{13} + x_{15} &\ge y_6 \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{13} + x_{15} &\ge y_7 \\
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{14} + x_{15} &\ge y_8 \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{12} + x_{14} + x_{15} &\ge y_9 \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{14} + x_{15} &\ge y_{10} \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{11} \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{12} \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_8 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{13} \\
x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{14} \\
x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{15} \\
x_2 + x_3 + x_4 + x_5 + x_6 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{16} \\
x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{17} \\
x_1 + x_3 + x_5 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{18} \\
x_2 + x_3 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{19} \\
x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &\ge y_{20} \\
\sum_{1 \le j \le 15} x_j &= \ell \\
x_i, y_j &\in \{0,1\}, \quad 1 \le i \le 15, \\
& \qquad\qquad 1 \le j \le 20.
\end{aligned}$$

TABLE 4.4: *Results obtained from solving the resource utilisation problem of the lottery* $\langle 6,4,3,2\rangle$ *after being modelled as an ILP formulation. The column labelled* $P(\boldsymbol{x})$ *represents the optimal resource utilisation for each possible playing set cardinality.*

ℓ	Solution variables with a value of 1	$P(\boldsymbol{x})$
1	$y_2, y_3, y_4, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}, y_{17}, y_{18}, y_{19}, y_{20}, x_{14}$	0.8
2	$y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}, y_{17}, y_{18}, y_{19}, y_{20}, x_3, x_{10}$	1

Solving the formulation above using LINGO 11 [28] yields the results in Table 4.4. It may be seen from these results, that when the cardinality of the participant's playing set is 1*, the largest proportion of government tickets that may possibly be covered is* $\frac{16}{20} = 0.8$*, while when the cardinality of the participant's playing set is* 2*, the largest proportion of government tickets that may possibly be covered is* $\frac{20}{20} = 1$*; this may be achieved by selecting the third and tenth ticket available for the participant to include into his/her playing set, i.e.* $\{\{1,2,3,6\},\{3,4,5,6\}\}$*.* ■

4.4 Analysis of results

As mentioned, the problem instances in the numerical examples of the previous sections were solved by means of LINGO 11 [28], using LINGO script files. There are many advantages of using LINGO 11 scripts. At the time of writing, LINGO 11 is a relatively new and extremely powerful tool which may be used to solve ILP problems. LINGO script files support the use of arrays, which allow for a large formulation to be expressed in a few, relatively short lines of code. An example of such a script file for the formulation of the resource utilisation problem for the lottery $\langle 6,4,3,2\rangle$ is presented in Code Example 4.1.

Code Example 4.1 (LINGO code example)

```
! Output the solution report generated by LINGO
SET TERSEO 1
! Set a solver time limit of 14400 second=4hrs
SET TIMLIM 14400
! Do not allow dialog screens to appear
SET ERRDLG 0
! Force the solver to use an iteration-based stopping limit
! for heuristics used at each node of the branch and bound tree
apiset 369 int 2
! Set heuristics iteration limit to 100
apiset 334 int 100
! Begins input of a new LINGO model
model:
data:
! The adjacency matrix has dimensions 20x15
numrow=20;
numcol=15;
enddata
! Define y variables to be the associated with the rows
! of the input matrix, and the x variables to be
```

```
! associated with the columns of the of the model
sets:
row/1..numrow/:y; !index i;
col/1..numcol/:x; !index j;
rxc(row,col):bij;
endsets
! Retieve the adjacency matrix from file
data:
bij=@file('matrix6432.txt');
enddata
! Objective function declaration
max=@sum(row(i):y(i))/numrow;
! The adjacency constraints
@for(row(i):@sum(col(j):bij(i,j)*x(j))>=y(i));
! The playing set cardinality must be fixed
@sum(col(j):x(j))=1.0;
! All variables in the formulation are binary
@for(col(j):@bin(x(j)));
@for(row(i):@bin(y(i)));
end
! Solve the model
GO
! Open a file
DIVERT 1.TXT
! Send solution to the file
SOLUTION
! Send total solver time to file
TIME
! Close solution file
RVRT
! The playing set cardinality increases by 1
ALTER ALL '1.0'2.0'
! Solve the model
GO
! Open a file
DIVERT 2.TXT
! Send solution to the file
SOLUTION
TIME
! Close solution file
RVRT
! Restore parameters
SET TERSEO 0
SET ECHOIN 0
```

■

In the above code, the resource utilisation problem of the lottery $\langle 6, 4, 3, 2\rangle$ is formulated as an ILP and is solved for $\ell = 1$, and then for $\ell = 2$. For each value of ℓ the results are written to a file, and in each case the execution time is limited to 4 hours (14 400 seconds). This time limit was also adopted for all 38 lottery instances in Table 4.5.

TABLE 4.5: *The small (non-isomorphic) lottery instances which are investigated in this chapter, together with the known complete lottery number, L_1, for each instance.*

$\langle m,n,t,k\rangle$	L_1	$\langle m,n,t,k\rangle$	L_1	$\langle m,n,t,k\rangle$	L_1	$\langle m,n,t,k\rangle$	L_1	$\langle m,n,t,k\rangle$	L_1
$\langle 6,3,3,2\rangle$	2	$\langle 8,4,4,3\rangle$	6	$\langle 9,4,5,2\rangle$	2	$\langle 10,4,3,2\rangle$	4	$\langle 10,5,3,2\rangle$	2
$\langle 7,3,3,2\rangle$	4	$\langle 9,3,3,2\rangle$	7	$\langle 9,4,5,3\rangle$	5	$\langle 10,4,4,2\rangle$	3	$\langle 10,5,4,2\rangle$	2
$\langle 7,3,4,2\rangle$	2	$\langle 9,3,4,2\rangle$	3	$\langle 9,4,6,3\rangle$	2	$\langle 10,4,4,3\rangle$	14	$\langle 10,5,4,3\rangle$	7
$\langle 8,3,3,2\rangle$	5	$\langle 9,3,5,2\rangle$	3	$\langle 10,3,3,2\rangle$	8	$\langle 10,4,5,2\rangle$	2	$\langle 10,5,5,2\rangle$	2
$\langle 8,3,4,2\rangle$	3	$\langle 9,3,6,2\rangle$	2	$\langle 10,3,4,2\rangle$	5	$\langle 10,4,5,3\rangle$	7	$\langle 10,5,5,3\rangle$	2
$\langle 8,3,5,2\rangle$	2	$\langle 9,4,3,2\rangle$	4	$\langle 10,3,5,2\rangle$	3	$\langle 10,4,6,2\rangle$	2	$\langle 10,5,5,4\rangle$	14
$\langle 8,4,3,2\rangle$	2	$\langle 9,4,4,2\rangle$	2	$\langle 10,3,6,2\rangle$	3	$\langle 10,4,6,3\rangle$	4		
$\langle 8,4,4,2\rangle$	2	$\langle 9,4,4,3\rangle$	9	$\langle 10,3,7,2\rangle$	2	$\langle 10,4,7,3\rangle$	2		

For each lottery instance, the incomplete lottery problem is formulated as an ILP and solved for each value of $\psi \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. The resource utilisation problem is also formulated as an ILP and solved for each possible value of $\ell \in \mathbb{N}$ until a value of $\Psi_\ell(m,n,t,k) = 1$ is found for each problem instance. The results appear in Tables B.1 and C.1. From these tables, 51 problem instances reached the time limit of 14 400 seconds. It was found through experimental analysis that if the time limit is not enforced, some problem instances may take many days or even weeks to solve.

TABLE 4.6: *Lottery instances which are isomorphic to the small lottery instances investigated in this chapter, as listed in Table 4.5. Each entry in this table is of the form $\langle m,n,t,k\rangle \cong \langle m, m-n, m-t, m+k-n-t\rangle$.*

$\langle 7,4,4,3\rangle \cong \langle 7,3,3,2\rangle$	$\langle 9,5,5,3\rangle \cong \langle 9,4,4,2\rangle$	$\langle 10,6,6,4\rangle \cong \langle 10,4,4,2\rangle$
$\langle 7,4,3,2\rangle \cong \langle 7,3,4,2\rangle$	$\langle 9,5,5,4\rangle \cong \langle 9,4,4,3\rangle$	$\langle 10,6,6,5\rangle \cong \langle 10,4,4,3\rangle$
$\langle 8,5,5,4\rangle \cong \langle 8,3,3,2\rangle$	$\langle 9,5,4,2\rangle \cong \langle 9,4,5,2\rangle$	$\langle 10,6,5,3\rangle \cong \langle 10,4,5,2\rangle$
$\langle 8,5,4,3\rangle \cong \langle 8,3,4,2\rangle$	$\langle 9,5,4,3\rangle \cong \langle 9,4,5,3\rangle$	$\langle 10,6,5,4\rangle \cong \langle 10,4,5,3\rangle$
$\langle 8,5,3,2\rangle \cong \langle 8,3,5,2\rangle$	$\langle 9,5,3,2\rangle \cong \langle 9,4,6,3\rangle$	$\langle 10,6,4,2\rangle \cong \langle 10,4,6,2\rangle$
$\langle 8,4,5,3\rangle \cong \langle 8,4,3,2\rangle$	$\langle 10,7,7,6\rangle \cong \langle 10,3,3,2\rangle$	$\langle 10,6,4,3\rangle \cong \langle 10,4,6,3\rangle$
$\langle 9,6,6,5\rangle \cong \langle 9,3,3,2\rangle$	$\langle 10,7,6,5\rangle \cong \langle 10,3,4,2\rangle$	$\langle 10,6,3,2\rangle \cong \langle 10,4,7,3\rangle$
$\langle 9,6,5,4\rangle \cong \langle 9,3,4,2\rangle$	$\langle 10,7,5,4\rangle \cong \langle 10,3,5,2\rangle$	$\langle 10,5,7,4\rangle \cong \langle 10,5,3,2\rangle$
$\langle 9,6,4,3\rangle \cong \langle 9,3,5,2\rangle$	$\langle 10,7,4,3\rangle \cong \langle 10,3,6,2\rangle$	$\langle 10,5,6,3\rangle \cong \langle 10,5,4,2\rangle$
$\langle 9,6,3,2\rangle \cong \langle 9,3,6,2\rangle$	$\langle 10,7,3,2\rangle \cong \langle 10,3,7,2\rangle$	$\langle 10,5,6,4\rangle \cong \langle 10,5,4,3\rangle$
$\langle 9,5,6,4\rangle \cong \langle 9,4,3,2\rangle$	$\langle 10,6,7,5\rangle \cong \langle 10,4,3,2\rangle$	

For integer programing problems, it is known that as the number of variables in the ILP formulation increases, the execution time may increase (even exponentially) because more branches in the branch-and-bound tree may need to be evaluated. A pessimistic indication of the number of branches contained in the branch-and-bound tree, and hence the number of candidate solutions which may need to be considered when solving the incomplete lottery problem, is

$$\sum_{\ell=1}^{L_1(m,n,t,k)} \binom{\binom{m}{n}}{\ell} \binom{\binom{m}{t}}{\left\lceil \psi \binom{m}{t} \right\rceil}. \tag{4.20}$$

The reasoning behind the expression in (4.20) is that, in the worst case, all $\binom{\binom{m}{n}}{\ell}$ possible playing sets may need to be considered, and each time a playing set is considered, each combination of $\left\lceil \psi \binom{m}{t} \right\rceil$ government tickets may need to be considered. Similarly, a pessimistic indication

of the number of branches which may have to be traversed in the branch-and-bound tree, and hence the number of candidate solutions which may need to be considered when solving for the resource utilisation problem is

$$\sum_{i=1}^{\binom{m}{t}} \binom{\binom{m}{t}}{i} \binom{\binom{m}{n}}{\ell}. \tag{4.21}$$

The reasoning behind the expression in (4.21) is that, in the worst case, all possible government tickets which could possibly have k numbers in common with at least one ticket in the playing set may need to be considered, and each time a set of government tickets is considered, all possible playing sets of cardinality ℓ may need to be considered.

4.4.1 Analysis of results for the incomplete lottery problem

As an example, the problem instances involving the lotteries $\langle 9, 4, 4, 3\rangle$ and $\langle 10, 5, 4, 3\rangle$ are formulated as ILP's and solved. The results are presented in Tables 4.7 and 4.8, respectively.

It may be seen in Figures 4.1 and 4.2 that as the desired win probability of the participant increases, the number of tickets required in the playing set displays a convex upward trend. This is an interesting observation for the participant, because it shows a relationship between the payoff and the participant's attitude towards risk. If the participant tends to be risk seeking, he/she participates in a lottery draw without fear of losing and might select a playing set of low cardinality. If the participant is risk-averse, he/she participates in a lottery draw with a great fear of losing and therefore would like to increase his/her probability of winning a k-prize. Therefore, a participant who is risk-averse may select a playing set of larger cardinality. The convex upward nature of the graph indicates that as the desired probability of winning a k-prize increases (implying a decrease in risk of losing), the playing set cardinality which achieves this desired win probability does not decrease.

TABLE 4.7: *Results obtained when solving for the value $L_\psi(9, 4, 4, 3)$ when the problem is formulated as an ILP. The first column contains the value of the required probability-of-win, ψ. The second column contains an example of a playing set which at least achieves the required probability-of-win value. The third column contains the actual probability of win value, ψ', achieved by the playing set in the second column. The fourth column contains the associated number of branches in the branch-and-bound tree (bounded above by (4.20)). Finally, the fifth column contains the time in seconds required to find the solution.*

ψ	Playing set	ψ'	Branches	Time
0.1	$\{\{1,2,3,4\}\}$	0.167	0	1.949
0.3	$\{\{1,2,5,6\},\{1,3,4,8\}\}$	0.333	0	1.669
0.5	$\{\{1,2,3,9\},\{1,4,6,7\},\{4,5,8,9\}\}$	0.5	0	0.429
0.6	$\{\{1,2,4,9\},\{1,3,5,9\},\{2,3,7,8\},\{3,4,6,8\}\}$	0.603	61	1.179
0.7	$\{\{1,2,5,9\},\{1,3,6,7\},\{1,6,8,9\},\{2,3,4,8\},$ $\{4,5,6,7\}\}$	0.738	420	2.039
0.8	$\{\{1,2,7,8\},\{1,4,5,8\},\{1,4,7,9\},\{2,3,4,8\},$ $\{2,5,6,9\},\{3,5,6,7\}\}$	0.825	781 103	14 400
0.9	$\{\{1,2,4,9\},\{1,3,5,7\},\{1,6,7,8\},\{2,3,4,8\},$ $\{2,3,6,9\},\{4,5,6,9\},\{5,7,8,9\}\}$	0.904	886 618	14 400
1	$\{\{1,2,5,9\},\{1,3,4,5\},\{1,3,6,7\},\{1,3,6,8\},$ $\{2,3,5,9\},\{2,4,6,9\},\{2,4,7,8\},\{4,7,8,9\},$ $\{5,6,7,8\}\}$	1	41 491	4 256.259

TABLE 4.8: *Results obtained when solving for the value $L_\psi(10, 5, 4, 3)$ when the problem is formulated as an ILP. The first column contains the value of the required probability-of-win value, ψ. The second column contains an example of a playing set which at least achieves the required probability-of-win value. The third column contains the actual probability of win value, ψ', achieved by the playing set in the second column. The fourth column contains the associated number of branches in the branch-and-bound tree. Finally, the fifth column contains the time in seconds required to find the solution.*

ψ	Playing set	ψ'	Branches	Time
0.2	$\{\{1,2,3,4,5\}\}$	0.262	0	1.040
0.5	$\{\{1,2,3,4,8\},\{2,5,6,7,9\}\}$	0.523	2	1.790
0.7	$\{\{1,3,4,6,9\},\{1,5,6,8,10\},\{2,4,7,9,10\}\}$	0.7	1	2.339
0.8	$\{\{1,3,7,8,10\},\{1,4,5,6,9\},\{2,3,4,5,7\},$ $\{2,3,5,8,10\}\}$	0.8	71	8.979
0.9	$\{\{1,3,4,5,6\},\{1,5,8,9,10\},\{2,3,7,8,9\},$ $\{2,4,6,7,8\},\{2,4,6,7,10\}\}$	0.919	51 446	14 400
1	$\{\{1,2,6,8,9\},\{1,3,4,5,7\},\{1,3,5,7,10\},$ $\{2,3,4,5,8\},\{2,3,4,8,10\},\{2,6,7,8,9\}$ $\{4,5,6,9,10\}\}$	1	117 287	14 400

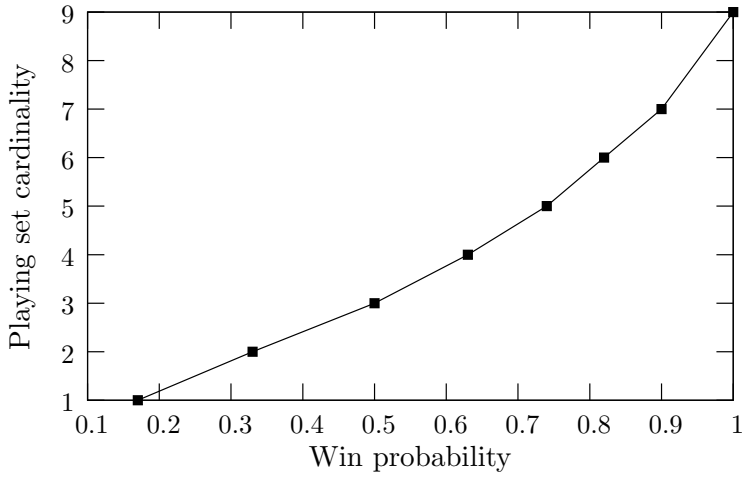


FIGURE 4.1: *Playing set cardinality as a function of win probability for the incomplete lottery problem associated with the lottery $\langle 9, 4, 4, 3\rangle$.*

Another aspect of interest is the execution time required to solve the incomplete lottery problem. It may be seen in Figures 4.3 and 4.4 that as the participant's desired win probability increases, it is often the case that the execution time of solving the problem is longer. One reason for this is that the number of different ticket combinations comprising the playing set which satisfy the participant's desire for a win probability of at least ψ, may become significantly larger (resulting in a larger number of computations required) as the desired win probability increases. In terms of solving the problems when modelled as ILPs, the execution time is affected by the number of branches in the branch-and-bound tree; however, it is also affected by the manner in which *LINGO* determines which variables to branch on. Typically, as may be seen in Figures 4.3 and 4.4, as the number of branches in the branch-and-bound tree increases, so does the execution time. However, this is not *always* the case. As an example, consider the cases when the optimal values of $L_{0.5}(10, 5, 4, 3)$ and $L_{0.7}(10, 5, 4, 3)$ are sought. It took approximately 1.8 seconds to solve for $L_{0.5}(10, 5, 4, 3)$ and the branch-and-bound tree consisted of 2 branches; however, it took approximately 2.3 seconds to solve for $L_{0.7}(10, 5, 4, 3)$ but the branch-and-bound tree consisted

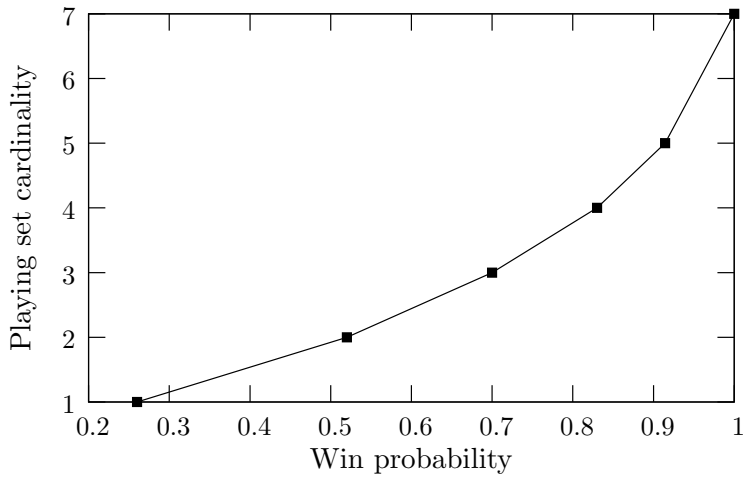


FIGURE 4.2: *Playing set cardinality as a function of win probability for the incomplete lottery problem associated with the lottery* $\langle 10, 5, 4, 3\rangle$.

of only 1 branch. The slight discrepancy in the execution time in those cases may be attributed to the above-mentioned manner in which *LINGO* determines which variables to branch on.

In Figure 4.3, the execution time taken to find the value of $L_1(9, 4, 4, 3)$ is 4 256.3 seconds, but in Figure 4.4, the execution time required to find the value of $L_1(10, 5, 4, 3)$ reaches the time limit of 14 400 seconds (or 4 hours), indicating that the problem was not solved to completion and an upper bound on $L_1(10, 5, 4, 3)$ was found. This may be explained by noting that only 41 491 branches are required in the branch-and-bound tree in order to find the value of $L_1(9, 4, 4, 3)$ (compared to at least 886 618 branches being required to find the value of $L_{0.9}(9, 4, 4, 3)$) and at least 117 287 branches are required in the branch-and-bound tree when the value of $L_1(10, 5, 4, 3)$ is sought (compared to at least 51 446 branches required to find the value of $L_1(10, 5, 4, 3)$). This explains the difference in the graphs in Figures 4.3 and 4.4.

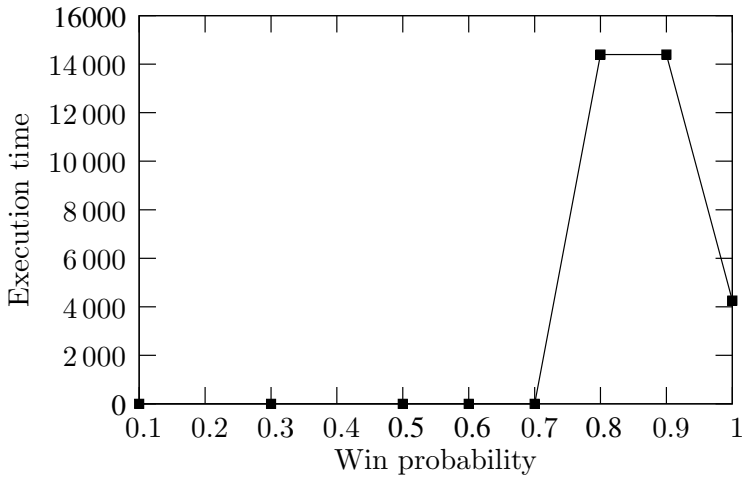


FIGURE 4.3: *Execution time as a function of win probability for the incomplete lottery problem associated with the lottery* $\langle 9, 4, 4, 3\rangle$.

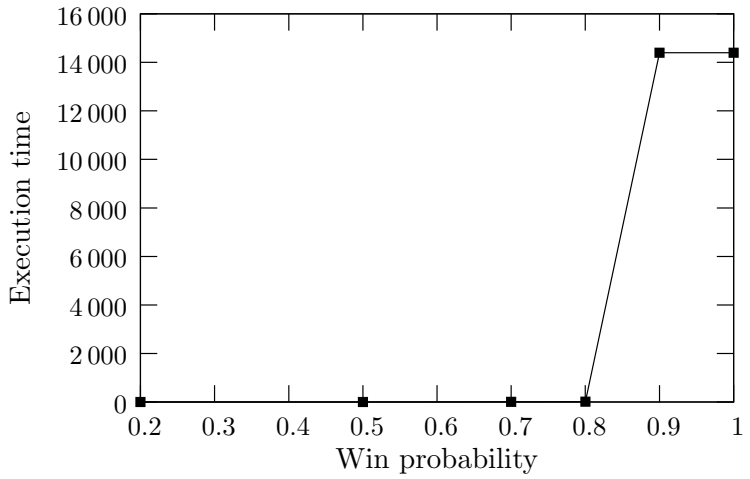


FIGURE 4.4: *Execution time as a function of win probability for the incomplete lottery problem associated with the lottery* $\langle 10, 5, 4, 3\rangle$.

TABLE 4.9: *Results obtained when solving for the value* $\Psi_\ell(9, 4, 4, 3)$ *when the problem is formulated as an ILP. The first column contains the fixed playing set cardinality* ℓ*. The second column contains an example of a playing set which achieves the probability-of-win value* Ψ_ℓ *in the third column. The fourth column contains the associated number of branches in the branch-and-bound tree. Finally, the fifth column contains the time in seconds required to find the solution.*

ℓ	Playing set	Ψ_ℓ	Branches	Time
1	$\{\{5,7,8,9\}\}$	0.167	0	0.319
2	$\{\{1,3,8,9\},\{3,4,6,7\}\}$	0.333	0	0.368
3	$\{\{1,3,5,7\},\{2,4,5,6\},\{6,7,8,9\}\}$	0.5	0	0.449
4	$\{\{1,2,3,9\},\{1,2,6,8\},\{3,4,5,6\},\{4,7,8,9\}\}$	0.635	325 782	9 819.239
5	$\{\{1,2,7,9\},\{1,6,7,8\},\{2,4,5,8\},\{3,4,6,9\},$ $\{3,5,7,8\}\}$	0.738	277 847	14 400
6	$\{\{1,2,3,6\},\{1,4,6,9\},\{2,4,6,7\},\{2,5,8,9\},$ $\{3,4,5,8\},\{3,7,8,9\}\}$	0.825	353 245	14 400
7	$\{\{1,3,6,7\},\{1,4,5,7\},\{1,4,6,8\},\{1,7,8,9\},$ $\{2,3,4,9\},\{2,3,5,8\},\{2,5,6,9\}\}$	0.913	428 601	14 400
8	$\{\{1,2,8,9\},\{1,3,4,9\},\{1,5,6,7\},\{2,3,6,7\},$ $\{2,4,5,6\},\{2,4,5,7\},\{3,5,8,9\},\{4,6,7,8\}\}$	0.960	657 794	144 00
9	$\{\{1,2,5,7\},\{1,3,4,7\},\{1,4,5,6\},\{1,6,8,9\},$ $\{2,3,4,9\},\{2,3,6,8\},\{2,4,7,8\},\{3,5,8,9\},$ $\{5,6,7,9\}\}$	1	156	1.559

4.4.2 Analysis of results for the resource utilisation problem

As an example, the results obtained from solving the two resource utilisation problem instances for the lotteries $\langle 9, 4, 4, 3\rangle$ and $\langle 10, 5, 4, 3\rangle$, adopting an ILP formulation approach, are presented in Tables 4.9 and 4.10, respectively.

The graphs of resource utilisation as a function of playing set cardinality, shown in Figures 4.5 and 4.6, roughly exhibit a concave downward shape. The reason for this shape is that, as the playing set cardinality increases, the neighbourhoods of those tickets in the playing set overlap more with each other, and subsequently the *improvement* in the resource utilisation tends to

TABLE 4.10: *Results obtained when solving for the value $\Psi_\ell(10,5,4,3)$ when the problem is formulated as an ILP. The first column contains the fixed playing set cardinality ℓ. The second column contains an example of a playing set which achieves the probability-of-win value Ψ_ℓ in the third column. The fourth column contains the associated number of branches in the branch-and-bound tree. Finally, the fifth column contains the time in seconds required to find the solution.*

ℓ	Playing set	Ψ_ℓ	Branches	Time
1	$\{\{1,2,3,4,5\}\}$	0.262	0	2.359
2	$\{\{1,2,3,5,6\},\{1,4,8,9,10\}\}$	0.524	0	4.689
3	$\{\{1,2,4,5,10\},\{3,5,7,8,9\},\{4,6,8,9,10\}\}$	0.7	56 333	14 400
4	$\{\{1,4,7,8,10\},\{1,5,6,7,9\},\{2,3,4,6,9\},$ $\{2,3,5,8,10\}\}$	0.876	82 174	14 400
5	$\{\{1,3,4,5,7\},\{1,3,6,9,10\},\{2,3,7,8,10\},$ $\{2,4,6,8,9\},\{2,5,6,8,9\}\}$	0.924	104 717	14 400
6	$\{\{1,2,5,6,9\},\{1,3,4,7,8\},\{1,5,7,9,10\},$ $\{2,3,4,8,10\},\{2,5,6,7,9\},\{3,4,6,8,10\}\}$	0.986	130 990	14 400
7	$\{\{1,2,3,4,8\},\{1,4,5,9,10\},\{1,4,7,9,10\},$ $\{1,5,6,7,10\},\{2,3,5,7,8\},\{2,4,6,8,9\},$ $\{3,4,6,9,10\}\}$	1	3	1.099

decrease. The associated execution time graphs are similar to those observed when solving the incomplete lottery problem; it is often the case that as the playing set cardinality increases, the time taken to find an optimal (or near optimal) solution increases. One possible reason for this phenomenon is that as the playing set cardinality increases, it is possible that more combinations of participant tickets are assessed. This implies that more iterations are required to build the branch-and-bound tree, and hence more calculations are carried out by the processor, thus causing the execution time to increase regardless of the speed of the processor. In Figure 4.6 the graph appears to be linear from point 2 to point 4. This may be explained as follows. The neighbourhoods of the tickets in the playing set of cardinality 2 have no tickets in common with each other, however, the neighbourhoods of the tickets of the playing set of cardinality 3 have, in total, 18 tickets in common (an increase of 18 tickets). Also, the neighbourhoods of the tickets of the playing set of cardinality 4 have, in total, 36 tickets in common (an increase of 18 tickets). Therefore, the increase in the number of tickets shared by the neighbourhoods of the tickets in the playing set is linear, hence the linear shape of the graph from $\ell = 2$ to $\ell = 4$. Also in Figure 4.6, there seems to be a "kink" in the graph where the playing set cardinality is equal to 5. This may be explained as follows. As mentioned, the neighbourhoods of the tickets of the playing set of cardinality 4 have, in total, 36 tickets in common. The neighbourhoods of the tickets in the playing set of cardinality 5 have, in total, 81 tickets in common (an increase of $81-36=45$), the neighbourhoods of the tickets in the playing set of cardinality 6 have, in total, 123 tickets in common (an increase of $123-81=42$), and neighbourhoods of the tickets in the playing set of cardinality 7 have, in total, 81 tickets in common (an increase of $175-123=52$). Therefore, the increase in the number of tickets shared by the neighbourhoods of the tickets in the playing set is *not* linear, hence the non-linear shape of the graph from $\ell = 4$ to $\ell = 7$.

4.5 Boundaries of feasibility via an ILP approach

In this section, the boundaries of feasibility of an ILP approach towards solving the incomplete lottery problem and the resource utilisation problem are explored. In Tables 4.7–4.10 in §4.4,

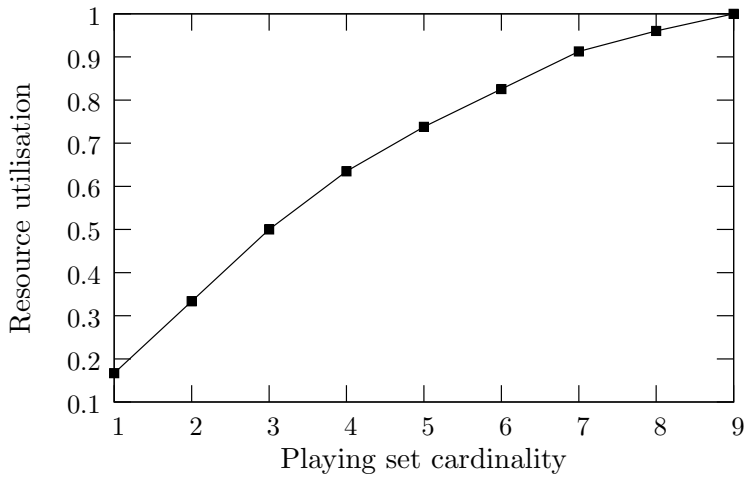


FIGURE 4.5: $\Psi_\ell(9,4,4,3)$ *as a function of playing set cardinality. The values of resource utilisation associated with playing set cardinalities of* 1, 2, 3, 4 *and* 9 *represent the optimal value of* $\Psi_\ell(9,4,4,3)$*, and the resource utilisation values associated playing set cardinalities of* 5, 6, 7 *and* 8 *represent lower bounds on the value of* $\Psi_\ell(9,4,4,3)$ *because execution time reached the predefined time limit of* 14 400 *seconds.*

it may be seen that certain small instances of the incomplete lottery problem and the resource utilisation problem require more than four hours to solve for an optimal solution. In this section, a problem instance which reached the four hour time limit is considered, and the ILP formulation of that problem is altered by fixing variables explicitly in the hope of finding a globally optimal solution within a relatively small amount of time. The same problem is also allowed to solve to completion, and the results are documented.

As an example, consider the case where the variables x_1 and x_2 are fixed explicitly in the formulation. They may possibly be fixed as $x_1 = 0, x_2 = 0$; as $x_1 = 1, x_2 = 1$; as $x_1 = 1, x_2 = 0$; or as $x_1 = 1, x_2 = 1$. The branch-and-bound algorithm will not be required to branch on these two variables. This therefore results in fewer branches in the branch-and-bound tree, which may result in a decrease in the execution time associated with solving instances of the incomplete lottery problem and the resource utilisation problem. If r variables are chosen to be fixed explicitly, 2^r different formulations have to be solved in order to find the best objective function value. If more variables are explicitly fixed, a shorter execution time may be required to find a solution. However, if more variables are explicitly fixed, an exponentially larger number of formulations have to be solved, which may require more processing time and memory. Due to time and resource limitations, only four variables are explicitly fixed in the example which follows. This requires $2^4 = 16$ problem formulations to be solved.

When selecting the four variables to be fixed explicitly, it may not be desirable to select the variables x_1, x_2, x_3, x_4 (which represent the first four tickets, arranged lexicographically) because it is highly unlikely that all those variables will be included in an optimal playing set due to the large overlap of their neighbourhoods. Therefore, a better technique would be a naive partitioning of $\Phi(\mathcal{U}_m, n)$ into four subsets of equal size, and using a variable from each subset. For example, in the lottery $\langle 10,5,4,3\rangle$, for which $\binom{10}{5} = 252$, the variables

$$x_1,\ x_{1+\frac{252}{4}},\ x_{1+\frac{252}{4}+\frac{252}{4}},\ x_{1+\frac{252}{4}+\frac{252}{4}+\frac{252}{4}}$$

i.e. x_1, x_{64}, x_{127}, x_{190} may be fixed explicitly. It may be more likely that the set of tickets represented by these variables will cover more government tickets than the set represented by

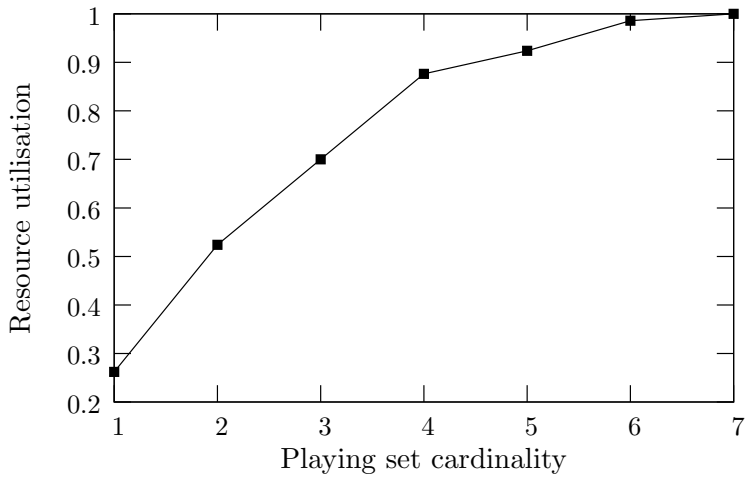


FIGURE 4.6: $\Psi_\ell(10,5,4,3)$ *as a function of playing set cardinality. The values of resource utilisation associated with playing set cardinalities of* 1, 2, *and* 7 *represent the optimal value of* $\Psi_\ell(10,5,4,3)$, *and the resource utilisation values associated playing set cardinalities of* 3, 4, 5 *and* 6 *represent lower bounds on the value of* $\Psi_\ell(10,5,4,3)$ *because execution time reached the predefined time limit of* 14 400 *seconds.*

the variables x_1, x_2, x_3, x_4. The above-mentioned partitioning technique will used in the discussion below.

Example 4.4 (Explicitly fixing a predetermined set of variables) *In Table 4.10, the resource utilisation problem corresponding to the value* $\ell = 5$ *was not solved to completion for the lottery* $\langle 10,5,4,3\rangle$. *Therefore, the four variables* x_1, x_{64}, x_{127} *and* x_{190} *are fixed. This implies that* $2^4 = 16$ *different problem instances are formulated as ILPs and solved to completion. The results appear in Table 4.11.*

It is known that the value of $\Psi_5(10,5,4,3)$ *is approximately* 0.923 810. *From Table 4.11 it is evident that if some variables are fixed explicitly, a solution equal to the known optimal solution to the problem may be obtained in a very short time. In the case of the lottery* $\langle 10,5,4,3\rangle$ *for* $\ell = 5$, *a solution equal to the optimal solution may be obtained after* 0.919 *seconds (when the variables* x_1, x_{127} *and* x_{190} *are fixed to* 1, *and the variable* x_{64} *is fixed to* 0*). However, it will only be known that a solution is optimal once all possible values of* x_1, x_{64}, x_{127} *and* x_{190} *have been considered. This approach is analogous to solving the problem in parallel.* ■

If the resource utilisation problem for the lottery $\langle 10,5,4,3\rangle$ when $\ell = 5$ is formulated as an ILP and solved to completion, without any variables being explicitly fixed, the results documented in Table 4.12 are obtained.

From Table 4.12 it may be seen that the problem solved to completion in approximately 15.3 days. When the four variables x_1, x_{64}, x_{127} and x_{190} are explicitely fixed, the total time execution of solving all $2^4 = 16$ problems is approximately 17.3 days. However, if the variables x_1, x_{127} and x_{190} are fixed to 1 and x_{64} is fixed to 0, a solution equal to the known optimal solution of approximately 0.923 810 is found in approximately 0.9 seconds. This result either implies that, should an optimal solution be sought, it is less time consuming to allow the problem to solve to completion without fixing any variables explicitly, *or* it may be possible to find an optimal solution or a near optimal solution, in less time by explicitly fixing a larger number of variables. In turn, this will require more resources but possibly less total execution time to find an optimal solution. Regardless, in a worst case scenario, it may yield viable bounds on $\Psi_\ell(m,n,t,k)$ in a feasible amount of time.

TABLE 4.11: *Results obtained when explicitly fixing four variables in the ILP formulation of the resource utilisation problem in which the value $\Psi_5(10, 5, 4, 3)$ is sought. The first four columns contain the values to which each of the four variables are fixed. The fifth column contains the associated number of branches in the branch-and-bound tree. The sixth column contains the time, in seconds, taken to find an optimal solution. Finally, the seventh column contains the objective function value achieved, i.e. the proportion of government tickets covered by the corresponding optimal playing set.*

x_1	x_{64}	x_{127}	x_{190}	Branches	Seconds	Objective
0	0	0	0	24 378 296	1 178 321.000	0.924
0	0	0	1	561 160	62 920.080	0.924
0	0	1	0	621 355	80 066.300	0.924
0	0	1	1	113 259	3 033.000	0.924
0	1	0	0	551 994	62 907.770	0.924
0	1	0	1	252 568	12 448.449	0.924
0	1	1	0	92 842	2 920.819	0.924
0	1	1	1	64	1.089	0.919
1	0	0	0	631 642	82 149.550	0.924
1	0	0	1	160 452	5 554.159	0.924
1	0	1	0	11 233	401.919	0.924
1	0	1	1	0	0.919	0.924
1	1	0	0	76 542	1 786.399	0.924
1	1	0	1	104	1.219	0.919
1	1	1	0	0	1.099	0.9
1	1	1	1	0	0.769	0.89

TABLE 4.12: *Results obtained when solving the problem instance for $\Psi_5(10, 5, 4, 3)$.*

Lottery	ℓ	Branches	Time (s)	Time (days)	Objective value
$\langle 10, 5, 4, 3 \rangle$	5	29 665 952	1 318 179.27	15.25	0.923 810

4.6 Chapter overview

The mathematical programming approach towards solving the complete lottery problem, the incomplete lottery problem, and the resource utilisation problem adopted in this chapter is useful in two situations. Firstly, the mathematical programming approach provides an exact answer. This means that, by examining the results of this approach, the participant knows exactly which playing set to construct that meets his/her requirements. Secondly, a feasible answer may be reached in a short amount of time, and it is often close to the optimal answer.

The mathematical programming approach does, however, have its shortcomings. It has been found in this chapter that as the parameters in the problems increase, the execution time tends to increase as well. The ultimate goal, when solving the lottery problem may be to find an optimal solution for both the incomplete lottery problem and the resource utilisation problem for the lottery $\langle 49, 6, 6, k \rangle$. Unfortunately, due to hardware limitations, it is not possible to employ the mathematical programming approach towards achieving this goal because the execution time will be too long, and the computers used may run out of memory before an optimal solution is reached. Also, it should be mentioned that the model presented in this chapter requires a file, which contains an adjacency matrix, to be provided as input. This file contains all the elements of the adjacency matrix, with spaces between elements in each row. If it is assumed that an element, and a space each take up one byte of data, the file for the lottery $\langle 49, 6, 6, k \rangle$ would

be $\left(\binom{49}{6} \times 2\right) \times \binom{49}{6}$ bytes large. This is approximately 356 terabytes large, which no single computer may store in memory. Currently, in a 32-bit Windows system, the maximum amount of memory which may be used on the computer is 3.5 gigabytes, and on a 64-bit Windows system, the maximum amount of memory which may be used by the computer is 16 gigabytes. Therefore, a feasible answer to $L_{\psi}(m, n, t, k)$ and $\Psi_{\ell}(m, n, t, k)$ may only be found using the mathematical programming approach if the file containing the adjacency matrix is within the physical storage limits of the computer on which it is solved. Consequently, that implies that the incomplete lottery problem and resource utilisation problem may only be solved to completion for small lottery instances by using the mathematical programming approach.

It may therefore be concluded that the mathematical programming approach is useful for finding bounds, or optimal solutions to *small* instances of the incomplete lottery problem or resource utilisation problem (currently problems for which $m \leq 10$). As computers become more powerful, the limitations on the mathematical programming approach will be less, and it may then be possible to find optimal solutions to larger problem instances. However, finding the value of $L_1(49, 6, 6, k)$ via a mathematical programming approach seems infeasible for the forseable future.

CHAPTER 5

Exhaustive Enumeration

Contents

In this chapter, an alternative solution method is presented for the incomplete lottery problem and the resource utilisation problem. In this method, all possible overlapping playing set structures are constructed. From these constructions the answers to both the incomplete lottery problem and the resource utilisation problem may be obtained for a given lottery instance. This exhaustive enumeration solution method was originally proposed by Gründlingh [16] for lottery instances where $n = t$. In §5.1 the method is reviewed and implemented for the more general case where $n \neq t$ in fulfilment of Thesis Objective III(a). In §5.2 a few numerical examples are presented of how the exhaustive enumeration method works and these examples are used to verify the correctness of the implementation, in fulfilment of Thesis Objective III(b). Thereafter, in §5.3, pseudocode examples of the implementation of the exhaustive enumeration method are presented. Following that, the same problem instances which were solved in Chapter 4 via an integer programming approach are solved in §5.4, this time using the exhaustive enumeration method instead. The reason for doing this is to verify the results of Chapter 4 in fulfilment of Thesis Objective III(c). The method is also used to find as many structurally non-isomorphic optimal solutions as possible to the problem instances originally investigated in Chapter 4, in fulfilment of Thesis Objective III(d).

5.1 The lottery tree

The exhaustive enumeration lottery tree method involves the iterative construction of a rooted tree data structure. The aim in this method is to find all non-isomorphic overlapping playing set structures for a given lottery instance of the incomplete lottery problem or the resource utilisation problem. Each node in the tree represents a unique overlapping playing set structure. A probability-of-win value is ultimately assigned to each playing set structure. The answer to both the incomplete lottery problem and the resource utilisation problem may be obtained by examining these probability-of-win values.

Level i ($1 \leq i \leq \ell$) in the lottery tree contains nodes which represent playing set structures of cardinality i tickets. The root node of the tree exists at level $i = 1$, and it represents an arbitrary playing set of cardinality 1. The next level in the tree, $i = 2$, contains one or more nodes containing overlapping playing set structures which represent the different ways in which a second ticket may be added to the playing set. This process of iteratively constructing the lottery tree continues until either a probability-of-win value of ψ is found to be associated with a node (in which case the incomplete lottery problem is solved) or until the lottery tree reaches a level corresponding to a fixed number of ℓ tickets (in which case the resource utilisation problem is solved).

5.1.1 Creating the nodes in the lottery tree

Suppose $\mathcal{L}_\ell = \{T_1, T_2, \ldots, T_\ell\}$ is a playing set of cardinality ℓ in the lottery $\langle m, n, t, k\rangle$. The lottery tree for this playing set has $\ell + 1$ levels, and the nodes on level ℓ of the tree represent potential $\Psi_\ell(m, n, t, k)$ playing set structures for the resource utilisation problem, or potential $L_\psi(m, n, t, k)$ playing set structures for the incomplete lottery problem.

Each node in the lottery tree is represented by a vector $\vec{\mathbf{X}}^{(\ell)} = (x^{(\ell)}_{(000\ldots0)}, x^{(\ell)}_{(000\ldots1)}, \ldots, x^{(\ell)}_{(111\ldots1)})$ which captures the overlapping ticket structure contained in that node. The entries in each vector $\vec{\mathbf{X}}^{(\ell)}$ may be interpreted as mentioned in §2.2.4. Except for the root node (which is always represented by the vector $\vec{\mathbf{X}}^{(1)} = (x^{(1)}_0, x^{(1)}_1) = (m - n, n)$), each vector at level $i < \ell$ of the lottery tree is derived from the vector in its parent node at level i according to the following rules:

1. All the entries in the vector $\vec{\mathbf{X}}^{(i+1)}$ must add up to m, because there are only m numbers available for the participant to include in his/her playing set for any given lottery $\langle m, n, t, k\rangle$.

2. The entries
$$x^{(i+1)}_{(100\ldots0)}, x^{(i+1)}_{(100\ldots1)}, \ldots, x^{(i+1)}_{(111\ldots1)} \tag{5.1}$$
in the vector $\vec{\mathbf{X}}^{(i+1)}$ must add up to n. These entries represent all the compartments in the overlapping playing set structure which include ticket $i+1$. Each ticket in the playing set must contain n numbers, therefore all the compartments including ticket $i + 1$ must collectively contain exactly n numbers.

3. The entries
$$x^{(i+1)}_{(000\ldots0)}, x^{(i+1)}_{(000\ldots1)}, \ldots, x^{(i+1)}_{(011\ldots1)}$$

in the vector $\vec{\mathbf{X}}^{(i+1)}$ must add up to $m-n$. These entries represent all the compartments in the overlapping playing set structure other than those compartments denoting ticket $i+1$. Ticket $i+1$ must contain n numbers, which implies that all the other compartments must contain $m-n$ numbers collectively.

4. Entry $x^{(i+1)}_{(111\ldots1)}$ of the vector $\vec{\mathbf{X}}^{(i+1)}$ must be less than n, for otherwise each ticket in the playing set would contain exactly the same numbers, in which case the playing set would have the same probability-of-win as a playing set of cardinality 1.

5. Each entry in (5.1) must be less than or equal to the entry corresponding to it in its parent vector $\vec{\mathbf{X}}^{(i)}$. This implies that

$$x^{(i+1)}_{(100\ldots0)} \leq x^{(i)}_{(000\ldots0)},\ x^{(i+1)}_{(100\ldots1)} \leq x^{(i)}_{(000\ldots1)},\ \ldots,\ x^{(i+1)}_{(111\ldots1)} \leq x^{(i)}_{(111\ldots1)}. \tag{5.2}$$

Each entry in (5.1) represents an overlap of ticket $i+1$ with the compartment in the corresponding overlapping playing set structure of the parent vector $\vec{\mathbf{X}}^{(i)}$.

5.1.2 Assigning probability-of-win values to the nodes

Once the exhaustive enumeration lottery tree has been constructed, it is possible to compute and assign a probability-of-win value to each node in the tree. This value represents the probability of winning a k-prize if that specific overlapping playing set structure were to be selected by the participant.

The probability-of-win value at a specific node is computed by analysing the children of that node, because the overlapping playing set structures represented by the children nodes may also represent the different ways in which a winning ticket, containing t numbers may overlap with the playing set structure represented by that node. This, together with the process described in §5.1.1 represents an additional step involved in the construction of the tree if $n \neq t$. The method presented in §5.1.1 of building children from a parent node $\vec{\mathbf{X}}^{(\ell)}$ results in the construction of vectors which represent the overlapping of a winning ticket of size n only. If the lottery instance is such that $n \neq t$, two sets of children have to be constructed for each node. The first set is the set of children described in §5.1.1, and these children may spawn new children. The second set of children represents the potential overlapping of the winning ticket of size t with the overlapping playing set structure of the parent node, and they are constructed in the same way as described in §5.1.1 with the value of n replaced by the value of t. From the second set of children, a test of whether or not a winning ticket of size t has at least k numbers in common with at least one ticket in playing set structure of the parent node is conducted. In [16], this is referred to as a so-called *domination test*. The second set of children are ultimately used to compute the probability-of-win value for the parent node, and they do not spawn new children. Once the probability-of-win value for the parent node has been computed, this second set of children is no longer required in the lottery tree structure.

Each child in the second set of children are assigned a boolean value of *true* or *false*, which is used to calculate the probability-of-win value of the parent node. A child from the second set of children is assigned the value *true* (and assigned to a set $\mathcal{D}_T$) if the overlapping winning ticket has at least k numbers in common with at least one of the tickets in the parent node's playing set, which implies that it passes the above-mentioned domination test. The child node is assigned a value of *false* (and assigned to a set $\mathcal{D}_F$) otherwise.

The number of ways to overlap a winning ticket with the overlapping playing set structure of the parent node (when interchanging the roles of the elements of $\mathcal{U}_m$) so as to guarantee a k-prize, divided by the total number of ways to overlap a winning ticket with the overlapping playing set structure of the parent node (when interchanging the roles of the elements of $\mathcal{U}_m$) is equal to the value of the probability-of-win if a playing set conforming to the overlapping playing set structure of the parent node is selected by the participant. Therefore, together with these boolean values, each child node is assigned a value equal to the multiplicity of its vector. The formula[1],

$$M(\vec{\mathbf{X}}^{(\ell+1)}) = \frac{m!}{\Pi_{j=0}^{2^\ell-1} x_j^{(\ell+1)}!}, \tag{5.3}$$

for the multiplicity of the vector results, because all the m entries in the overlapping playing structure may be interchanged (there are $m!$ ways to do this). However, the number of entries in each compartment remains the same and interchanging the entries within a compartment has no effect on the playing set represented by the overlapping playing set structure. Therefore, the multiplicity in (5.3) represents the number of structurally different overlappings that are possible when interchanging the roles of the elements of $\mathcal{U}_m$. The number of ways of overlapping a winning ticket with the overlapping playing set structure of the parent node (when interchanging the roles of the elements of $\mathcal{U}_m$) so as to guarantee a k-prize is

$$\sum_{\vec{\mathbf{X}}^{(\ell+1)} \in \mathcal{D}_T} M(\vec{\mathbf{X}}^{(\ell+1)}),$$

and the total number of ways of overlapping a winning ticket with the overlapping playing set structure of the parent node (when interchanging the roles of the elements of $\mathcal{U}_m$) is

$$\sum_{\vec{\mathbf{X}}^{(\ell+1)} \in \mathcal{D}_T \cup \mathcal{D}_F} M(\vec{\mathbf{X}}^{(\ell+1)}).$$

Using these values, the probability-of-win value for the parent node may then be calculated as

$$\Psi_{\vec{\mathbf{X}}^{(\ell)}} = \frac{\sum_{\vec{\mathbf{X}}^{(\ell+1)} \in \mathcal{D}_T} M(\vec{\mathbf{X}}^{(\ell+1)})}{\sum_{\vec{\mathbf{X}}^{(\ell+1)} \in \mathcal{D}_T \cup \mathcal{D}_F} M(\vec{\mathbf{X}}^{(\ell+1)})}.$$

This value is assigned to the parent node, and represents the fraction of government tickets covered by a playing set represented by that overlapping ticket structure. This means that

$$\Psi_{\vec{\mathbf{X}}^{(\ell)}} = \frac{|\bigcup_{v \in \mathcal{L}_\psi(m,n,t,k)} \mathcal{N}[v]|}{\binom{m}{t}},$$

where v is a ticket in the overlapping playing set structure represented by the vector $\vec{\mathbf{X}}^{(\ell)}$.

5.2 Numerical examples

Three numerical examples (involving small problem instances) of the use of the exhaustive enumeration lottery tree method described above are presented in this section.

[1] In this formula, the convention $0! = 1$ is used.

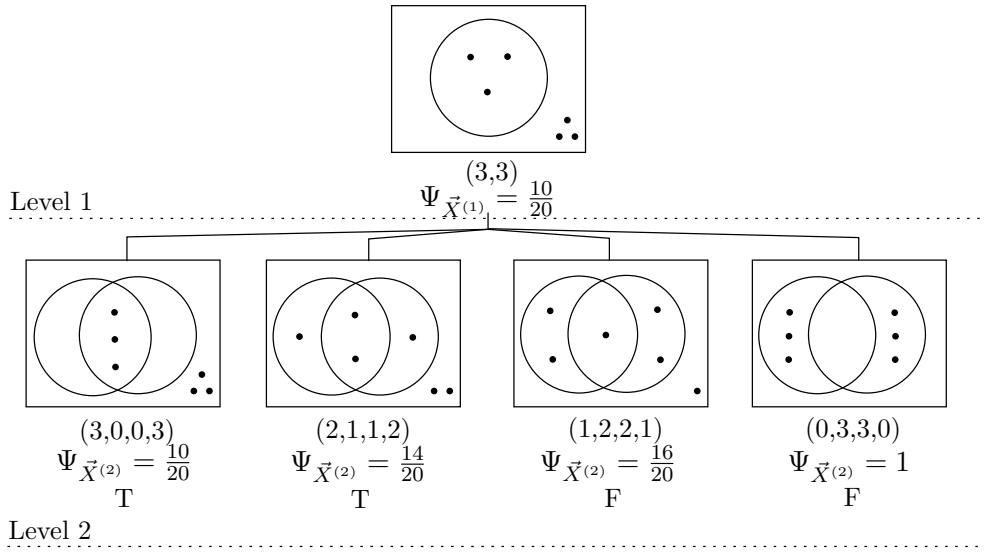


FIGURE 5.1: *The exhaustive enumeration lottery tree for the lottery* $\langle 6, 3, 3, 2\rangle$. *The letter "T" placed below a node indicates that the corresponding overlapping playing set structure passes the domination test in §5.1.2, and the letter "F" indicates otherwise.*

5.2.1 The lottery $\langle 6, 3, 3, 2\rangle$

The first aspect that must be noted regarding this problem instance is that since $n = t = 3$, there is no need to construct a second set of children for each node in the exhaustive enumeration lottery tree. The root of the tree is a playing ticket containing 3 numbers, with $m-n = 6-3 = 3$ numbers remaining. The first two levels of the lottery tree for this problem instance appear in Figure 5.1. The associated vector and the boolean value appears below each node. That is, the domination test was performed in Level 2 (by adding a single 3-set, depicting the government winning ticket, to the playing set structure) and yielded $D_T = \{(2, 1, 1, 2), (3, 0, 0, 3)\}$, while $D_F = \{(0, 3, 3, 0), (1, 2, 2, 1)\}$

It is found that $M(\vec{\mathbf{X}}^{(2)}) = 20$, $M(\vec{\mathbf{X}}^{(2)}) = 180$, $M(\vec{\mathbf{X}}^{(2)}) = 180$ and $M(\vec{\mathbf{X}}^{(2)}) = 20$. Once the value of $M(\vec{\mathbf{X}}^{(2)})$ corresponding to each node's vector has been computed, it may be used together with the boolean values associated with each node to compute the probability-of-win value for the parent node, which is represented by the vector $\vec{\mathbf{X}}^{(1)} = (3, 3)$, as

$$\Psi_{\vec{\mathbf{X}}^{(1)}} = \frac{\sum_{\vec{\mathbf{X}}^{(2)} \in \mathcal{D}_T} M(\vec{\mathbf{X}}^{(2)})}{\sum_{\vec{\mathbf{X}}^{(2)} \in \mathcal{D}_T \cup \mathcal{D}_F} M(\vec{\mathbf{X}}^{(2)})} = \frac{20 + 180}{20 + 180 + 180 + 20} = \frac{200}{400} = 0.5.$$

In Table 5.1 ticket P_1 represents a single ticket in the participant's playing set. The remaining tickets in the table, labelled G_i $(1 \leq i \leq 20)$ represent all the tickets from which the government may choose the single winning lottery ticket. There are 10 tickets ($G_1, \ldots, G_7$ and $G_{11}, \ldots, G_{13}$) out of the possible 20 government tickets which have $k = 2$ or more numbers in common with the participant's ticket. Therefore, if the participant wishes to form a playing set of only one ticket, the chance of winning a 2-prize is $\frac{10}{20} = 0.5$. This verifies the results emanating from the exhaustive enumeration lottery tree method in Figure 5.1. Note that if the participant had instead selected a playing set consisting of two tickets, these tickets may be selected in such a way that *all* the government tickets have at least $k = 2$ numbers in common with the participant's ticket, implying that $L_1(6, 3, 3, 2) = 2$ and $\Psi_2(6, 3, 3, 2) = 1$.

TABLE 5.1: *Tabular representation of one participant ticket, P_1, together with all government tickets G_i ($1 \leq i \leq 20$) for the lottery $\langle 6, 3, 3, 2 \rangle$. The column labelled O contains values which indicate the number of overlapping elements each ticket G_i has with ticket P_1.*

	1	2	3	4	5	6	O
P_1	×	×	×				
G_1	×	×	×				3
G_2	×	×		×			2
G_3	×	×			×		2
G_4	×	×				×	2
G_5	×		×	×			2
G_6	×		×		×		2
G_7	×		×			×	2
G_8	×			×	×		1
G_9	×			×		×	1
G_{10}	×				×	×	1
G_{11}		×	×	×			2
G_{12}		×	×		×		2
G_{13}		×	×			×	2
G_{14}		×		×	×		1
G_{15}		×		×		×	1
G_{16}		×			×	×	1
G_{17}			×	×	×		1
G_{18}			×	×		×	1
G_{19}			×		×	×	1
G_{20}				×	×	×	0

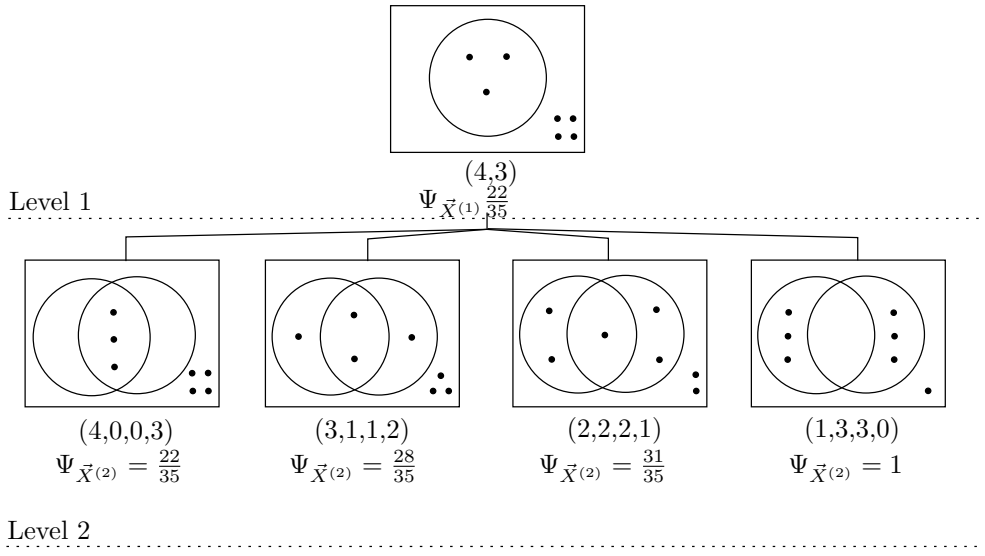


FIGURE 5.2: *The exhaustive enumeration lottery tree for the lottery* $\langle 7, 3, 4, 2\rangle$.

5.2.2 The lottery $\langle 7, 3, 4, 2\rangle$

Since $n = 3 \neq 4 = t$, two sets of children are constructed for each node in the exhaustive enumeration lottery tree in this problem instance. The first set of children contains nodes which represent the overlapping of an additional participant ticket of size $n = 3$ with respect to the existing playing set. The second set of children contains nodes which represent the overlapping of a winning government ticket of size $t = 4$ with the playing set.

The root node, together with its first set of children is presented in Figure 5.2. This set of children may spawn new children. The root node, together with its second set of children, is presented in Figure 5.3. This set of children is used to compute the probability-of-win value of the root node. Recall that the second set of children does not spawn new children.

It is found that $M(\vec{\mathbf{X}}^{(2)}) = 140$, $M(\vec{\mathbf{X}}^{(2)}) = 630$, $M(\vec{\mathbf{X}}^{(2)}) = 420$ and $M(\vec{\mathbf{X}}^{(2)}) = 35$. Once the $M(\vec{\mathbf{X}}^{(2)})$ value for each node's vector has been calculated, it may be used together with the boolean values associated with each node to compute the probability-of-win value for the parent node, which is represented by the vector $(4, 3)$, as

$$\Psi_{\vec{\mathbf{X}}^{(1)}} = \frac{\sum_{\vec{\mathbf{X}}^{(2)} \in \mathcal{D}_T} M(\vec{\mathbf{X}}^{(2)})}{\sum_{\vec{\mathbf{X}}^{(2)} \in \mathcal{D}_T \cup \mathcal{D}_F} M(\vec{\mathbf{X}}^{(2)})} = \frac{140 + 630}{140 + 630 + 420 + 35} = \frac{770}{1225} \approx 0.629.$$

In Table 5.2, ticket P_1 represents a single participant's ticket of size $n = 3$. The remaining rows in the table, labelled G_i $(1 \leq i \leq 35)$, represent all the tickets from which the government chooses the winning lottery ticket of size $t = 4$. There are 22 tickets ($G_1, \ldots, G_{16}$ and $G_{21}, \ldots, G_{26}$) out of the possible 35 government tickets which have $k = 2$ or more numbers in common with the participant's ticket. Therefore, if the participant wishes to form a playing set of only one ticket, the chance of winning a 2-prize is $\frac{22}{35} \approx 0.6286$. This verifies the results emanating from the exhaustive enumeration lottery tree in Figure 5.2. Note that if the participant had instead selected a playing set consisting of two tickets, they may be selected in such a way that *all* the government tickets will have at least $k = 2$ numbers in common with the participant's ticket.

TABLE 5.2: *Tabular representation of one participant ticket P_1, together with all government tickets G_i, $1 \leq i \leq 35$ for the lottery $\langle 7, 3, 4, 2 \rangle$. The column labelled O contains values which indicate the number of overlapping numbers each ticket G_i has with ticket P_1.*

	1	2	3	4	5	6	7	O
P_1	×	×	×					
G_1	×	×	×	×				3
G_2	×	×	×		×			3
G_3	×	×	×			×		3
G_4	×	×	×				×	3
G_5	×	×		×	×			2
G_6	×	×		×		×		2
G_7	×	×		×			×	2
G_8	×	×			×	×		2
G_9	×	×			×		×	2
G_{10}	×	×				×	×	2
G_{11}	×		×	×	×			2
G_{12}	×		×	×		×		2
G_{13}	×		×	×			×	2
G_{14}	×		×		×	×		2
G_{15}	×		×		×		×	2
G_{16}	×		×			×	×	2
G_{17}	×			×	×	×		1
G_{18}	×			×	×		×	1
G_{19}	×			×		×	×	1
G_{20}	×				×	×	×	1
G_{21}		×	×	×	×			2
G_{22}		×	×	×		×		2
G_{23}		×	×	×			×	2
G_{24}		×	×		×	×		2
G_{25}		×	×		×		×	2
G_{26}		×	×			×	×	2
G_{27}		×		×	×	×		1
G_{28}		×		×	×		×	1
G_{29}		×		×		×	×	1
G_{30}		×			×	×	×	1
G_{31}			×	×	×	×		1
G_{32}			×	×	×		×	1
G_{33}			×	×		×	×	1
G_{34}			×		×	×	×	1
G_{35}				×	×	×	×	0

TABLE 5.3: *Tabular representation of participant tickets P_1 and P_2, together with all government tickets G_i, $1 \leq i \leq 35$ in the lottery $\langle 7, 5, 4, 3 \rangle$. The column labelled O represents the maximum amount of numbers ticket G_i has in common with either ticket P_1 or P_2. That is, $O_i = \max\{|P_1 \cap G_i|, |P_2 \cap G_i|\}$ for all $i = 1, \ldots, 35$.*

	1	2	3	4	5	6	7	O
P_1	×	×	×	×	×			
P_2	×	×	×			×	×	
G_1	×	×	×	×				4
G_2	×	×	×		×			4
G_3	×	×	×			×		4
G_4	×	×	×				×	4
G_5	×	×		×	×			4
G_6	×	×		×		×		3
G_7	×	×		×			×	3
G_8	×	×			×	×		3
G_9	×	×			×		×	3
G_{10}	×	×				×	×	4
G_{11}	×		×	×	×			4
G_{12}	×		×	×		×		3
G_{13}	×		×	×			×	3
G_{14}	×		×		×	×		3
G_{15}	×		×		×		×	3
G_{16}	×		×			×	×	4
G_{17}	×			×	×	×		3
G_{18}	×			×	×		×	3
G_{19}	×			×		×	×	3
G_{20}	×				×	×	×	3
G_{21}		×	×	×	×			4
G_{22}		×	×	×		×		3
G_{23}		×	×	×			×	3
G_{24}		×	×		×	×		3
G_{25}		×	×		×		×	3
G_{26}		×	×			×	×	4
G_{27}		×		×	×	×		3
G_{28}		×		×	×		×	3
G_{29}		×		×		×	×	3
G_{30}		×			×	×	×	3
G_{31}			×	×	×	×		3
G_{32}			×	×	×		×	3
G_{33}			×	×		×	×	3
G_{34}			×		×	×	×	3
G_{35}				×	×	×	×	2

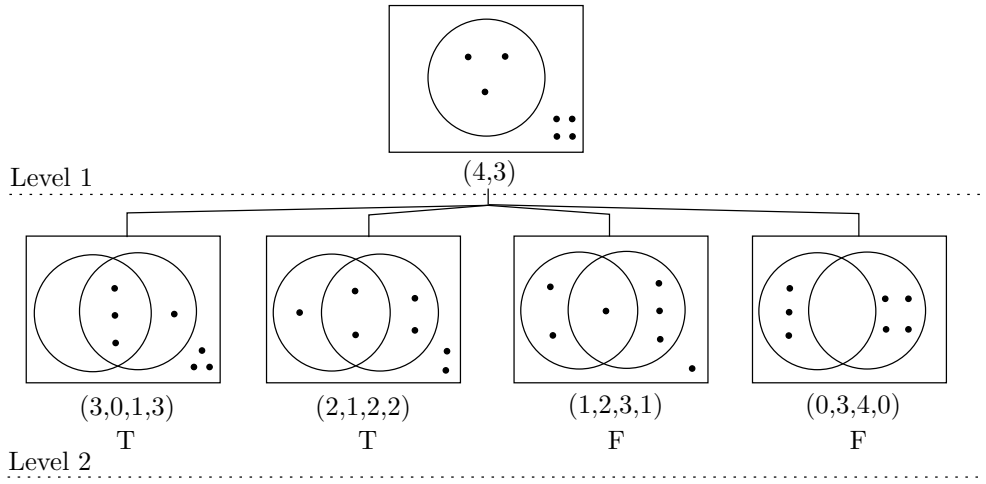


FIGURE 5.3: *The root node of the exhaustive enumeration lottery tree, together with the second set of children for the lottery* $\langle 7, 3, 4, 2\rangle$. *The letter "T" placed below a node indicates that the corresponding overlapping playing set structure passes the domination test in §5.1.2, and the letter "F" indicates otherwise.*

5.2.3 The lottery $\langle 7, 5, 4, 3\rangle$

The lottery $\langle 7, 5, 4, 3\rangle$ is not as trivial to analyse as the previous two lotteries. The smallest playing set yielding a probability-of-win value of 1 has a cardinality of three. Therefore, the lottery tree possesses three levels, as indicated in Figure 5.4.

The three levels of the exhaustive enumeration lottery tree associated with the lottery $\langle 7, 5, 4, 3\rangle$ is presented in Figure 5.4. This set of children may spawn new children. The second set of children of the overlapping playing set structure $(0, 2, 2, 3)$ is presented in Figure 5.5. This set of children is used to compute the probability-of-win value of their parent node (the node representing the overlapping playing set structure $(0, 2, 2, 3)$). Recall that the second set of children does not spawn new children.

It is found that $M(\vec{\mathbf{X}}^{(3)}) = 420$, $M(\vec{\mathbf{X}}^{(3)}) = 630$, $M(\vec{\mathbf{X}}^{(3)}) = 420$, $M(\vec{\mathbf{X}}^{(3)}) = 2\,520$, $M(\vec{\mathbf{X}}^{(3)}) = 1\,260$, $M(\vec{\mathbf{X}}^{(3)}) = 630$, $M(\vec{\mathbf{X}}^{(3)}) = 1\,260$ and $M(\vec{\mathbf{X}}^{(3)}) = 210$. Once the $M(\vec{\mathbf{X}}^{(3)})$ value corresponding to each node's vector has been calculated, it may be used together with the boolean values associated with each node to compute the probability-of-win value for the parent node, the node corresponding to the vector $(0, 2, 2, 3)$, as

$$\begin{aligned}\Psi_{\vec{\mathbf{X}}^{(2)}} = \frac{\sum_{\vec{\mathbf{X}}^{(3)}\in\mathcal{D}_T} M(\vec{\mathbf{X}}^{(3)})}{\sum_{\vec{\mathbf{X}}^{(3)}\in\mathcal{D}_T\cup\mathcal{D}_F} M(\vec{\mathbf{X}}^{(3)})} &= \frac{420+630+420+2\,520+1\,260+630+1\,260}{420+630+420+2\,520+1\,260+630+1\,260+210} \\ &= \frac{7\,140}{7\,350} \approx 0.971.\end{aligned}$$

In Table 5.3 the top two tickets, labelled P_1 and P_2, represent the tickets in the participant's playing set. The remaining rows in the table, labelled G_i $(1 \leq i \leq 35)$, represent all the tickets from which the government may choose the winning lottery ticket. There are 34 tickets out of the possible 35 government tickets which have at least $k = 3$ or more numbers in common with the any of the participants tickets (government ticket G_{35} is the only ticket which does

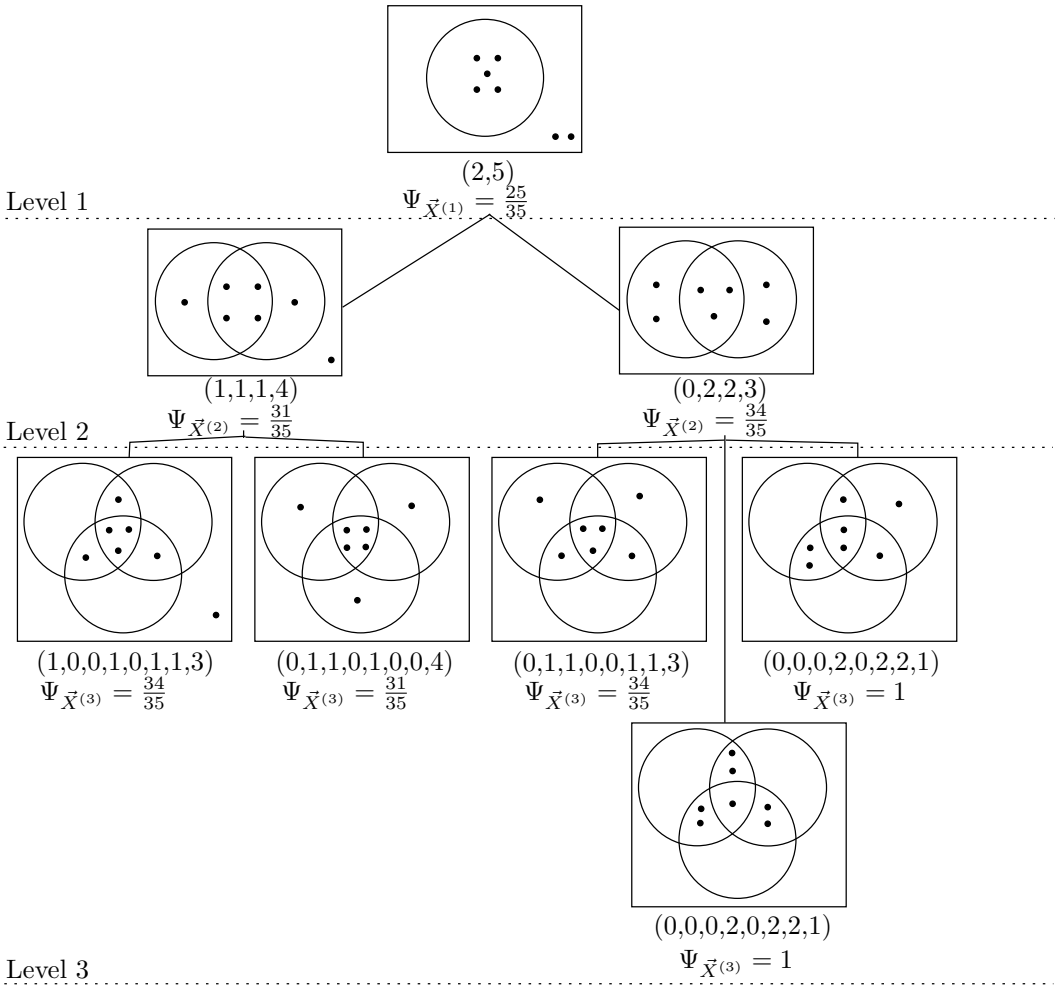


FIGURE 5.4: *The exhaustive enumeration lottery tree for the lottery* $\langle 7, 5, 4, 3\rangle$.

not have at least three numbers in common with any of the participants tickets). Therefore, if the participant wishes to select a playing set, which consists of two tickets and conforms to the overlapping playing set structure represented by the vector $(0, 2, 2, 3)$, the probability of winning a 3-prize is equal to $\frac{34}{35} \approx 0.971$. This verifies the results which may be computed from the lottery tree shown in Figures 5.4 and 5.5. Note that if the participant had instead purchased a playing set consisting of three tickets, they may be selected in such a way that all the government tickets will have at least $k = 3$ numbers in common with the participant's ticket.

5.3 Implementation

In this section, the implementation of the exhaustive enumeration lottery tree method described above is presented in the form of pseudocode listings.

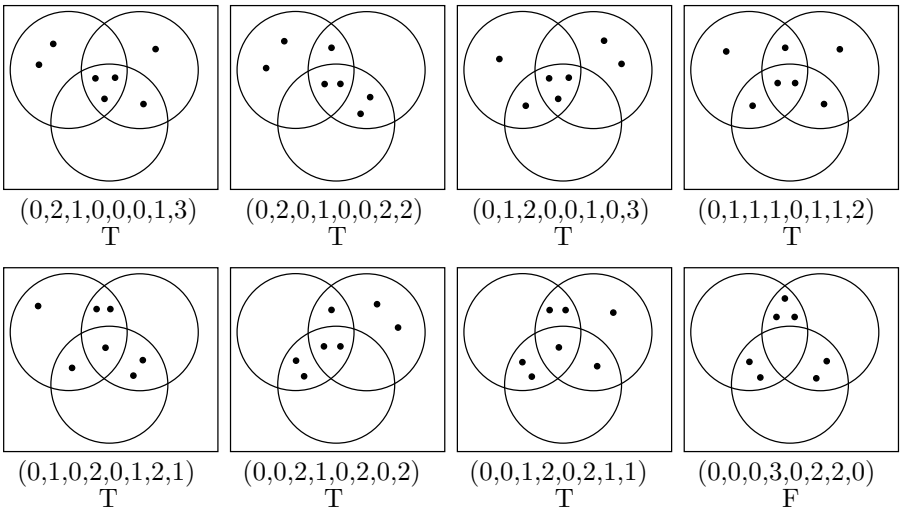


FIGURE 5.5: *The second set of children for the vector* $(0, 2, 2, 3)$ *in the exhaustive enumeration lottery tree for the lottery* $\langle 7, 5, 4, 3\rangle$*. The letter "T" placed below a node indicates that the associated overlapping playing set structure passes the domination test in §5.1.2, and the letter "F" indicates otherwise.*

5.3.1 Pruning of the lottery tree

If the method described in §5.1.1 is used, some nodes in the exhaustive enumeration lottery tree are duplicates of each other. As the level of the tree increases, the number of duplicates increases exponentially. Therefore, many more nodes may be considered for insertion into the exhaustive enumeration lottery tree than is necessary, rendering the computational cost of the method unnecessarily large. An example of a set of duplicates from the lottery $\langle 9, 4, 5, 3\rangle$ is shown in Figure 5.6. For each overlapping playing set structure containing ℓ tickets, $\ell!$ duplicate overlapping playing set structures exist, and only one of these duplicate structures needs to be added to the lottery tree. For each of these duplicate vectors, a unique *ancestor* vector

$$\vec{\mathbf{Y}}^{(\ell)} = (\overbrace{x^{(1)}_{(1)}}^{T_1}, \overbrace{x^{(2)}_{(10)}, x^{(2)}_{(11)}}^{T_2}, \overbrace{x^{(3)}_{(100)}, x^{(3)}_{(101)}, x^{(3)}_{(110)}, x^{(3)}_{(111)}}^{T_3}, \ldots, \overbrace{x^{(\ell)}_{(10\ldots0)}, x^{(\ell)}_{(10\ldots1)}, \ldots, x^{(\ell)}_{(11\ldots1)}}^{T_\ell}) \tag{5.4}$$

is constructed. The length of this vector is $2^{\ell-1}$, and it is used to track the addition of new tickets at each level of the exhaustive enumeration lottery tree. The only duplicate vector inserted into the exhaustive enumeration lottery tree is the one with the lexicographically smallest ancestor vector associated with it.

The lottery tree may be pruned further by applying the following pruning rules, established by Gründlingh [16], at any level i of the lottery tree:

1. If there are more than t entries in the compartment excluding all tickets, any winning ticket of size t, which contains t of the numbers not used in any playing set ticket, $\vec{\mathbf{X}}^{(i)}$ will not represent an $L_1(m, n, t, k)$-set for $\langle m, n, t, k\rangle$. In other words, if $x^{(i)}_{(000\ldots0)_2} \geq (L_1(m, n, t, k) - i + 1)t$, then the overlapping ticket structure corresponding to the vector $\vec{\mathbf{X}}^{(i)}$ will not yield any $L_1(m, n, t, k)$-sets in the subtree having $\vec{\mathbf{X}}^{(i)}$ as root and may therefore be pruned from the lottery tree.

2. If the sum of all the entries in each exclusive ticket compartment is at least t, a winning ticket of size t may exist which might not have k numbers in common with any

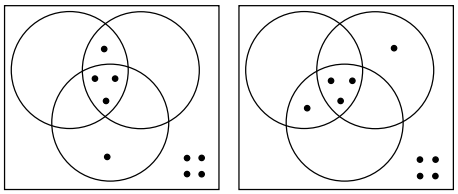


FIGURE 5.6: *Duplicate overlapping playing set structures from the exhaustive enumeration lottery tree for the lottery* $\langle 9, 4, 5, 3\rangle$. *For the overlapping structure vector* $\vec{\boldsymbol{X}}^{(3)} = (4, 0, 0, 1, 1, 0, 0, 3)$, *the corresponding ancestor vector is* $\vec{\boldsymbol{Y}}^{(3)} = (4, 0, 4, 1, 0, 0, 3)$, *and for the overlapping structure vector* $\vec{\boldsymbol{X}}^{(3)} = (4, 0, 1, 0, 0, 1, 0, 3)$, *the corresponding ancestor vector is* $\vec{\boldsymbol{Y}}^{(3)} = (4, 1, 3, 0, 1, 0, 3)$. *It may be seen that* $(4, 0, 4, 1, 0, 0, 3)$ *is lexicographically smaller than* $(4, 1, 3, 0, 1, 0, 3)$. *Hence the vector* $(4, 0, 1, 0, 0, 1, 0, 3)$ *is not inserted into the lottery tree, but the vector* $(4, 0, 0, 1, 1, 0, 0, 3)$ *is inserted into the tree if it passes further validity tests.*

playing set ticket implying that the structure does not represent an $L_1(m, n, t, k)$-set for $\langle m, n, t, k\rangle$. In other words, if $\min\{x^{(i)}_{(100\ldots0)_2}, k-1\}+\ldots+\min\{x^{(i)}_{(000\ldots1)_2}, k-1\}+x^{(i)}_{(000\ldots0)_2} \geq (L_1(m, n, t, k) - i + 1)t$, then the overlapping ticket structure corresponding to the vector $\vec{\mathbf{X}}^{(i)}$ will not yield any $L_1(m, n, t, k)$-sets in the subtree which has $\vec{\mathbf{X}}^{(i)}$ as root and may therefore be pruned from the lottery tree.

Of the following pruning rules (also established by Gründlingh), rule (1) may be applied to the nodes at level ℓ of the lottery tree, and rules (2) and (3) may be applied to level $\ell - 1$ of the lottery tree.

1. If $\min\{x^{(\ell)}_{(100\ldots0)_2}, k-1\} + \ldots + \min\{x^{(\ell)}_{(000\ldots1)_2}, k-1\} + x^{(\ell)}_{(000\ldots0)_2} \geq t$, then the overlapping playing set structure represented by the vector $\vec{\mathbf{X}}^{(\ell)}$ are not complete lottery sets, and may be pruned from the tree.

2. If $x_0^{(\ell-1)} \geq n + 1$, then all possible overlapping playing set structures represented by the vector $\vec{\mathbf{X}}^{(\ell-1)}$ may be pruned from the tree.

3. If $\min\{x^{(\ell)}_{(100\ldots0)_2}, k-1\} + \ldots + \min\{x^{(\ell)}_{(000\ldots1)_2}, k-1\} + x^{(\ell)}_{(000\ldots0)_2} \geq (n+t)$, then all possible overlapping playing set structures represented by the vector $\vec{\mathbf{X}}^{(\ell-1)}$ are not lottery sets, and may hence be pruned from the tree.

5.3.2 Pseudocode

The exhaustive enumeration lottery tree method described above was implemented in the programming language C#. The nodes in the exhaustive enumeration lottery tree are represented by a class called `Node`. In terms of object orientated programming, a class is a custom-made data structure which may represent a person, place or entity. In this case, the class named `Node` represents a node in the exhaustive enumeration lottery tree. A class contains descriptions of the attributes (known as member variables) and abilities (methods) of the person, place or entity which it represents. The class named `Node` has various member variables (a vector, `arrX`, representing the overlapping playing set structure, the binary variable and probability-of-win

value associated with the overlapping playing set structure, and a list of the children associated with that node in the exhaustive enumeration lottery tree) and methods (functions which retrieve and set the values of the vector `arrX`, the binary variable, the probability-of-win value associated with the overlapping playing set structure, and the list of children) associated with it.

Algorithm 5.1: Main

Data: This algorithm drives the entire program

Result: The resource utilisation for each level of the tree (playing set cardinality) is printed out.

1 Initialize values for m, n, t, k and ℓ;
2 Construct new object of type `Node`, called `rootNode` which consists of two elements. This represents the root of the exhaustive enumeration lottery tree. The first element equals $m - n$, and the second element equals n;
3 Call the function, LevelK;
4 Print execution time out to screen;
5 Print out the tree `/* An optional command */`
6 ;
7 Print out the best resource utilisation associated with each level of the lottery tree;

Algorithm 5.2: calcv

Data: arrY, arrX, sz

Result: A boolean value which indicates if a valid child vector has been constructed in array arrY, from the parent vector which is represented by the array arrX.

`/* Fill in values of` $x^{(i+1)}_{(100\ldots0)}, x^{(i+1)}_{(100\ldots1)}, \ldots, x^{(i+1)}_{(111\ldots1)}$ `for` $arrY$`. */`
1 **for** $i = 0$; $i < sz$ **do**
2 $a = arrX[sz - i] + 1$;
3 $arrY[2 \times sz - i] = iter \bmod a$;
4 $iter = iter/a$;
5 **end**
`/* All possible child vectors for` $arrX$ `have been created. */`
6 **if** $iter > 0$ **then**
7 return `false`;
8 **end**
`/* Fill in the values of` $x^{(i+1)}_{(000\ldots0)}, x^{(i+1)}_{(000\ldots1)}, \ldots, x^{(i+1)}_{(011\ldots1)}$ `for` $arrY$`. */`
9 **for** $i = 0$; $i < sz$ **do**
10 $arrY[i] = arrX[i] - arrY[sz + i]$;
11 **end**
12 return `true`;

The list of children associated with each node is represented by a variable named `children` and is stored as a member variable in `Node`. This variable is an `ArrayList` data type. An `ArrayList` data type stores references to objects which are used in the implementation. In this case the references refer to the children of a parent node in the exhaustive enumeration lottery tree. The vector $\vec{\mathbf{X}}^{(\ell)}$ in each node is represented by an integer array named `arrX`. The binary value (described in §5.1.2) associated with each node in the tree is represented by a boolean

Algorithm 5.3: LevelK

Data: parentNode, sz, max

Result: Builds the lottery tree and stores it in memory.

```
1  arrX = array corresponding to parentNode;
2  if max = 0 then
3      return;
4  end
5  iter = 0;
6  Let arrY be a new array of size sz × 2;
7  while calcv(sz, arrY, arrX, iter) do
8      sum = sum of all the elements in the second half of arrY;
9      if sum = n and Last element of arrY ≤ n then
10         isValid=true;
11         for Each ticket in the parent node do
12             Set kSum = the sum of the elements which the new ticket has in common with the
               current ticket from the parent node;
13             if kSum >= k then
14                 Set child node's boolean value to true;
15             end
16             if kSum >= n then
17                 isValid=false;
18             end
19             level2Vector = ancestor of arrY at level 2;
20             if kSum < level2Vector[11_2] then
21                 isValid=false;
22             end
23         end
24         if sum == t then
25             Call AddTempChild of parentNode;
26         end
27         if toPrune = false and isValid = true then
28             if isDuplicate=false then
29                 Call LevelK(n, newChild, sz2, max - 1);
30             end
31         end
32     end
33     if sum = t and Last element of arrY ≤ n AND n ≠ t then
34         for Each ticket in the parent node do
35             Set kSum = the sum of the elements which the new ticket has in common with the
               current ticket from the parent node;
36             if kSum >= k then
37                 Set child node's boolean value to true;
38             end
39         end
40         Call AddTempChild of parentNode;
41     end
42 end
43 iter = iter + 1;
```

Algorithm 5.4: isDuplicate

Data: `vector`, `vectorSize`

Result: A boolean value, which has the value true if a duplicate of the received vector is already in the lottery tree, and false otherwise.

```
1  Construct a temporary vector representing the same elements as the received vector, vector;
2  while i < Factorial(numtickets) do
3      Construct a permuation of length equal to numtickets = log vectorSize/log 2;
4      Permute all binary strings (which represent the received vector placeholders) according the
       the values of the permutation;
5      Construct a new vector D using the permuted binary strings as placeholders. This will be
       a duplicate vector of the received vector;
6      if GetVectorY(D) lexicograpically less than GetVectorY(receivedvector) then
7          return true;
8      end
9      return false;
10 end
```

value named `TorF`. This value is only assigned to a node if it is part of the second set of children (also described in §5.1.2) of a parent node.

The method `AddChild` in `Node` receives a reference to a child node as input and adds it to the list represented by the variable, `children`. The method `setarrX` receives an array as input, and copies its values one by one to `arrX` for the given node. The method `setTorF` assigns a value to the variable `TorF` associated with the node, depending on the value of the boolean value received. The method `getTorF` returns the variable `TorF` associated with the node. The method `getArrX` returns a reference to the array `arrX` of the given node, and the method `printarrX` outputs the array `arrX` to the screen. Finally, the method `getPsi` returns the probability-of-win value associated with a given node.

Of the many functions in the program that are responsible for its various actions, the ones presented below in pseudocode form are the most important functions. The function named `Main` drives the entire program. This function is responsible for initialising all the necessary variables which are used in the program, and for calling the function, `LevelK`, which constructs the exhaustive enumeration lottery tree.

The function `LevelK` is recursive. It is responsible for the construction of both sets of children for each level of the exhaustive enumeration lottery tree as well as for adding both sets of children to the tree. All the ways in which a ticket of size n or size t may be overlapped with an overlapping playing set structure are considered. This is achieved by considering all possible vectors (which represent overlapping playing set structures) of length `sz`$\times 2$ which may emanate from the vector of length `sz`. If the sum of the elements in (5.1) equals n, the vector may be added to the tree, provided that it is not pruned from the tree according to the pruning rules presented in §5.3.1 and if the sum of those elements equals t; it is added to the second set of children of the parent node. The function `LevelK` calls a boolean function which returns the value `true` for every possible child vector that it produces, and the value `false` once it has produced all the possible child vectors of a given parent vector. This boolean function is named `calcv`.

The function `calcv` computes the different permutations of the values of (5.1), according to (5.2). If a *repeating set* $\{0, 1, 0, 1, 0, 1, 0, 1 \ldots\}$ is constructed, the formula $i \pmod 2$ represents

Algorithm 5.5: toPrune

Data: vector

Result: A boolean value which has the value true, if the received vector must be omitted from the lottery tree.

```
1  if vector[0] ≥ (ℓ − currentlevel + 1) ∗ t then
2      return true;
3  end
4  if min(vector[10...0], k − 1) + ... + min(vector[0...01], k − 1) + vector[00...0] ≥
   (ℓ − currentlevel + 1) × t then
5      return true;
6  end
7  if vector.level = ℓ then
8      if min(vector[10...0], k − 1) + ... + min(vector[0...01], k − 1) + vector[00...0] ≥ t then
9          return true;
10     end
11 end
12 if vector.level = ℓ − 1 then
13     if vector[0] ≥ n + 1 then
14         return true;
15     end
16     if min(vector[10...0], k − 1) + ... + min(vector[0...01], k − 1) + vector[00...0] ≥ n + t
       then
17         return true;
18     end
19 end
```

each element in the *repeating set*, where i is the zero-based index. Similarly, if the *repeating set* $\{0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, \ldots\}$ is constructed in which two 1's always follow two 0's, then each element in it is represented by the formula $\lfloor i/2 \rfloor \pmod 2$. Consider the vector $(1, 2, 2, 1)$ from which children vectors, are constructed. The values, (5.2), of those children appear in Table 5.4 and are computed in the function `calcv` by means of the use of an iterator i which is analogous to a zero-based vector of a repeating set in a column of the table. Each column in the table is a *repeating set* and each *repeating set* contains the possible values of each variable in (5.1). The possible values, according to (5.1), of $x^{(3)}_{(111)}$ are 0 or 1, and they appear in Table 5.4 one after the other. The possible values of $x^{(3)}_{(110)}$ are 0, 1 or 2 and they appear in Table 5.4 in twos. The reason for this is that the value of $x^{(3)}_{(110)}$ may only be changed once the two possible permutations of $x^{(3)}_{(111)}$ have appeared. Likewise, in the column with the heading $x^{(3)}_{(100)}$, eighteen 0's, followed by eighteen 1's, appear. The reason for this is that the value of $x^{(3)}_{(100)}$ may only change once all possible permutations of the values of the variables $x^{(3)}_{(111)}$, $x^{(3)}_{(110)}$ and $x^{(3)}_{(101)}$ appear (from the vector $(1, 2, 2, 1)$, there are $(2 + 1) \times (2 + 1) \times (1 + 1) = 18$ possible permutations of the variables $x^{(3)}_{(111)}$, $x^{(3)}_{(110)}$ and $x^{(3)}_{(101)}$). The values in Table 5.4 are computed as

$$x^{(3)}_{(111)} = i \ \left(\mod (x^{(2)}_{(11)} + 1) \right), \tag{5.5}$$

TABLE 5.4: *All the possible permutations of the values (5.1) belonging to the child nodes of the vector* $(1,2,2,1)$ *in* $\langle 6,3,t,k\rangle$*, according to (5.2) and computed by means of iterator* i*.*

i	$x^{(3)}_{100}$	$x^{(3)}_{101}$	$x^{(3)}_{110}$	$x^{(3)}_{111}$	i	$x^{(3)}_{100}$	$x^{(3)}_{101}$	$x^{(3)}_{110}$	$x^{(3)}_{111}$
0	0	0	0	0	18	1	0	0	0
1	0	0	0	1	19	1	0	0	1
2	0	0	1	0	20	1	0	1	0
3	0	0	1	1	21	1	0	1	1
4	0	0	2	0	22	1	0	2	0
5	0	0	2	1	23	1	0	2	1
6	0	1	0	0	24	1	1	0	0
7	0	1	0	1	25	1	1	0	1
8	0	1	1	0	26	1	1	1	0
9	0	1	1	1	27	1	1	1	1
10	0	1	2	0	28	1	1	2	0
11	0	1	2	1	29	1	1	2	1
12	0	2	0	0	30	1	2	0	0
13	0	2	0	1	31	1	2	0	1
14	0	2	1	0	32	1	2	1	0
15	0	2	1	1	33	1	2	1	1
16	0	2	2	0	34	1	2	2	0
17	0	2	2	1	35	1	2	2	1

$$x^{(3)}_{(110)} = \left\lfloor \frac{i}{(x^{(2)}_{(11)}+1)} \right\rfloor \ \Big(\mathrm{mod}\ (x^{(2)}_{(10)}+1)\Big), \tag{5.6}$$

$$x^{(3)}_{(101)} = \left\lfloor \frac{i}{(x^{(2)}_{(11)}+1)(x^{(2)}_{(10)}+1)} \right\rfloor \ \Big(\mathrm{mod}\ (x^{(2)}_{(01)}+1)\Big), \text{ and} \tag{5.7}$$

$$x^{(3)}_{(100)} = \left\lfloor \frac{i}{(x^{(2)}_{(11)}+1)(x^{(2)}_{(10)}+1)(x^{(2)}_{(01)}+1)} \right\rfloor \ \Big(\mathrm{mod}\ (x^{(2)}_{(00)}+1)\Big) \tag{5.8}$$

for all $0 \leq i \leq (x^{(2)}_{(11)}+1)(x^{(2)}_{(10)}+1)(x^{(2)}_{(01)}+1)(x^{(2)}_{(000)}+1)-1$. The expressions (5.5)–(5.8) may be generalised to

$$x^{(\ell+1)}_{(11\ldots1)} = i \ \Big(\mathrm{mod}\ (x^{(\ell)}_{(11\ldots1)}+1)\Big), \tag{5.9}$$

$$x^{(\ell+1)}_{(11\ldots0)} = \left\lfloor \frac{i}{(x^{(\ell)}_{(11\ldots1)}+1)} \right\rfloor \ \Big(\mathrm{mod}\ (x^{(\ell)}_{(11\ldots0)}+1)\Big), \ldots, \tag{5.10}$$

$$x^{(\ell+1)}_{(10\ldots1)} = \left\lfloor \frac{i}{(x^{(\ell)}_{(11\ldots1)}+1)\times\ldots\times(x^{(\ell)}_{(0\ldots10)}+1)} \right\rfloor \ \Big(\mathrm{mod}\ (x^{(\ell)}_{(10\ldots1)}+1)\Big) \text{ and} \tag{5.11}$$

$$x^{(\ell+1)}_{(10\ldots0)} = \left\lfloor \frac{i}{(x^{(\ell)}_{(11\ldots1)}+1)\times\ldots\times(x^{(\ell)}_{(00\ldots1)}+1)} \right\rfloor \left(\mathrm{mod}\ (x^{(\ell)}_{(00\ldots0)}+1)\right), \tag{5.12}$$

for all $0 \leq i \leq (x^{(\ell)}_{(11)}+1)\times\ldots\times(x^{(\ell)}_{(001)}+1)-1$.

Due to space constraints, comments are omitted from Algorithm 5.3 and replaced by the following description of the algorithm. In line 1, the array named `arrX` is defined as the array from the parent node. In lines 2 to 4, the stopping condition for the recursive algorithm `LevelK` is specified. In line 6, a new array, named `arrY` is defined as the array of a child node of the parent node which contains `arrX`. In line 8, all the elements in the newly added ticket are counted. In line 9 it is tested whether n elements occur in the newly added ticket, in which case it represents a valid ticket in the participant's playing set (and if $n = t$, it also represents a valid overlapping of a government ticket with the playing set in the parent node). In line 13, it is tested whether the newly added ticket has k numbers in common with at least one participant ticket. If the test is passed, the boolean value for the node corresponding to the newly created vector is assigned the value `true`. In line 16, it is tested whether the new ticket contains the same elements as any of the other tickets. If this is the case, the overlapping playing set structure represented by `arrY` is not valid, and may be omitted from the exhaustive enumeration lottery tree. In lines 20 to 22, it is tested whether the overlapping playing structure has not previously been explored in the exhaustive enumeration lottery tree. In line 24, it is tested whether the child node should be added to the second set of children. Furthermore, it is tested in lines 27 to 28, whether the overlapping playing set structure may be added to the lottery tree as a child in the first set of children. In line 33, it is tested whether the newly added ticket contains t elements which implies that it represents the overlapping of a government ticket with the participant's playing set in the parent node. In line 40, the child node is added to the second set of children of the parent node.

Another function which is of interest is a boolean function called `isDuplicate`. This function returns the value `true` if a vector is found to be a duplicate vector, and it returns `false` otherwise. This function is mentioned in §5.3.1. The `isDuplicate` function is defined in Algorithm 5.4. An efficient permutation class is used to construct the permutations for the `isDuplicate` function. This class was obtained via the Microsoft Developer Network website [30]. The function `isDuplicate` refers to a function `GetVectorY` which constructs the ancestor vector $\vec{\mathbf{Y}}^{(\ell)}$ as described in (5.4) for the vector $\vec{\mathbf{X}}^{(\ell)}$ at level ℓ.

5.4 Results

Solutions (in overlapping structural vector notation) for all non-isomorphic lottery problem instances for $m \leq 10$ and using up to $\ell = 6$ tickets, as found by the exhaustive enumeration lottery tree method described in §5.3 are presented in Table 5.5. Due to space constraints, commas are omitted from the vector representations of the overlapping playing set structures. Therefore, in the following table, the vector representation of an overlapping playing set structure is of the form $(x^{(\ell)}_{(000\ldots0)}x^{(\ell)}_{(000\ldots1)}\ldots x^{(\ell)}_{(111\ldots1)})$. A value of 0.000 in the fourth column (which indicates execution time) implies that the execution time was less than one thousandth of a second.

TABLE 5.5: *Solutions (in overlapping structural vector notation) for all non-isomorphic lottery problem instances in which $m \leq 10$, as found by the exhaustive enumeration lottery tree method described in §5.3. The entries in the first and second columns represent the lottery instance and the fixed playing set cardinality, respectively, for which a maximum resource utilisation value is sought. The third column, labelled "Ψ_ℓ" contains the associated resource utilisation number obtained via the exhaustive enumeration lottery tree method. The fourth column contains the required execution time and the fifth column contains all the non-isomorphic overlapping playing set structures for the problem instance.*

Lottery	ℓ	Ψ_ℓ	Time (secs)	Overlapping playing set structure
$\langle 6,3,3,2\rangle$	1	0.5	0.000	(33)
	2	1	0.000	(0330)
$\langle 7,3,3,2\rangle$	1	0.371	0.000	(43)
	2	0.743	0.000	(1330)
	3	0.914	0.000	(01301200)
	4	1	0.031	(0030010001001100)
				(0020020001101000)
$\langle 7,3,4,2\rangle$	1	0.629	0.000	(43)
	2	1	0.000	(1330)
$\langle 8,3,3,2\rangle$	1	0.286	0.000	(53)
	2	0.571	0.000	(2330)
	3	0.786	0.000	(02302100)
	4	0.893	0.047	(0030110002001000)
	5	1	1.141	(00300100000010000000100002000000)
$\langle 8,3,4,2\rangle$	1	0.5	0.000	(53)
	2	0.871	0.000	(2330)
	3	1	0.000	(02302100)
$\langle 8,3,5,2\rangle$	1	0.714	0.000	(53)
	2	1	0.000	(2330)
$\langle 8,4,3,2\rangle$	1	0.5	0.000	(44)
	2	1	0.000	(0440)
$\langle 8,4,4,2\rangle$	1	0.757	0.000	(44)
	2	1	0.000	(1331)
				(0440)
$\langle 8,4,4,3\rangle$	1	0.243	0.000	(44)
	2	0.486	0.000	(1331)
				(0440)
	3	0.671	0.000	(01211210)
	4	0.857	0.094	(0001021001201000)
	5	0.914	4.234	(0001001000100100001001000100100 0)
				(0000001001200000000102000000100 0)
	6	1	314.359	(0000000000000100000100000020000000000110010000000000100000000000)
$\langle 9,3,3,2\rangle$	1	0.226	0.000	(63)
	2	0.452	0.000	(3330)
	3	0.679	0.000	(03303000)
	4	0.774	0.063	(0130110011001000)
	5	0.893	2.250	(00300100010010000100100010000000)
	6	0.952	159.516	(00000100011000000100200000000000002000001000000000000000000000000)
				(00000100011000000110100000000000001010001000000000000000000000000)
$\langle 9,3,4,2\rangle$	1	0.405	0.000	(63)
	2	0.738	0.000	(3330)
	3	1	0.000	(03303000)
$\langle 9,3,5,2\rangle$	1	0.595	0.000	(63)
	2	0.929	0.000	(3330)
	3	1	0.000	(03303000)
$\langle 9,3,6,2\rangle$	1	0.774	0.000	(63)
	2	1	0.000	(3330)
$\langle 9,4,3,2\rangle$	1	0.405	0.000	(54)
	2	0.81	0.000	(1440)
	3	0.952	0.000	(01401300)
	4	1	0.141	(0030020002001010)
				(0020021002101000)
				(0040010001001200)
				(0030020001101100)
				(0020030001201000)
$\langle 9,4,4,2\rangle$	1	0.643	0.000	(54)
	2	1	0.000	(1440)
$\langle 9,4,4,3\rangle$	1	0.167	0.000	(54)
	2	0.333	0.000	(2331)

Table 5.5 – continued from previous page

Lottery	ℓ	Ψ_ℓ	Time (secs)	Overlapping playing set structure
				(1440)
	3	0.5	0.016	(02212110)
	4	0.635	0.203	(0011111002101000)
	5	0.738	11.672	(00010110001010000010100002000000)
				(00010100011010000010101001000000)
				(00000110021000000011100000001000)
				(00000110011010000011100001000000)
	6	0.825	1403.594	(00000001001000000100010000001000000010000110000000100000000000000)
				(00000100011000000001000000001000000000100100000000101000000000000)
				(00000010020000000001000000001000000001000010000000101000000000000)
$\langle 9,4,5,2\rangle$	1	0.833	0.000	(54)
	2	1	0.000	(2331)
				(1440)
$\langle 9,4,5,3\rangle$	1	0.357	0.000	(54)
	2	0.714	0.000	(1440)
	3	0.857	0.016	(02212110)
				(02301210)
				(01401300)
	4	0.976	0.219	(0020030001201000)
	5	1	12.250	(00010110001010000010100002000000)
				(00000110012000000110010000001000)
				(00100010021000000010020000001000)
				(00000110021000000020010000001000)
				(00000210002000000010100002000000)
				(00400000000001000000010001001100)
				(00300000001001000000020001001000)
				(00100300001000000010000001101000)
				(00100200002000000010010001001000)
				(00000300002000000020000001001000)
				(00000200002010000020010001000000)
				(00000200003000000010110001000000)
$\langle 9,4,6,3\rangle$	1	0.595	0.000	(54)
	2	1	0.000	(1440)
$\langle 10,3,3,2\rangle$	1	0.183	0.000	(73)
	2	0.367	0.000	(4330)
	3	0.55	0.000	(13303000)
	4	0.667	0.063	(0220210020100000)
				(0220300011100000)
	5	0.767	2.719	(01103000011000000110000010000000)
	6	0.85	217.641	(00100100010010001020000000000000001001000100000000000000000000000)
				(00100100101000000110100000000000001001000100000000000000000000000)
				(00001100101000000110100000000000001100000100000000000000000000000)
				(00003000011000000110000000000000011000000000000000000000010000000)
				(00002000011000000110100000000000011000001000000000000000000000000)
$\langle 10,3,4,2\rangle$	1	0.333	0.000	(73)
	2	0.624	0.000	(4330)
	3	0.871	0.000	(13303000)
	4	0.957	0.063	(0130300012000000)
	5	1	1.234	(00303000010000000100000011000000)
				(00203000020000000110000010000000)
$\langle 10,3,5,2\rangle$	1	0.5	0.000	(73)
	2	0.833	0.000	(4330)
	3	1	0.000	(13303000)
$\langle 10,3,6,2\rangle$	1	0.667	0.000	(73)
	2	0.957	0.000	(4330)
	3	1	0.000	(22302100)
				(13303000)
$\langle 10,3,7,2\rangle$	1	0.817	0.000	(73)
	2	1	0.000	(4330)
$\langle 10,3,8,2\rangle$	1	0.933	0.000	(73)
	2	1	0.000	(5221)
				(4330)
$\langle 10,4,3,2\rangle$	1	0.333	0.000	(64)
	2	0.667	0.000	(2440)
	3	0.867	0.016	(02402200)
	4	1	0.156	(0040020002002000)
$\langle 10,4,4,2\rangle$	1	0.548	0.000	(64)
	2	0.924	0.000	(2440)

Table 5.5 – continued from previous page

Lottery	ℓ	Ψ_ℓ	Time (secs)	Overlapping playing set structure
	3	1	0.000	(03303001)
				(03302110)
				(02402200)
$\langle 10,4,4,3\rangle$	1	0.119	0.000	(64)
	2	0.238	0.000	(3331)
				(2440)
	3	0.357	0.000	(12212110)
				(03303001)
				(03302110)
	4	0.476	0.328	(0111111011101000)
	5	0.595	23.313	(00010110011010000110100010000000)
	6	0.662	3825.359	(00000110001010000010000001000000000100001000000000100100000000000)
$\langle 10,4,5,2\rangle$	1	0.738	0.000	(64)
	2	1	0.000	(2440)
$\langle 10,4,5,3\rangle$	1	0.262	0.000	(64)
	2	0.524	0.000	(2440)
	3	0.714	0.000	(03302110)
	4	0.833	0.359	(0111111011101000)
	5	0.952	24.875	(0220011001102000)
	6	0.976	3920.016	(00010110011010000110100010000000)
				(00000000100101000001010000100000000101000010000000100000000000000)
				(00000000102000000002000000000100000002000001000000100000000000000)
$\langle 10,4,6,2\rangle$	1	0.881	0.000	(64)
	2	1	0.000	(3331)
				(2440)
$\langle 10,4,6,3\rangle$	1	0.452	0.000	(64)
	2	0.829	0.000	(2440)
	3	0.971	0.000	(02402200)
	4	1	0.313	(0030111003001000)
				(0040110002001100)
				(0040020002002000)
				(0030120002101000)
$\langle 10,4,7,3\rangle$	1	0.667	0.000	(64)
	2	1	0.000	(2440)
$\langle 10,5,3,2\rangle$	1	0.5	0.000	(55)
	2	1	0.000	(0550)
$\langle 10,5,4,2\rangle$	1	0.738	0.000	(55)
	2	1	0.016	(1441)
				(0550)
$\langle 10,5,4,3\rangle$	1	0.262	0.000	(55)
	2	0.524	0.000	(1441)
				(0550)
	3	0.7	0.000	(02211220)
	4	0.876	0.625	(0001022002201000)
	5	0.924	53.328	(00010010001010100010020002000000)
				(00010010002001000010020001001000)
	6	0.986	10658.047	(00000000000001000001000000300000000002100100000000001000000000000)
$\langle 10,5,5,2\rangle$	1	0.897	0.000	(55)
	2	1	0.000	(2332)
				(1441)
				(0550)
$\langle 10,5,5,3\rangle$	1	0.5	0.000	(55)
	2	1	0.000	(0550)
$\langle 10,5,5,4\rangle$	1	0.103	0.000	(55)
	2	0.206	0.000	(2332)
				(1441)
				(0550)
	3	0.31	0.016	(11121220)
				(02212111)
				(02211220)
	4	0.413	0.563	(0011110102101010)
				(0011021002101010)
				(0001022002201000)
	5	0.516	51.125	(00000101011000100011100001001000)
	6	0.619	10498.375	(00000001000000100001000001001000000001000110000000101000000000000)

5.5 Chapter overview

In this chapter, the exhaustive enumeration lottery tree, first presented by Gründlingh [16] for the case $n = t$ was reviewed and adapted for the case $n \neq t$. The implementation was also presented in the form of a pseudocode listing. This solution method was implemented in order to compare its results to those obtained in Chapter 4. The results in Table 5.5 may be compared to the results in Table C. The resource utilisation numbers in Table C confirm the results in Table 5.5. This implies that both methods are viable for solving small instances of both the incomplete lottery problem and the resource utilisation problem.

It may be seen that the results in Table 5.5 were only solved for values of $1 \leq \ell \leq 6$. The reason for this is that the exhaustive enumeration lottery tree tends to grow exponentially as the value of ℓ increases. If the problem is solved using the exhaustive enumeration lottery tree method for $\ell > 6$, it would typically take longer than 14 400 seconds to find solutions, which is longer than the time limit used in the ILP solution method. This presents a limitation on the exhaustive enumeration lottery tree method, because only playing sets of a limited cardinality may be considered. It may therefore be useful develop a method in which playing sets of a larger cardinality may be considered. This is explored in the chapter which follows.

CHAPTER 6

Overlapping playing set structures: An approximation approach

Contents

In this chapter, the overlapping playing set structures presented in Chapter 1, and used as the nodes in the exhaustive enumeration lottery tree in Chapter 5, are considered again. The manner in which the elements are distributed amongst the compartments of the overlapping playing set structure is considered to give rise to feasible (or infeasible) solutions to the incomplete lottery problem and the resource utilisation problem. Good feasible solutions to these problems may be found by means of (meta)heuristics.

In §6.1, the combinatorial optimisation problem to be solved (approximately) is formulated, in fulfilment of Thesis Objective IV(a). Various approximation algorithms (such as random search, simulated annealing, tabu search and genetic algorithms) are presented as possible solution approaches to the above-mentioned combinatorial optimisation problem in fulfilment of Thesis objective IV(b). These algorithms are applied to the same small problem instances considered in Chapters 4 and 5 in fulfilment of Thesis objective IV(c). The chapter closes with a comparison of the algorithms mentioned above in terms of their performance, in fulfilment of Thesis objective IV(d).

6.1 Problem definition

Consider an overlapping playing set structure representing a playing set of cardinality ℓ and consisting of 2^ℓ compartments denoted by the entries in the vector $(x^{(\ell)}_{(000\ldots0)}, x^{(\ell)}_{(000\ldots1)}, \ldots, x^{(\ell)}_{(111\ldots1)})$. There are m elements which may be placed inside these compartments in such a way that for each ticket in the overlapping playing set structure, the sum of the elements in the compartments which belong to that ticket equals n, *i.e.*

$$\begin{aligned} x^{(\ell)}_{(0\ldots1)} + x^{(\ell)}_{(0\ldots11)} + \ldots + x^{(\ell)}_{(111\ldots1)} &= n, \\ x^{(\ell)}_{(0\ldots10)} + x^{(\ell)}_{(0\ldots11)} + \ldots + x^{(\ell)}_{(111\ldots1)} &= n, \\ \vdots &\quad \vdots \\ x^{(\ell)}_{(10\ldots0)} + x^{(\ell)}_{(10\ldots1)} + \ldots + x^{(\ell)}_{(111\ldots1)} &= n. \end{aligned}$$

The objective is to maximise the probability-of-win value (also known as the resource utilisation number, $\Psi_\ell(m, n, t, k)$) of the overlapping playing set structure which is computed in the same way as described in §5.1.2.

The algorithms in this chapter are implemented with the goal of achieving a high resource utilisation value, given a fixed playing set cardinality ℓ, within an execution time that is less than those expended by the methods presented in Chapters 4 and 5. If a resource utilisation value of 1 is achieved, an upper bound $L_1(m, n, t, k) \leq \ell$ on the complete lottery number for the given lottery instance is found. If a resource utilisation $\psi < 1$ is achieved, a lower bound $\psi \leq \Psi_\ell(m, n, t, k)$ on the resource utilisation number for the given lottery instance is found, and hence an upper bound $L_\psi(m, n, t, k) \leq \ell$ on the incomplete lottery number is also found.

The objective is therefore to find a good approximation of the resource utilisation number associated with a feasible playing set of cardinality ℓ for a lottery $\langle m, n, t, k\rangle$.

6.2 A random search algorithm

The random search algorithm presented in pseudocode form as Algorithm 6.1 involves the repeated construction of feasible randomly generated overlapping playing set structures of a given cardinality ℓ. In Figure 6.1, each point in the graph represents the resource utilisation of such a random construction, hence yielding a lower bound on the resource utilisation number $\Psi_7(12, 5, 7, 4)$. It may be seen in this graph that the random search never converges to some solution quality, which is typical behaviour of a random search algorithm. In Figure 6.1, the highest resource utilisation value achieved by Algorithm 6.1 for $\ell = 7$ tickets in the lottery $\langle 12, 5, 7, 4\rangle$ is $\frac{754}{792} \approx 0.952$ at iteration 354.

Assuming the generation of a random overlapping playing set structure has a worst-case complexity of $\mathcal{O}(2^\ell)$ and that the computation of a probability-of-win value has a worst-case complexity of

$$\mathcal{O}\left(\binom{2^\ell + t - 1}{t}\right),$$

the worst-case complexity of I iterations of the random search algorithm is

$$\mathcal{O}\left(2^\ell \binom{2^\ell + t - 1}{t} I\right).$$

Algorithm 6.1: Random search

Data: The lottery parameters m, n, t and k, the number of tickets ℓ, and the number of algorithmic iterations required I.

Result: The highest probability-of-win value found during the I iterations.

```
for i = 0; i ≤ I do
    CurrentSolution = new random solution;
    if Evaluate(CurrentSolution) > Evaluate(BestSolution) then
        BestSolution = CurrentSolution;
    end
end
return BestSolution;
```

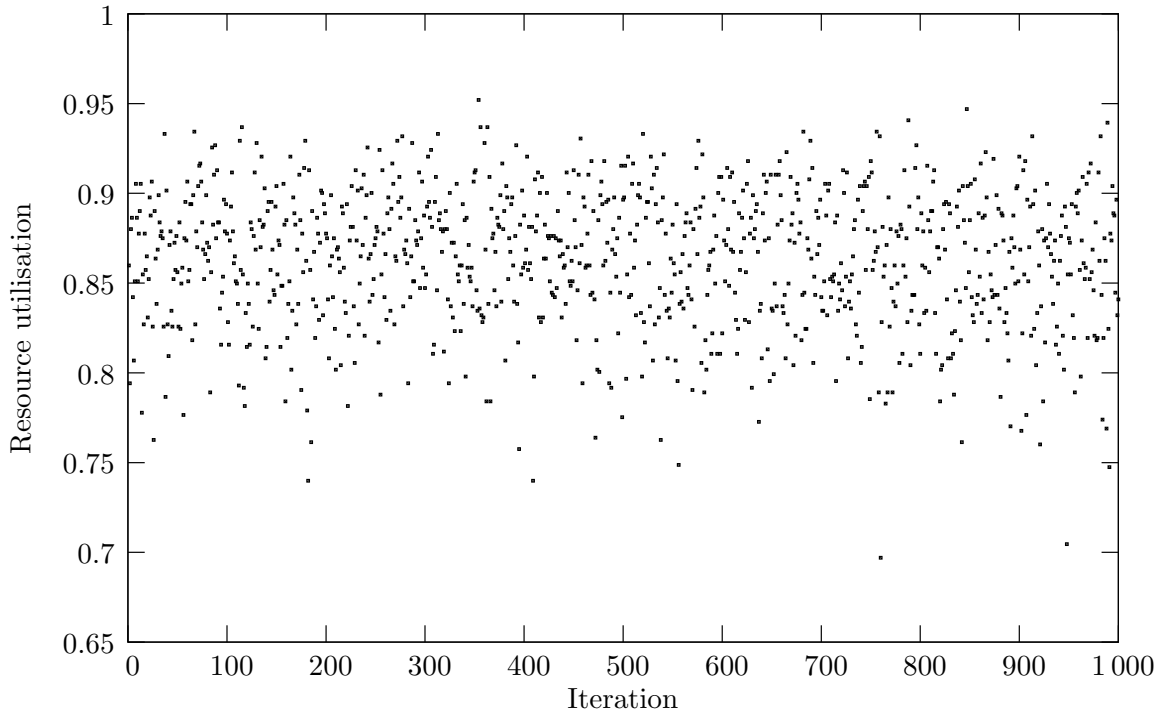


FIGURE 6.1: *Resource utilisation as a function of random search iterations. Each point represents a lower bound on the resource utilisation number* $\Psi_7(12, 5, 7, 4)$ *for each of the* 1 000 *iterations of the random search algorithm.*

6.3 Two local search algorithms

A local search algorithm involves the exploration of the solution space of a combinatorial optimisation problem by traversing small regions of the solution space at a time. In order to accomplish this, a neighbourhood function is defined for each solution. The neighbourhood function applies a small transformation to a solution in order to obtain a set of new solutions, known as the neighbourhood of the original solution.

Some primitive local search algorithms are prone to eventually explore the same neighbourhood of a solution continually. In this situation, the search algorithm is said to be *trapped at a local optimum*. In this section, two local search algorithms (namely *simulated annealing*, and *tabu search*) are presented. Both these algorithms are designed to be able to escape from local optima. They are used to obtain a lower bound on the resource utilisation number $\Psi_\ell(m, n, t, k)$ by repeatedly constructing different overlapping playing set structures for the specific lottery problem instance, keeping track of the best resource utilisation achieved by ℓ tickets.

Algorithm 6.2: firstSwap

Data: A vector *vec*, representing an overlapping playing set structure and whose neighbourhood is to be explored.
Result: A list of neighbourhood vectors of *vec*.

```
1  for i < vec.Length do
2      for j < vec.Length do
3          Move one element from compartment i to compartment j;
4          Call secondSwap(vec);
5          Restore vec to its original form;
6      end
7  end
```

Algorithm 6.3: secondSwap

Data: An infeasible vector *vec*, and the zero-based vector indices i and j representing the compartments in the overlapping playing set structure in which no elements are added or removed.
Result: The neighbourhood vectors of *vec*.

```
1  for a < vec.Length do
2      for b < vec.Length do
3          if a ≠ i and a ≠ j and b ≠ i and b ≠ j then
4              Move one element from compartment a to compartment b;
5              if the new vector is feasible then
6                  Compute probability-of-win value;
7                  Add to neighbourhood;
8              end
9          end
10     end
11 end
```

6.3.1 The neighbourhood of a solution

Local search methods involve the successive exploration of the neighbourhoods of solutions. These neighbourhoods are formed by applying a so-called neighbourhood function. Given a feasible overlapping playing set structure, one element from compartment i may be moved to each compartment $j \neq i$, according to Algorithm 6.2. This move may cause the overlapping playing set structure to become infeasible. Therefore, the next step is to move elements between the compartments in the overlapping playing set structure in order to regain a feasible solution. Once an element is moved from compartment i or to compartment j according to Algorithm 6.2, for each compartment (except compartments i or j) which contains at least one element, an attempt is made to move an element from that compartment to all the other compartments (except compartments i or j), one compartment at a time, according to Algorithm 6.3. All the resulting *feasible* overlapping playing set structures make up the *neighbourhood* of the original structure.

The following example illustrates how the neighbourhood of the overlapping playing set structure (represented by the vector $(4,1,0,0,0,0,1,2)$) may be constructed.

Example 6.1 (The neighbourhood of $(4,1,0,0,0,0,1,2)$) *For the lottery $\langle 8,3,t,k\rangle$, Algorithm 6.2 results in the construction of the vectors* $(5,0,0,0,0,0,1,2)$, $(5,1,0,0,0,0,0,2)$, $(5,1,0,0,0,0,1,1)$, $(3,2,0,0,0,0,1,2)$, $(4,2,0,0,0,0,0,2)$, $(4,2,0,0,0,0,1,1)$, $(3,1,1,0,0,0,1,2)$, $(4,0,1,0,0,0,1,2)$, $(4,1,1,0,0,0,0,2)$, $(4,1,1,0,0,0,1,1)$, $(3,1,0,1,0,0,1,2)$, $(4,0,0,1,0,0,1,2)$, $(4,1,0,1,0,0,0,2)$, $(4,1,0,1,0,0,1,1)$, $(3,1,0,0,1,0,1,2)$, $(4,0,0,0,1,0,1,2)$, $(4,1,0,0,1,0,0,2)$, $(4,1,0,0,1,0,1,1)$, $(3,1,0,0,0,1,1,2)$, $(4,0,0,0,0,1,1,2)$, $(4,1,0,0,0,1,0,2)$, $(4,1,0,0,0,1,1,1)$, $(3,1,0,0,0,0,2,2)$, $(4,0,0,0,0,0,2,2)$, $(4,1,0,0,0,0,2,1)$, $(3,1,0,0,0,0,1,3)$, $(4,0,0,0,0,0,1,3)$, $(4,1,0,0,0,0,0,3)$. *Each of the above vectors represents an infeasible solution and is therefore used as input to Algorithm 6.3. As an example, consider the vector* $(3,1,0,0,1,0,1,2)$ *taken as input to Algorithm 6.3. The solutions which result from applying Algorithm 6.3 to this vector appear graphically in Figure 6.2. In the figure, it may be seen that the only two feasible solutions which result from applying Algorithm 6.3 to the vector* $(3,1,0,0,1,0,1,2)$ *are the vectors* $(3,1,1,0,1,0,0,2)$ *and* $(3,1,0,1,1,0,1,1)$ *— denoted by means of bold-faced frames in Figure 6.2. Likewise, Algorithm 6.3 is applied to all the above vectors which result as output from Algorithm 6.2. All the feasible solutions which result from applying Algorithm 6.3 are considered to be in the neighbourhood of the solution represented by the vector* $(4,1,0,0,0,0,1,2)$ *— this neighbourhood is shown in Figure 6.3.* ■

6.3.2 Simulated annealing

The metaheuristic *simulated annealing* is based on a cooling process in thermodynamics. In order to grow a crystal structure, a certain material is heated until is has reached a molten state. The temperature to which this molten material is exposed, is then reduced in a controlled manner until the molten material regains a solid state. If cooling takes place at a rate which is undesirably quick, the resulting structure of the solid material may be unstable; therefore the material is allowed to cool slowly. Simulated annealing is a metaheuristic that mimics this cooling process and consists of a mechanism which allows a local search algorithm to escape local optima. This is achieved by specifying a temperature variable, $T \geq 0$, and a cooling ratio, r $(0 < r \leq 1)$. At each iteration, the temperature variable is multiplied by the cooling ratio in order to decrease its temperature value. Given an overlapping playing set structure S_1 and its neighbour S_2, if the probability-of-win value associated with S_2 is larger than the probability-

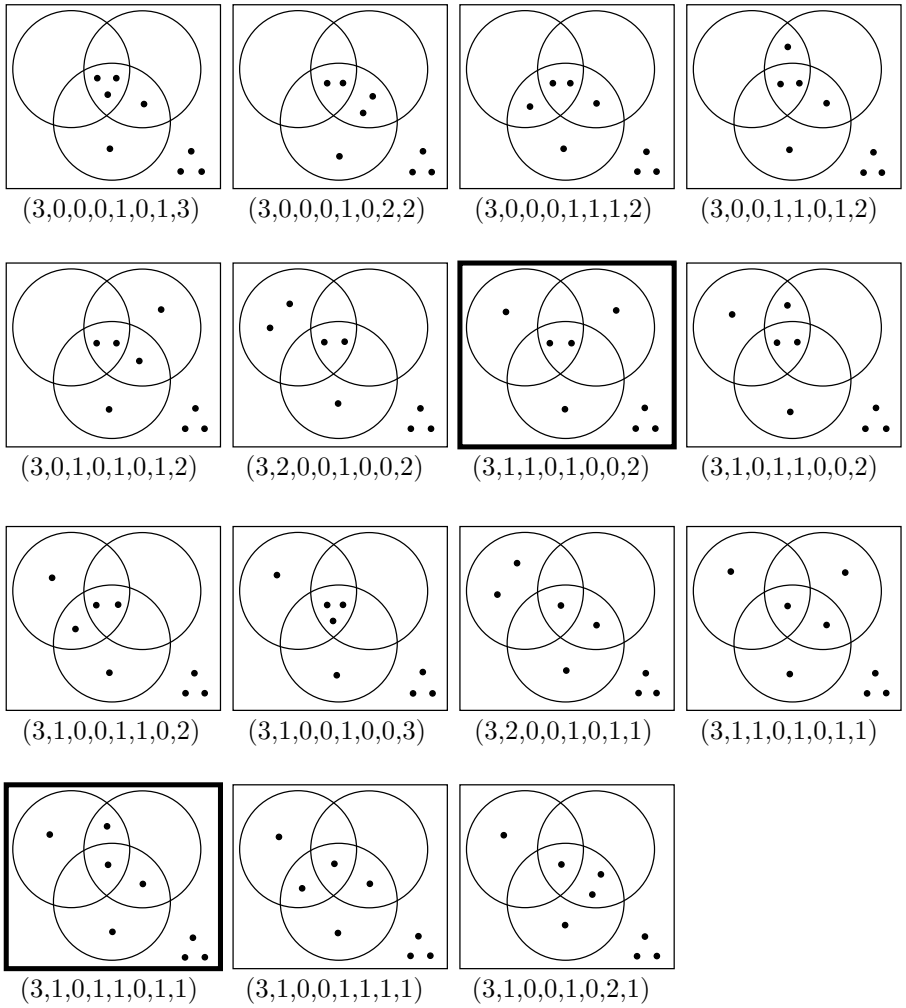


FIGURE 6.2: *"Overlapping playing set structures" for the lottery* $\langle 8, 3, t, k\rangle$ *resulting from applying Algorithm 6.3 to the vector* $(3, 1, 0, 0, 1, 0, 1, 2)$.

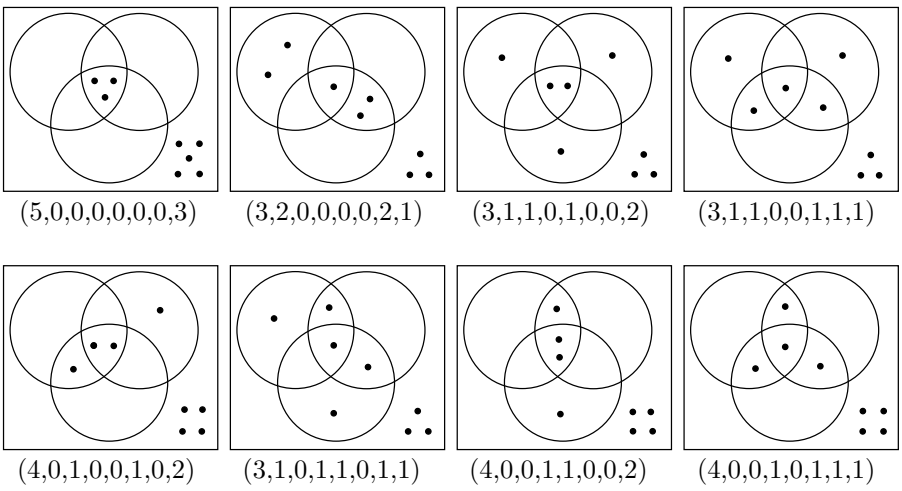


FIGURE 6.3: *The neighbourhood of the overlapping playing set structure for the lottery* $\langle 8, 3, t, k\rangle$ *represented by the vector* $(4, 1, 0, 0, 0, 0, 1, 2)$.

of-win value associated with S_1, a transition from S_1 to S_2 is made, otherwise the transition to the inferior solution S_2 from S_1 is made with probability

$$p_{escape} = \exp\left[\frac{\text{Evaluate}(S_2) - \text{Evaluate}(S_1)}{T}\right].$$

This implies that Algorithms 6.2 and 6.3 are not run to completion, but are only used to find a different overlapping playing set structure whose neighbourhood can be explored. If the transition occurs, the neighbourhood of the overlapping playing set structure S_2 is explored — and this neighbourhood may be in a vastly different area of the solution space compared to where the neighbourhood of the current feasible overlapping playing set structure is, thereby enabling the search to escape from a local optimum.

Algorithm 6.4: Simulated annealing

Data: The lottery parameters m, n, t and k, the playing set cardinality ℓ, the minimum temperature T_{min}, the maximum temperature T_{max}, the cooling rate r.

Result: The highest resource utilisation number found via the Simulated annealing heuristic.

```
1  StartSolution ← A randomly generated solution.
2  Initialize T ← T_max.
3  CurrentSolution ← StartSolution.
4  Evaluate(CurrentSolution).
5  for i ≤ I do
6      Select NewSolution from the neighbourhood of CurrentSolution.
7      if Evaluate(NewSolution) > Evaluate(CurrentSolution) then
       CurrentSolution ← NewSolution.
8      else if Random[0,1) < e^((Evaluate(NewSolution)−Evaluate(CurrentSolution))/T then
9          CurrentSolution ← NewSolution.
10     if T > T_min then
11         T ← r × T.
12     else
13         T ← T_max.
14     end
15 end
```

Three parameters are fixed for the simulated annealing algorithm. These are the maximum temperature, the minimum temperature, and the cooling rate. The values of these parameters have a significant impact on the performance of the algorithm with regards to how quickly a good solution is obtained, how widely the solution space is explored, and the actual resource utilisation of the best solution obtained by the algorithm. These parameters are problem-specific, which means that an investigation is required for each problem in order to find the best values for the parameters. In terms of the problem investigated in this chapter, if the temperature cools at a relatively quick rate, the solution space is more widely searched, because a transition may occur from a current solution more often to one which is inferior in terms of resource utilisation. If the value of the temperature cools slowly, the neighbourhood of each solution is searched more intensively with the result that the solution space is not widely explored. It is ideal to find a balance between searching the neighbourhood of a solution as intensively as possible, and searching the solution space as widely as possible. Due to the large size of the solution space, if an attempt is made to search it as widely as possible, it may occur that only weak solutions are encountered, because not enough time is allowed to search the neighbourhood of those solutions for improvements. However, if the neighbourhood

of each solution is searched thoroughly each time a solution is found, it may occur that only a small portion of the solution space is explored, which is not ideal. In order to decide which values to use for the variables, the value of the minimum temperature may be fixed at 0.01, and the values of the maximum temperature and the cooling rate may be varied. From Table 6.1, it may be seen that a value of $r = 0.99$ is ideal, meaning that the temperature cools slowly, which implies that the neighbourhood of each solution is explored intensively. As the number of iterations increases, causing the temperature to drop, the probability of choosing a neighbouring solution which is inferior to the current solution grows. Also, if d is the difference between the probability-of-win value associated with a current solution and the probability-of-win value associated with a neighbouring solution which is less fit than it, as d increases, the probability of choosing an inferior neighbour increases.

In Figure 6.4, each point represents the resource utilisation associated with the overlapping playing set structure being considered during an iteration of Algorithm 6.4 when a playing set of cardinality $\ell = 7$ is considered in the lottery $\langle 12, 5, 7, 4\rangle$. From iteration 1 to 22, the resource utilisation value increases while the value of the temperature variable T slowly decreases, causing the probability of a transition to an inferior solution to slowly increase. Due to the increasingly higher probability of a transition occurring to an inferior solution, the resource utilisation value appears to decrease from iteration 58 to 128. Once the temperature variable T reaches its minimum allowable value, T_{min} it is reset to its original maximum value, T_{max}. The resulting increased value of the temperature variable T causes the probability of a transition to an inferior solution to be close to zero, which leads to the gradual increase of the resource utilisation value from iteration 182 to 215. This process is repeated several times throughout the 1 000 iterations. In Figure 6.4 the highest resource utilisation achieved via the simulated annealing algorithm is 0.985.

In the simulated annealing algorithm, the worst-case complexity is achieved if the entire neighbourhood of a solution is explored. This implies that both Algorithms 6.2 and 6.3 are always executed to completion. Algorithm 6.2 is completed with a worst-case complexity of

$$\mathcal{O}\left(2\binom{2^\ell}{2}\right)$$

and Algorithm 6.3 is similar to Algorithm 6.2, except that for each iteration of Algorithm 6.3, the solution must be validated. Therefore, the worst-case complexity of Algorithm 6.3 is

$$\mathcal{O}\left(2\binom{2^\ell}{2}\binom{2^\ell + t - 1}{t}\right)$$

which implies that, the worst-case time complexity of I iterations of the simulated annealing approach is

$$\mathcal{O}\left(4\binom{2^\ell}{2}^2\binom{2^\ell + t - 1}{t}I\right).$$

6.3.3 Tabu search

Tabu search is another metaheuristic method incorporating a mechanism which allows a local search to escape local optima. This method employs a short-term memory structure containing information on moves whose reversals are not allowed (or are considered tabu) for a predetermined number of iterations. A tabu move is defined as follows. Given a transition from a

TABLE 6.1: *Comparison of the largest resource utilisation value ψ obtained using $\ell = 7$ tickets in the lotteries $\langle 10, 4, 5, 3\rangle$ and $\langle 12, 5, 7, 4\rangle$ and the number of iterations I it takes to find that value, under the conditions that the maximum temperature T_{max} and the cooling rate r of the simulated annealing heuristic are varied.*

$\langle 10, 4, 5, 3\rangle, \ell = 7$								
	$T_{max} = 100$		$T_{max} = 75$		$T_{max} = 50$		$T_{max} = 25$	
r	ψ	I	ψ	I	ψ	I	ψ	I
0.25	0.956	5	0.976	82	0.956	3	0.952	34
0.5	0.937	6	1.000	72	0.881	5	0.937	6
0.75	1.000	13	1.000	41	1.000	38	1.000	10
0.99	1.000	15	1.000	19	1.000	179	1.000	13

$\langle 12, 5, 7, 4\rangle, \ell = 7$								
	$T_{max} = 100$		$T_{max} = 75$		$T_{max} = 50$		$T_{max} = 25$	
r	ψ	I	ψ	I	ψ	I	ψ	I
0.25	0.919	4	0.886	4	0.831	5	0.856	2
0.5	0.860	93	0.918	7	0.913	7	0.905	5
0.75	0.944	313	0.948	203	0.902	40	0.878	40
0.99	0.985	22	0.985	34	0.985	23	0.985	24

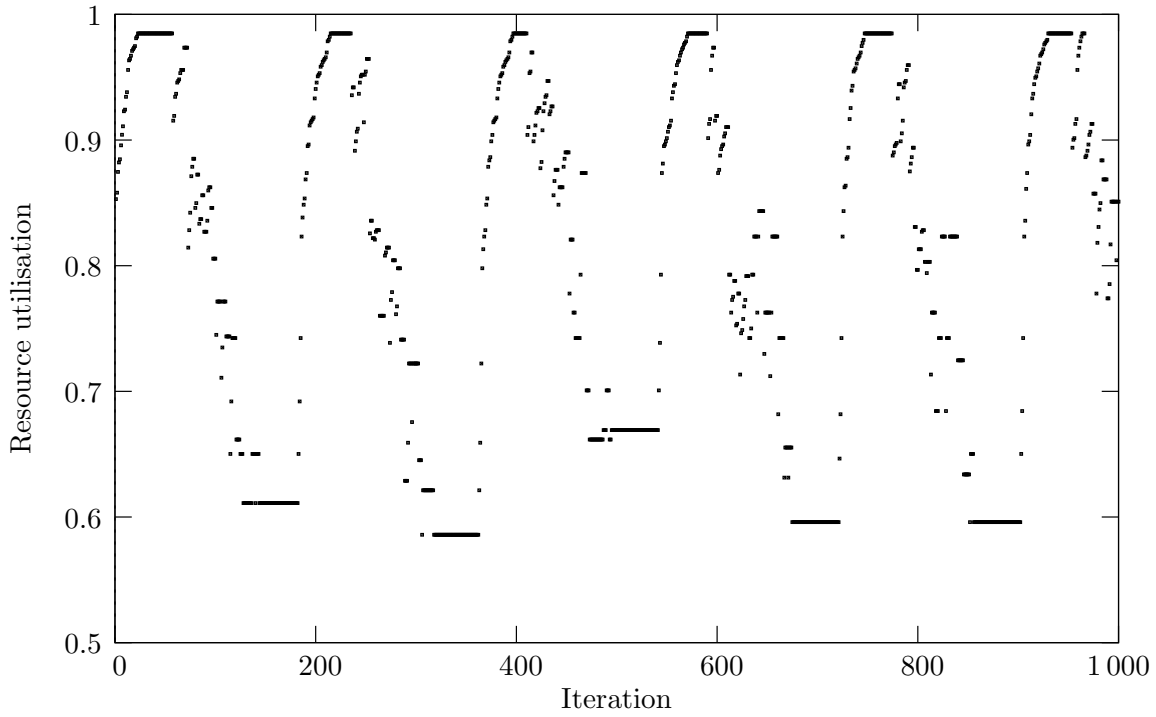


FIGURE 6.4: *Lower bounds on the resource utilisation number $\Psi_7(12, 5, 7, 4)$ as a function of simulated annealing iterations. Each point represents the resource utilisation value associated with the overlapping playing set structure considered during an iteration of Algorithm 6.4.*

solution (*i.e.* an overlapping playing set structure) x_1 to a solution x_2, it is tabu to make the transition back from x_2 to x_1. By classifying these transitions as tabu and accumulating them in a so-called tabu list, the algorithm is prevented (for a certain number of iterations) from repeatedly looping through solutions which have already been explored. This results in a more widespread search of the solution space. It is, however, desirable to search each region of a solution space thoroughly. Therefore, once the tabu list has reached a certain length (called the *tabu tenure*), the entries that have been inside it the longest are removed, one entry per iteration in a first-in-first-out manner. By removing entries from the tabu list, previously explored sections of the solution space may be explored again. If the tabu list contains many entries, the search through the solution space is widely spread, but not thorough, while if the tabu list contains comparatively few entries, the search through the solution space is thorough, but not necessarily widespread. It is ideal to achieve a balance between a widespread search and a thorough search under the restriction that a limited number of search iterations are carried out. In order to determine an ideal tabu list length, its value was varied between the values $\{20, 40, 60, 80, 100\}$ when searching for good approximations of the values $\Psi_7(10, 4, 5, 3)$ and $\Psi_7(12, 5, 7, 4)$. The number of distinct solutions found and the largest resource utilisation number uncovered was recorded. From Table 6.2, the largest number of distinct solutions is encountered with a tabu list length of 60 for both lottery problem instances. This implies that if a relatively widespread search is required, a tabu list length of 60 may be used when searching for resource utilisation numbers of small lottery instances.

Algorithm 6.5: Tabu search

Data: The lottery parameters m, n, t and k, the playing set cardinality ℓ, and the tabu tenure T.

Result: The highest resource utilisation number found via the tabu search heuristic.

```
1  StartSolution = random solution.
2  Initialize T.
3  CurrentSolution = StartSolution.
4  Evaluate(CurrentSolution).
5  for i ≤ I do
6      Select best neighbour, bestNeighbour, of CurrentSolution from non-tabu neighbours of
       CurrentSolution.
7      CurrentSolution = bestNeighbour.
8      Record the moves bestNeighbour ← CurrentSolution and
       CurrentSolution ← bestNeighbour as tabu.
9      if the tabu list contains more than T items then
10         Remove the oldest item from the list.
11     end
12 end
```

In Figure 6.5, each point in the graph represents the resource utilisation associated with an overlapping playing set structure considered during iteration i of Algorithm 6.5 when a playing set of cardinality 7 is considered in the lottery $\langle 12, 5, 7, 4\rangle$. It can be seen in this graph that the resource utilisation value rapidly converges to a high value, and then fluctuates slightly as the neighbourhoods of solutions are iteratively explored. The resource utilisation value does not decrease dramatically because only the most superior neighbour of each solution is considered. From Figure 6.5, highest resource utilisation number achieved via the tabu search algorithm is 0.985.

TABLE 6.2: *Bounds (ψ) on the resource utilisation numbers $\Psi_7(10,4,5,3)$ and $\Psi_7(12,5,7,4)$ obtained via a tabu search, and the number of distinct solutions encountered (I) for different values of the tabu tenure, T.*

Lottery		$T=20$	$T=40$	$T=60$	$T=80$	$T=100$
$\langle 10,4,5,3\rangle$	I	12	18	19	18	18
	ψ	1	1	1	1	1
$\langle 12,5,7,4\rangle$	I	15	18	21	18	19
	ψ	0.985	0.985	0.985	0.985	0.985

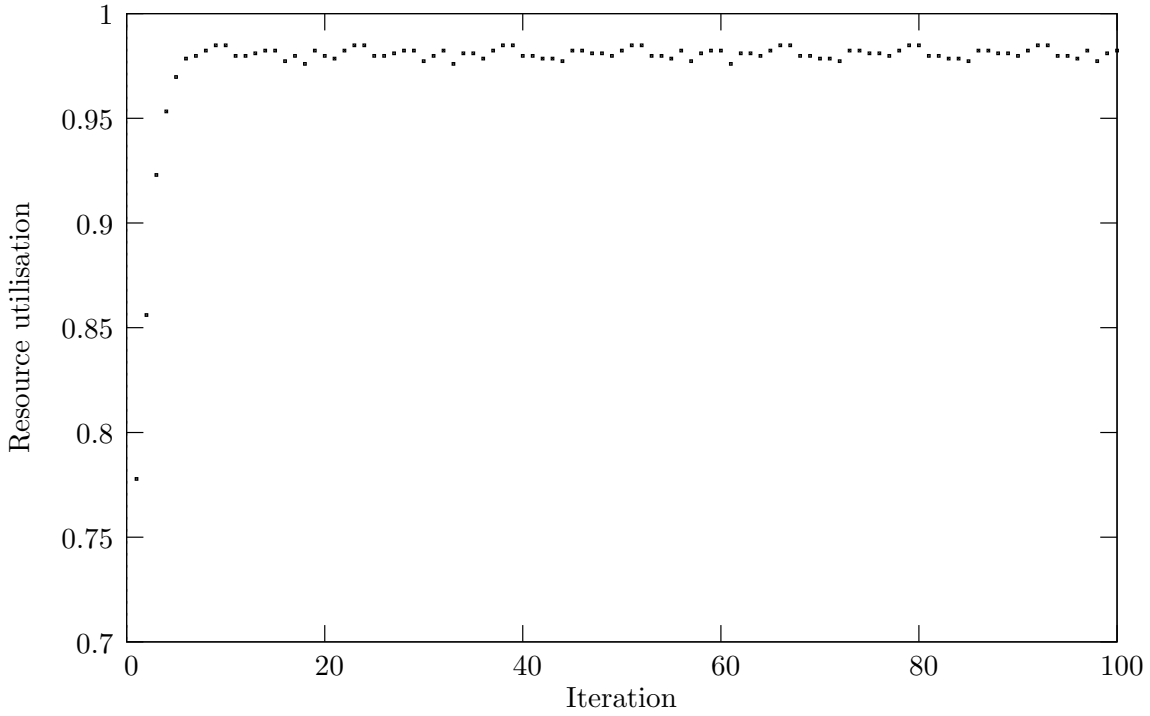


FIGURE 6.5: *Lower bounds on the resource utilisation number $\Psi_7(12,5,7,4)$ as a function of tabu search iterations.*

The tabu search is similar to the simulated annealing approach with respect to worst-case complexity. In the worst-case, the entire neighbourhood of each solution considered may need to be explored. This implies that the worst-case complexity of I iterations of the tabu search metaheuristic is

$$\mathcal{O}\left(4\binom{2^\ell}{2}^2\binom{2^\ell+t-1}{t}I\right).$$

6.4 A genetic algorithmic approach

Genetic algorithms are based on the theory of evolutionary biology which states that the strongest individuals in a population typically are the ones that thrive, and the weaker individuals eventually die and are replaced by even stronger individuals thus leading to an increasingly stronger population as time passes. In this section, a classical genetic algorithm is presented and applied in search of bounds on resource utilisation numbers for small lottery instances. This

algorithm is then enhanced by means of a hybridisation technique.

6.4.1 A classical genetic algorithm

In the context of the resource utilisation problem, an individual in the population is defined as a vector representing an overlapping playing set structure, and the fitness (measure of strength or goodness) of an individual is defined by the probability-of-win value associated with that vector. The higher the probability-of-win value associated with a given vector, the fitter the individual, and the higher the chance that the individual may survive as time passes. New individuals are introduced into the population in two ways. Firstly, a proportion of the population is selected to breed and produce new offspring by combining certain solutions in the hope of constructing new solutions in a previously unexplored region of the solution space. Individuals are chosen for breeding such that it is more likely for the more fit individuals to be chosen than for the less fit individuals to be chosen. Breeding takes place by considering two vectors from the set of solutions comprising the population and constructing two solutions (which may initially be infeasible) by combining randomly selected tickets from the one parent vector $\vec{p}_1$ with randomly selected tickets from another parent vector $\vec{p}_2$, and *visa versa*. This process is described in more detail in Algorithm 6.7. These newly created children vectors may not represent feasible solutions and if that is the case, Algorithm 6.8 may be applied to the infeasible vectors in order to find feasible ones. The resulting vectors which represent feasible solutions are then added to the population in such a way that they replace weaker individuals.

Algorithm 6.6: Classical genetic algorithm

Data: Lottery parameters m, n, t and k, the playing set cardinality ℓ, the probability of a mutation occurring p_{mutate}, *The* proportion of the population exposed to breeding p_{breed}, and the total number of iterations to be carried out, I.

Result: The highest probability-of-win value found by the classic genetic algorithm.

```
1  Generate a population of randomly generated solutions.
2  Evaluate the fitness of each individual.
3  for i < I do
4      Generate value 0 ≤ p_random ≤ 1.
5      if p_random ≤ p_mutate then
6          Select a solution from the population and mutate it.
7      else
8          Select a proportion p_breed of the solutions in the population and breed new individuals
           from them.
9      end
10     Evaluate the fitness of the new solution(s).
11     Replace the worse ranked solutions in the population with new solutions if they are more
       fit.
12 end
13 Return the probability-of-win value of the strongest individual in the population.
```

The following example illustrates the mating of two individuals denoted by the vectors $\vec{p}_1 = (1, 2, 2, 0, 2, 0, 0, 1)$ and $\vec{p}_2 = (3, 1, 1, 0, 1, 0, 0, 2)$ representing overlapping playing set structures in the lottery $\langle 8, 3, t, k\rangle$.

Example 6.2 (Mating of individuals in a lottery $\langle 8, 3, t, k\rangle$) *Assume two individuals in a population represent overlapping playing set structures in the lottery $\langle 8, 3, t, k\rangle$ denoted by the*

Algorithm 6.7: Mate

Data: Parent vector $\vec{p}_1$, Parent vector $\vec{p}_2$.
Result: Vectors $\vec{c}_1$ and $\vec{c}_2$ representing the children of $\vec{p}_1$ and $\vec{p}_2$, respectively.

R_a = randomly generated list of tickets which $\vec{c}_1$ will inherit from $\vec{p}_2$, and $\vec{c}_2$ will inherit from $\vec{p}_1$
R_b = list of tickets which $\vec{c}_1$ will inherit from $\vec{p}_1$, and $\vec{c}_2$ will inherit from $\vec{p}_2$
/* For all the tickets listed in R_a, fill $\vec{c}_1$ and $\vec{c}_2$ with elements from the tickets $\vec{p}_2$ and $\vec{p}_1$, respectively. */
for $i = 0$; $i < R_a.Length$ **do**
    for $j = 0$; $j < \vec{p}_1.Length$ **do**
        if $j \& 2^{R_a[i]} = R_a[i]$ **then**
            $\vec{c}_1[j] = \vec{p}_2[j]$
            $\vec{c}_2[j] = \vec{p}_1[j]$
        end
    end
end
/* For all the tickets listed in R_b, fill $\vec{c}_1$ and $\vec{c}_2$ with elements from the tickets $\vec{p}_1$ and $\vec{p}_2$, respectively until m elements are contained in $\vec{c}_1$ and $\vec{c}_2$. */
for $i = 0$; $i < R_b.Length$ **do**
    for $j = 0$; $j < \vec{p}_1.Length$ **do**
        if $j \& 2^{R_b[i]} = R_a[i]$ **then**
            if $Sum(\vec{c}_1) + \vec{c}_1[j] \leq m$ **then**
                $\vec{c}_1[j] = \vec{p}_1[j]$
            else
                $\vec{c}_1[j] = m - Sum(\vec{c}_1)$
            end
            if $Sum(\vec{c}_2) + \vec{c}_2[j] \leq m$ **then**
                $\vec{c}_2[j] = \vec{p}_2[j]$
            else
                $\vec{c}_2[j] = m - Sum(\vec{c}_2)$
            end
        end
    end
end
/* If there are still less than m elements in the child overlapping playing set structure, add the remaining necessary elements to compartment $x^{(\ell)}_{(000...0)}$. */
if $Sum(\vec{c}_1) < m$ **then**
    $\vec{c}_1[0] = \vec{c}_1[0] + (m - Sum(\vec{c}_1))$
end
if $Sum(\vec{c}_2) < m$ **then**
    $\vec{c}_2[0] = \vec{c}_2[0] + (m - Sum(\vec{c}_2))$
end

vectors $\vec{p}_1 = (1,2,2,0,2,0,0,1)$ *and* $\vec{p}_2 = (3,1,1,0,1,0,0,2)$. *Ticket* 2 *is randomly chosen to be the ticket inherited by child* $\vec{c}_1$ *from parent* $\vec{p}_2$, *and by child* $\vec{c}_2$ *from parent* $\vec{p}_1$. *Therefore, every entry in* $\vec{p}_2$ *which represents a compartment in ticket* 2 *is copied to the corresponding entry in* $\vec{c}_1$, *and every entry in* $\vec{p}_1$ *which represents a compartment in ticket* 2 *is copied to the corresponding entry in* $\vec{c}_2$. *At this stage,* $\vec{c}_1 = (\ ,\ ,1,0,\ ,\ ,0,2)$ *and* $\vec{c}_2 = (\ ,\ ,2,0,\ ,\ ,0,1)$. *Elements still need to be inserted into compartments*

$$x^{(3)}_{(000)},\ x^{(3)}_{(001)},\ x^{(3)}_{(100)} \text{ and } x^{(3)}_{(101)} \tag{6.1}$$

of $\vec{c}_1$ *and* $\vec{c}_2$. *The elements from the compartments in (6.1) of* $\vec{p}_1$ *are copied to matching compartments in* $\vec{c}_1$, *and likewise, the elements from the compartments in (6.1) of* $\vec{p}_2$ *are copied to matching compartments in* $\vec{c}_2$. *The resulting vectors are* $\vec{c}_1 = (1,2,1,0,2,0,0,2)$ *and* $\vec{c}_2 = (3,1,2,0,1,0,0,1)$. *The vector* $\vec{c}_1$ *is infeasible, and it is not possible to construct a feasible solution by shifting an element from one of the compartments in (6.1) to another. However, if an element is shifted from compartment* $x^{(3)}_{(000)}$ *to* $x^{(3)}_{(101)}$ *of* $\vec{c}_2$, *a feasible solution in the form of* $\vec{c}_2 = (2,1,2,0,1,1,0,1)$ *is obtained, and this solution therefore represents a child of the solutions* $\vec{p}_1$ *and* $\vec{p}_2$. ■

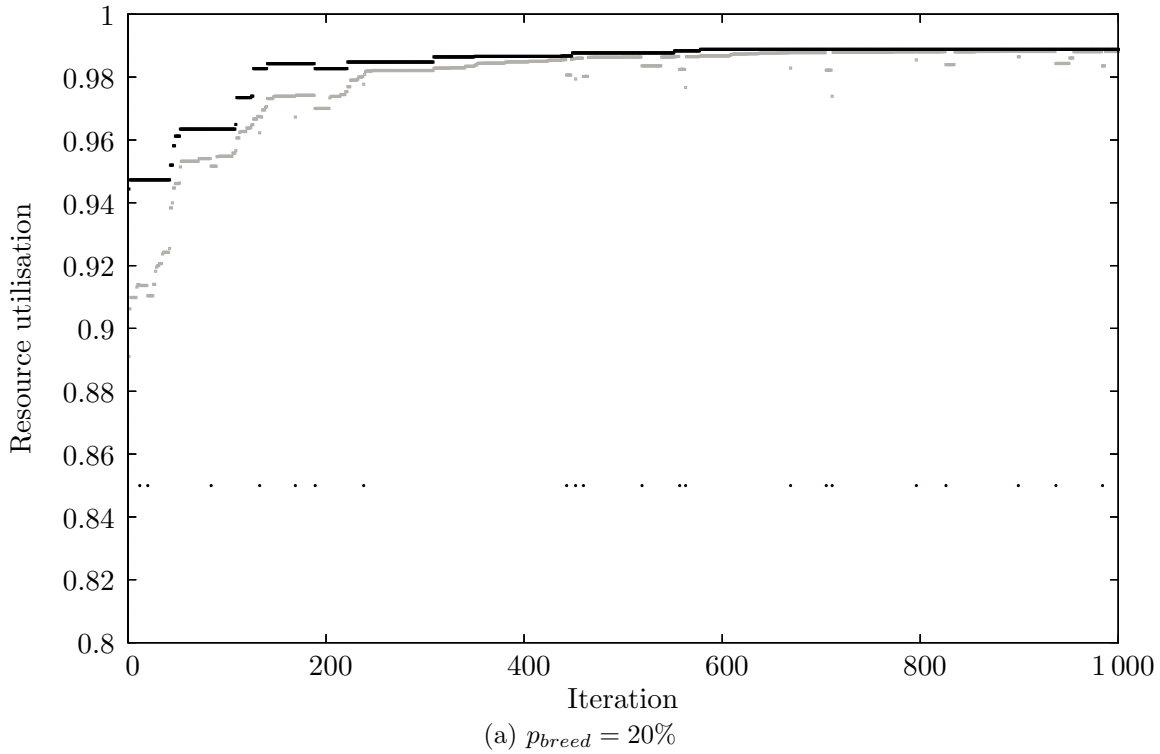
The second method of introducing new solutions into the population is by way of a mutation process, meaning that if a solution is changed in some way, the result is a different solution which may be in a previously unexplored region of the solution space. A mutation operation is applied to a solution by randomly distributing the elements of a solution in the population, and this occurs with small probability.

There are three parameters in the classical genetic algorithm which are of interest, namely p_{mutate} (the probability of a solution being mutated at any given iteration), p_{breed} (the proportion of solutions in the population which are exposed to breeding), and the population size. The value of p_{mutate} is typically set to 0.01, which implies that, on average, a mutation should occur approximately one out of every hundred iterations. However, as may be seen in Figure 6.6, that may vary depending on the randomly generated value of p_{random}. As the value of p_{breed} increases (Figure 6.6 shows the population maximum/average fitness for values of $p_{breed} \in \{20\%, 40\%, 60\%, 80\%\}$), so does the execution time of the algorithm, which implies that it is ideal for the value of p_{breed} to be relatively small. However, as may be seen in Figure 6.6, the bound on the resource utilisation number obtained appears to converge to a relatively high value quicker as the value of p_{breed} increases. The population size should be a small fraction of

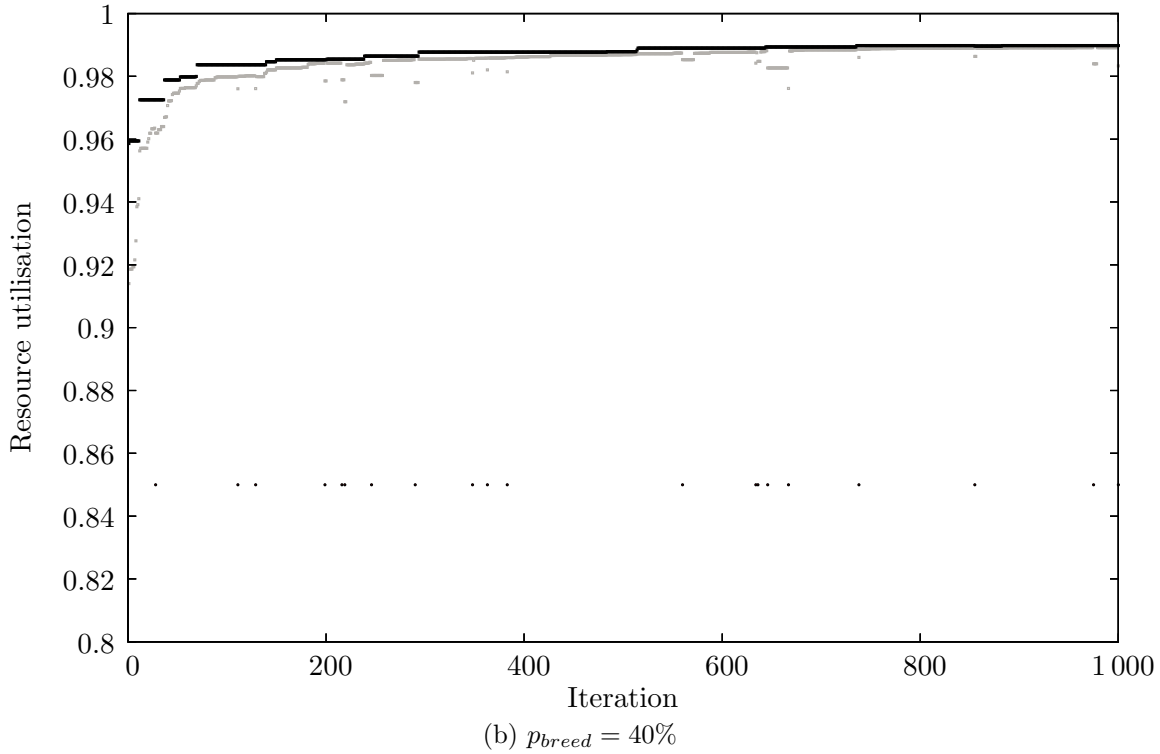
Algorithm 6.8: Finding a feasible child

Data: Infeasible solution $\vec{c}_i$, resulting from the mating of parent solutions $\vec{p}_i$ and $\vec{p}_j$.
Result: Feasible child vectors.

```
1 for each compartment a in the structure represented by c_i inherited from p_i do
2     for each compartment b ≠ a in c_i inherited from p_i do
3         Move an element from compartment a into compartment b.
4         if a feasible solution results then
5             Add the feasible solution to the population.
6         end
7         Move an element from compartment b back into compartment a.
8     end
9 end
```

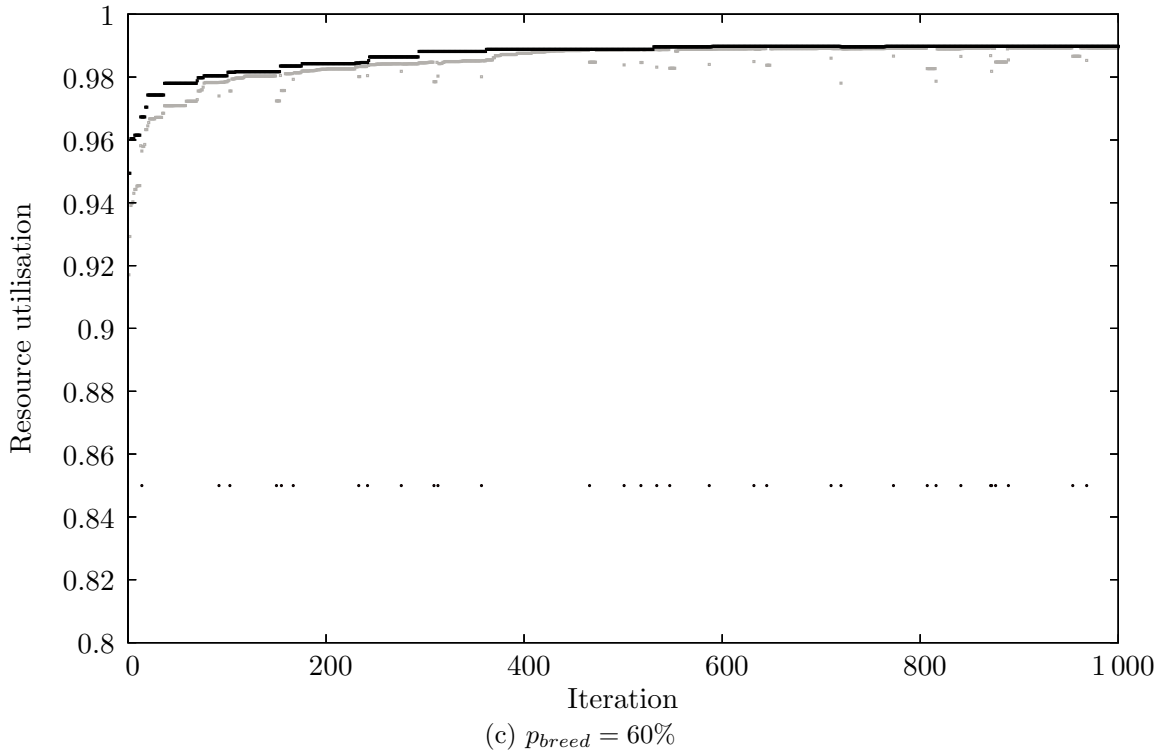


(a) $p_{breed} = 20\%$

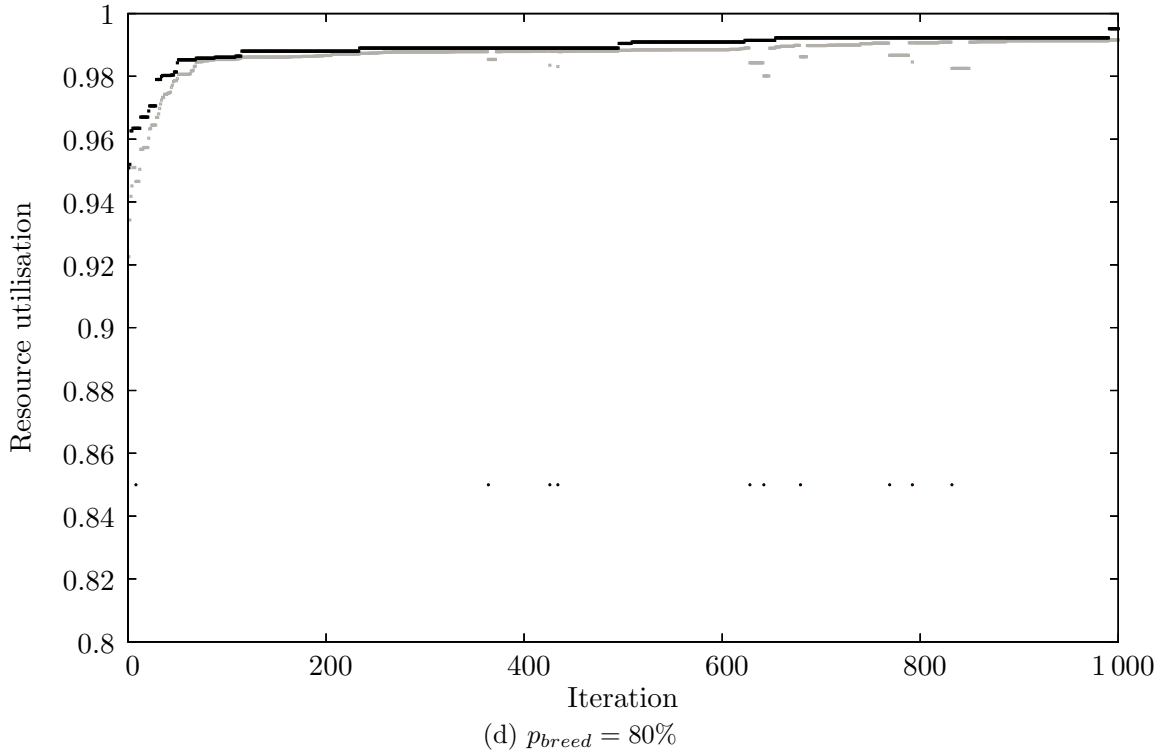


(b) $p_{breed} = 40\%$

FIGURE 6.6: *Bounds on the resource utilisation number as a function of classical genetic algorithm iterations when a playing set of cardinality 6 is considered for the lottery* $\langle 17, 7, 5, 3 \rangle$. *A population size of 20 is used, with a value of* $p_{mutate} = 1\%$. *A dot at* $\Psi = 0.85$ *indicates that a mutation took place at that iteration. The black points indicate the best bound on the resource utilisation number achieved (the fitness of the fittest individual in the population) at that iteration, and the grey points indicate the average population fitness.*

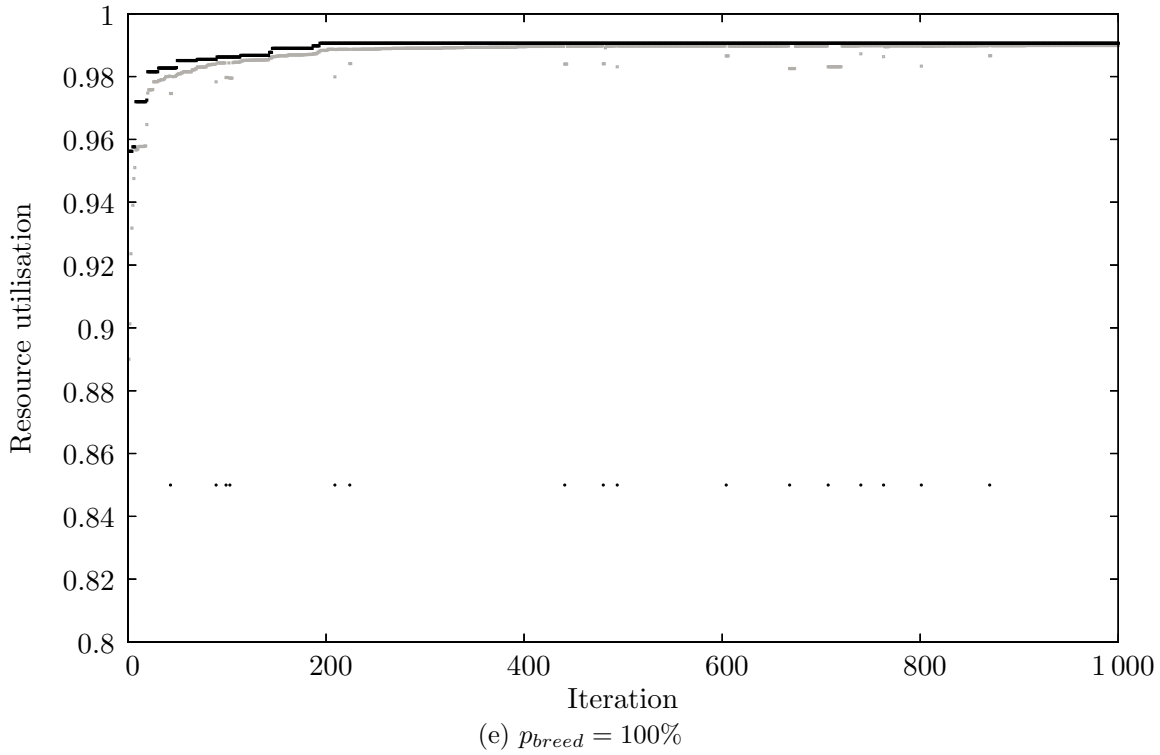


(c) $p_{breed} = 60\%$



(d) $p_{breed} = 80\%$

Figure 6.6: — *continued from the previous page* — *Bounds on the resource utilisation number as a function of classical genetic algorithm iterations when a playing set of cardinality* 6 *is considered for the lottery* $\langle 17, 7, 5, 3 \rangle$. *A population size of* 20 *is used, with a value of* $p_{mutate} = 1\%$. *A dot at* $\Psi = 0.85$ *indicates that a mutation took place at that iteration. The black points indicate the best bound on the resource utilisation number achieved (the fitness of the fittest individual in the population) at that iteration, and the grey points indicate the average population fitness.*



(e) $p_{breed} = 100\%$

FIGURE 6.6: — *continued from the previous page* — *Bounds on the resource utilisation number as a function of classical genetic algorithm iterations when a playing set of cardinality* 6 *is considered for the lottery* $\langle 17, 7, 5, 3\rangle$. *A population size of* 20 *is used, with a value of* $p_{mutate} = 1\%$. *A dot at* $\Psi = 0.85$ *indicates that a mutation took place at that iteration. The black points indicate the best bound on the resource utilisation number achieved (the fitness of the fittest individual in the population) at that iteration, and the grey points indicate the average population fitness.*

the size of the solution space of the problem. If the population size is large, an optimal solution to the problem may exist in the initial population constructed in line 1 of Algorithm 6.6. If this occurs, the evolutionary property of the classical genetic algorithm is no longer required, because an optimal solution has already been found. The population size should be relatively small because the execution time of the algorithm increases as the size of the population increases.

During each iteration of the classical genetic algorithm, solutions may either undergo a process of breeding or mutation. The breeding process involves copying values to two vectors of size ℓ. This occurs with a worst-case complexity of $\mathcal{O}(2\ell)$. Elements are then swapped between the resulting vectors, which occurs with a worst-case complexity of

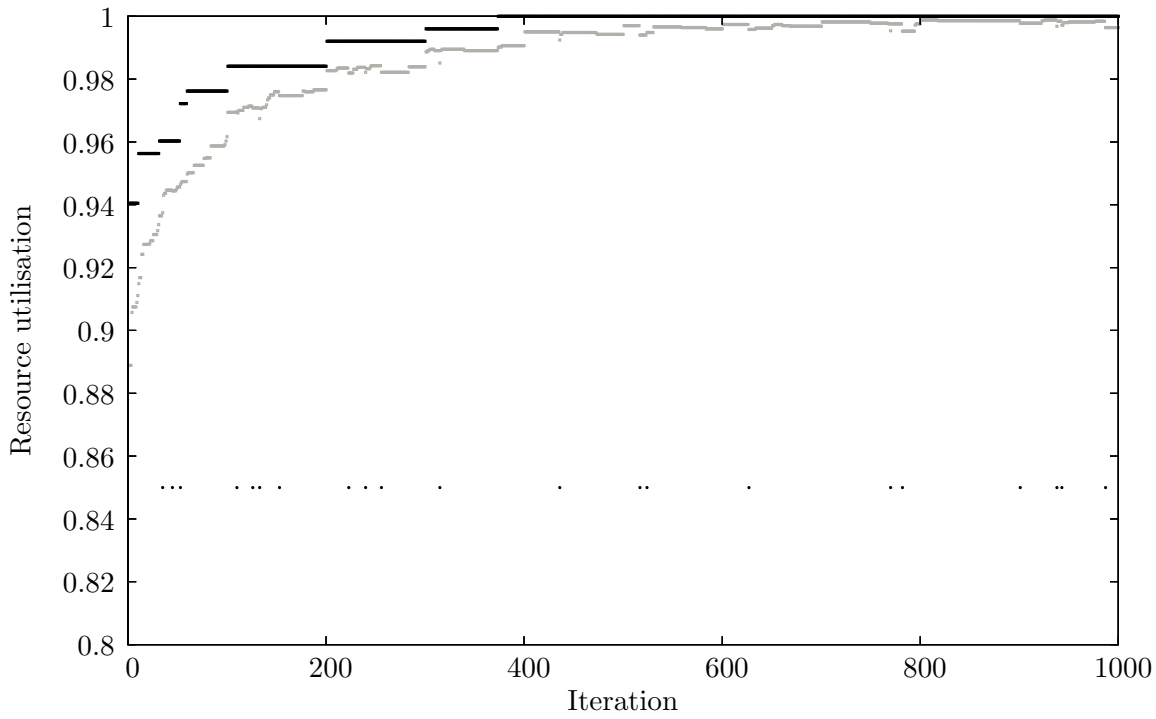
$$\mathcal{O}\left(2\binom{2^\ell}{2}\binom{2^\ell + t - 1}{t}\right),$$

implying that the breeding process has a worst-case complexity of

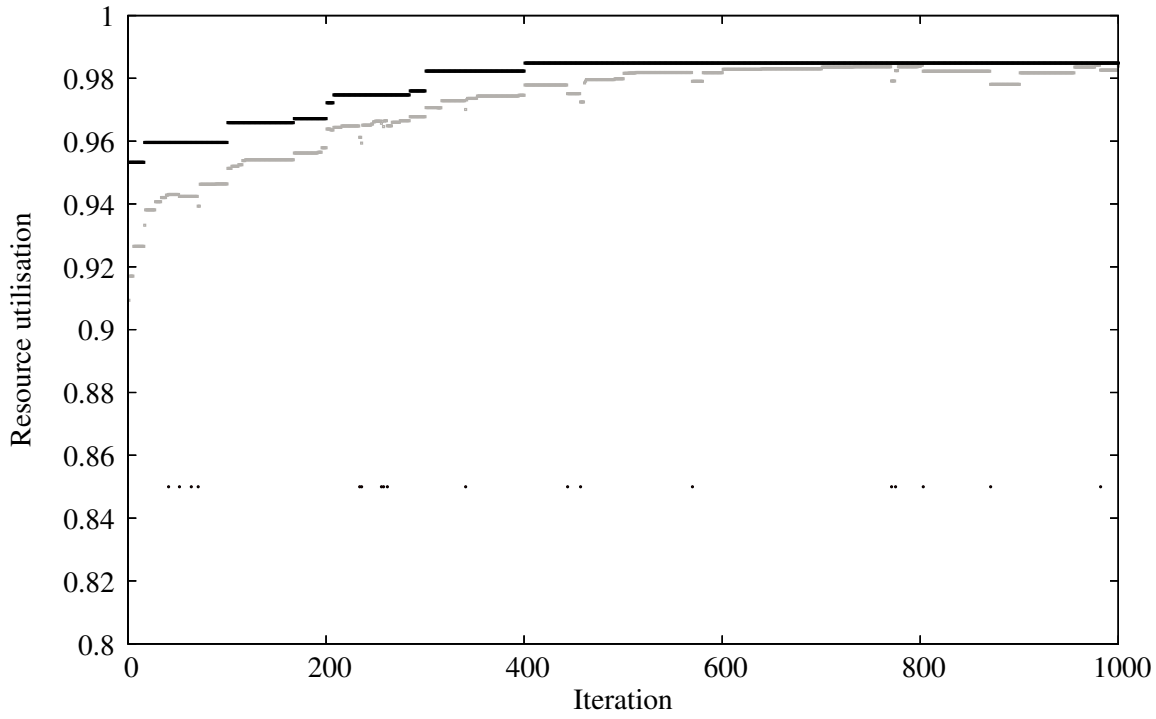
$$\mathcal{O}\left(4\ell\binom{2^\ell}{2}\binom{2^\ell + t - 1}{t}\right).$$

The mutation process involves changing the value of two elements with a worst-case complexity of $\mathcal{O}(2)$. Following this, elements are swapped between the compartments with a worst-case complexity of

$$\mathcal{O}\left(2\binom{2^\ell}{2}\binom{2^\ell + t - 1}{t}\right),$$



(a) *The lottery* $\langle 10, 4, 5, 3\rangle$



(b) *The lottery* $\langle 12, 5, 7, 4\rangle$

Figure 6.7: *Bounds on the resource utilisation number as a function of iterations of a genetic algorithm combined with a local search when a playing set of cardinality* 7 *is considered for the lotteries* $\langle 10, 4, 5, 3\rangle$ *and* $\langle 12, 5, 7, 4\rangle$*. A population size of* 20 *is used, with a value of* $p_{mutate} = 1\%$*. A dot at* $\Psi = 0.85$ *indicates that a mutation took place at that iteration. The black points indicate the best bound on the resource utilisation number achieved (the fitness of the fittest individual in the population) at that iteration, and the grey points indicate the average population fitness.*

implying that the worst-case complexity of the mutation process is

$$\mathcal{O}\left(4\binom{2^\ell}{2}\binom{2^\ell+t-1}{t}\right).$$

The maximum probability-of-win value obtained via the classical genetic algorithm for the lottery $\langle 12, 5, 7, 4\rangle$ is approximately 0.975, compared to the value of approximately 0.985 achieved via the tabu search and simulated annealing methods with a playing set cardinality of 7, and approximately 0.963, compared to the value of approximately 1 achieved via the tabu search and simulated annealing methods, for the lottery $\langle 10, 4, 5, 3\rangle$ with a playing set cardinality of 7. Hence the classical genetic algorithm method is not as effective in finding lower bounds on the resource utilisation number as are the tabu search method and the simulated annealing method. The following section contains a suggestion for improving the classical genetic algorithm.

6.4.2 A genetic algorithm combined with a local search

It may be assumed that, in a population, individuals are able to observe their neighbours and learn what their good traits are. By doing this, an individual is able to improve its own fitness.

Algorithm 6.9: Genetic algorithm combined with a local search

Data: Lottery parameters m, n, t and k, the playing set cardinality ℓ, the probability of a mutation occurring p_{mutate}, *The* proportion of the population exposed to breeding p_{breed}, The maximum number of iterations required, I.

Result: The highest probability-of-win value found by the genetic algorithm.

1 Generate a population of randomly generated solutions;
2 Evaluate the fitness of each individual;
3 **for** $i < I$ **do**
4 Generate value $0 \leq p_{random} \leq 1$;
5 **if** $p_{random} \leq p_{mutate}$ **then**
6 Select a solution from the population and mutate it;
7 **else**
8 Select a proportion p_{breed} of the solutions in the population and breed new individuals from them;
9 **end**
10 Evaluate the fitness of the new solution(s);
11 Replace the worse ranked solutions in the population with new solutions if they are more fit;
12 **if** $i = I \times 0.1$ **then**
13 **for** *each individual* P *in the population* **do**
14 Select $NewSolution$ from neighbourhood of individual P;
15 **if** $Evaluate(NewSolution) > Evaluate(P)$ **then**
16 $P \leftarrow NewSolution$;
17 Go to step 13;
18 **end**
19 **end**
20 **end**
21 **end**
22 Return the probability-of-win value of the strongest individual in the population;

This process may be applied in a genetic algorithm in order to enhance the fitness of individuals over time. That is, after a certain number of iterations, the individuals in the population may be allowed to examine their neighbours, and if a neighbour is found to have a higher fitness than the individual, the individual may assume the same properties as its fitter neighbour implying that the solution will have improved its fitness. Algorithm 6.9 illustrates the steps involved in this process.

The bounds $\Psi_7(10, 4, 5, 3) \geq 1$ and $\Psi_7(12, 5, 7, 4) \geq 0.985$ are found via the genetic algorithm combined with a local search as shown in Figure 6.7. This implies that by including a local search into Algorithm 6.6, the performance of the algorithm is improved in terms of solution quality. Algorithm 6.9 is, however, a more computationally expensive algorithm due to the addition of the local search, resulting in a worst case complexity of

$$\mathcal{O}\left(4\binom{2^\ell}{2}\binom{2^\ell + t - 1}{2}(\ell + 1)\, I\right).$$

6.5 Results obtained for small lottery instances

In this section, the local search and metaheuristic search techniques presented in this chapter are applied to the same small lottery problem instances considered in the previous chapters. The methods are compared to each other with respect to solution quality achieved and execution time expended. This is done in order to determine which method is best suited to the resource utilisation problem. Through experimentation, the tabu method (presented in §6.3.3) requires large average execution times per iteration in comparison to the simulated Annealing method (presented in §6.3.2), and the genetic algorithms (presented in §6.4). The reason for these large average execution times per iteration in the tabu search is that, with each solution encountered, the entire neighbourhood is explored in order to find the best neighbour. In the case of the simulated annealing local search technique, the entire neighbourhood is not necessarily explored for each solution encountered, because as time goes by, it becomes more likely for a suboptimal solution to be selected as the current solution. For this reason, a run of 100 iterations of the tabu search heuristic is compared to a run of 1 000 iterations of the simulated annealing heuristic, the classical genetic algorithm and the genetic algorithm combined with a local search. The results are shown in Table 6.3.

Table 6.3: *Lower bounds on the resource utilisation number, $\Psi_\ell(m, n, t, k)$, and the time (in seconds) taken to complete* 100 *iterations of the tabu search, and* 1 000 *iterations of the simulated annealing algorithm and both genetic algorithms. The entries in the column labelled Algorithm 6.4 result from the simulated annealing algorithm. The entries in the column labelled Algorithm 6.5 result from the tabu search algorithm. The entries in the column labelled Algorithm 6.6 result from the classical genetic algorithm. The entries in the column labelled Algorithm 6.9 result from the genetic algorithm combined with a local search.*

Lottery	ℓ	Algorithm 6.4		Algorithm 6.5		Algorithm 6.6		Algorithm 6.9	
		Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time
$\langle 6, 3, 3, 2\rangle$	2	1	0.078	1	0.016	1	0.063	1	0.125
$\langle 7, 3, 3, 2\rangle$	2	0.743	0.109	0.743	0.016	0.743	0.109	0.743	0.125
$\langle 7, 3, 3, 2\rangle$	3	0.914	0.906	0.914	0.078	0.914	0.297	0.914	0.547
$\langle 7, 3, 3, 2\rangle$	4	1	5.375	1	1.250	1	0.609	1	2.500
$\langle 7, 3, 4, 2\rangle$	2	1	0.125	1	0.016	1	0.078	1	0.203
$\langle 8, 3, 3, 2\rangle$	2	0.571	0.141	0.571	0.000	0.571	0.141	0.571	0.125

Table 6.3 – continued from previous page

Lottery	ℓ	Algorithm 6.4		Algorithm 6.5		Algorithm 6.6		Algorithm 6.9	
		Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time
$\langle 8,3,3,2\rangle$	3	0.786	1.250	0.786	0.109	0.786	0.219	0.786	1.313
$\langle 8,3,3,2\rangle$	4	0.893	6.063	0.893	1.531	0.893	1.531	0.893	4.656
$\langle 8,3,3,2\rangle$	5	1	23.969	1	11.266	0.964	6.766	1	21.672
$\langle 8,3,4,2\rangle$	2	0.871	0.156	0.871	0.000	0.871	0.531	0.871	0.141
$\langle 8,3,4,2\rangle$	3	1	1.359	1	0.125	1	0.563	1	0.922
$\langle 8,3,5,2\rangle$	2	1	0.141	1	0.016	1	0.125	1	0.266
$\langle 8,4,3,2\rangle$	2	1	0.078	1	0.000	1	0.109	1	0.094
$\langle 8,4,4,2\rangle$	2	1	0.281	1	0.016	1	0.141	1	0.188
$\langle 8,4,4,3\rangle$	2	0.486	0.266	0.486	0.000	0.486	0.156	0.486	0.125
$\langle 8,4,4,3\rangle$	3	0.671	4.281	0.671	0.141	0.671	0.500	0.671	0.547
$\langle 8,4,4,3\rangle$	4	0.857	11.656	0.857	2.000	0.857	1.266	0.857	3.563
$\langle 8,4,4,3\rangle$	5	0.914	56.141	0.914	14.078	0.914	2.203	0.914	17.141
$\langle 8,4,4,3\rangle$	6	1	270.859	1	94.063	0.971	3.406	1	109.531
$\langle 9,3,3,2\rangle$	2	0.452	0.188	0.452	0.000	0.452	0.078	0.452	0.234
$\langle 9,3,3,2\rangle$	3	0.679	0.891	0.679	0.125	0.679	1.438	0.679	1.500
$\langle 9,3,3,2\rangle$	4	0.774	16.109	0.774	1.875	0.774	4.359	0.774	11.484
$\langle 9,3,3,2\rangle$	5	0.893	61.219	0.893	15.172	0.893	22.344	0.893	42.563
$\langle 9,3,3,2\rangle$	6	0.952	334.406	0.952	107.047	0.952	38.422	0.952	239.609
$\langle 9,3,3,2\rangle$	7	1	1 861.875	1	849.219	1	126.609	1	1 182.813
$\langle 9,3,4,2\rangle$	2	0.738	0.156	0.738	0.000	0.738	0.172	0.738	0.281
$\langle 9,3,4,2\rangle$	3	1	0.891	1	0.141	1	0.828	1	1.969
$\langle 9,3,5,2\rangle$	2	0.929	0.172	0.929	0.000	0.929	0.109	0.929	0.406
$\langle 9,3,5,2\rangle$	3	1	2.609	1	0.063	1	1.234	1	1.281
$\langle 9,3,6,2\rangle$	2	1	0.141	1	0.000	1	0.141	1	0.125
$\langle 9,4,3,2\rangle$	2	0.81	0.156	0.81	0.016	0.81	0.188	0.81	0.125
$\langle 9,4,3,2\rangle$	3	0.952	1.156	0.952	0.172	0.952	0.453	0.952	0.531
$\langle 9,4,3,2\rangle$	4	1	8.016	1	2.203	1	0.641	1	3.594
$\langle 9,4,4,2\rangle$	2	1	0.172	1	0.016	1	0.094	1	0.109
$\langle 9,4,4,3\rangle$	2	0.333	0.344	0.333	0.016	0.333	0.234	0.333	0.094
$\langle 9,4,4,3\rangle$	3	0.5	6.406	0.5	0.156	0.5	0.234	0.5	1.031
$\langle 9,4,4,3\rangle$	4	0.635	46.953	0.635	2.953	0.635	1.641	0.635	7.531
$\langle 9,4,4,3\rangle$	5	0.738	181.625	0.738	19.656	0.738	3.688	0.738	32.172
$\langle 9,4,4,3\rangle$	6	0.825	578.297	0.825	133.406	0.817	7.141	0.825	179.875
$\langle 9,4,4,3\rangle$	7	0.913	2 365.469	0.913	935.734	0.849	14.516	0.913	920.031
$\langle 9,4,4,3\rangle$	8	0.96	11 601.953	0.96	7 088.859	0.905	62.109	0.96	4 109.625
$\langle 9,4,4,3\rangle$	9	0.992	79 370.781	1	57 678.875	0.937	388.813	0.992	42 970.266
$\langle 9,4,5,2\rangle$	2	1	0.375	1	0.031	1	0.094	1	0.109
$\langle 9,4,5,3\rangle$	2	0.714	0.188	0.714	0.016	0.714	0.141	0.714	0.250
$\langle 9,4,5,3\rangle$	3	0.857	1.875	0.857	0.219	0.857	0.484	0.857	1.234
$\langle 9,4,5,3\rangle$	4	0.976	9.422	0.976	2.297	0.976	2.234	0.976	4.547
$\langle 9,4,5,3\rangle$	5	1	53.563	1	16.813	1	3.281	1	23.484
$\langle 9,4,6,3\rangle$	2	1	0.156	1	0.000	1	0.125	1	0.375
$\langle 10,3,3,2\rangle$	2	0.367	0.172	0.367	0.016	0.367	0.563	0.367	0.438
$\langle 10,3,3,2\rangle$	3	0.55	1.406	0.55	0.109	0.55	3.750	0.55	3.063
$\langle 10,3,3,2\rangle$	4	0.667	18.078	0.667	0.767	0.667	10.547	0.667	20.578
$\langle 10,3,3,2\rangle$	5	0.767	114.156	0.767	18.781	0.767	79.078	0.767	106.750

Table 6.3 – continued from previous page

Lottery	ℓ	Algorithm 6.4		Algorithm 6.5		Algorithm 6.6		Algorithm 6.9	
		Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time
$\langle 10,3,3,2\rangle$	6	0.85	543.094	0.85	126.594	0.85	768.281	0.85	803.234
$\langle 10,3,3,2\rangle$	7	0.917	2 808.391	0.917	1 033.578	0.917	1 048.984	0.917	2 914.484
$\langle 10,3,3,2\rangle$	8	1	12 988.734	1	8 444.875	0.967	7 142.938	0.967	24 712.563
$\langle 10,3,4,2\rangle$	2	0.624	0.203	0.624	0.016	0.624	0.094	0.624	0.359
$\langle 10,3,4,2\rangle$	3	0.871	1.359	0.871	0.094	0.871	2.672	0.871	2.828
$\langle 10,3,4,2\rangle$	4	0.957	10.078	0.957	1.953	0.957	28.266	0.957	20.859
$\langle 10,3,4,2\rangle$	5	1	60.984	1	15.578	1	54.234	1	109.063
$\langle 10,3,5,2\rangle$	2	0.833	0.188	0.833	0.000	0.833	0.281	0.833	0.328
$\langle 10,3,5,2\rangle$	3	1	1.547	1	0.109	1	1.750	1	4.172
$\langle 10,3,6,2\rangle$	2	0.957	0.172	0.957	0.000	0.957	0.453	0.957	0.047
$\langle 10,3,6,2\rangle$	3	1	3.328	1	0.094	1	3.328	1	7.031
$\langle 10,3,7,2\rangle$	2	1	0.172	1	0.000	1	0.109	1	0.438
$\langle 10,4,3,2\rangle$	2	0.667	0.188	0.667	0.016	0.667	0.250	0.667	0.188
$\langle 10,4,3,2\rangle$	3	0.867	1.922	0.867	0.188	0.867	0.922	0.867	1.469
$\langle 10,4,3,2\rangle$	4	1	6.766	1	2.641	1	2.391	1	5.250
$\langle 10,4,4,2\rangle$	2	0.924	0.219	0.924	0.016	0.924	0.234	0.924	0.109
$\langle 10,4,4,2\rangle$	3	1	2.656	1	0.234	1	1.359	1	1.031
$\langle 10,4,4,3\rangle$	2	0.238	0.516	0.238	0.016	0.238	0.313	0.238	0.234
$\langle 10,4,4,3\rangle$	3	0.357	3.094	0.357	0.141	0.357	1.016	0.357	1.484
$\langle 10,4,4,3\rangle$	4	0.476	69.484	0.476	4.078	0.476	3.641	0.476	10.984
$\langle 10,4,4,3\rangle$	5	0.595	457.688	0.595	32.453	0.595	7.734	0.595	69.078
$\langle 10,4,4,3\rangle$	6	0.662	1 182.031	0.662	192.188	0.662	25.891	0.662	255.266
$\langle 10,4,4,3\rangle$	7	0.743	3 995.625	0.743	1 320.203	0.729	30.813	0.743	1 324.766
$\langle 10,4,4,3\rangle$	8	0.81	15 223.688	0.81	9 149.922	0.762	167.734	0.81	8 833.672
$\langle 10,4,4,3\rangle$	9	0.862	49 910.156	0.862	73 266.906	0.814	1 692.641	0.862	44 787.828
$\langle 10,4,5,2\rangle$	2	1	0.219	1	0.047	0.964	0.172	1	0.344
$\langle 10,4,5,3\rangle$	2	0.524	0.219	0.524	0.000	0.488	0.172	0.524	0.219
$\langle 10,4,5,3\rangle$	3	0.714	4.641	0.714	0.297	0.714	0.625	0.714	2.375
$\langle 10,4,5,3\rangle$	4	0.833	38.031	0.833	3.203	0.833	3.094	0.833	8.828
$\langle 10,4,5,3\rangle$	5	0.952	319.766	0.952	24.156	0.952	5.625	0.952	45.391
$\langle 10,4,5,3\rangle$	6	0.976	909.313	0.976	159.656	0.964	13.641	0.976	254.188
$\langle 10,4,5,3\rangle$	7	1	2 875.984	1	1 208.156	0.992	34.297	1	1 043.734
$\langle 10,4,6,2\rangle$	2	1	0.500	1	0.016	1	0.203	1	0.188
$\langle 10,4,6,3\rangle$	2	0.829	0.203	0.829	0.000	0.743	0.203	0.829	0.281
$\langle 10,4,6,3\rangle$	3	0.971	2.219	0.971	0.281	0.971	1.047	0.971	0.719
$\langle 10,4,6,3\rangle$	4	1	20.141	1	3.063	1	3.344	1	6.500
$\langle 10,4,7,3\rangle$	2	1	0.172	1	0.016	1	0.172	1	0.188
$\langle 10,5,3,2\rangle$	2	1	0.125	1	0.000	1	0.078	1	0.094
$\langle 10,5,4,2\rangle$	2	1	0.344	1	0.000	1	0.188	1	0.172
$\langle 10,5,4,3\rangle$	2	0.524	0.391	0.524	0.016	0.524	0.078	0.524	0.141
$\langle 10,5,4,3\rangle$	3	0.7	7.938	0.7	0.328	0.7	0.969	0.7	1.438
$\langle 10,5,4,3\rangle$	4	0.876	22.844	0.876	3.438	0.876	1.984	0.876	6.344
$\langle 10,5,4,3\rangle$	5	0.924	146.594	0.924	27.625	0.876	2.953	0.924	39.500
$\langle 10,5,4,3\rangle$	6	0.986	380.781	0.986	166.359	0.938	6.734	0.986	147.656
$\langle 10,5,4,3\rangle$	7	1	3 077.766	1	1 157.031	0.971	45.875	1	1 006.766

Table 6.3 – continued from previous page

Lottery	ℓ	Algorithm 6.4		Algorithm 6.5		Algorithm 6.6		Algorithm 6.9	
		Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time	Ψ_ℓ	Time
$\langle 10, 5, 5, 2\rangle$	2	1	0.578	1	0.000	1	0.250	1	0.125
$\langle 10, 5, 5, 3\rangle$	2	1	0.141	1	0.000	0.857	0.047	1	0.078
$\langle 10, 5, 5, 4\rangle$	2	0.206	0.641	0.206	0.016	0.206	0.125	0.206	0.266
$\langle 10, 5, 5, 4\rangle$	3	0.31	15.750	0.31	0.234	0.31	0.516	0.31	1.953
$\langle 10, 5, 5, 4\rangle$	4	0.413	86.531	0.413	5.141	0.413	1.641	0.413	11.391
$\langle 10, 5, 5, 4\rangle$	5	0.516	593.563	0.516	35.578	0.516	3.578	0.516	57.375
$\langle 10, 5, 5, 4\rangle$	6	0.619	1 641.922	0.619	206.719	0.587	9.250	0.619	261.000
$\langle 10, 5, 5, 4\rangle$	7	0.675	5 019.859	0.675	1 290.531	0.659	42.406	0.675	1 549.859
$\langle 10, 5, 5, 4\rangle$	8	0.746	18 644.688	0.762	9 341.703	0.683	641.734	0.762	8 111.516
$\langle 10, 5, 5, 4\rangle$	9	0.806	76 113.703	0.806	73 801.094	0.73	11 701.578	0.802	36 976.766

6.6 Tabu search applied to larger lotteries

Burger, *et al.* [5] provided a table of larger lottery instances for which the bounds on $L_1(m, n, t, k)$ could not be improved via the exhaustive enumeration lottery tree solution method. Some of those lottery instances are considered in this section and the tabu search solution method is applied to them.

It known that $6 \leq L_1(17, 7, 5, 3) \leq 7$ [5]. When applying the tabu search solution method to the overlapping playing set structure of the lottery $\langle 17, 7, 5, 3\rangle$ for 6 tickets, the best resource utilisation value found is approximately 0.995. Hence the tabu search approach also cannot be used to decide whether the value of $L_1(17, 7, 5, 3)$ is 6 or 7. Similarly, for the lottery $\langle 18, 9, 6, 4\rangle$, the best resource utilisation value found for 6 tickets via the tabu search approach is approximately 0.997 so again the tabu search approach also cannot be used to decide whether the value of $L_1(18, 9, 6, 4)$ is 6 or 7.

However, the tabu search solution method was also applied to the lottery $\langle 20, 10, 4, 3\rangle$, for which it was previously known that $6 \leq L_1(20, 10, 4, 3) \leq 8$. For $\ell = 6$, the tabu search yields a resource utilisation value of 1. This implies that $L_1(20, 10, 4, 3) = 6$. An optimal overlapping playing set structure is given by the vector $\vec{\mathbf{X}}^{(6)} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0)$.

6.7 Chapter overview

In this chapter, various approximation algorithms were applied to the problem of distributing the elements of an overlapping playing set structure amongst its compartments in an attempt to find an overlapping playing set structure that has a high probability-of-win value associated with it. Firstly, a method of randomly exploring the solution space was investigated. The results from this solution method are poor, as expected, because a random search method, although sometimes useful and quick to deliver results, incorporates no search intelligence. Following the investigation of the random search method, two local search metaheuristics were presented, namely a simulated annealing algorithm and a tabu search. As expected, these two methods perform better than the random search method. Finally, two genetic algorithms were presented in the hope that they may offer improved performance compared to the local search methods.

From Table 6.3, it may be seen that the classical genetic algorithm (presented in §6.4.1) performs worse in terms of solution quality, compared to the other methods, but it is the least computationally expensive method. If a local search is incorporated into the genetic algorithm (as described in §6.4.2), the solution quality of the genetic algorithm improves, but the method is computationally more expensive. The simulated annealing algorithm (presented in §6.3.2) performs the worst out of the two local search methods presented in terms of solution quality, but it is computationally less expensive than the tabu search. The tabu search, presented in §6.3.3, performs the best out of all the methods presented in this chapter in terms of solution quality, but is by far the most expensive method computationally. It may therefore be concluded that if a solution is sought within a short timespan, the genetic algorithm may be applied, while if a good quality solution is sought, the tabu search method should be used.

Furthermore, the exhaustive enumeration lottery tree method, presented in Chapter 5, is a good method for finding (optimal) solutions to the incomplete lottery problem and the resource utilisation problem, but it is also a time consuming method, especially for problem instances in which the playing set cardinality, ℓ, is greater than 6. For example, consider the lottery $\langle 10, 4, 5, 3\rangle$ in Table 5.5. To find the value of $\Psi_6(10, 4, 5, 3) \approx 0.976$, the exhaustive enumeration lottery tree method takes approximately 3 920.02 seconds, but from Table 6.3, the tabu search method took approximately 159.656 seconds to find that same value. This indicates that an approximate approach towards finding solutions to the incomplete lottery problem and the resource utilisation problem may be an attractive alternative. The reason for this is that there are many different overlapping playing set structures which possess the same probability-of-win value, implying that not all regions of the solution space have to be explored, as is done in the exhaustive enumeration lottery tree method.

Finally, it is shown in §6.6 that the solution methods presented in this chapter are indeed useful, in the sense that they may sometimes be used to compute previously unknown lottery numbers exactly, such as $L_1(20, 10, 4, 3) = 6$.

CHAPTER 7

On the expected number of lottery winners

Contents

On the 15^{th} of March 2003, there were 33 winners of the jackpot prize in the South African National Lottery, and on the 7^{th} of February 2009, there were 18 winners of the jackpot prize [32]. Due to the extreme perceived unlikelihood of these events, independent investigations into alleged irregularities were called for by members of the public [14, 42]. It is the aim of this chapter to clarify how unlikely the outcomes of the 15^{th} of March 2003 and 7^{th} of February 2009 lottery draws actually are. The following definition of a so-called *Type A ticket* is adopted in the remainder of this chapter.

Type A ticket. A lottery ticket containing numbers which are randomly selected according to a uniform distribution.

In this chapter, the probability is computed that $\lambda \in \mathbb{N}$ participants win a lottery draw in which the winning government ticket is of type A and all the participants' tickets are also of type A, in fulfilment of Thesis Objective V(a). Those same probability values are then recomputed by means of simulation for validation purposes, in fulfilment of Thesis Objective V(b). Following that, the probability is computed that λ participants win a lottery draw in which the winning government ticket is still of type A, while the numbers chosen by some proportion $\theta \in (0, 1]$ of participants during the construction of their playing sets are *not* of type A, but conform to some pre-defined probability distribution, in fulfilment of Thesis Objective V(c).

This chapter proceeds under two assumptions. The first assumption is that each lottery participant selects a playing set of *distinct* tickets. This assumption implies that if the winning

government ticket has k or more numbers in common with x participant tickets, then exactly x participants share the winnings, because no participant selects the same ticket more than once. The second assumption is that the winning government ticket is a type A ticket. This assumption is substantiated by Haigh [18], who concluded that statistical tests show no significant evidence that the numbers drawn for a government ticket in the United Kingdom national lottery are not uniformly random[1].

In §7.1 an analysis is conducted of the expected number of winners (of the jackpot prize) in the lottery $\langle 49, 6, 6, 6\rangle$ under the naive assumption that the playing sets constructed by participants contain only type A tickets. Further analysis of the number of winners of the jackpot prize in the lottery $\langle 49, 6, 6, 6\rangle$ is conducted in §7.2 under the assumption that numbers chosen by some proportion θ of participants during their playing set constructions are *not* chosen uniformly, but are rather chosen according to some pre-defined statistical distribution. An analysis of the number of winners in the lottery $\langle 49, 6, 6, 6\rangle$ naturally inspires §7.3, in which an analysis is conducted of the number of k-prize winners in draws of smaller lottery instances (*i.e.* not necessarily the jackpot).

7.1 A naive analysis of the lottery $\langle 49, 6, 6, 6\rangle$

In this section, the probability that λ participants win (the jackpot prize) in the lottery $\langle 49, 6, 6, 6\rangle$ is computed, based on the assumption that all participant tickets are of type A.

7.1.1 An analytical approach

The probability, $P(N, \lambda)$, of λ successes occurring in a sequence of N identical, independent binary trials, each with a probability p of success, is

$$P(N, \lambda) = \binom{N}{\lambda} p^{\lambda}(1-p)^{N-\lambda}, \tag{7.1}$$

which is the probability mass function of the binomial distribution [37]. If N participant tickets are selected independently in the lottery $\langle 49, 6, 6, 6\rangle$, then $P(N, \lambda)$ in (7.1) is the probability that the jackpot is shared by λ participants, where $p = 1/\binom{49}{6}$. Values of $P(N, \lambda)$ are presented for $N = 7\,500\,000$ and $N = 20\,000\,000$ in Table 7.1. These values of N are motivated by the facts that in draws in which relatively few participants purchase tickets, approximately $N = 7\,500\,000$ tickets are purchased, and in draws in which a relatively large number of participants purchase tickets approximately $N = 20\,000\,000$ tickets are purchased [31].

As the number of participants in a lottery draw increases, it becomes more likely for a larger number of participants to share a prize. This explains why $P(7\,500\,000, \lambda) < P(20\,000\,000, \lambda)$ for $\lambda \geq 1$; if $20\,000\,000$ tickets are purchased in total, it is more likely that one or more participants share the jackpot prize than in the corresponding case where only $7\,500\,000$ tickets are purchased.

[1] The UK lottery is of the form $\langle 49, 6, 6, k\rangle$, and the process of drawing numbers for the winning government ticket is the same as the one used in the South African National Lottery, *Lotto*.

TABLE 7.1: *The probability, computed from (7.1), of λ participants winning in the lottery $\langle 49, 6, 6, 6\rangle$ in which $N = 7\,500\,000$ and $N = 20\,000\,000$ type A tickets are selected by participants.*

λ	$P(7.5M, \lambda)$	$P(20M, \lambda)$
0	5.849×10^{-1}	2.393×10^{-1}
1	3.137×10^{-1}	3.421×10^{-1}
2	8.412×10^{-2}	2.447×10^{-1}
3	1.504×10^{-2}	1.167×10^{-1}
4	2.017×10^{-3}	4.171×10^{-2}
5	2.163×10^{-4}	1.193×10^{-2}
10	3.174×10^{-10}	2.361×10^{-6}
15	3.909×10^{-17}	3.921×10^{-11}
18	1.232×10^{-21}	2.343×10^{-14}

7.1.2 A simulation approach

The values in Table 7.1 may be verified by means of a Monte Carlo simulation approach. In such a simulation approach the value $E(N, \lambda)$, which represents the expected number of government tickets covered[2] λ times by the N participant tickets, is computed. The algorithm used for the simulation is presented in Algorithm 7.1.

Algorithm 7.1: Monte Carlo simulation of lottery draws for the lottery $\langle m, n, t, k\rangle$

Data: The total number of tickets purchased N, the lottery parameters m, n, t and k, and the number of required iterations I.
Result: Approximations of the values $E(N, \lambda)$ and $P(N, \lambda)$

```
for i = 1; i ≤ I do
    Create a list L, of N participant tickets of size n
    for Each government ticket G_i do
        S_i ← the number tickets in L which have at least k numbers in common with G_i
    end
    for Each λ ≥ 0 do
        W_λ = W_λ + the number of occurrences of λ in the list S = {S_1, S_2, ..., S_C(m,t)}
    end
end
Output E(N, λ) ≈ W_λ / I
Output P(N, λ) ≈ W_λ / [I C(m,t)]
```

It may be seen from the results in Table 7.1 that the above simulation should run approximately $1/1.232 \times 10^{-21} \approx 8.118 \times 10^{20}$ iterations on expectation for $N = 7\,500\,000$ (and $1/2.343 \times 10^{-14} \approx 4.268 \times 10^{13}$ iterations for $N = 20\,000\,000$) in order for an iteration to occur in which $\lambda = 18$ winners are recorded. The values obtained from the simulation for $N = 7\,500\,000$ and $N = 20\,000\,000$ appear in Table 7.2 and they correspond satisfactorily to the values in Table 7.1. It may also be seen in Table 7.2 that, in order to obtain results which are correct to three decimal places for $0 \leq \lambda \leq 10$, a thousand simulation iterations is sufficient. The reason for this is that, after a thousand simulation iterations, the *average* value of $P(N, \lambda)$ resembles the value of $P(N, \lambda)$ obtained in §7.1.1.

[2]Recall that a government ticket is covered by a participant ticket if it has k or more numbers in common with it.

TABLE 7.2: *The value, $E(N,\lambda)$, which represents the expected number of government tickets covered λ times by $N = 7\,500\,000$ or $N = 20\,000\,000$ participant tickets, respectively, and the probability, $P(N,\lambda)$, of λ participants winning a draw of the lottery $\langle 49,6,6,6\rangle$ in which $N = 7\,500\,000$ or $N = 20\,000\,000$ tickets are purchased by participants, computed by means of Algorithm 7.1.*

λ	$E(7.5M,\lambda)$	$P(7.5M,\lambda)$	λ	$E(20M,\lambda)$	$P(20M,\lambda)$
0	8 178 986.561	5.849×10^{-1}	0	3 345 742.237	2.393×10^{-1}
1	4 386 647.152	3.137×10^{-1}	1	4 785 089.710	3.422×10^{-1}
2	1 176 339.187	8.412×10^{-2}	2	3 421 808.674	2.447×10^{-1}
3	210 329.311	1.504×10^{-2}	3	1 631 406.761	1.167×10^{-1}
4	28 198.500	2.017×10^{-3}	4	583 289.045	4.171×10^{-2}
5	3 022.829	2.162×10^{-4}	5	166 846.584	1.193×10^{-2}
10	0.007	5.006×10^{-10}	10	32.934	2.355×10^{-6}
15	0	0	15	0.001	7.151×10^{-11}
18	0	0	18	0	0

Approximately $E(N,\lambda)$ out of $\binom{49}{6}$ government tickets are covered λ times by the N purchased tickets, which implies that the probability, $P(N,\lambda)$, that a randomly selected government ticket is covered λ times is $E(N,\lambda)/\binom{m}{t}$. From Tables 7.1 and 7.2, it may be seen that the probability of 18 participants sharing the prize in the lottery $\langle 49,6,6,6\rangle$ is approximately 2.34×10^{-14} which is one million times smaller than the probability that a single participant wins the jackpot with one ticket. Therefore, by analysing the number of jackpot winners in the lottery $\langle 49,6,6,6\rangle$ in such a naive way as in this section, it may seem quite valid that members of the public view the number of jackpot winners in the South African National Lottery draw on the 15^{th} of March 2003 and the 7^{th} of February 2009 suspiciously.

7.2 A more realistic analysis of winners in the lottery $\langle 49,6,6,6\rangle$

It has often been noted that certain number combinations are more popular than others among lottery players, in which case the assumption that participants select only type A tickets no longer holds. In this section, the situation is considered in which the numbers chosen by participants during construction of their playing set tickets are *not* chosen uniformly, but where they are rather chosen according to some pre-defined statistical distribution pattern. The aim of this section is to show that, under this assumption, the value of $P(N,\lambda)$ is significantly larger than the values reported in §7.1 under the previous assumption that participants only select type A tickets, thereby casting some doubt on the legitimacy of calls for an investigation into irregularities in the South African National Lottery as a result of the events of the 15^{th} of March 2003 and the 7^{th} of February 2009.

7.2.1 Popular number choices in the lottery $\langle 49,6,6,k\rangle$

Popular techniques employed by lottery participants to choose numbers for inclusion in their tickets include a variety of methods. It has been speculated that many people base their numbers on special dates, such as birthdays or well-known holidays [45]. For this reason, the numbers $1,\ldots,31$ are referred to as *calendar numbers*. Other participants may attempt to spread the 6 numbers that they include in a lottery ticket as evenly as possible over the physical ticket that they fill in during a purchase, because they (incorrectly) assume that it better represents

a random number selection.

Another method of constructing playing sets is to select tickets containing numbers which form an arithmetic sequence. The number choices in these tickets either arise from a purposeful selection of an arithmetic sequence by the participant, or by a selection of numbers which form some sort of geometric pattern on the physical lottery ticket. To understand this, consider Figure 7.1, which shows the physical layout of a ticket on which a participant in the South African National Lottery, *Lotto*, indicates his/her number choices. Some participants may decide to play the numbers in a specific row, column or diagonal. If a participant were to select ticket numbers in this way, then the participant's playing ticket numbers form an arithmetic sequence. For example, if the participant chooses the numbers in the middle column, from the top, the ticket $\{3, 8, 13, 18, 23, 28\}$ results. This is an arithmetic sequence in which the difference between any two successive numbers is exactly 5.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40
41	42	43	44	45
46	47	48	49	

FIGURE 7.1: *Physical layout of a lottery ticket for the South African National Lottery, Lotto, which is of the form* $\langle 49, 6, 6, k\rangle$*, where* $3 \leq k \leq 6$.

Based on the above observations about popular number choices by lottery participants, the following definitions are adopted in the remainder of this chapter:

Type $B(\kappa)$ **ticket.** A lottery ticket in which the difference between any two numbers is *no less than* κ.

Type $C(\kappa)$ **ticket.** A lottery ticket containing κ *calendar numbers* and in which the remaining numbers are non-calendar numbers.

Participants often select type $B(\kappa)$ tickets in a bid to spread the numbers in their tickets over the spectrum $\{1, \ldots, 49\}$ as much as possible. Tickets of this form arise when a participant plays the numbers in a diagonal of the ticket, together with another number, such as $\{15, 19, 23, 27, 31, 49\}$, which is a type $B(4)$ ticket.

In order to count the number of type $B(\kappa)$ tickets in the lottery $\langle 49, 6, 6, k\rangle$, consider the ordered set containing the numbers 1 to 49. This set may be partitioned into 13 ordered subsets, namely $\mathcal{C}_1$, $\mathcal{T}_1$, $\mathcal{C}_2$, $\mathcal{T}_2$, $\mathcal{C}_3$, $\mathcal{T}_3$, $\mathcal{C}_4$, $\mathcal{T}_4$, $\mathcal{C}_5$, $\mathcal{T}_5$, $\mathcal{C}_6$, $\mathcal{T}_6$, $\mathcal{C}_7$, where each subset $\mathcal{T}_i$ has cardinality 1 and represents the i^{th} number in the type $B(\kappa)$ ticket. The cardinality of each subset $\mathcal{C}_i$ is at least $\kappa - 1$, for $2 \leq i \leq 6$ (no such restriction is imposed on $\mathcal{C}_1$ and $\mathcal{C}_7$), and each number in $\mathcal{C}_i$ is less than the number in T_j, for all $1 \leq i \leq j \leq 6$. Furthermore, each number in C_i is greater than the number in T_{i-1}, for all $2 \leq i \leq 7$. The number of ways in which to partition the set of 49 numbers thus equals the number of ways in which $49 - [6 + 5 \times (\kappa - 1)]$ indistinguishable balls may be placed into 7 distinguishable urns. Here, the balls represent numbers and the urns

represent the sets $\mathcal{C}_1, \ldots, \mathcal{C}_7$. To compute this number, consider Figure 7.2 in which $C_1, \ldots, C_7$ represent the seven urns into which the balls must be placed. If exactly $\kappa - 1$ balls are initially placed into urns C_2 to C_6, $49 - [6 + 5 \times (\kappa - 1)]$ balls remain to be placed inside urns $C_1, \ldots, C_7$ in any order. There are $\binom{q+p-1}{p}$ ways in which p indistinguishable balls may be placed into q distinguishable urns [21]. Taking $p = 49 - [6 + 5 \times (\kappa - 1)]$ and $q = 7$, it therefore follows that there are $\binom{54-5\kappa}{48-5\kappa}$ distinct ways in which a type $B(\kappa)$ ticket may be selected.

$$C_1 \quad \bullet \quad C_2 \quad \bullet \quad C_3 \quad \bullet \quad C_4 \quad \bullet \quad C_5 \quad \bullet \quad C_6 \quad \bullet \quad C_7$$

FIGURE 7.2: *Six balls (represented by the black dots) and seven urns $C_1, \ldots, C_7$ which in total contain $43 = (49 - 6)$ balls, under the condition that $C_2, \ldots, C_6$ each contains at least $\kappa - 1$ balls.*

Finally, there are $\binom{31}{\kappa} \times \binom{49-31}{6-\kappa}$ distinct ways in which a lottery participant may select a type $C(\kappa)$ ticket.

Table 7.3 contains the number of $B(\kappa)$ and $C(\kappa)$ tickets in the lottery $\langle 49, 6, 6, k \rangle$ for $1 \leq \kappa \leq 10$.

TABLE 7.3: *The number, $\binom{54-5\kappa}{48-5\kappa}$, of distinct type $B(\kappa)$ tickets, and the number, $\binom{31}{\kappa}\binom{18}{6-\kappa}$, of distinct type $C(\kappa)$ tickets in the lottery $\langle 49, 6, 6, k \rangle$.*

κ	Type $B(\kappa)$ tickets	Type $C(\kappa)$ tickets
1	13 983 816	265 608
2	7 059 052	1 422 900
3	3 262 623	3 667 920
4	1 344 904	4 814 145
5	475 020	3 058 398
6	134 596	736 281
7	27 132	0
8	3 003	0
9	84	0
10	0	0

7.2.2 An analysis of winners assuming popular number choices

One of the most intuitive participant number choices to investigate is the case where a certain proportion of participants purchase type $B(\kappa)$ tickets. The winning ticket from the draw which took place on the 7th of February 2009 was of the form $\{3, 13, 17, 28, 37, 49\}$. In this ticket, the smallest difference between any two numbers is 4. Therefore, a simulation was conducted under the assumption that at least a proportion θ of the tickets purchased are constructed as type $B(4)$ tickets (while the remaining tickets are of type A). There are $\binom{54-5\times4}{48-5\times4} = \binom{34}{28} = 1\,344\,904$ type $B(4)$ tickets. During each iteration of the simulation, a proportion, θ, of the selected participant tickets comprise a uniform selection from these 1 344 904 type $B(4)$ tickets, and the remaining participant tickets comprise a uniform selection of type A tickets. This implies then, that *at least* a proportion θ of the tickets purchased by participants in a lottery draw consists of type $B(4)$ tickets. A sensitivity analysis was conducted by varying the value of θ so as to establish which values of θ resulted in realistic values for $E(N, 18)$. A realistic value for $E(N, 18)$ would be one from which it may be deduced that once every few years (say 10) the case where 18 participants share the lottery prize is expected. The results obtained in this sensitivity analysis are shown in Tables 7.4–7.7.

TABLE 7.4: *The expected number of government tickets covered λ times, $E(7.5M, \lambda)$, where it is assumed that a certain proportion, θ, of participants select type $B(4)$ tickets in the lottery $\langle 49, 6, 6, 6\rangle$ and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 7 500 000 tickets are selected. Results computed via Algorithm 7.1.*

$E(7.5M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	8 179 037.160	9 071 921.059	9 964 207.330
$\lambda = 1$	4 386 555.697	3 515 090.902	2 644 007.257
$\lambda = 2$	1 176 379.263	956 930.983	737 558.133
$\lambda = 3$	210 324.210	197 641.628	185 204.247
$\lambda = 4$	28 202.331	63 589.947	99 084.086
$\lambda = 5$	3 024.103	48 175.372	93 336.893
$\lambda = 10$	0.006	8 161.263	16 314.057
$\lambda = 15$	0.000	122.420	244.819
$\lambda = 18$	0.000	4.367	8.607

$E(7.5M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	10 858 070.290	11 750 609.396	12 644 001.130
$\lambda = 1$	1 771 636.512	900 449.567	28 396.219
$\lambda = 2$	518 009.930	298 496.325	79 160.603
$\lambda = 3$	172 413.394	159 837.730	147 141.277
$\lambda = 4$	134 365.872	169 843.604	205 148.631
$\lambda = 5$	138 494.732	183 692.651	228 819.865
$\lambda = 10$	24 478.928	32 637.092	40 813.944
$\lambda = 15$	366.736	486.947	609.882
$\lambda = 18$	12.836	17.195	21.663

TABLE 7.5: *The probability, $P(7.5M, \lambda)$, of λ participants winning the jackpot prize in the lottery $\langle 49, 6, 6, 6\rangle$ where it is assumed that a certain proportion, θ, of participants select type $B(4)$ tickets and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 7 500 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(7.5M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	5.849×10^{-1}	6.487×10^{-1}	7.126×10^{-1}
$\lambda = 1$	3.137×10^{-1}	2.514×10^{-1}	1.891×10^{-1}
$\lambda = 2$	8.412×10^{-2}	6.843×10^{-2}	5.274×10^{-2}
$\lambda = 3$	1.504×10^{-2}	1.413×10^{-2}	1.324×10^{-2}
$\lambda = 4$	2.017×10^{-3}	4.547×10^{-3}	7.086×10^{-3}
$\lambda = 5$	2.163×10^{-4}	3.445×10^{-3}	6.675×10^{-3}
$\lambda = 10$	4.291×10^{-10}	5.836×10^{-4}	1.166×10^{-3}
$\lambda = 15$	0	8.754×10^{-6}	1.751×10^{-5}
$\lambda = 18$	0	3.123×10^{-7}	6.155×10^{-7}

$P(7.5M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	7.765×10^{-1}	8.403×10^{-1}	9.042×10^{-1}
$\lambda = 1$	1.267×10^{-1}	6.439×10^{-2}	2.031×10^{-3}
$\lambda = 2$	3.704×10^{-2}	2.135×10^{-2}	5.661×10^{-3}
$\lambda = 3$	1.233×10^{-2}	1.143×10^{-2}	1.052×10^{-2}
$\lambda = 4$	9.609×10^{-3}	1.215×10^{-2}	1.467×10^{-2}
$\lambda = 5$	9.904×10^{-3}	1.314×10^{-2}	1.636×10^{-2}
$\lambda = 10$	1.751×10^{-3}	2.334×10^{-3}	2.919×10^{-3}
$\lambda = 15$	2.623×10^{-5}	3.482×10^{-5}	4.361×10^{-5}
$\lambda = 18$	9.179×10^{-7}	1.230×10^{-6}	1.549×10^{-6}

TABLE 7.6: *The expected number of government tickets covered λ times, $E(20M, \lambda)$, where it is assumed that a certain proportion, θ, of participants select type $B(4)$ tickets in the lottery $\langle 49, 6, 6, 6\rangle$ and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 20 000 000 tickets are selected. Results computed via Algorithm 7.1.*

$E(20M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	3 345 743.519	5 204 399.123	7 057 888.958
$\lambda = 1$	4 785090.071	3 828 019.761	2 876 274.307
$\lambda = 2$	3 421 836.583	2 737 512.513	2 053 018.346
$\lambda = 3$	1 631 336.593	1 305 128.650	977 866.419
$\lambda = 4$	583 325.571	466 836.077	350 112.849
$\lambda = 5$	166 855.802	134 063.104	101 542.400
$\lambda = 10$	32.958	13 671.408	27 490.540
$\lambda = 15$	0.000	27 539.397	54 917.059
$\lambda = 18$	0.000	18 494.808	36 879.270
$E(20M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	8 918 020.819	10 778 985.129	12 638 912.472
$\lambda = 1$	1 918 460.625	958 783.941	6.873
$\lambda = 2$	1 368 360.499	684 094.673	51.378
$\lambda = 3$	651 877.667	326 201.717	256.015
$\lambda = 4$	233 522.385	117 329.149	955.162
$\lambda = 5$	68 436.833	35 622.695	2 836.785
$\lambda = 10$	41 192.074	54 665.176	68 229.095
$\lambda = 15$	82 528.460	110 141.552	137 690.007
$\lambda = 18$	55 396.733	73 954.836	92 500.876

TABLE 7.7: *The probability, $P(20M, \lambda)$, of λ participants winning the jackpot prize in the lottery $\langle 49, 6, 6, 6\rangle$ where it is assumed that a certain proportion, θ, of participants purchase type $B(4)$ tickets and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 20 000 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(20M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	2.393×10^{-1}	3.722×10^{-1}	5.047×10^{-1}
$\lambda = 1$	3.422×10^{-1}	2.737×10^{-1}	2.057×10^{-1}
$\lambda = 2$	2.447×10^{-1}	1.958×10^{-1}	1.468×10^{-1}
$\lambda = 3$	1.167×10^{-1}	9.333×10^{-2}	6.993×10^{-2}
$\lambda = 4$	4.171×10^{-2}	3.338×10^{-2}	2.504×10^{-2}
$\lambda = 5$	1.193×10^{-2}	9.587×10^{-3}	7.261×10^{-3}
$\lambda = 10$	2.357×10^{-6}	9.777×10^{-4}	1.966×10^{-3}
$\lambda = 15$	0	1.969×10^{-3}	3.928×10^{-3}
$\lambda = 18$	0	1.326×10^{-3}	2.637×10^{-3}
$P(20M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	6.378×10^{-1}	7.708×10^{-1}	9.038×10^{-1}
$\lambda = 1$	1.371×10^{-1}	6.856×10^{-2}	4.915×10^{-7}
$\lambda = 2$	9.785×10^{-2}	4.892×10^{-2}	3.674×10^{-6}
$\lambda = 3$	4.662×10^{-2}	2.333×10^{-2}	1.831×10^{-5}
$\lambda = 4$	1.670×10^{-2}	8.390×10^{-3}	6.830×10^{-5}
$\lambda = 5$	4.894×10^{-3}	2.547×10^{-3}	2.029×10^{-4}
$\lambda = 10$	2.946×10^{-3}	3.909×10^{-3}	4.879×10^{-3}
$\lambda = 15$	5.902×10^{-3}	7.876×10^{-3}	9.846×10^{-3}
$\lambda = 18$	3.961×10^{-3}	5.289×10^{-3}	6.615×10^{-3}

A sensitivity analysis was also conducted with respect to the proportion θ of participants who select tickets which contain calendar numbers. The winning ticket on the 7th of February 2009, $\{3, 13, 17, 28, 37, 49\}$, contains four calendar numbers. Out of the $\binom{49}{6}$ possible tickets from which the participant may choose, there are $\binom{31}{4} \times \binom{49-31}{2} = 4\,814\,145$ distinct type $C(4)$ tickets. Table 7.9 contains estimates of the values of $P(7.5M, \lambda)$, and Table 7.10 contains estimates of the values of $P(20M, \lambda)$. Tables containing approximations of $E(7.5M, \lambda)$ and $E(20M, \lambda)$ are omitted for the sake of brevity.

The following definition is also adopted in the remainder of this chapter.

Type $D(\kappa, c)$ ticket. A lottery ticket in which the difference between any two numbers is no less than κ, *and* exactly c of the numbers are calendar numbers.

The winning ticket in the South African National Lottery draw of the 7th of February 2009 (*i.e.* $\{3, 13, 17, 28, 37, 49\}$) is a type $D(4, 4)$ ticket. From Table 7.8, there are 671 859 distinct type $D(4, 4)$ tickets.

A simulation was finally conducted under the assumption that at least a proportion θ of the tickets purchased are type $D(4, 4)$ tickets for lottery draws in which 7 500 000 or 20 000 000 tickets are purchased in total. Table 7.11 contains estimates of the the values of $P(7.5M, \lambda)$, while Table 7.12 contains estimates of the the values of $P(20M, \lambda)$.

The number of distinct type $D(\kappa, c)$ tickets in the lottery $\langle 49, 6, 6, k\rangle$ may be computed via a direct counting approach [15], yielding

$$\begin{aligned} D(\kappa, c) &= \binom{31 - (c-1)(\kappa-1) - (\kappa-1)}{c} \binom{18 - (n-c-1)(\kappa-1)}{n-c} \\ &\quad + \sum_{i=1}^{\kappa-1} \binom{31 - (c-2)(\kappa-1) - (\kappa-1+i)}{c-1} \binom{18 - (n-c-1)(\kappa-1) - (\kappa-1)}{n-c} \end{aligned}$$

which, for n=6, simplifies to

$$\begin{aligned} D(\kappa, c) &= \binom{31 + c - c\kappa)}{c} \binom{23 + c(\kappa-1) - 5\kappa}{6-c} \\ &\quad + \sum_{i=1}^{\kappa-1} \binom{30 + c - i + \kappa - c\kappa}{c-1} \binom{23 + i + c(\kappa-1) - 6\kappa}{6-c}. \end{aligned} \tag{7.2}$$

7.2.3 Interpretation of simulation results

In this section, the results in Tables 7.1–7.12 are analysed. Consider the bipartite lottery graph $\mathbf{G}\langle m, n, t, k\rangle$, with $\binom{m}{t}$ government tickets (each of degree r) and $\binom{m}{n}$ participant tickets (of which s are special tickets[3]) as its vertices. In any given draw a government ticket which is chosen as the winning ticket is covered by $u \in \{u_1, u_2, \ldots, u_z\}$ special tickets. Let w_j be the number of government tickets which are covered by u_j $(1 \leq j \leq z)$ special participant tickets. Assume that at least a proportion θ of the N tickets purchased in total are special tickets. Hence the expected number of times that each special ticket is played is

$$\frac{\theta N}{s} + \frac{N - \theta N}{\binom{m}{n}},$$

[3]A special ticket is defined as either a type A, $B(\kappa)$, $C(\kappa)$ or $D(\kappa, c)$ ticket.

TABLE 7.8: *The number of distinct type $D(\kappa, c)$ tickets in the lottery $\langle 49, 6, 6, k\rangle$, computed via (7.2).*

	$c=0$	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$
$\kappa=1$	18 564	265 608	1 422 900	3 667 920	4 814 145	3 058 398	736 281
$\kappa=2$	1 716	61 347	583 219	2 006 550	2 737 800	1 438 190	230 230
$\kappa=3$	28	7 490	188 535	994 500	1 432 739	585 067	54 264
$\kappa=4$	0	169	40 865	430 459	671 859	193 544	8 008
$\kappa=5$	0	0	3 934	153 254	270 782	46 585	462
$\kappa=6$	0	0	0	40 684	87 444	6 467	1
$\kappa=7$	0	0	0	6 969	19 904	259	0
$\kappa=8$	0	0	0	525	2 478	0	0
$\kappa=9$	0	0	0	0	84	0	0
$\kappa=10$	0	0	0	0	0	0	0

TABLE 7.9: *The probability, $P(7.5M, \lambda)$, of λ participants winning the jackpot prize where it is assumed that a certain proportion, θ, of participants purchase type $C(4)$ tickets in the lottery $\langle 49, 6, 6, 6\rangle$ and where the remaining proportion $1-\theta$ of participants select type A tickets, if a total of 7 500 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(7.5M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	5.845×10^{-1}	6.136×10^{-1}	6.422×10^{-1}
$\lambda = 1$	3.137×10^{-1}	2.735×10^{-1}	2.334×10^{-1}
$\lambda = 2$	8.412×10^{-2}	8.490×10^{-2}	8.566×10^{-2}
$\lambda = 3$	1.504×10^{-2}	2.117×10^{-2}	2.730×10^{-2}
$\lambda = 4$	2.017×10^{-3}	5.170×10^{-3}	8.327×10^{-3}
$\lambda = 5$	2.163×10^{-4}	1.282×10^{-3}	2.348×10^{-3}
$\lambda = 10$	4.291×10^{-10}	3.350×10^{-7}	6.786×10^{-7}
$\lambda = 15$	0	0	7.151×10^{-11}
$\lambda = 18$	0	0	0

$P(7.5M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	6.709×10^{-1}	6.995×10^{-1}	7.282×10^{-1}
$\lambda = 1$	1.933×10^{-1}	1.532×10^{-1}	1.129×10^{-1}
$\lambda = 2$	8.643×10^{-2}	8.720×10^{-2}	8.798×10^{-2}
$\lambda = 3$	3.342×10^{-2}	3.955×10^{-2}	4.568×10^{-2}
$\lambda = 4$	1.148×10^{-2}	1.463×10^{-2}	1.779×10^{-2}
$\lambda = 5$	3.412×10^{-3}	4.476×10^{-3}	5.544×10^{-3}
$\lambda = 10$	1.006×10^{-6}	1.337×10^{-6}	1.694×10^{-6}
$\lambda = 15$	0	0	7.151×10^{-11}
$\lambda = 18$	0	0	0

while the expected number of times that each non-special ticket will be played is

$$\frac{N - \theta N}{\binom{m}{n}}.$$

If a winning ticket is selected which is covered by u special tickets, it is expected that there will be

$$u\left(\frac{\theta N}{s} + \frac{N - \theta N}{\binom{m}{n}}\right) + (r-u)\left(\frac{N - \theta N}{\binom{m}{n}}\right) = \frac{u\theta N}{s} + \frac{r(N - \theta N)}{\binom{m}{n}} \tag{7.3}$$

winners. The probability that a winning government ticket is selected which is covered by $u = u_j$ special tickets equals $w_j/\binom{m}{t}$. Therefore, the expected number of winners in a lottery draw is

$$\sum_{j=1}^{z} \frac{w_j}{\binom{m}{t}} \left(\frac{u_j \theta N}{s} + \frac{r(N - \theta N)}{\binom{m}{n}} \right). \tag{7.4}$$

The case of type A tickets only

If it is assumed that each participant constructs only type A tickets, Tables 7.1 and 7.2 are applicable. From these tables it may be seen that if 7 500 000 tickets are purchased in total, there is a 1.232×10^{-21} probability of eighteen participants winning the lottery. However, if 20 000 000 tickets are purchased in total, this probability rises to 2.343×10^{-14}.

If 7 500 000 tickets are purchased in a draw of the lottery $\langle 49, 6, 6, 6\rangle$, the expected number of winners per draw, (7.4), is equal to $7\,500\,000/\binom{49}{6} \approx 0.536$. Therefore, in this case, the highest value of $P(7.5M, \lambda)$ should occur where $\lambda \approx 0.536$ and the value of $P(7.5M, \lambda)$ should decrease as λ increases, depending on the value of (7.3). Likewise, if 20 000 000 tickets were to be purchased, the expected number of winners per draw would rise to $20\,000\,000/\binom{49}{6} \approx 1.430$. Therefore, in this case, the highest value of $P(20M, \lambda)$ should occur where $\lambda \approx 1.430$ and the value of $P(20M, \lambda)$ should decrease as the difference between λ and 1.430 increases. These expected growth patterns of $P(7.5M, \lambda)$ and $P(20M, \lambda)$ as a function of λ may be seen in Figure 7.3, which is a plot of the results of Table 7.1.

TABLE 7.10: *The probability, $P(20M, \lambda)$ of λ participants winning the jackpot prize in the lottery $\langle 49, 6, 6, 6\rangle$ where it is assumed that a certain proportion, θ, of participants purchase type $C(4)$ tickets and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 20 000 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(20M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	2.393×10^{-1}	3.236×10^{-1}	4.080×10^{-1}
$\lambda = 1$	3.422×10^{-1}	2.783×10^{-1}	2.143×10^{-1}
$\lambda = 2$	2.447×10^{-1}	2.051×10^{-1}	1.655×10^{-1}
$\lambda = 3$	1.167×10^{-1}	1.062×10^{-1}	9.582×10^{-2}
$\lambda = 4$	4.171×10^{-2}	4.678×10^{-2}	5.185×10^{-2}
$\lambda = 5$	1.193×10^{-2}	2.068×10^{-2}	2.945×10^{-2}
$\lambda = 10$	2.356×10^{-6}	4.574×10^{-4}	9.138×10^{-4}
$\lambda = 15$	0	1.572×10^{-6}	3.136×10^{-6}
$\lambda = 18$	0	2.074×10^{-8}	4.562×10^{-8}

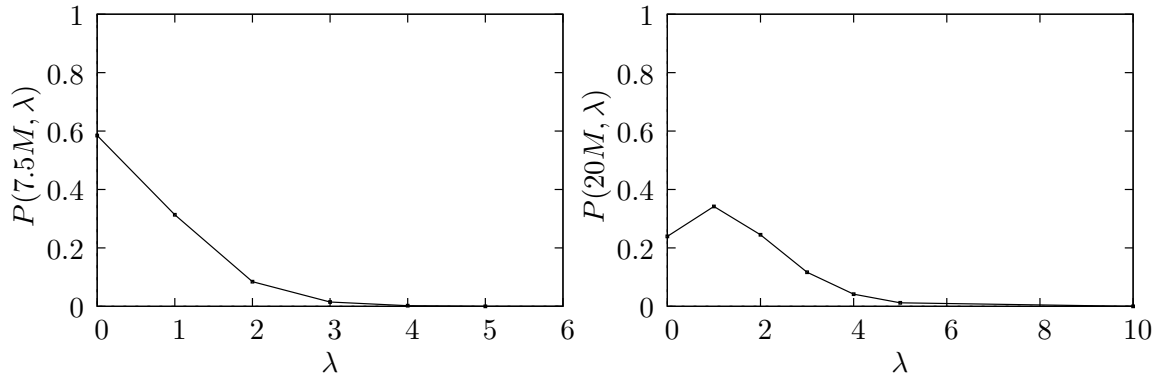
$P(20M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	4.923×10^{-1}	5.768×10^{-1}	6.611×10^{-1}
$\lambda = 1$	1.505×10^{-1}	8.640×10^{-2}	2.245×10^{-2}
$\lambda = 2$	1.259×10^{-1}	8.624×10^{-2}	4.663×10^{-2}
$\lambda = 3$	8.541×10^{-2}	7.499×10^{-2}	6.457×10^{-2}
$\lambda = 4$	5.692×10^{-2}	6.120×10^{-2}	6.706×10^{-2}
$\lambda = 5$	3.820×10^{-2}	4.696×10^{-2}	5.572×10^{-2}
$\lambda = 10$	1.368×10^{-3}	1.825×10^{-3}	2.281×10^{-3}
$\lambda = 15$	4.690×10^{-6}	6.314×10^{-6}	7.796×10^{-6}
$\lambda = 18$	6.829×10^{-8}	9.382×10^{-8}	1.1770×10^{-7}

TABLE 7.11: *The probability, $P(7.5M, \lambda)$, of λ participants winning the jackpot prize where it is assumed that a certain proportion, θ, of participants purchase type $D(4, 4)$ tickets in the lottery $\langle 49, 6, 6, 6\rangle$ and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 7 500 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(7.5M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	5.849×10^{-1}	6.583×10^{-1}	7.316×10^{-1}
$\lambda = 1$	3.137×10^{-1}	2.510×10^{-1}	1.883×10^{-1}
$\lambda = 2$	8.412×10^{-2}	6.730×10^{-2}	5.047×10^{-2}
$\lambda = 3$	1.504×10^{-2}	1.206×10^{-2}	9.091×10^{-3}
$\lambda = 4$	2.017×10^{-3}	1.701×10^{-3}	1.402×10^{-3}
$\lambda = 5$	2.166×10^{-4}	3.698×10^{-4}	5.396×10^{-4}
$\lambda = 10$	2.145×10^{-10}	1.129×10^{-3}	2.255×10^{-3}
$\lambda = 15$	0	5.432×10^{-4}	1.084×10^{-3}
$\lambda = 18$	0	1.543×10^{-4}	3.076×10^{-4}
$P(7.5M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	8.051×10^{-1}	8.785×10^{-1}	9.520×10^{-1}
$\lambda = 1$	1.255×10^{-1}	6.274×10^{-2}	7.626×10^{-6}
$\lambda = 2$	3.368×10^{-2}	1.686×10^{-2}	4.248×10^{-5}
$\lambda = 3$	6.111×10^{-3}	3.135×10^{-3}	1.580×10^{-4}
$\lambda = 4$	1.071×10^{-3}	7.565×10^{-4}	4.409×10^{-4}
$\lambda = 5$	6.774×10^{-4}	8.307×10^{-4}	9.849×10^{-4}
$\lambda = 10$	3.387×10^{-3}	4.514×10^{-3}	5.645×10^{-3}
$\lambda = 15$	1.630×10^{-3}	2.173×10^{-3}	2.715×10^{-3}
$\lambda = 18$	4.629×10^{-4}	6.170×10^{-4}	7.715×10^{-4}

TABLE 7.12: *The probability, $P(20M, \lambda)$, of λ participants winning the jackpot prize in the lottery $\langle 49, 6, 6, 6\rangle$ where it is assumed that a certain proportion, θ, of participants purchase type $D(4, 4)$ tickets and where the remaining proportion $1 - \theta$ of participants select type A tickets, if a total of 20 000 000 tickets are selected. Results computed via Algorithm 7.1.*

$P(20M, \lambda)$	$\theta = 0\%$	$\theta = 20\%$	$\theta = 40\%$
$\lambda = 0$	2.392×10^{-1}	3.818×10^{-1}	5.243×10^{-1}
$\lambda = 1$	3.421×10^{-1}	2.738×10^{-1}	2.053×10^{-1}
$\lambda = 2$	2.447×10^{-1}	1.958×10^{-1}	1.468×10^{-1}
$\lambda = 3$	1.167×10^{-1}	9.333×10^{-2}	7.000×10^{-2}
$\lambda = 4$	4.171×10^{-2}	3.337×10^{-2}	2.503×10^{-2}
$\lambda = 5$	1.193×10^{-2}	9.545×10^{-3}	7.158×10^{-3}
$\lambda = 10$	2.372×10^{-6}	2.064×10^{-6}	1.759×10^{-6}
$\lambda = 15$	0	1.105×10^{-5}	2.211×10^{-5}
$\lambda = 18$	0	5.968×10^{-5}	1.194×10^{-4}
$P(20M, \lambda)$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$\lambda = 0$	6.669×10^{-1}	8.094×10^{-1}	9.520×10^{-1}
$\lambda = 1$	1.369×10^{-1}	6.844×10^{-2}	0
$\lambda = 2$	9.788×10^{-2}	4.894×10^{-2}	0
$\lambda = 3$	4.666×10^{-2}	2.333×10^{-2}	0
$\lambda = 4$	1.669×10^{-2}	8.344×10^{-3}	2.145×10^{-10}
$\lambda = 5$	4.774×10^{-3}	2.386×10^{-3}	1.073×10^{-9}
$\lambda = 10$	1.468×10^{-6}	1.166×10^{-6}	8.548×10^{-7}
$\lambda = 15$	3.327×10^{-5}	4.441×10^{-5}	5.543×10^{-5}
$\lambda = 18$	1.789×10^{-4}	2.388×10^{-4}	2.984×10^{-4}



(a) $N = 7\,500\,000$ (b) $N = 20\,000\,000$

FIGURE 7.3: *Graphs of $P(7.5M, \lambda)$ and $P(20M, \lambda)$ as a function of the number of winners in the lottery $\langle 49, 6, 6, 6\rangle$ under the assumption that participants purchase type A tickets.*

The case of type A and type $B(4)$ tickets

Tables 7.4 and 7.5 contain the values of $E(7.5M, \lambda)$ and $P(7.5M, \lambda)$, respectively, for the case in which at least a proportion θ of the tickets purchased by participants are of type $B(4)$. The winning government ticket may either be covered by a single type $B(4)$ participant ticket (which implies that, on expectation, there will be

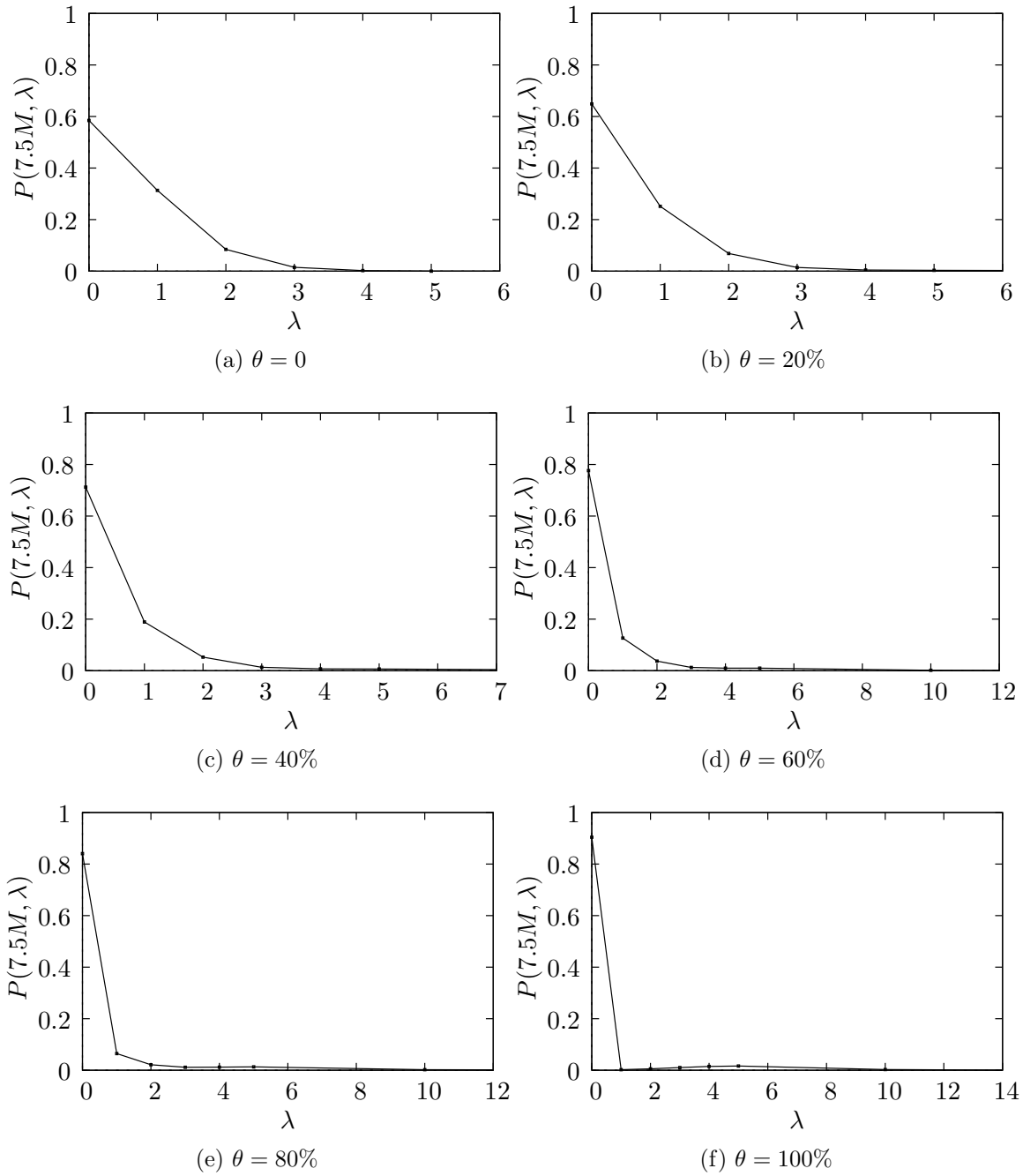
$$\frac{7\,500\,000\theta}{1\,344\,904} + \frac{7\,500\,000(1-\theta)}{\binom{49}{6}}$$

winners), or a single participant ticket which is not of type $B(4)$ (which implies that, on expectation, there will be

$$\frac{7\,500\,000(1-\theta)}{\binom{49}{6}}$$

winners). Table 7.13 contains these values for different values of θ, and Figure 7.6 illustrates the growth patterns inherent to the values in Table 7.13. The values in Table 7.5 are expected to be distributed around the value in (7.4), taking the possible values of (7.3) into consideration. For example, for the value of $\theta = 40\%$, the largest values of $P(7.5M, \lambda)$ appear where $\lambda \approx 0.536$, and they start to decrease rapidly for $\lambda \geq 2.552$, the expected maximum number of winners from Table 7.13. Of course, values appear for $\lambda > 2.552$ because that value is only the *expected* maximum number of winners. For example, if 7 500 000 tickets are purchased in total, it is expected that a type $B(4)$ ticket will be purchased approximately 2.552 times on average. However, it may occur that one specific type $B(4)$ ticket is purchased many more times than another one. If a winning government ticket is selected which is covered by that type $B(4)$ ticket, there may be significantly more winners than expected, which could distort the results somewhat. The growth patterns of $P(7.5M, \lambda)$ as a function of λ may be seen in Figure 7.4, which is a plot of the results of Table 7.5.

A similar analysis may be conducted in order to explain the nature of the values in Tables 7.6 and 7.7. For example, consider the case where $\theta = 80\%$. From Table 7.14, it may be seen that the expected number of winners is 1.430, but the maximum expected number of winners is 12.183 and the minimum expected number of winners is 0.286. Therefore, with a probability of 0.904, there will be approximately 0.286 winners, and with a probability of 0.096 there will be



(a) $\theta = 0$ (b) $\theta = 20\%$

(c) $\theta = 40\%$ (d) $\theta = 60\%$

(e) $\theta = 80\%$ (f) $\theta = 100\%$

FIGURE 7.4: *$P(7.5M, \lambda)$ as a function of the number, λ, of winners of the lottery $\langle 49, 6, 6, 6 \rangle$. From the results in Table 7.5, under the assumption that a certain proportion, θ, of participants purchase type $B(4)$ tickets and the remaining proportion, $1 - \theta$, of participants select type A tickets, if a total of 7 500 000 tickets are selected.*

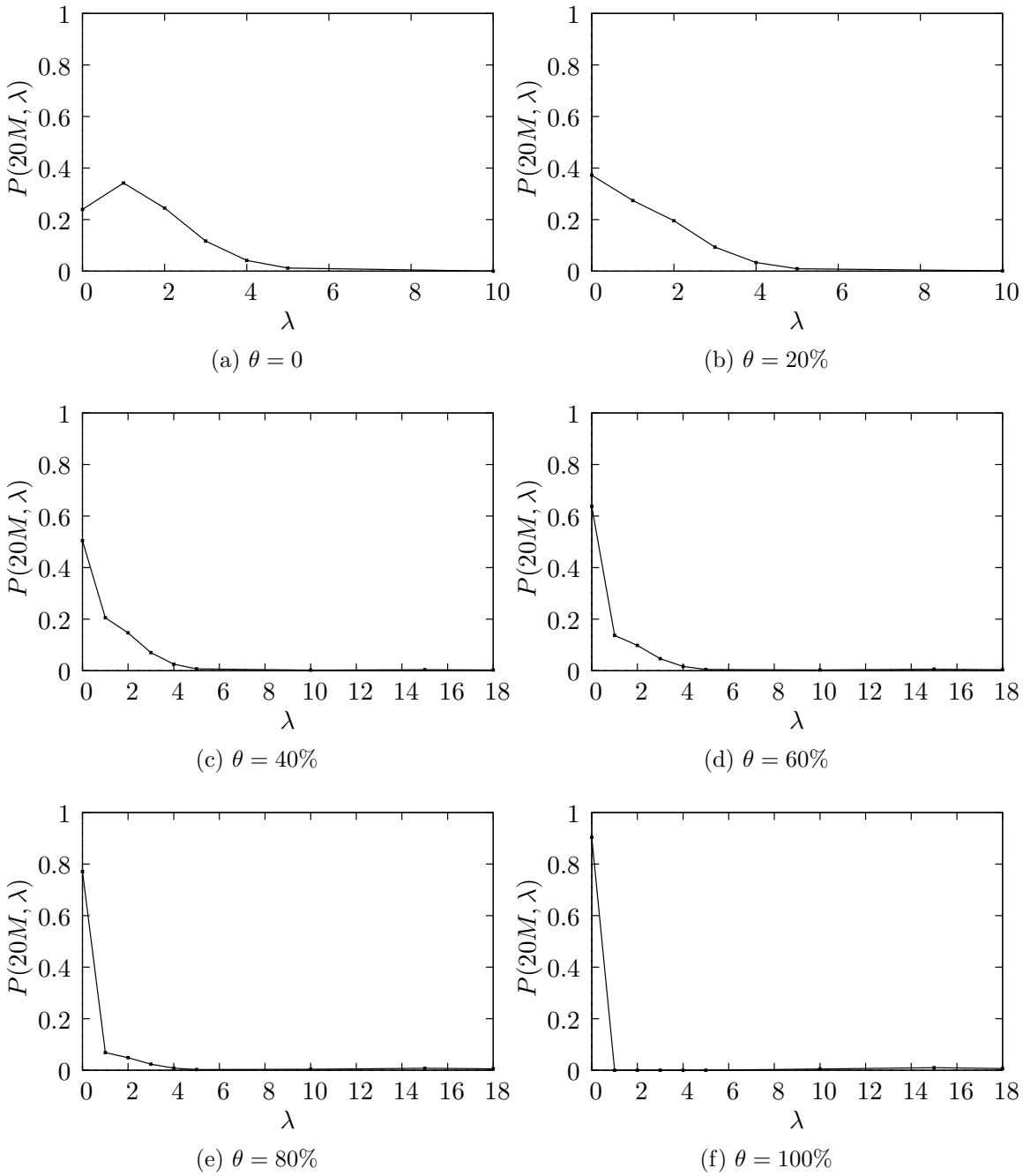


FIGURE 7.5: $P(20M, \lambda)$ *as a function of the number,* λ*, of winners of the lottery* $\langle 49, 6, 6, 6\rangle$*. From the results in Table 7.7, under the assumption that a certain proportion,* θ*, of participants purchase type* $B(4)$ *tickets and the remaining proportion,* $1 - \theta$*, of participants select type A tickets, if a total of* 20 000 000 *tickets are selected.*

TABLE 7.13: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 7\,500\,000$ participant tickets played are of type $B(4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	0.536	0.429	0.322	0.215	0.107	0	0.904
1	0.536	1.544	2.552	3.560	4.569	5.577	0.096
	0.536	0.536	0.536	0.536	0.536	0.536	

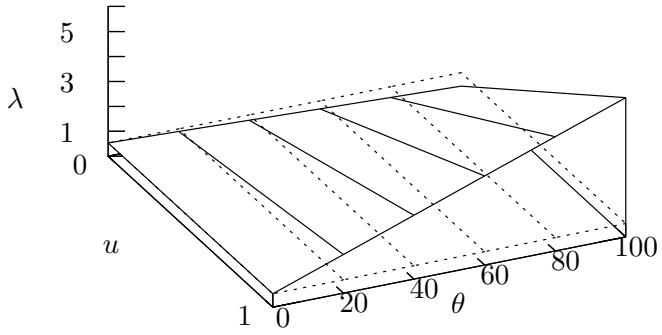


FIGURE 7.6: *The number of winners λ as a function of the values of u and θ for the lottery $\langle 49, 6, 6, 6\rangle$ in the situation where a proportion θ of a total of 7 500 000 tickets purchased are tickets of type $B(4)$. Here, u denotes the number of special participant tickets which may cover the winning government ticket.*

approximately 12.183 winners which explains the increase in the value of $P(20M, \lambda)$ for values of λ close to 12.183. As another example, consider the case where $\theta = 100\%$. In this case, if a winning government ticket is purchased which is not covered by a type $B(4)$ participant ticket, there will be no winners because only type $B(4)$ tickets are played. Because there is approximately a $(\binom{49}{6} - 1\,344\,904)/\binom{49}{6} = 0.904$ probability of this occurring, $P(20M, 0) \approx 0.904$. There is a probability of approximately 0.096 that approximately 14.871 participants will share the winnings. Therefore, the value of $P(20M, \lambda)$ will be relatively high near $\lambda = 14.871$, as may be seen in Figure 7.16. The growth pattern of $P(20M, \lambda)$ as a function of λ may be seen in Figure 7.5, which is a plot of the results of Table 7.7.

TABLE 7.14: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 20\,000\,000$ participant tickets played are of type $B(4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	1.430	1.144	0.858	0.572	0.286	0	0.904
1	1.430	4.118	6.807	9.495	12.183	14.871	0.096
	1.430	1.430	1.430	1.430	1.430	1.430	

The case of type A and type $C(4)$ tickets

Alternatively, assume that a certain proportion of participants in a lottery draw purchase type $C(4)$ tickets. If $N = 7\,500\,000$ and it is assumed that $\theta = 80\%$, it follows by Table 7.15 that the

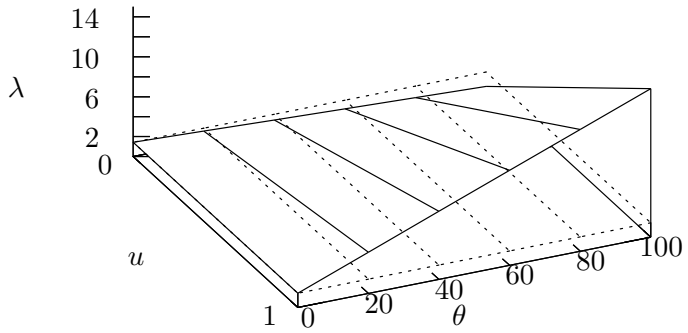


FIGURE 7.7: *The number of winners λ as a function of the values of u and θ for the lottery $\langle 49, 6, 6, 6\rangle$ in the situation where a proportion θ of a total of 20 000 000 tickets purchased are tickets of type $B(4)$. Here, u denotes the number of special participant tickets which may cover the winning government ticket.*

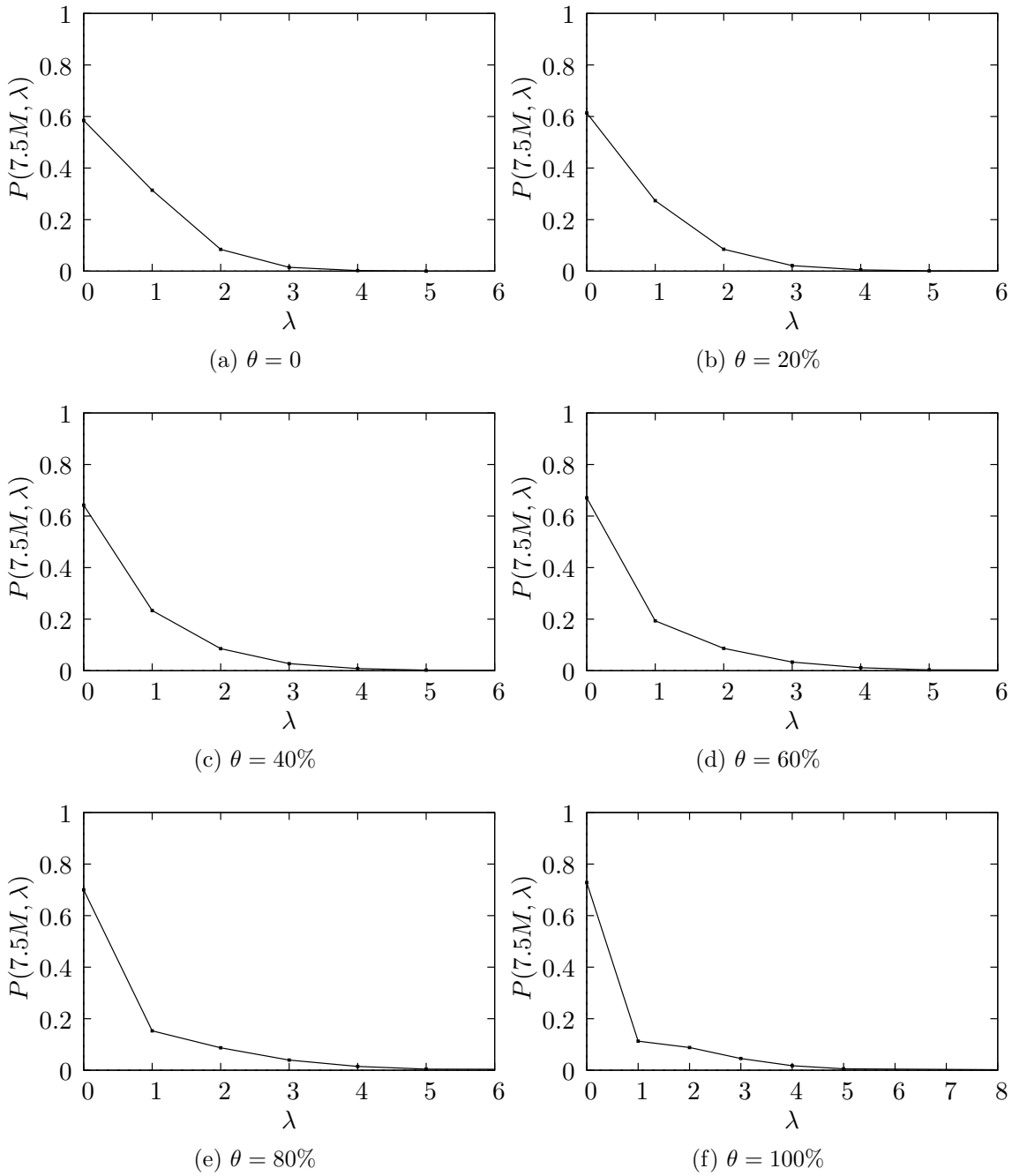
expected number of winners of the jackpot is 0.536, the expected minimum number of winners in a draw is 0.107, and the expected maximum number of winners is 1.354. This explains why the values of $P(7.5M, \lambda)$ steadily decrease from $\lambda = 0$. In the case where 20 000 000 tickets are purchased, if $\theta = 20\%$, and if a winning government ticket is chosen which is not covered by a participant ticket of type $C(4)$ it is expected that there will be 1.144 winners. With a probability of 0.656, a government ticket is selected which is not covered by a participant ticket of type $C(4)$, which explains the relatively high value of $P(20M, 0) = 6.136 \times 10^{-1}$. Due to the value of the expected number of winners, and the expected minimum and maximum number of winners being very close to each other, the value of $P(20M, \lambda)$ decreases as λ increases, for all values of $\lambda \geq 1.975$. The growth patterns of $P(7.5M, \lambda)$ and $P(20M, \lambda)$ as functions of λ may be seen in Figures 7.8 and 7.9, which are plots of the results of Tables 7.9 and 7.10.

TABLE 7.15: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 7\,500\,000$ participant tickets played are of type $C(4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	0.536	0.429	0.322	0.215	0.107	0	0.656
1	0.536	0.741	0.945	1.149	1.354	1.558	0.344
	0.536	0.536	0.536	0.536	0.536	0.536	

TABLE 7.16: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 20\,000\,000$ participant tickets played are of type $C(4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	1.430	1.144	0.858	0.572	0.286	0	0.656
1	1.430	1.975	2.520	3.065	3.610	4.154	0.344
	1.430	1.430	1.430	1.430	1.430	1.430	

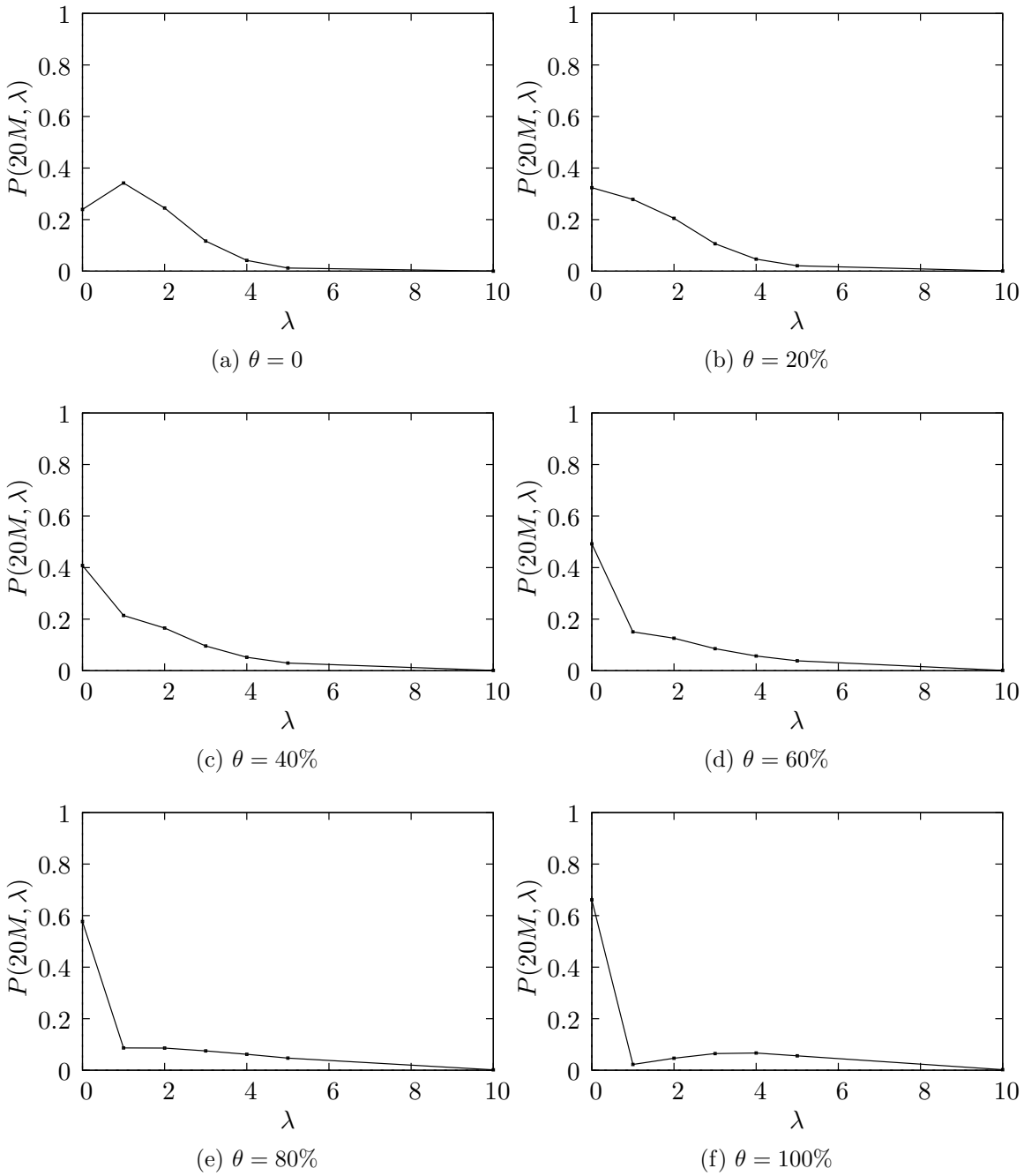


(a) $\theta = 0$ (b) $\theta = 20\%$

(c) $\theta = 40\%$ (d) $\theta = 60\%$

(e) $\theta = 80\%$ (f) $\theta = 100\%$

FIGURE 7.8: *$P(7.5M, \lambda)$ as a function of the number, λ, of winners of the lottery $\langle 49, 6, 6, 6\rangle$. From the results in Table 7.9 under the assumption that a certain proportion, θ, of participants purchase type $C(4)$ tickets and the remaining proportion, $1 - \theta$, of participants select type A tickets, if a total of 7 500 000 tickets are selected.*



(a) $\theta = 0$ (b) $\theta = 20\%$

(c) $\theta = 40\%$ (d) $\theta = 60\%$

(e) $\theta = 80\%$ (f) $\theta = 100\%$

FIGURE 7.9: *$P(20M, \lambda)$ as a function of the number, λ, of winners of the lottery $\langle 49, 6, 6, 6\rangle$. From the results in Table 7.10 under the assumption that a certain proportion, θ, of participants purchase type $C(4)$ tickets and the remaining proportion, $1 - \theta$, of participants select type A tickets, if a total of* 20 000 000 *tickets are selected.*

TABLE 7.17: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 7\,500\,000$ participant tickets played are of type $D(4, 4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	0.536	0.429	0.322	0.215	0.107	0	0.952
1	0.536	2.662	4.787	6.912	9.038	11.163	0.048
	0.536	0.536	0.536	0.536	0.536	0.536	

TABLE 7.18: *The expected number of winners in (7.3) for a draw of the lottery $\langle 49, 6, 6, 6\rangle$ if a winning government ticket is selected which is covered by $u = u_j$ special participant tickets and it is assumed that at least a proportion θ of the $N = 20\,000\,000$ participant tickets played are of type $D(4, 4)$. The bottom row of the table contains the values from (7.4). The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
0	1.430	1.144	0.858	0.572	0.286	0	0.952
1	1.430	7.098	12.765	18.433	24.101	29.768	0.048
	1.430	1.430	1.430	1.430	1.430	1.430	

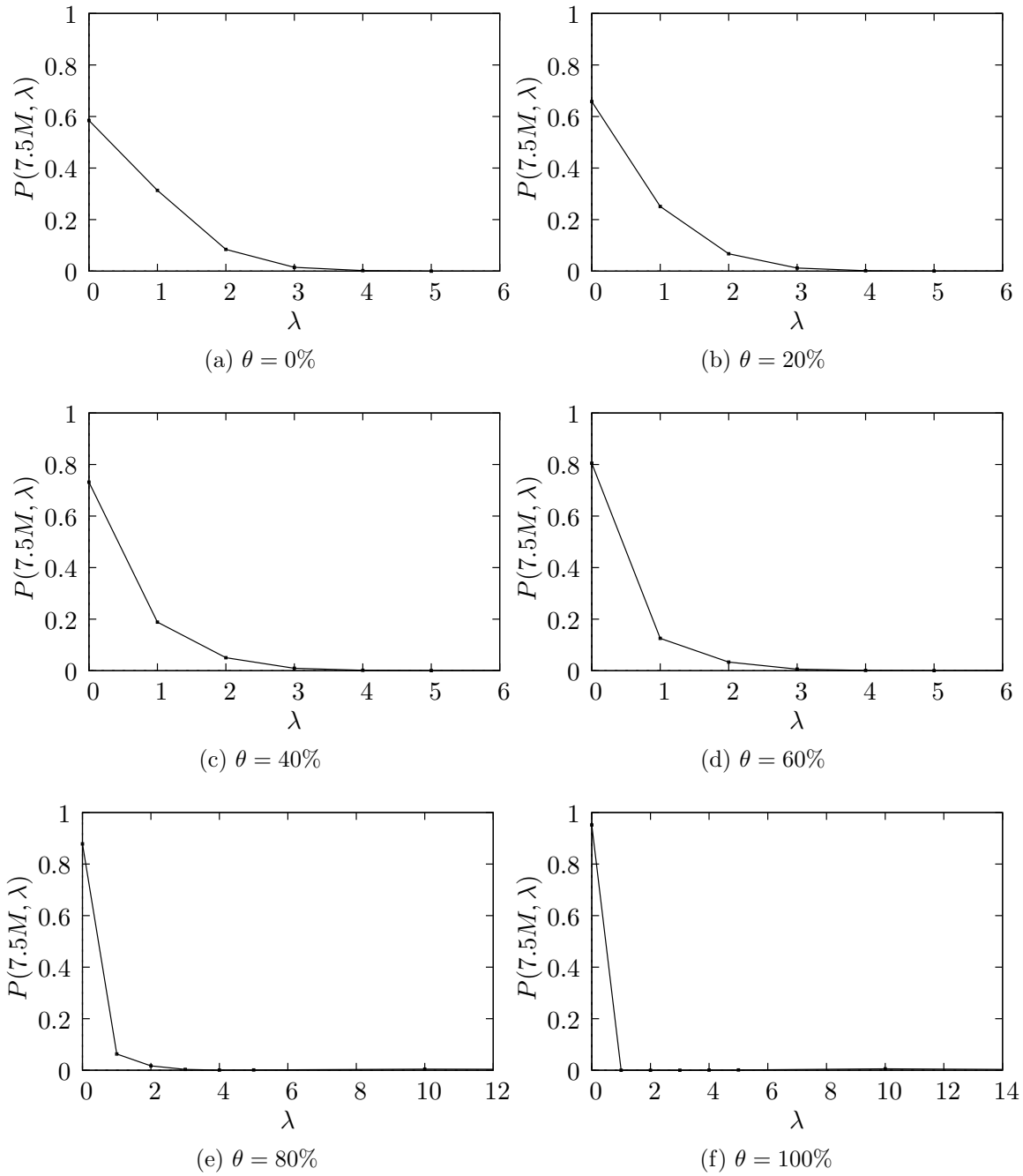
The case of type A and type $D(4, 4)$ tickets

In Tables 7.11 and 7.12, values of $P(N, \lambda)$ appear for the case in which a certain proportion θ of the tickets purchased by participants are type $D(4, 4)$. The nature of the values in Tables 7.11 and 7.12 may be explained by examining the entries in Tables 7.17 and 7.18. For example, if $N = 7\,500\,000$ tickets are purchased and it is assumed that $\theta = 60$, it may be seen from Table 7.17 that the expected number of winners is 0.536, the expected maximum number of winners is 6.912 and the expected minimum number of winners is 0.215. The probability of approximately 0.536 participants winning is 0.952. This implies that it is likely that a small number of participants may win simultaneously, which is substantiated by the values of $P(7.5M, 0)$ and $P(7.5M, 1)$ because $P(7.5M, 0) + P(7.5M, 1) \approx 0.930$. The value of $P(7.5M, \lambda)$ should decrease as λ increases for $\lambda \geq 0.215$. However, a steady decrease may only occur for values of $\lambda \geq 6.912$. The growth patterns of $P(7.5M, \lambda)$ and $P(20M, \lambda)$ as functions of λ may be seen in Figures 7.10 and 7.11, which are plots of the results of Tables 7.11 and 7.12.

7.2.4 Expected waiting time between extreme events

In this section, the results of the previous sections are used to establish the expected waiting time between two successive events of at least 18 participants having to share the jackpot in the lottery $\langle 49, 6, 6, 6\rangle$. The event of 18 participants winning the lottery may be perceived to be unlikely. Therefore this event is called an *extreme event*.

If 20 000 000 tickets are purchased in total, it follows by Tables 7.1 and 7.2 that, on average, eighteen participants will win the lottery once every 4.268×10^{13} lottery draws. There are two lottery draws per week in the South African National Lottery, which implies that there are $52 \times 2 = 104$ lottery draws per year. Therefore, if it is assumed that for every draw, 20 000 000 tickets are purchased in total, it may be expected that, on average, eighteen participants will win the lottery approximately once every 4.104×10^{11} years. Likewise, eighteen participants

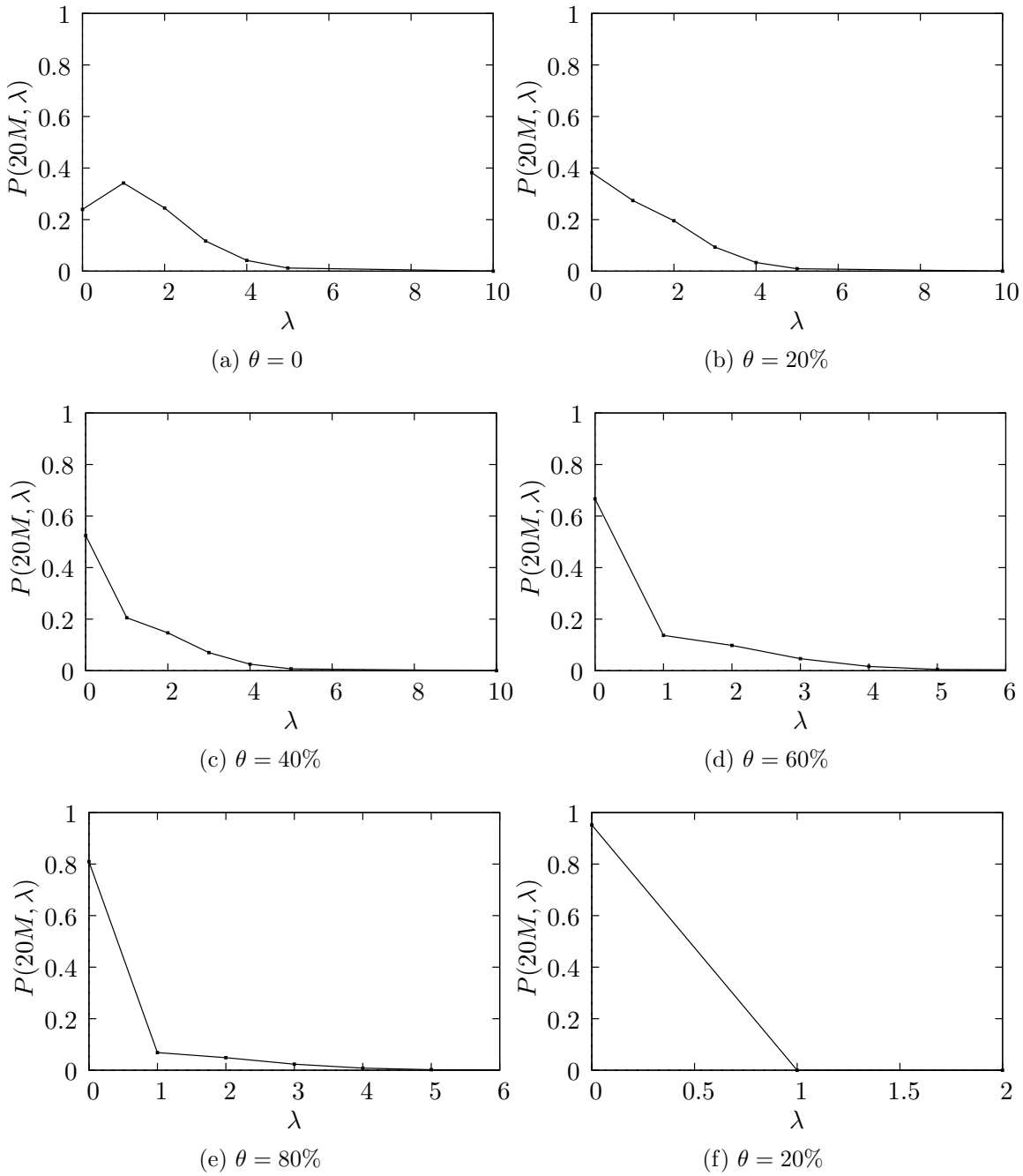


(a) $\theta = 0\%$ (b) $\theta = 20\%$

(c) $\theta = 40\%$ (d) $\theta = 60\%$

(e) $\theta = 80\%$ (f) $\theta = 100\%$

FIGURE 7.10: *$P(7\,500\,000, \lambda)$ as a function of the number, λ, of winners in the lottery $\langle 49, 6, 6, 6\rangle$. From the results in Table 7.11, under the assumption that a certain proportion, θ, of participants purchase type $D(4, 4)$ tickets and the remaining proportion, $1 - \theta$, of participants select type A tickets, if a total of 7 500 000 tickets are selected.*



(a) $\theta = 0$ (b) $\theta = 20\%$

(c) $\theta = 40\%$ (d) $\theta = 60\%$

(e) $\theta = 80\%$ (f) $\theta = 20\%$

FIGURE 7.11: *Graphs of* $P(20\,000\,000, \lambda)$ *as a function of the number,* λ*, of winners in the lottery* $\langle 49, 6, 6, 6\rangle$*. From the results in Table 7.12 under the assumption that a certain proportion,* θ*, of participants purchase type* $D(4, 4)$ *tickets and the remaining proportion,* $1 - \theta$*, of participants select type A tickets, if a total of* 20 000 000 *tickets are selected.*

are expected to win the lottery once every 7.805×10^{18} years if 7 500 000 tickets purchased in total for every lottery draw. However, these expectations hold under the assumption that all participants select tickets of type A.

Table 7.19 contains values of $P(N, \lambda = 18)$ and $P(N, \lambda \geq 18)$ when it is assumed that a certain proportion θ of the tickets purchased are of type $B(4)$. From this table, it may be seen that if a small number of tickets (such as 7 500 000) is purchased, then it is unlikely that 18 participants will win the lottery jackpot. However, if a large number of tickets (such as 20 000 000) is purchased, it is likely that *exactly* 18 participants will share the jackpot approximately every 7.27 years if $\theta = 20\%$, and within a shorter time span if θ is larger. Also, if $\theta = 20\%$, it may be deduced that *at least* 18 participants are likely to share the jackpot prize approximately every 2.08 years. This number is close to being a reflection of reality, because on the dates 31 October 2001, 15 March 2003 and 7 February 2009, there were 19, 33, and 18 winners, respectively. Therefore, in approximately 7.3 years there were 3 occurrences in which at least 18 participants won the jackpot ($7.3/3 = 2.433$).

The values in Table 7.20 show that, under the assumption of a certain proportion of the population purchasing tickets which are of type $C(4)$, it is highly unlikely that 18 participants will win the jackpot in the lottery $\langle 49, 6, 6, 6 \rangle$ within the average person's lifetime [10]. This occurrence has, however taken place three times since 31 October 2001, which indicates that the assumption that a certain proportion $\theta \in \{0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\}$ of participants purchase tickets of type $C(4)$ may not be an accurate reflection of reality.

The values in Table 7.21 represent the case in which a certain proportion, θ, of the tickets purchased are of type $D(4, 4)$. From this table, if 7 500 000 tickets are purchased, it is expected to take between 12.46 (if $\theta = 100\%$) and 62.33 years (if $\theta = 20\%$) for a draw to occur in which *exactly* 18 participants win the jackpot. For draws in which it is assumed that 20 000 000 tickets are purchased, it may take even longer for a draw to occur in which *exactly* 18 participants win the jackpot. However, it may take between 5.52 (if $\theta = 100\%$) and 27.62 years (if $\theta = 20\%$) for a draw to occur in which *at least* 18 participants win the jackpot if 7 500 000 tickets are purchased, and it may take between 0.20 (if $\theta = 100\%$) or 1.01 years (if $\theta = 20\%$) for a draw to occur in which *at least* 18 participants win the jackpot if 20 000 000 tickets are purchased.

Finally, from data obtained from [32], the average number of tickets purchased per draw is 13 718 105. It therefore follows by Table 7.22, in which it is assumed that a certain proportion, θ, of the tickets purchased are of type $B(4)$, that it is expected to take between 12.05 (if $\theta = 100\%$) or 60.36 years (if $\theta = 20\%$) for a draw to occur in which *exactly* 18 participants win the jackpot if 13 718 105 tickets are purchased, while it is expected to take between 5.87 (if $\theta = 100\%$) or 29.43 years (if $\theta = 20\%$) for a draw to occur in which *at least* 18 participants win the jackpot if 13 718 105 tickets are purchased.

7.3 An analysis of small lottery instances

It is computationally infeasible to conduct a similar investigation to the one in the previous sections on the lottery $\langle 49, 6, 6, k \rangle$ where $3 \leq k \leq 5$. Therefore, in order to investigate the number of concurrent lottery winners of a k-prize (not necessarily the jackpot prize), small lottery instances are considered. Tables 7.23 and 7.24 contain values, computed via Algorithm 7.1, of $P(N, \lambda)$ for the small lotteries $\langle 8, 3, 5, 2 \rangle$ and $\langle 9, 4, 5, 3 \rangle$, respectively, if it is assumed that at least a proportion θ of the N tickets purchased are type $B(2)$ tickets.

Note that under the assumption of each participant purchasing tickets of type A only and the

winning government ticket also being of type A, the binomial distribution probability mass function (7.1) may be used to compute the value $P(N, \lambda)$, where

$$p = \frac{\sum_{i=k}^{n} \binom{t}{i}\binom{m-t}{n-i}}{\binom{m}{n}} \tag{7.5}$$

denotes the probability of a single participant ticket yielding a k-prize.

As mentioned before, the values of $P(N, \lambda)$ are distributed around the value of (7.4) and depend on the possible values of the expression in (7.3). From Tables 7.23 and 7.24, it may be seen that for a given value of θ, the value of $P(N, \lambda)$ increases as λ increases up to a certain point after which the value of $P(N, \lambda)$ decreases as λ increases. This phenomenon may be explained by examining Tables 7.25 and 7.26 which contain values of λ for each possible value of u and θ for the lotteries $\langle 8, 3, 5, 2\rangle$ and $\langle 9, 4, 5, 3\rangle$, respectively.

TABLE 7.19: *The probability of either exactly* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$*, or at least* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$ *is presented in the second column. Columns three and four contain the expected number of draws and the expected number of years, respectively, between successive occurrences of these events. These results hold under the assumption that a proportion* θ *of participants purchase tickets of type* $B(4)$*, and that either* 7 500 000 *or* 20 000 000 *tickets are purchased in total.*

θ	$P(7.5M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	3.123×10^{-7}	3 202 151.85	30 789.92
40	6.155×10^{-7}	1 624 703.29	15 622.15
60	9.179×10^{-7}	1 089 421.93	10 475.21
80	1.230×10^{-6}	813 246.15	7 819.67
100	1.549×10^{-6}	645 515.28	6 206.88
θ	$P(7.5M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	4.346×10^{-7}	2 300 726.56	22 122.37
40	8.594×10^{-7}	1 163 572.64	11 188.20
60	1.291×10^{-6}	774 340.55	7 445.58
80	1.726×10^{-6}	579 231.88	5 569.54
100	2.164×10^{-6}	462 213.79	4 444.36
θ	$P(20M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	1.323×10^{-3}	756.09	7.27
40	2.637×10^{-3}	379.18	3.65
60	3.961×10^{-3}	252.43	2.43
80	5.289×10^{-3}	189.09	1.82
100	6.615×10^{-3}	151.18	1.45
θ	$P(20M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	4.622×10^{-3}	216.36	2.08
40	9.217×10^{-3}	108.50	1.04
60	1.384×10^{-2}	72.25	0.69
80	1.848×10^{-2}	54.12	0.52
100	2.311×10^{-2}	43.27	0.42

TABLE 7.20: *The probability of either exactly* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$*, or at least* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$ *is presented in the second column. Columns three and four contain the expected number of draws and the expected number of years, respectively, between successive occurrences of these events. These results hold under the assumption that a proportion* θ *of the participants' tickets are of type* $C(4)$*, and that either* 7 500 000 *or* 20 000 000 *tickets are purchased in total.*

θ	$P(7.5M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	0	∞	∞
40	0	∞	∞
60	0	∞	∞
80	0	∞	∞
100	0	∞	∞
θ	$P(7.5M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	0	∞	∞
40	0	∞	∞
60	0	∞	∞
80	0	∞	∞
100	0	∞	∞
θ	$P(20M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	2.074×10^{-8}	48 219 960.17	463 653.46
40	4.562×10^{-8}	21 918 192.54	210 751.85
60	6.829×10^{-8}	14 642 746.28	140 795.64
80	9.382×10^{-8}	10 658 401.43	102 484.63
100	1.177×10^{-7}	8 495 671.46	81 689.15
θ	$P(20M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	2.674×10^{-8}	37 389 880.19	359 518.08
40	5.749×10^{-8}	17 392 806.13	167 238.52
60	8.638×10^{-8}	11 576 006.58	111 307.76
80	1.201×10^{-7}	8 323 700.01	80 035.58
100	1.504×10^{-7}	6 646 300.38	63 906.73

TABLE 7.21: *The probability of either exactly* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$*, or at least* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$ *is presented in the second column. Columns three and four contain the expected of draws and the expected number of years, respectively, between successive occurrences of these events. These results hold under the assumption that in a proportion* θ *of the participants' tickets are of type* $D(4, 4)$*, and that either* 7 500 000 *or* 20 000 000 *tickets are purchased in total.*

θ	$P(7.5M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	1.542×10^{-4}	6 482.48	62.33
40	3.076×10^{-4}	3 250.76	31.26
60	4.629×10^{-4}	2 160.23	20.77
80	6.169×10^{-4}	1 620.80	15.58
100	7.715×10^{-4}	1 296.15	12.46
θ	$P(7.5M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	3.4813×10^{-4}	2 872.21	27.62
40	6.946×10^{-4}	1 439.53	13.84
60	1.044×10^{-3}	957.08	9.20
80	1.392×10^{-3}	718.14	6.91
100	1.740×10^{-3}	574.42	5.52
θ	$P(20M, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	5.967×10^{-5}	16 757.30	161.13
40	1.193×10^{-4}	8 377.45	80.55
60	1.789×10^{-4}	5 588.50	53.74
80	2.388×10^{-4}	4 186.85	40.26
100	2.983×10^{-4}	3 351.66	32.23
θ	$P(20M, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	9.531×10^{-3}	104.92	1.01
40	1.906×10^{-2}	52.46	0.50
60	2.859×10^{-2}	34.97	0.34
80	3.812×10^{-2}	26.23	0.25
100	4.765×10^{-2}	20.98	0.20

TABLE 7.22: *The probability of either exactly* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$*, or at least than* 18 *participants winning the lottery* $\langle 49, 6, 6, 6\rangle$ *is presented in the second column. Columns three and four contain the expected number of draws and years, respectively, between successive occurrences of these events. These results hold under the assumption that a proportion,* θ*, of participant tickets are of type* $B(4)$*, and that the average number of tickets (*13 718 105*) has been purchased.*

θ	$P(13\,718\,105, \lambda = 18)$	Draws	Years
0	0	∞	∞
20	1.592×10^{-4}	6 277.89	60.36
40	3.190×10^{-4}	3 134.52	30.14
60	4.779×10^{-4}	2 092.31	20.12
80	6.373×10^{-4}	1 568.88	15.09
100	7.979×10^{-4}	1 253.21	12.05
θ	$P(13\,718\,105, \lambda \geq 18)$	Draws	Years
0	0	∞	∞
20	3.267×10^{-4}	3 060.69	29.43
40	6.544×10^{-4}	1 528.18	14.69
60	9.805×10^{-4}	1 019.84	9.81
80	1.308×10^{-3}	764.53	7.35
100	1.637×10^{-3}	610.84	5.87

In table 7.25, it may be seen that the value of (7.4) decreases as the value of θ increases. However, the minimum expected number of winners decreases as θ increases, and the maximum expected number of winners increases as θ increases. This is illustrated graphically in Figure 7.12. As an example, consider a draw of the lottery $\langle 8, 3, 5, 2\rangle$ in which 800 tickets are purchased, and $\theta = 60\%$. It is expected that, on average, 537.143 participants will win a 2-prize, the expected minimum number of winners is 468.571, and the expected maximum number of winners is 612.571. Also, from Table 7.25, there is a probability if approximately $0.232 + 0.268 = 0.5$ of either 516.571 or 540.571 participants winning the lottery draw. Therefore, it is expected that the value of $P(800, \lambda)$ is relatively large around the values of $\lambda = 516.571$ and $\lambda = 540.571$. From Table 7.23, it may be seen that the larger values of $P(800, \lambda)$ are indeed between $\lambda = 500$ and $\lambda = 600$, but the value of $P(800, \lambda)$ increases as the value of λ moves closer to 600.

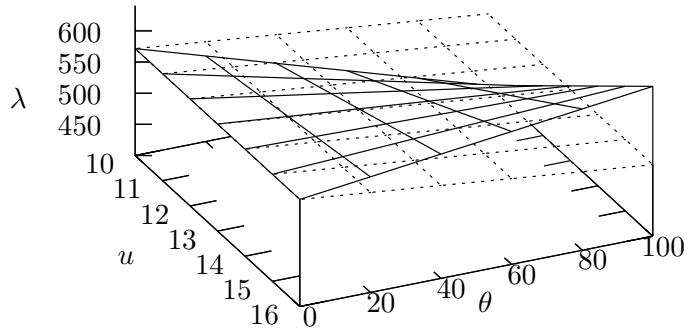


FIGURE 7.12: *The number of winners* λ *as a function of the values of* u *and* θ *for the lottery* $\langle 8, 3, 5, 2\rangle$ *in the situation where a proportion* θ *of the* $100 \times m$ *tickets purchased are of type* $B(2)$*. The graph drawn in solid lines represents the value of* λ *as* θ *and* u *vary, while the graph drawn in dotted lines represents the expected value of* λ*.*

In Table 7.26, the situation is slightly different in the sense that the value of (7.4) remains

TABLE 7.23: *The probability, $P(N, \lambda)$, of λ participants winning a 2-prize in the lottery $\langle 8, 3, 5, 2\rangle$ in which a proportion θ of the tickets purchased are type $B(2)$ tickets. Computed from Algorithm 7.1*

λ	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$N = m$						
0	4.566×10^{-5}	4.738×10^{-5}	5.743×10^{-5}	7.875×10^{-5}	1.161×10^{-4}	1.729×10^{-4}
1	8.852×10^{-4}	9.335×10^{-4}	1.077×10^{-3}	1.320×10^{-3}	1.708×10^{-3}	2.239×10^{-3}
2	7.787×10^{-3}	7.993×10^{-3}	8.653×10^{-3}	9.789×10^{-3}	1.138×10^{-2}	1.340×10^{-2}
3	3.884×10^{-2}	3.934×10^{-2}	4.074×10^{-2}	4.302×10^{-2}	4.596×10^{-2}	4.948×10^{-2}
4	1.214×10^{-1}	1.216×10^{-1}	1.221×10^{-1}	1.229×10^{-1}	1.238×10^{-1}	1.249×10^{-1}
5	2.429×10^{-1}	2.420×10^{-1}	2.392×10^{-1}	2.348×10^{-1}	2.290×10^{-1}	2.221×10^{-1}
6	3.036×10^{-1}	3.024×10^{-1}	2.989×10^{-1}	2.933×10^{-1}	2.857×10^{-1}	2.764×10^{-1}
7	2.168×10^{-1}	2.171×10^{-1}	2.181×10^{-1}	2.196×10^{-1}	2.211×10^{-1}	2.222×10^{-1}
8	6.779×10^{-2}	6.863×10^{-2}	7.108×10^{-2}	7.525×10^{-2}	8.123×10^{-2}	8.912×10^{-2}
$N = 10 \times m$						
0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
10	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
20	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
30	≈ 0	≈ 0	≈ 0	3.214×10^{-7}	3.357×10^{-6}	2.623×10^{-5}
40	2.550×10^{-5}	5.302×10^{-5}	1.900×10^{-4}	6.573×10^{-4}	1.816×10^{-3}	3.838×10^{-3}
50	2.086×10^{-2}	2.264×10^{-2}	2.650×10^{-2}	3.025×10^{-2}	3.242×10^{-2}	3.304×10^{-2}
60	7.937×10^{-2}	7.782×10^{-2}	7.331×10^{-2}	6.645×10^{-2}	5.861×10^{-2}	4.286×10^{-2}
70	3.522×10^{-4}	5.206×10^{-4}	1.208×10^{-3}	3.017×10^{-3}	6.787×10^{-3}	1.249×10^{-2}
80	≈ 0	≈ 0	≈ 0	≈ 0	1.786×10^{-8}	2.321×10^{-7}
$N = 100 \times m$						
0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
100	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
200	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
300	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
400	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	1.929×10^{-5}
500	3.571×10^{-8}	5.679×10^{-6}	5.115×10^{-4}	1.965×10^{-3}	2.622×10^{-3}	2.836×10^{-3}
600	2.508×10^{-3}	6.469×10^{-3}	9.126×10^{-3}	8.161×10^{-3}	7.791×10^{-3}	8.211×10^{-3}
700	≈ 0	≈ 0	≈ 0	≈ 0	1.161×10^{-6}	3.876×10^{-4}
800	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

TABLE 7.24: *The probability, $P(N, \lambda)$, of λ participants winning a 3-prize in the lottery $\langle 9, 4, 5, 3\rangle$ in which a proportion θ of the tickets purchased are type $B(2)$ tickets. Computed from Algorithm 7.1*

λ	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$
$N = m$						
0	1.875×10^{-2}	1.989×10^{-2}	2.321×10^{-2}	2.894×10^{-2}	3.778×10^{-2}	5.060×10^{-2}
1	9.379×10^{-2}	9.593×10^{-2}	1.017×10^{-1}	1.106×10^{-1}	1.216×10^{-1}	1.329×10^{-1}
2	2.083×10^{-1}	2.080×10^{-1}	2.070×10^{-1}	2.048×10^{-1}	2.004×10^{-1}	1.931×10^{-1}
3	2.701×10^{-1}	2.666×10^{-1}	2.573×10^{-1}	2.430×10^{-1}	2.253×10^{-1}	2.057×10^{-1}
4	2.250×10^{-1}	2.226×10^{-1}	2.157×10^{-1}	2.051×10^{-1}	1.918×10^{-1}	1.773×10^{-1}
5	1.250×10^{-1}	1.255×10^{-1}	1.264×10^{-1}	1.274×10^{-1}	1.276×10^{-1}	1.264×10^{-1}
6	4.635×10^{-2}	4.780×10^{-2}	5.185×10^{-2}	5.790×10^{-2}	6.519×10^{-2}	7.273×10^{-2}
7	1.104×10^{-2}	1.187×10^{-2}	1.430×10^{-2}	1.834×10^{-2}	2.409×10^{-2}	3.130×10^{-2}
8	1.531×10^{-3}	1.743×10^{-3}	2.392×10^{-3}	3.628×10^{-3}	5.690×10^{-3}	8.849×10^{-3}
9	9.579×10^{-5}	1.139×10^{-4}	1.847×10^{-4}	3.407×10^{-4}	6.406×10^{-4}	1.199×10^{-3}
$N = 10 \times m$						
0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	1.190×10^{-7}
10	3.175×10^{-8}	5.952×10^{-7}	1.350×10^{-5}	2.065×10^{-4}	1.811×10^{-3}	7.272×10^{-3}
20	2.148×10^{-3}	4.089×10^{-3}	1.051×10^{-2}	1.944×10^{-2}	2.433×10^{-2}	2.507×10^{-2}
30	7.946×10^{-2}	7.236×10^{-2}	5.831×10^{-2}	4.515×10^{-2}	3.540×10^{-2}	2.850×10^{-2}
40	1.982×10^{-2}	2.405×10^{-2}	2.953×10^{-2}	2.946×10^{-2}	2.633×10^{-2}	2.305×10^{-2}
50	5.605×10^{-5}	2.331×10^{-4}	1.559×10^{-3}	5.757×10^{-3}	1.121×10^{-2}	1.352×10^{-2}
60	≈ 0	6.349×10^{-8}	2.175×10^{-6}	5.301×10^{-5}	5.801×10^{-4}	2.986×10^{-3}
70	≈ 0	≈ 0	≈ 0	1.587×10^{-8}	3.016×10^{-7}	1.203×10^{-5}
80	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
90	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
$N = 10 \times m$						
0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
100	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	3.138×10^{-4}
200	≈ 0	≈ 0	1.724×10^{-5}	1.852×10^{-3}	3.117×10^{-3}	1.041×10^{-3}
300	9.217×10^{-3}	1.191×10^{-2}	7.704×10^{-3}	4.944×10^{-3}	4.761×10^{-3}	4.930×10^{-3}
400	1.587×10^{-8}	1.740×10^{-4}	3.006×10^{-3}	2.876×10^{-3}	4.063×10^{-3}	1.740×10^{-3}
500	≈ 0	≈ 0	≈ 0	1.574×10^{-5}	1.868×10^{-3}	9.157×10^{-4}
600	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	4.365×10^{-7}
700	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
800	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
900	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

TABLE 7.25: *The expected number of winners in a lottery draw for all possible values of u and θ for the lottery $\langle 8, 3, 5, 2\rangle$. The bottom row of the table contains the values from (7.4). It is assumed that at least a proportion θ of the tickets purchased are of type $B(2)$. The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
10	571.429	537.143	502.857	468.571	434.286	400	0.054
11	571.429	545.143	518.857	492.571	466.286	440	0.125
12	571.429	553.143	534.857	516.571	498.286	480	0.232
13	571.429	561.143	550.857	540.571	530.286	520	0.268
14	571.429	569.143	566.857	564.571	562.286	560	0.179
15	571.429	577.143	582.857	588.571	594.286	600	0.107
16	571.429	585.143	598.857	612.571	626.286	640	0.036
	571.429	560.000	548.571	537.143	525.714	514.286	

constant as θ increases. The reason is that for this problem instance, (7.4) reduces to

$$\frac{rN}{\binom{m}{n}}, \tag{7.6}$$

a value independent of θ. Also, from Table 7.26, the minimum expected number of winners decreases as θ increases, and the maximum expected number of winners increases as θ increases. This is illustrated graphically in Figure 7.13. As an example, consider a draw of the lottery $\langle 9, 4, 5, 3\rangle$ in which 900 tickets are purchased, and $\theta = 20\%$. The expected number of winners is 321.429, the minimum expected number of winners is 281, and the maximum expected number of winners is 365. This implies that the value of $P(900, \lambda)$ should be largest where λ is approximately 321.429 and the value of $P(900, \lambda)$ decreases as λ decreases from 321.429 to 281 or as λ increases to 365. As λ decreases from 281 or increases from 365, the value $P(900, \lambda)$ should converge to zero.

TABLE 7.26: *The expected number of winners in a lottery draw for all possible values of u and θ for the lottery $\langle 9, 4, 5, 3\rangle$. The bottom row of the table contains the values from (7.4). A certain proportion θ of the participants are assumed to purchase tickets of type $B(2)$. The rightmost column contains the probability that a government ticket is selected which is covered by $u = u_j$ special participant tickets.*

u	$\theta = 0$	$\theta = 20\%$	$\theta = 40\%$	$\theta = 60\%$	$\theta = 80\%$	$\theta = 100\%$	$w_j/\binom{m}{t}$
2	321.429	281.143	240.857	200.571	160.286	120	0.056
3	321.429	293.143	264.857	236.571	208.286	180	0.119
4	321.429	305.143	288.857	272.571	256.286	240	0.206
5	321.429	317.143	312.857	308.571	304.286	300	0.175
6	321.429	329.143	336.857	344.571	352.286	360	0.143
7	321.429	341.143	360.857	380.571	400.286	420	0.151
8	321.429	353.143	384.857	416.571	448.286	480	0.079
9	321.429	365.143	408.857	452.571	496.286	540	0.071
	321.429	321.429	321.429	321.429	321.429	321.429	

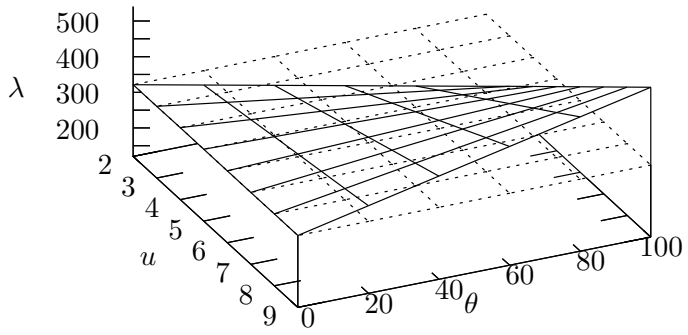


FIGURE 7.13: *The number of winners λ as a function of the values of u and θ for the lottery $\langle 9, 4, 5, 3\rangle$ in the situation where a proportion θ of the $100 \times m$ tickets purchased are of type $B(2)$. The graph drawn in solid lines represents the value of λ as θ and u vary, while the graph drawn in dotted lines represents the expected value of λ.*

In the lottery $\langle 8, 3, 5, 2\rangle$, $\Psi_1(8, 3, 5, 2) \approx 0.714$, and in the lottery $\langle 9, 4, 5, 3\rangle$, $\Psi_1(9, 4, 5, 3) \approx 0.357$. Therefore, because it is easier for a participant in the lottery $\langle 8, 3, 5, 2\rangle$ to win a 2-prize with a single ticket than it is for a participant to win a 3-prize in the lottery $\langle 9, 4, 5, 3\rangle$ with a single ticket, the expected number of winners in a draw of the lottery $\langle 8, 3, 5, 2\rangle$ is larger than the expected number of winners in a draw of the lottery $\langle 9, 4, 5, 3\rangle$. This is confirmed in Tables 7.25 and 7.26.

It is useful for the participants in a small lottery to know what the probabilities are of them having to share a k-prize with $\lambda-1$ other participants. This information may be used to provide the participant with an estimate of the prize money that they may receive should they win the k-prize. If the values of $P(N,\lambda)$ and T_k (the total value of the prize pool for the k-prize) are known, they may be used by participants in a lottery draw to compute their expected return if they were to construct playing sets conforming to a certain overlapping playing set structure which guarantees a probability-of-win value of $\Psi_\ell(m,n,t,k)$. In general, a participant's expected return in a lottery $\langle m,n,t,k\rangle$ is

$$\Psi_\ell(m,n,t,k)\sum_{\lambda=1}^{N}P(N,\lambda)\frac{T_k}{\lambda}. \tag{7.7}$$

Example 7.1 (Expected winnings in draw of the lottery $\langle 10,4,5,3\rangle$) *Assume a participant purchases a playing set of cardinality* 4*. If the playing set conforms to the overlapping playing set structure* $(0,1,1,1,1,1,1,0,1,1,1,0,1,0,0,0)$*, then the participant has a probability of approximately* 0.83 *of winning a k-prize. However, assume it is known that approximately* 1 000 *tickets are purchased altogether by participants in this particular draw. Let* T_3 *denote the prize pool for the* 3*-prize. From (7.7), the participant's expected return may be computed as*

$$0.83\sum_{\lambda=1}^{1\,000}P(1\,000,\lambda)\frac{T_3}{\lambda}.$$

If the prize pool for the 3*-prize,* T_3 *equals* R100 000*, it may be computed from (7.7) that the expected return is approximately* R317.81*.* ■

7.4 Chapter overview

This chapter commenced with the computation, by means of an analytical approach and by a simulation approach, of the probability that λ participants win a lottery draw under the assumptions that the winning government ticket is of type A and all the participants' tickets are also of type A. Specific attention was afforded to the probability, $P(N,\lambda=18)$, that $\lambda=18$ participants win a draw of the lottery $\langle 49,6,6,6\rangle$. It was found that if $N=7\,500\,000$ type A tickets are purchased in total, $P(7.5M,18)=1.232\times10^{-21}$, and if $N=20\,000\,000$ type A tickets are purchased in total, $P(20M,18)=2.343\times10^{-14}$. These values are so small that the event of eighteen participants winning a draw of the lottery $\langle 49,6,6,6\rangle$ may be perceived to be unlikely.

The chapter then proceeded under the assumption that a certain proportion θ of tickets purchased in a draw of the lottery $\langle 49,6,6,6\rangle$ are type $B(4)$ tickets and the other tickets purchased are type A tickets. It was found that $1.323\times10^{-3}\leq P(20M,18)\leq 6.615\times10^{-3}$ for $20\%\leq\theta\leq100\%$. This led to the deduction that, under the assumption that a proportion θ of the tickets purchased are type $B(4)$ tickets, the event of eighteen participants winning a draw is much more likely, than when it is assumed that only type A tickets are purchased by participants, and that the assumption made may be a close reflection of reality.

A further analysis was conducted in this chapter under the assumption that a certain proportion θ of tickets purchased are type $C(4)$ tickets and the other tickets purchased are type A tickets. The values of $P(N,\lambda)$ were not found to be a close reflection of reality. Finally, an analysis

was conducted under the assumption that a certain proportion θ of tickets purchased are type $D(4,4)$ tickets and the other tickets purchased are type A tickets. The resulting values of $P(N,\lambda)$ were also found to not be a reflection of reality.

From the analyses conducted on the expected number of winners in the lottery $\langle 49,6,6,6\rangle$ it may be concluded that the event of 18 or more participants winning a draw concurrently is likely under the assumption that a certain proportion θ of tickets purchased are type $B(4)$ tickets and the other tickets purchased are type A tickets.

Finally, it was also demonstrated in this chapter that the value $P(N,\lambda)$ is useful in the sense that it may be used to compute a participant's expected win per draw. This allows the participant to know whether or not partaking in a lottery draw is financially viable.

CHAPTER 8

Conclusion

Contents

This chapter contains a summary of the work contained in the previous chapters of this thesis. Novel contributions to the literature on the two problems considered in this thesis are highlighted, and the scope for future work is discussed.

8.1 Thesis summary

The focus in this thesis is on two combinatorial optimisation problems involving lotteries. These problems are known as the incomplete lottery problem (see Definition 1.1) and the resource utilisation problem (see Definition 1.2). In Chapter 2, a concise summary was given of previous work on these two problems. More specifically, the work by Burger *et al.* [5, 6], Gründlingh [16] and Li & van Rees [22, 24, 25] constitute a significant contribution to the study of these two problems and has formed the foundation upon which this thesis was built.

In Chapter 3, lotteries were modelled as bipartite graphs and from this model, an upper bound in (3.14) on the complete lottery number for problem instances in which $n \neq t$ was established. This upper bound is analogous to the upper bound on the cardinality of a minimum covering set of a bipartite graph, obtained via the greedy covering algorithm (Algorithm 3.1) by Astratian, *et al.* [2]. The bound in (3.14) was computed for small and large lottery instances and compared to bounds mostly obtained via graph domination theory [16], for small and large lottery instances in which $n = t$. The bound in (3.14) was also computed for small lottery instances in which $n \neq t$ and compared to the known values of the complete lottery number $L_1(m, n, t, k)$ of those lottery instances. The bound in (3.14), was generalised in (3.17) for the case of the incomplete lottery number and was compared to known values of $L_\psi(m, n, t, k)$ for small lottery instances. It was found that this new upper bound performed well for lotteries in which $n = t$, compared to the upper bounds obtained via graph domination theory, but it did not improve on the recursive bound for lotteries in (2.12) by Li & van Rees [27]. However, it was noted in Chapter 3 that the greedy bound (3.14) may be computed for any lottery $\langle m, n, t, k\rangle$, while the bound (2.12) by Li & van Rees [27] can only be computed for $L_1(m, n, t, k)$ if the values of the complete lottery

numbers $L_1(m-1, n-1, t-1, k-1)$ and $L_1(m-1, n, t, k)$ are known or if good bounds on these numbers are known. The difference between the bound in (3.17) on the incomplete lottery number and the value of $L_\psi(m, n, t, k)$ was found to be larger for small values of ψ, and as ψ increased, the difference was non-increasing, indicating that the bound improves as ψ increases.

In Chapter 4, a mathematical programming approach towards solving the incomplete lottery problem and the resource utilisation problem for the general case where $n \neq t$ was presented, building on previous work by Gründlingh [16] and by Jans & Degraeve [20] in which mathematical programming formulations were derived for the complete lottery problem, the incomplete lottery problem and the resource utilisation problem for lottery instances of the form $\langle m, n, n, k\rangle$. The new formulations were applied to small lottery instances in order to compute the values of $L_\psi(m, n, t, k)$ and $\Psi_\ell(m, n, t, k)$, and to compute bounds on the values of $L_\psi(m, n, t, k)$ and $\Psi_\ell(m, n, t, k)$ in cases where it was not practically feasible to obtain exact values. A data file containing the adjacency matrix (with dimensions $\binom{m}{t} \times \binom{m}{n}$) used in the formulations was required, and the size of this file increased dramatically, to such an extent that it was not practically feasible to solve problems with large parameters. The conclusion from this chapter was that, although a mathematical programming approach is a viable solution method, it is only practically feasible for small lottery instances — it is, for example, highly unlikely that exact values of $L_\psi(49, 6, 6, k)$ and $\Psi_\ell(49, 6, 6, k)$ will be found using this solution method in the near future.

Chapter 5 contained a review of work by Gründlingh [16] on an exhaustive enumeration method, based on the iterative construction of a rooted tree data structure in which nodes represent overlapping playing set structures for the case where $n = t$. In that chapter, the exhaustive enumeration lottery tree was generalised to the general case in which $n \neq t$. The solution method was described in detail, and the description was followed by three practical examples. The exhaustive enumeration lottery tree solution method was applied to the same small lottery instances investigated in Chapter 4. This was done in order to compare the mathematical programming solution method to the exhaustive enumeration lottery tree solution method in terms of execution time and practical feasibility and to validate the results of Chapter 4. It was found in Chapter 5 that the exhaustive enumeration solution method performs better than the mathematical programming approach in terms of execution time. The exhaustive enumeration lottery tree approach is also preferable because from it, all the overlapping playing set structures resulting in an optimal resource utilisation are determined, while only one playing set structure is determined via the mathematical programming approach. It was found in Chapter 5, however, that as the playing set cardinality exceeds 6, the exhaustive enumeration lottery tree solution method takes many hours, or even days, to find a solution, which leads to the conclusion that the exhaustive enumeration solution method is practically infeasible as a solution method for problem instances involving large playing set cardinalities. For the small lottery instances investigated, solutions to both the incomplete lottery problem and the resource utilisation problem were found via the mathematical programming approach for problems involving larger playing set cardinalities. It was therefore concluded that the mathematical programming approach is an appropriate exact solution method for lottery problem instances in which the parameters are small, but the cardinality of an optimal (or near optimal) playing set is relatively large, while the exhaustive enumeration lottery tree solution method is appropriate for lottery problem instances in which the cardinality of an optimal (or near optimal) solution is small, but the parameters are relatively large.

As mentioned, the execution time of the exhaustive enumeration lottery tree solution method presented in Chapter 5 increases dramatically as the associated playing set cardinality in the problem instance increases. Therefore, various (meta)heuristic approaches towards solving the

incomplete lottery problem and the resource utilisation problem were considered in Chapter 6. Gründlingh [16] also considered (meta)heuristic approaches towards solving these problems by investigating how a playing set of pre-specified cardinality may be constructed by iteratively adding different tickets to the playing set. In Chapter 6, an alternative method was presented in which a solution is represented by an overlapping playing set structure instead of by specific playing set tickets and the associated probability-of-win value depends on the manner in which the elements in the overlapping playing set structure were distributed amongst its compartments. A total of five (meta)heuristics were implemented, which resulted in the construction of different overlapping playing set structures by exploring the different ways in which the elements in the overlapping playing set structure may be redistributed amongst its compartments. The (meta)heuristic which performed the best in terms of solution quality was the tabu search method. This method was, however, found to be computationally expensive. The method that was computationally the least expensive was a genetic algorithm; however, the solution quality achieved via this method was not good. The solutions achieved for small problem instances via the tabu search method were the same in terms of the probability of win value obtained, but they were found in much shorter execution times compared to the exhaustive enumeration lottery tree solution method presented in Chapter 5. Hence the solution methods presented in Chapter 6 are good alternatives to the solution methods presented in Chapters 4 and 5 when a solution that is close to optimal is sought within a relatively short execution time. In some cases, the solution methods presented in Chapter 6 did not achieve optimal solutions, but they provided good bounds on the values of $L_{\psi}(m, n, t, k)$ and $\Psi_{\ell}(m, n, t, k)$. However, in other cases involving larger problem instances, optimal solutions were found — such as the newly established result that $L_1(20, 10, 4, 3) = 6$.

A relatively new contribution to the literature on lotteries was presented in Chapter 7. In this chapter, an investigation into the value of the probability $P(N, \lambda)$ of λ participants concurrently winning a k-prize in a lottery draw was conducted. As a case study, the jackpot prize in the South African National Lottery, a lottery of the form $\langle 49, 6, 6, 6\rangle$, was considered. In the draw which took place on the 7^{th} February 2009, 18 participants shared the jackpot prize, a seemingly unrealistic number compared to the average number, 1.002, of jackpot winners per draw. The probability $P(N, \lambda = 18)$ was found to be extremely small under the assumption that participants construct their playing sets from numbers which are randomly selected according to a uniform distribution. An alternative assumption was, however, made that not all players construct their playing sets from numbers which were randomly selected according to a uniform distribution [18, 19, 45], but rather that at least a proportion θ of the participants construct their playing sets from a pool of special tickets — an example of which is tickets constructed in such a way that the difference between any two numbers in the ticket is at least κ (a so-called type $B(\kappa)$ ticket) — the value of $P(N, \lambda = 18)$ thus increased dramatically for $\kappa = 4$, compared to its value under the original assumption. From an analysis of the entries in Table 7.19 it is expected that 18 or more players may win the jackpot concurrently once every 7.27 years under the assumption that 20 000 000 tickets are purchased in total for every draw and at least a proportion $\theta = 20\%$ of the tickets bought are type $B(4)$ tickets. Following the analysis of the lottery $\langle 49, 6, 6, 6\rangle$, an analysis of the value of $P(N, \lambda)$ associated with smaller lottery instances, in which the k-prize is not the jackpot prize, was conducted. It was found that if the winning government ticket is covered by a special ticket, a relatively large number of participants are expected to share the k-prize, while if the winning goverment ticket is not covered by a special ticket, a relatively small number of participants are expected to share the k-prize. In this chapter, the value $P(N, \lambda)$ was also used to compute a participant's expected return per lottery draw if the prize pool for a k-prize is known.

8.2 Novel contributions in this thesis

In Chapter 3, a new bound on the incomplete lottery number was established. This bound is not computationally expensive to compute and it may be applied to lotteries in which $n \neq t$. It performs well compared to bounds obtained via graph domination theory for small and large lotteries.

In Chapter 4, the incomplete lottery problem and the resource utilisation problem were modelled as integer programming models, and solved. Using this solution approach, optimal solutions to the incomplete lottery problem and the resource utilisation problem for small problem instances were computed. However, due to current limits on the computational ability of computers, it was found that a mathematical programming approach may not be practically feasible when used to seek answers to incomplete lottery and resource utilisation problem instances in which the parameters are relatively large (*i.e.* when $m > 10$).

In Chapter 5, the exhaustive enumeration lottery tree solution method of Burger, *et al.* [5] and Gründlingh [16] was generalised to the case where $n \neq t$ and it was compared in terms of execution times and practical feasibility with respect to the mathematical programming approach of solving the incomplete lottery problem and the resource utilisation problem. It was found that the exhaustive enumeration lottery tree solution method is suitable for finding (optimal) solutions to problem instances in which the value of the associated playing set cardinality is relatively small and the parameters of the problem are relatively large ($m > 10$). In contrast, it was found that the mathematical programming solution method is a better solution method for problem instances when an optimal solution is sought in which the playing set cardinality is relatively large, but the parameters in the problem are small ($m \leq 10$).

In Chapter 6, various metaheuristic approaches towards solving the incomplete lottery problem and the resource utilisation problem were presented. These solution methods involve the distribution of the elements in an overlapping playing set structure amongst its compartments. It was shown that this approach may be used to find previously unknown (optimal) solutions to the incomplete lottery problem and the resource utilisation problem, such as the novel result that $L_1(20, 10, 4, 3) = 6$.

It was shown in Chapter 7 that the event of 18 (or more) participants winning a draw of the South African National Lottery, *Lotto*, is possible and indeed likely under the assumption that participants *do not* select the numbers in their playing sets randomly according to a uniform distribution. The probability $P(N, \lambda)$ of λ participants winning a lottery draw concurrently was also computed for various lottery problem instances and it was shown how this value may be used to compute a participant's expected return per lottery draw.

8.3 Future work

This section contains a list of options for future work involving improvements or elaborations based on the work done in this thesis:

1. As mentioned, the difference between the upper bound on the incomplete lottery number (3.17) and the actual value of $L_\psi(m, n, t, k)$ is large for small values of ψ and it is non-increasing as ψ increases. Therefore scope for future work exists with respect to the formulation of an upper bound on the incomplete lottery number in which the difference between the upper bound and $L_\psi(m, n, t, k)$ is closer to zero for small values of ψ.

2. From Chapter 3, the bound in (3.17) does not improve on the recursive bound, (2.12), by Li & van Rees [26]. Therefore, further scope with respect to future work involves the formulation of a bound which improves on that bound by Li & van Rees.

3. In Chapter 4 it was found that a major drawback of the mathematical programming approach is the large execution time required to find values of $L_\psi(m, n, t, k)$ or $\Psi_\ell(m, n, t, k)$ for large lottery problem instances on a single computer. Therefore, scope for future work entails the use of a large number of computers in parallel for solving the incomplete lottery and resource utilisation problems for larger problem instances upon setting a large set of variables explicitly.

4. Another major drawback of the mathematical programming solution approach in Chapter 4 is that a file containing the adjacency matrix used in the integer programming formulations is required to be stored in memory. For large problem instances, it is not practically feasible to store this file in memory. Possible future work may involve investigations into the possible compression of the size of the adjacency matrix, or a more efficient storage of the information contained in the matrix.

5. In Chapter 5, a major drawback of the exhaustive enumeration lottery tree solution method was found to be its execution time which increased exponentially as the depth of the tree increases. This limits the solution method to only being able to achieve a solution within a feasible time frame if the cardinality of the playing set is small. Scope for future work exists with regards to investigations into the incorporation of further pruning rules, such as those described in §5.3.1, which may lead to a decrease the number of nodes constructed in the exhaustive enumeration lottery tree.

6. In Chapter 6 methods were described which involved determining bounds on the incomplete lottery number and the resource utilisation number algorithmically. It was found that, by applying these methods to certain lottery instances, previously unknown incomplete lottery and resource utilisation numbers may be found. Future work may entail using these or similar metaheuristic solution methods, or improvements thereof, in order to find currently unknown incomplete lottery, and resource utilisation numbers.

7. In Chapter 7, the value of $P(N, \lambda)$ was computed for the lottery $\langle 49, 6, 6, 6\rangle$. Using Algorithm 7.1, it was found that an investigation into the value of $P(N, \lambda)$ for the lottery $\langle 49, 6, 6, k\rangle$ $(3 \leq k < 6)$ was too computationally expensive. Scope for future work involves formulating improved methods of computing or estimating the value of $P(N, \lambda)$ for the lottery $\langle 49, 6, 6, k\rangle$ $(3 \leq k < 6)$.

8. Chapter 7 also included an investigation into how participants select the numbers included in their playing sets. It is not known exactly how participants select the numbers in their playing sets because, in most countries, that information is classified as confidential. Therefore, assumptions were made in Chapter 7, based on findings in the literature, as to how lottery participants choose their playing sets. Simon Cox, from the University of Southampton, analysed data originating from 113 draws of the UK lottery and compared the winning numbers from each draw with the number of participant tickets which had four, five or six numbers in common with that winning ticket [41]. From that analysis, he was able to determine which numbers appear to be most popular amongst the participants of the UK lottery. Scope for future work exists with regards to a similar investigation to the one conducted by Cox, and using the results from that investigation to reconstruct the N tickets purchased in simulations of the South African National Lottery draw. This may provide a more insightful analysis of the value of $P(N, \lambda)$.

References

[1] Arnautov VI, 1947, *Estimation of the exterior stability number of a graph by means of the minimal degree of the vertices* (Russian), Prikladnaya Matematika i Programmirovanie, **11**, pp. 3–8.

[2] Astratian AS, Denley TMJ & Häggkvist R, 1998, *Bipartite graphs and their applications*, Cambridge University Press, Cambridge.

[3] Bate JA, 1978, *A generalized covering problem*, PhD Thesis, University of Manitoba, Manitoba.

[4] Brouwer AE & Voorhoeve M, 1979, *Turan theory and the lotto problem*, Mathematical Center Tracts, **106**, pp. 99–105.

[5] Burger AP, Gründlingh WR & Van Vuuren JH, 2004, *Towards a characterisation of lottery set overlapping structures*, Ars Combinatoria, **84**, pp. 105–128.

[6] Burger AP, Gründlingh WR & Van Vuuren JH, 2006, *Two combinatorial problems involving lottery schemes: Algorithmic determination of solution sets*, Utilitas Mathematica, **70**, pp. 33–70.

[7] Caen D, 1983, *Extension of a theorem of Moon and Moser on complete subgraphs*, Ars Combinatoria, **16**, pp. 5–10.

[8] Caprara A, Toth P & Fischcetti M, 1998, *Algorithms for the set covering problem*, Annals of Operations Research, **98**, pp. 352–371.

[9] Caro Y & Roditty Y, 1985, *On the vertex-independence number and star decomposition of graphs*, Ars Combinatoria, **20**, pp. 167–180.

[10] Central intelligence agency, 2009, *The world factbook*, [Online], [Cited October 1st, 2009], Available from `https://www.cia.gov/library/publications/the-world-factbook/rankorder/2102rank.html`

[11] Clark WE, Fisher DC, Shekhtman B & Suen S, 1998, *Upper bounds for the domination number of a graph*, Congressus Numerantium, **132**, pp. 99–123.

[12] Encyclopædia britannica, 2009, *Lottery*, [Online], [Cited October 1st, 2009], Available from `http://www.britannica.com/EBchecked/topic/348555/lottery`

[13] Gidani, 2009, *Gidani, the National Lottery's operator*, [Online], [Cited October 1st, 2009], Available from `http://www.gidani.co.za/home.asp`

[14] Gilfillan D, 2009, *18 Winners suggests a lotto scam*, Cape Argus, 16 February, p. 14.

[15] Gründlingh WR, 2009, Past student of *Stellenbosch University*, [Personal Communication], Contactable at wgrundlingh@gmail.com.

[16] Gründlingh WR, 2004, *Two new combinatorial problems involving dominating sets for lottery schemes*, PhD Thesis, University of Stellenbosch, Stellenbosch.

[17] GTECH Corporation, 2009, *Lottery*, [Online], [Cited October 1st, 2009], Available from http://www. gtech.com/products_services/lottery.asp

[18] Haigh J, 2007, *The statistics of the National Lottery*, Journal of the Royal Statistical Society (Series A), **160(2)**, pp. 187–206.

[19] Henze N & Riedwyl H, 1998, *How to win more: Strategies for increasing a lottery win*, Illustrated Edition, A K Peters Ltd, Natick (MA).

[20] Jans R & Degraeve Z, 2007, *A note on a symmetrical set covering problem: The lottery problem*, European Journal of Operations Research, **186**, pp. 104–110.

[21] Johnson JL, 2008, *Probability and statistics for computer science*, Illustrated Edition, Wiley-IEEE, Hoboken (NJ).

[22] Li PC, 1999, *Some results on lotto designs*, PhD Thesis, University of Manitoba, Manitoba.

[23] Li PC, 2007, *Ben Li's lotto tables page*, [Online], [Cited October 16th, 2009], Available from http://www.cs.umanitoba.ca/∼lipakc/lottotables.html

[24] Li PC & Van Rees J, 1999, *Lower bounds on lotto designs*, Congressus Numerantium, **141**, pp. 5–30.

[25] Li PC & Van Rees J, 2002, *Lotto design tables*, Journal of Combinatorial Designs, **10**, pp. 335–359.

[26] Li PC & Van Rees J, 2004, *New constructions of lotto designs*, Utilitas Mathematica, **58**, pp. 45–64.

[27] Li PC & Van Rees J, 2007, *Lotto designs*, pp. 512–519 in Colbourn CJ & Dinitz JH (EDS), *Handbook of combinatorial designs*, CRC Press, Boca Raton (FL).

[28] LINDO Systems, 2008, *LINGO 11.0 — Optimization modeling software for linear, nonlinear, and integer programming*, [Online], [Cited October 21st, 2009], Available from http://www.lindo.com

[29] Marcu D, 1986, *An upperbound on the domination number of a graph*, Mathematica Scandinavica, **59(1)**, pp. 41–44.

[30] McCaffrey J, *Using permutations in .NET for improved systems security*, [Online], [Cited October 1st, 2009], Available from http://msdn.microsoft.com/en-us/library /aa302371.aspx

[31] National Lottery, 2009, *Facts and figures*, [Online], [Cited October 1st, 2009], Available from http://www.nationallottery.co.za/lotto_home/facts_figures.asp

[32] National Lottery, 2009, *Lotto results*, [Online], [Cited October 1st, 2009], Available from http://www.nationallottery.co.za/lotto_home/results.asp?type=1

[33] Nurmela KJ & Östergård PRJ, 1993, *Upper bounds for covering designs by simulated annealing*, Congressus Numerantuim, **96**, pp 93–111.

[34] Payan C, 1975, *Sur le nombre d'absorption d'un graphe simple* (French), Cahiers du Centre d'Etudes de Recherche Operationnelle, **17**, pp. 307–317.

[35] Ray R, 2008, *The history of the lottery*, [Online], [Cited October 1st, 2009], Available from `http://www.winningwithnumbers.com/lottery/history/`

[36] Reed BA, 1996, *Paths, stars and the number three*, Combinatorics, Probability and Computing, **5(3)**, pp. 277–295.

[37] Ross S, 2002, *A first course in probability*, 6th Edition, Prentice-Hall, Upper Saddle River (NJ).

[38] Schönheim J, 1964, *On coverings*, Pacific Journal of Mathematics, **14**, pp. 1405–1411.

[39] Schrijver A, 1979, *Packing and covering in combinatorics*, Mathematisch Centrum, Amsterdam.

[40] Soanes C & Stevenson A, 2003, *Oxford dictionary of English*, 2nd Edition, Oxford University Press, Oxford.

[41] Thomson H, 2009, *What's luck got to do with it?*, New Scientist, August, pp 35–39.

[42] Tromp B, 2009, *Probe into Lotto player's luck*, The Star, 16 February, p. 3.

[43] Wikipedia, 2009, *Lottery*, [Online], [Cited October 1st, 2009], Available from `http://en.wikipedia.org/wiki/Lottery`

[44] Wikipedia, 2009, *South African National Lottery*, [Online], [Cited October 1st, 2009], Available from `http://en.wikipedia.org/wiki/South_African_National_Lottery`

[45] Zaman A & Marsaglia G, 1990. *Random selection of subsets with specified element probabilities*, Communications in Statistics: Theory and Methods, **19**, pp. 4419–4434.

APPENDIX A

Some known lottery numbers

TABLE A.1: *An excerpt from Ben Li's lotto tables page [23]. In this table, all known complete lottery numbers, denoted by the symbol L_1, are presented for lotteries in which $m > \{n, t\} > k$ and $5 \leq m \leq 10$.*

m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1
5	3	3	2	2	8	5	5	3	2	9	5	8	3	1	10	3	9	2	1	10	7	7	5	3
5	3	4	2	1	8	5	5	4	5	9	5	8	4	1	10	4	3	2	4	10	7	7	6	8
5	4	3	2	1	8	5	6	2	1	9	6	3	2	2	10	4	4	2	3	10	7	8	2	1
5	4	4	2	1	8	5	6	3	1	9	6	4	2	2	10	4	4	3	14	10	7	8	3	1
5	4	4	3	1	8	5	6	4	3	9	6	4	3	3	10	4	5	2	2	10	7	8	4	1
6	3	3	2	2	8	5	7	2	1	9	6	5	2	1	10	4	5	3	7	10	7	8	5	1
6	3	4	2	2	8	5	7	3	1	9	6	5	3	2	10	4	6	2	2	10	7	8	6	3
6	3	5	2	1	8	5	7	4	1	9	6	5	4	3	10	4	6	3	4	10	7	9	2	1
6	4	3	2	2	8	6	3	2	2	9	6	6	2	1	10	4	7	2	2	10	7	9	3	1
6	4	4	2	1	8	6	4	2	1	9	6	6	3	1	10	4	7	3	2	10	7	9	4	1
6	4	4	3	3	8	6	4	3	3	9	6	6	4	3	10	4	8	2	1	10	7	9	5	1
6	4	5	2	1	8	6	5	2	1	9	6	6	5	7	10	4	8	3	2	10	7	9	6	1
6	4	5	3	1	8	6	5	3	1	9	6	7	2	1	10	4	9	2	1	10	8	3	2	2
6	5	3	2	1	8	6	5	4	3	9	6	7	3	1	10	4	9	3	1	10	8	4	2	1
6	5	4	2	1	8	6	6	2	1	9	6	7	4	1	10	5	3	2	2	10	8	4	3	3
6	5	4	3	1	8	6	6	3	1	9	6	7	5	3	10	5	4	2	2	10	8	5	2	1
6	5	5	2	1	8	6	6	4	1	9	6	8	2	1	10	5	4	3	7	10	8	5	3	1
6	5	5	3	1	8	6	6	5	4	9	6	8	3	1	10	5	5	2	2	10	8	5	4	3
6	5	5	4	1	8	6	7	2	1	9	6	8	4	1	10	5	5	3	2	10	8	6	2	1
7	3	3	2	4	8	6	7	3	1	9	6	8	5	1	10	5	5	4	14	10	8	6	3	1
7	3	4	2	2	8	6	7	4	1	9	7	3	2	2	10	5	6	2	2	10	8	6	4	1
7	3	5	2	2	8	6	7	5	1	9	7	4	2	1	10	5	6	3	2	10	8	6	5	4
7	3	6	2	1	8	7	3	2	1	9	7	4	3	3	10	5	6	4	7	10	8	7	2	1
7	4	3	2	2	8	7	4	2	1	9	7	5	2	1	10	5	7	2	1	10	8	7	3	1
7	4	4	2	2	8	7	4	3	1	9	7	5	3	1	10	5	7	3	2	10	8	7	4	1
7	4	4	3	4	8	7	5	2	1	9	7	5	4	3	10	5	7	4	2	10	8	7	5	1
7	4	5	2	1	8	7	5	3	1	9	7	6	2	1	10	5	8	2	1	10	8	7	6	4
7	4	5	3	2	8	7	5	4	1	9	7	6	3	1	10	5	8	3	1	10	8	8	2	1
7	4	6	2	1	8	7	6	2	1	9	7	6	4	1	10	5	8	4	2	10	8	8	3	1
7	4	6	3	1	8	7	6	3	1	9	7	6	5	4	10	5	9	2	1	10	8	8	4	1
7	5	3	2	2	8	7	6	4	1	9	7	7	2	1	10	5	9	3	1	10	8	8	5	1
7	5	4	2	1	8	7	6	5	1	9	7	7	3	1	10	5	9	4	1	10	8	8	6	1

Table A.1 – continued from previous page

m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1	m	n	t	k	L_1
7	5	4	3	3	8	7	7	2	1	9	7	7	4	1	10	6	3	2	2	10	8	8	7	5
7	5	5	2	1	8	7	7	3	1	9	7	7	5	1	10	6	4	2	2	10	8	9	2	1
7	5	5	3	1	8	7	7	4	1	9	7	7	6	4	10	6	4	3	4	10	8	9	3	1
7	5	5	4	3	8	7	7	5	1	9	7	8	2	1	10	6	5	2	2	10	8	9	4	1
7	5	6	2	1	8	7	7	6	1	9	7	8	3	1	10	6	5	3	2	10	8	9	5	1
7	5	6	3	1	9	3	3	2	7	9	7	8	4	1	10	6	5	4	7	10	8	9	6	1
7	5	6	4	1	9	3	4	2	3	9	7	8	5	1	10	6	6	2	1	10	8	9	7	1
7	6	3	2	1	9	3	5	2	3	9	7	8	6	1	10	6	6	3	2	10	9	3	2	1
7	6	4	2	1	9	3	6	2	2	9	8	3	2	1	10	6	6	4	3	10	9	4	2	1
7	6	4	3	1	9	3	7	2	2	9	8	4	2	1	10	6	6	5	14	10	9	4	3	1
7	6	5	2	1	9	3	8	2	1	9	8	4	3	1	10	6	7	2	1	10	9	5	2	1
7	6	5	3	1	9	4	3	2	4	9	8	5	2	1	10	6	7	3	1	10	9	5	3	1
7	6	5	4	1	9	4	4	2	2	9	8	5	3	1	10	6	7	4	2	10	9	5	4	1
7	6	6	2	1	9	4	4	3	9	9	8	5	4	1	10	6	7	5	4	10	9	6	2	1
7	6	6	3	1	9	4	5	2	2	9	8	6	2	1	10	6	8	2	1	10	9	6	3	1
7	6	6	4	1	9	4	5	3	5	9	8	6	3	1	10	6	8	3	1	10	9	6	4	1
7	6	6	5	1	9	4	6	2	2	9	8	6	4	1	10	6	8	4	1	10	9	6	5	1
8	3	3	2	5	9	4	6	3	2	9	8	6	5	1	10	6	8	5	3	10	9	7	2	1
8	3	4	2	3	9	4	7	2	1	9	8	7	2	1	10	6	9	2	1	10	9	7	3	1
8	3	5	2	2	9	4	7	3	2	9	8	7	3	1	10	6	9	3	1	10	9	7	4	1
8	3	6	2	2	9	4	8	2	1	9	8	7	4	1	10	6	9	4	1	10	9	7	5	1
8	3	7	2	1	9	4	8	3	1	9	8	7	5	1	10	6	9	5	1	10	9	7	6	1
8	4	3	2	2	9	5	3	2	2	9	8	7	6	1	10	7	3	2	2	10	9	8	2	1
8	4	4	2	2	9	5	4	2	2	9	8	8	2	1	10	7	4	2	2	10	9	8	3	1
8	4	4	3	6	9	5	4	3	5	9	8	8	3	1	10	7	4	3	3	10	9	8	4	1
8	4	5	2	2	9	5	5	2	2	9	8	8	4	1	10	7	5	2	1	10	9	8	5	1
8	4	5	3	2	9	5	5	3	2	9	8	8	5	1	10	7	5	3	2	10	9	8	6	1
8	4	6	2	1	9	5	5	4	9	9	8	8	6	1	10	7	5	4	3	10	9	8	7	1
8	4	6	3	2	9	5	6	2	1	9	8	8	7	1	10	7	6	2	1	10	9	9	2	1
8	4	7	2	1	9	5	6	3	2	10	3	3	2	8	10	7	6	3	1	10	9	9	3	1
8	4	7	3	1	9	5	6	4	4	10	3	4	2	5	10	7	6	4	3	10	9	9	4	1
8	5	3	2	2	9	5	7	2	1	10	3	5	2	3	10	7	6	5	5	10	9	9	5	1
8	5	4	2	2	9	5	7	3	1	10	3	6	2	3	10	7	7	2	1	10	9	9	6	1
8	5	4	3	3	9	5	7	4	2	10	3	7	2	2	10	7	7	3	1	10	9	9	7	1
8	5	5	2	1	9	5	8	2	1	10	3	8	2	2	10	7	7	4	1	10	9	9	8	1

APPENDIX B

Incomplete lottery problem modelled as an ILP

TABLE B.1: *Results obtained when solving for the value $L_\psi(m, n, t, k)$ when the incomplete lottery problem is formulated as an ILP. The first column contains the value of the required probability-of-win value, ψ. The second column contains an example of a playing set which at least achieves the required probability-of-win value. The third column contains the actual probability of win value, ψ', associated with the playing set in the second column. The fourth column contains the associated number of branches in the branch-and-bound tree. Finally, the fifth column contains the time in seconds required to find the solution.*

ψ	Playing set	ψ'	Branches	Time
	$\langle 6,3,3,2\rangle$			
0.5	$\{\{1,2,3\}\}$	0.5	0	0.019
1	$\{\{1,4,6\},\{2,3,5\}\}$	1	0	0.009
	$\langle 7,3,3,2\rangle$			
0.3	$\{\{1,2,3\}\}$	0.371	0	0.029
0.7	$\{\{1,4,5\},\{2,3,7\}\}$	0.743	0	0.039
0.9	$\{\{1,2,4\},\{2,4,5\},\{3,6,7\}\}$	0.914	10	0.109
1	$\{\{1,2,3\},\{1,3,7\},\{2,3,7\},\{4,5,6\}\}$	1	9	0.199
	$\langle 7,3,4,2\rangle$			
0.6	$\{\{1,2,3\}\}$	0.629	0	0.029
1	$\{\{1,4,5\},\{2,3,6\}\}$	1	0	0.009
	$\langle 8,3,3,2\rangle$			
0.2	$\{\{1,2,3\}\}$	0.286	0	0.029
0.5	$\{\{1,7,8\},\{2,4,5\}\}$	0.571	0	0.049
0.7	$\{\{1,2,7\},\{2,6,8\},\{3,4,5\}\}$	0.786	4	0.109
0.8	$\{\{1,2,3\},\{1,5,7\},\{4,6,8\},\{5,7,8\}\}$	0.857	151	1.869
1	$\{\{1,3,5\},\{1,3,6\},\{1,3,8\},\{2,4,7\},\{5,6,8\}\}$	1	11	0.389
	$\langle 8,3,4,2\rangle$			
0.5	$\{\{1,2,3\}\}$	0.5	0	0.029
0.8	$\{\{1,2,4\},\{3,6,7\}\}$	0.871	0	0.099
1	$\{\{1,2,3\},\{4,5,8\},\{4,6,7\}\}$	1	0	0.079
	$\langle 8,3,5,2\rangle$			
0.7	$\{\{1,2,3\}\}$	0.714	0	0.069
1	$\{\{1,3,5\},\{2,4,7\}\}$	1	0	0.099
	$\langle 8,4,3,2\rangle$			
0.5	$\{\{1,2,3,4\}\}$	0.5	0	0.069
1	$\{\{1,2,4,8\},\{3,5,6,7\}\}$	1	0	0.079

Table B.1 – continued from previous page

ψ	Playing set	ψ'	Branches	Time
	$\langle 8,4,4,2\rangle$			
0.7	$\{\{1,2,3,4\}\}$	0.757	0	0.109
1	$\{\{1,3,7,8\},\{2,3,4,5\}\}$	1	0	0.069
	$\langle 8,4,4,3\rangle$			
0.2	$\{\{1,2,3,4\}\}$	0.243	0	0.079
0.4	$\{\{1,2,5,6\},\{1,3,4,7\}\}$	0.486	0	0.059
0.6	$\{\{1,3,5,6\},\{2,4,5,6\},\{3,4,7,8\}\}$	0.671	31	0.169
0.8	$\{\{1,4,5,6\},\{1,5,6,8\},\{2,3,4,8\},\{2,3,6,7\}\}$	0.8	17	0.219
0.9	$\{\{1,2,3,5\},\{1,4,6,7\},\{2,3,7,8\},\{2,4,5,6\},$ $\{3,5,7,8\}\}$	0.9	315	2.979
1	$\{\{1,2,3,5\},\{1,4,5,6\},\{1,4,7,8\},\{2,3,6,7\},$ $\{2,3,6,8\},\{4,5,7,8\}\}$	1	221	3.189
	$\langle 9,3,3,2\rangle$			
0.2	$\{\{1,2,3\}\}$	0.226	0	0.049
0.4	$\{\{1,2,3\},\{1,4,5\}\}$	0.405	0	0.079
0.6	$\{\{1,3,5\},\{2,3,7\},\{4,6,9\}\}$	0.631	9	0.319
0.7	$\{\{2,3,7\},\{4,5,6\},\{4,5,8\},\{6,8,9\}\}$	0.726	27	0.249
0.8	$\{\{1,2,3\},\{2,4,7\},\{4,8,9\},\{5,6,8\},\{5,7,9\}\}$	0.869	1 870 660	14 400
0.9	$\{\{1,4,6\},\{1,4,9\},\{2,3,8\},\{2,5,7\},\{3,7,8\},$ $\{5,6,9\}\}$	0.952	1 628 839	14 400
1	$\{\{1,3,9\},\{1,7,9\},\{2,4,8\},\{2,5,8\},\{2,6,8\},$ $\{3,7,8\},\{4,5,6\}\}$	1	1245	11.849
	$\langle 9,3,4,2\rangle$			
0.4	$\{\{1,2,3\}\}$	0.405	0	0.219
0.7	$\{\{1,2,4\},\{5,7,8\}\}$	0.738	5	0.529
1	$\{\{1,3,4\},\{2,5,9\},\{6,7,8\}\}$	1	0	0.169
	$\langle 9,3,5,2\rangle$			
0.5	$\{\{1,2,3\}\}$	0.595	0	0.109
0.9	$\{\{2,5,9\},\{6,7,8\}\}$	0.929	2	0.599
1	$\{\{1,3,4\},\{2,5,9\},\{2,6,8\}\}$	1	33	2.979
	$\langle 9,3,6,2\rangle$			
0.7	$\{\{1,2,3\}\}$	0.774	0	0.109
1	$\{\{1,3,4\},\{2,5,9\}\}$	1	0	0.119
	$\langle 9,4,3,2\rangle$			
0.4	$\{\{1,2,3,4\}\}$	0.405	0	0.089
0.8	$\{\{1,2,3,5\},\{4,6,7,8\}\}$	0.810	0	0.189
0.9	$\{\{1,5,8,9\},\{2,3,4,6\},\{6,7,8,9\}\}$	0.929	3	0.429
1	$\{\{1,2,3,9\},\{1,4,5,8\},\{2,3,8,9\},\{4,5,6,7\}\}$	1	800	9.329
	$\langle 9,4,4,2\rangle$			
0.6	$\{\{1,2,3,4\}\}$	0.643	0	0.189
1	$\{\{1,2,6,9\},\{3,5,7,8\}\}$	1	0	0.379
	$\langle 9,4,4,3\rangle$			
0.1	$\{\{1,2,3,4\}\}$	0.167	0	1.949
0.3	$\{\{1,2,5,6\},\{1,3,4,8\}\}$	0.333	0	1.669
0.5	$\{\{1,2,3,9\},\{1,4,6,7\},\{4,5,8,9\}\}$	0.5	0	0.429
0.6	$\{\{1,2,4,9\},\{1,3,5,9\},\{2,3,7,8\},\{3,4,6,8\}\}$	0.603	61	1.179
0.7	$\{\{1,2,5,9\},\{1,3,6,7\},\{1,6,8,9\},\{2,3,4,8\},$	0.738	420	2.039

Table B.1 – continued from previous page

ψ	Playing set	ψ'	Branches	Time
	$\{4,5,6,7\}\}$			
0.8	$\{\{1,2,7,8\},\{1,4,5,8\},\{1,4,7,9\},\{2,3,4,8\},$ $\{2,5,6,9\},\{3,5,6,7\}\}$	0.825	781 103	14 400
0.9	$\{\{1,2,4,9\},\{1,3,5,7\},\{1,6,7,8\},\{2,3,4,8\},$ $\{2,3,6,9\},\{4,5,6,9\},\{5,7,8,9\}\}$	0.904	886 618	14 400
1	$\{\{1,2,5,9\},\{1,3,4,5\},\{1,3,6,7\},\{1,3,6,8\},$ $\{2,3,5,9\},\{2,4,6,9\},\{2,4,7,8\},\{4,7,8,9\},$ $\{5,6,7,8\}\}$	1	41 491	4 256.259
	$\langle 9,4,5,2\rangle$			
0.8	$\{\{1,2,3,4\}\}$	0.833	0	0.369
1	$\{\{1,3,6,9\},\{2,5,6,7\}\}$	1	0	0.269
	$\langle 9,4,5,3\rangle$			
0.3	$\{\{1,2,3,4\}\}$	0.357	0	0.249
0.7	$\{\{1,4,7,8\},\{3,5,6,9\}\}$	0.714	0	0.289
0.8	$\{\{1,2,3,4\},\{1,2,3,5\},\{6,7,8,9\}\}$	0.857	187	1.049
0.9	$\{\{1,2,3,6\},\{1,3,4,8\},\{2,5,6,8\},\{4,5,7,9\}\}$	0.968	37 610	776.399
1	$\{\{1,2,3,8\},\{1,2,4,5\},\{2,3,5,8\},\{3,6,7,9\},$ $\{6,7,8,9\}\}$	1	5 750	162.759
	$\langle 9,4,6,3\rangle$			
0.5	$\{\{1,2,3,4\}\}$	0.595	0	0.109
1	$\{\{1,3,6,9\},\{4,5,7,8\}\}$	1	0	3.779
	$\langle 10,3,3,2\rangle$			
0.1	$\{\{1,2,3\}\}$	0.183	0	0.199
0.3	$\{\{1,9,10\},\{2,5,7\}\}$	0.367	0	0.169
0.5	$\{\{2,7,9\},\{3,4,5\},\{6,8,10\}\}$	0.55	1	0.249
0.6	$\{\{1,2,3\},\{1,8,9\},\{4,5,9\},\{6,7,8\}\}$	0.633	1	0.239
0.7	$\{\{1,2,5\},\{1,3,9\},\{2,3,8\},\{4,6,9\},$ $\{6,7,10\}\}$	0.758	471 987	14 400
0.8	$\{\{1,4,10\},\{1,5,7\},\{2,3,8\},\{3,4,9\},$ $\{5,6,10\},\{6,7,9\}\}$	0.85	433 376	14 400
0.9	$\{\{1,2,3\},\{1,3,9\},\{2,3,9\},\{4,5,8\},\{4,6,7\},$ $\{5,7,10\},\{6,8,10\}\}$	0.9	753 363	14 400
1	$\{\{1,2,10\},\{1,3,10\},\{1,7,10\},\{2,3,7\},$ $\{4,5,6\},\{4,5,8\},\{4,5,9\},\{6,8,9\}\}$	1	1 317	35.090
	$\langle 10,3,4,2\rangle$			
0.3	$\{\{1,2,3\}\}$	0.333	0	0.439
0.6	$\{\{1,2,6\},\{4,8,9\}\}$	0.624	143	1.419
0.8	$\{\{1,4,7\},\{2,3,6\},\{8,9,10\}\}$	0.871	376	4.269
0.9	$\{\{1,2,9\},\{3,4,7\},\{3,6,10\},\{5,8,10\}\}$	0.943	242 745	14 400
1	$\{\{1,3,4\},\{1,3,5\},\{1,4,5\},\{2,7,10\},$ $\{6,8,9\}\}$	1	1 193	84.389
	$\langle 10,3,5,2\rangle$			
0.5	$\{\{1,2,3\}\}$	0.5	0	0.229
0.8	$\{\{2,4,10\},\{7,8,9\}\}$	0.833	475	10.749
1	$\{\{1,6,10\},\{2,4,8\},\{3,5,9\}\}$	1	0	2.709
	$\langle 10,3,6,2\rangle$			
0.6	$\{\{1,2,3\}\}$	0.667	0	0.549

Table B.1 – continued from previous page

ψ	Playing set	ψ'	Branches	Time
0.9	$\{\{1,5,6\},\{4,8,9\}\}$	0.957	1	1.419
1	$\{\{1,2,5\},\{1,3,4\},\{7,9,10\}\}$	1	53	11.249
	$\langle 10,3,7,2\rangle$			
0.8	$\{\{1,2,3\}\}$	0.817	0	0.199
1	$\{\{1,3,4\},\{2,7,10\}\}$	1	0	0.309
	$\langle 10,4,3,2\rangle$			
0.3	$\{\{1,2,3,4\}\}$	0.333	0	0.239
0.6	$\{\{1,3,4,9\},\{2,5,6,10\}\}$	0.667	0	0.529
0.8	$\{\{1,3,4,9\},\{2,6,7,10\},\{4,5,6,8\}\}$	0.85	5	0.859
1	$\{\{1,2,3,5\},\{4,6,8,9\},\{4,7,8,10\},\{6,7,9,10\}\}$	1	0	0.719
	$\langle 10,4,4,2\rangle$			
0.5	$\{\{1,2,3,4\}\}$	0.548	0	0.809
0.9	$\{\{1,2,6,9\},\{3,5,7,8\}\}$	0.924	91	4.459
1	$\{\{1,2,3,8\},\{2,4,5,6\},\{2,7,9,10\}\}$	1	65	90.379
	$\langle 10,4,4,3\rangle$			
0.1	$\{\{1,2,3,4\}\}$	0.119	0	0.299
0.2	$\{\{1,2,3,4\},\{4,6,7,10\}\}$	0.238	0	0.689
0.3	$\{\{1,3,4,7\},\{2,5,6,9\},\{6,7,8,10\}\}$	0.357	0	0.759
0.4	$\{\{1,3,7,8\},\{1,5,6,10\},\{2,6,8,9\},\{3,4,5,9\}\}$	0.476	0	0.809
0.5	$\{\{1,2,7,8\},\{1,3,5,8\},\{1,4,6,7\},\{2,3,4,9\},$ $\{2,4,6,8\}\}$	0.5	37	1.089
0.6	$\{\{1,2,5,6\},\{1,3,4,9\},\{1,5,7,9\},\{2,3,7,8\},$ $\{4,6,7,8\},\{6,7,9,10\}\}$	0.619	181	1.589
0.7	$\{\{1,2,3,4\},\{1,2,7,8\},\{1,4,5,6\},\{1,6,9,10\},$ $\{2,5,8,9\},\{3,6,7,8\},\{4,7,9,10\}\}$	0.719	117 214	14 400
0.8	$\{\{1,2,3,4\},\{1,3,5,10\},\{1,6,7,8\},\{2,5,7,9\},$ $\{2,6,8,10\},\{3,6,7,9\},\{4,5,6,9\},\{4,7,8,10\}\}$	0.81	148 392	14 400
0.9	$\{\{1,2,4,9\},\{1,3,9,10\},\{1,4,5,8\},\{1,7,8,10\},$ $\{2,3,5,7\},\{2,3,6,8\},\{2,5,6,10\},\{3,4,6,7\},$ $\{4,8,9,10\},\{5,6,7,9\}\}$	0.924	117 420	14 400
1	$\{\{1,2,3,4\},\{1,2,4,5\},\{1,5,7,9\},\{1,6,7,10\},$ $\{1,8,9,10\},\{2,3,7,8\},\{2,4,6,8\},\{2,5,9,10\},$ $\{2,6,7,9\},\{3,4,6,9\},\{3,4,7,10\},\{3,5,6,8\},$ $\{3,5,6,10\},\{4,5,7,8\}\}$	1	178 558	14 400
	$\langle 10,4,5,2\rangle$			
0.7	$\{\{1,2,3,4\}\}$	0.738	0	0.719
1	$\{\{1,3,8,9\},\{2,4,7,10\}\}$	1	0	5.249
	$\langle 10,4,5,3\rangle$			
0.2	$\{\{1,2,3,4\}\}$	0.262	0	0.390
0.5	$\{\{1,2,8,9\},\{3,4,6,7\}\}$	0.524	0	0.790
0.7	$\{\{1,5,6,10\},\{2,3,6,8\},\{4,7,9,10\}\}$	0.714	399	11.780
0.8	$\{\{1,2,4,8\},\{2,3,7,9\},\{3,8,9,10\},\{4,5,6,10\}\}$	0.825	366	10.110
0.9	$\{\{1,2,4,9\},\{1,5,6,10\},\{2,3,8,10\},\{3,4,7,9\},$ $\{5,6,7,8\}\}$	0.937	86 679	14 400
1	$\{\{1,2,3,6\},\{1,5,7,9\},\{2,4,8,10\},\{3,4,5,9\},$ $\{3,7,8,10\},\{4,6,7,8\},\{5,6,9,10\}\}$	1	103 263	14 400
	$\langle 10,4,6,2\rangle$			

Table B.1 – continued from previous page

ψ	Playing set	ψ'	Branches	Time
0.8	$\{\{1,2,3,4\}\}$	0.881	0	0.619
1	$\{\{1,2,3,10\},\{2,4,5,6\}\}$	1	0	2.579
	$\langle 10,4,6,3\rangle$			
0.4	$\{\{1,2,3,4\}\}$	0.452	0	0.419
0.8	$\{\{1,5,6,7\},\{3,8,9,10\}\}$	0.829	0	1.469
0.9	$\{\{1,4,9,10\},\{2,5,9,10\},\{3,6,7,8\}\}$	0.971	3 767	2 128.359
1	$\{\{1,2,3,5\},\{1,2,3,6\},\{2,4,5,6\},\{7,8,9,10\}\}$	1	4 442	1 063.409
	$\langle 10,4,7,3\rangle$			
0.6	$\{\{1,2,3,4\}\}$	0.667	0	0.489
1	$\{\{1,3,4,7\},\{2,8,9,10\}\}$	1	0	11.459
	$\langle 10,5,3,2\rangle$			
0.5	$\{\{1,2,3,4,5\}\}$	0.5	0	0.219
1	$\{\{1,4,6,8,10\},\{2,3,5,7,9\}\}$	1	0	0.199
	$\langle 10,5,4,2\rangle$			
0.7	$\{\{1,2,3,4,5\}\}$	0.738	0	0.609
1	$\{\{1,3,4,8,10\},\{5,6,7,9,10\}\}$	1	0	50.979
	$\langle 10,5,4,3\rangle$			
0.2	$\{\{1,2,3,4,5\}\}$	0.262	0	1.040
0.5	$\{\{1,2,3,4,8\},\{2,5,6,7,9\}\}$	0.523	2	1.790
0.7	$\{\{1,3,4,6,9\},\{1,5,6,8,10\},\{2,4,7,9,10\}\}$	0.7	1	2.339
0.8	$\{\{1,3,7,8,10\},\{1,4,5,6,9\},\{2,3,4,5,7\},$ $\{2,3,5,8,10\}\}$	0.8	71	8.979
0.9	$\{\{1,3,4,5,6\},\{1,5,8,9,10\},\{2,3,7,8,9\},$ $\{2,4,6,7,8\},\{2,4,6,7,10\}\}$	0.919	51 446	14 400
1	$\{\{1,2,6,8,9\},\{1,3,4,5,7\},\{1,3,5,7,10\},$ $\{2,3,4,5,8\},\{2,3,4,8,10\},\{2,6,7,8,9\}$ $\{4,5,6,9,10\}\}$	1	117 287	14 400
	$\langle 10,5,5,2\rangle$			
0.8	$\{\{1,2,3,4,5\}\}$	0.897	0	0.909
1	$\{\{1,2,3,5,7\},\{1,3,4,6,10\}\}$	1	0	3.159
	$\langle 10,5,5,3\rangle$			
0.5	$\{\{1,2,3,4,5\}\}$	0.5	0	0.469
1	$\{\{1,2,5,6,8\},\{3,4,7,9,10\}\}$	1	0	0.459
	$\langle 10,5,5,4\rangle$			
0.1	$\{\{1,2,3,4,5\}\}$	0.103	0	0.749
0.2	$\{\{1,2,3,4,5\},\{1,4,6,7,10\}\}$	0.206	0	1.099
0.3	$\{\{1,2,6,7,10\},\{1,3,7,8,9\},\{2,3,4,5,6\}\}$	0.310	23	1.379
0.4	$\{\{1,2,6,7,9\},\{1,3,5,6,10\},\{1,4,5,7,8\},$ $\{2,3,4,8,9\}\}$	0.413	0	1.089
0.5	$\{\{1,2,4,6,8\},\{1,3,5,7,8\},\{1,4,7,9,10\},$ $\{2,3,6,7,10\},\{5,6,8,9,10\}\}$	0.516	65	3.029
0.6	$\{\{1,2,3,4,10\},\{1,3,7,8,9\},\{1,4,5,6,8\},$ $\{2,4,6,7,9\},\{2,5,7,8,10\},\{3,5,6,9,10\}\}$	0.619	263	8.489
0.7	$\{\{1,2,4,7,10\},\{1,2,5,8,9\},\{1,3,5,6,7\},$ $\{1,4,6,8,9\},\{2,3,4,7,9\},\{2,3,7,9,10\},$ $\{2,4,5,6,10\},\{3,4,5,8,10\}\}$	0.714	106 521	14 400
0.8	$\{\{1,2,3,5,8\},\{1,2,4,8,10\},\{1,2,6,7,9\},$	0.802	50 636	14 400

Table B.1 – continued from previous page

ψ	Playing set	ψ'	Branches	Time
0.9	$\{\{1,3,4,5,7\},\{1,3,6,9,10\},\{2,4,5,6,10\},$ $\{2,5,7,9,10\},\{3,6,7,8,10\},\{4,5,6,8,9\}\}$ $\{\{1,2,4,7,8\},\{1,2,6,7,9\},\{1,3,4,5,10\},$ $\{1,3,4,6,7\},\{1,4,6,9,10\},\{1,5,6,8,9\},$ $\{2,3,4,5,9\},\{2,3,6,8,10\},\{2,5,6,7,10\},$ $\{3,7,8,9,10\},\{4,5,6,7,8\},\{5,7,8,9,10\}\}$	0.905	47 069	14 400
1	$\{\{1,2,3,4,6\},\{1,2,4,8,10\},\{1,2,5,6,10\},$ $\{1,3,5,7,8\},\{1,3,7,9,10\},\{1,4,5,7,9\},$ $\{1,6,7,8,9\},\{2,3,4,8,9\},\{2,3,5,6,9\},$ $\{2,5,8,9,10\},\{3,4,5,7,10\},\{3,6,7,8,10\},$ $\{4,5,6,7,8\},\{4,6,7,9,10\}\}$	1	59 592	14 400

APPENDIX C

Resource utilisation problem modelled as an ILP

TABLE C.1: *Results obtained when solving for the value $\Psi_\ell(m,n,t,k)$ when the resource utilisation problem is formulated as an ILP for small lottery instances. The first column contains the fixed playing set cardinality ℓ. The second column contains an example of a playing set which achieves the probability-of-win value Ψ_ℓ in the third column. The fourth column contains the associated number of branches in the branch-and-bound tree. Finally, the fifth column contains the time in seconds required to find the solution.*

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
	$\langle 6,3,3,2\rangle$			
1	$\{\{1,2,3\}\}$	0.5	0	0.009
2	$\{\{1,2,6\},\{2,3,5\}\}$	1	0	0.019
	$\langle 7,3,3,2\rangle$			
1	$\{\{3,5,7\}\}$	0.371	0	0.029
2	$\{\{1,2,4\},\{3,5,7\}\}$	0.743	0	0.039
3	$\{\{1,3,4\},\{1,3,7\},\{2,5,6\}\}$	0.914	156	0.669
4	$\{\{1,3,5\},\{1,4,7\},\{2,3,6\},\{4,5,7\}\}$	1	0	0.029
	$\langle 7,3,4,2\rangle$			
1	$\{\{4,5,7\}\}$	0.629	0	0.049
2	$\{\{1,2,3\},\{4,6,7\}\}$	1	0	0.039
	$\langle 8,3,3,2\rangle$			
1	$\{\{5,7,8\}\}$	0.286	0	0.059
2	$\{\{1,3,4\},\{2,7,8\}\}$	0.571	0	0.089
3	$\{\{1,5,7\},\{2,4,6\},\{3,7,8\}\}$	0.786	19	0.409
4	$\{\{1,4,5\},\{2,3,6\},\{2,7,8\},\{3,7,8\}\}$	0.893	1 508	5.589
5	$\{\{1,2,5\},\{1,4,5\},\{1,5,8\},\{2,4,8\},\{3,6,7\}\}$	1	0	0.059
	$\langle 8,3,4,2\rangle$			
1	$\{\{2,3,8\}\}$	0.5	0	0.099
2	$\{\{1,4,7\},\{2,6,8\}\}$	0.871	12	0.969
3	$\{\{1,3,8\},\{2,5,7\},\{4,6,7\}\}$	1	0	0.069
	$\langle 8,3,5,2\rangle$			
1	$\{\{3,5,6\}\}$	0.714	0	0.079
2	$\{\{1,5,8\},\{3,4,7\}\}$	1	0	0.059

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
	$\langle 8,4,3,2\rangle$			
1	$\{\{4,5,7,8\}\}$	0.5	0	0.089
2	$\{\{1,2,5,6\},\{3,4,7,8\}\}$	1	0	0.079
	$\langle 8,4,4,2\rangle$			
1	$\{\{3,4,6,8\}\}$	0.757	0	0.109
2	$\{\{1,2,6,8\},\{3,4,5,6\}\}$	1	0	0.089
	$\langle 8,4,4,3\rangle$			
1	$\{\{4,5,7,8\}\}$	0.243	0	0.109
2	$\{\{1,2,5,8\},\{3,4,6,7\}\}$	0.486	0	0.109
3	$\{\{1,2,3,4\},\{1,3,5,8\},\{4,6,7,8\}\}$	0.671	12 776	76.799
4	$\{\{1,2,5,6\},\{1,3,4,8\},\{2,3,4,7\},\{5,6,7,8\}\}$	0.857	1 685	15.859
5	$\{\{1,2,3,8\},\{1,2,6,7\},\{3,4,5,6\},\{4,5,6,8\},$ $\{4,5,7,8\}\}$	0.914	34 210	131.859
6	$\{\{1,2,3,6\},\{1,4,7,8\},\{1,5,7,8\},\{2,3,4,5\},$ $\{2,4,5,6\},\{3,6,7,8\}\}$	1	14	0.209
	$\langle 9,3,3,2\rangle$			
1	$\{\{1,2,9\}\}$	0.226	0	0.119
2	$\{\{2,3,4\},\{5,7,8\}\}$	0.452	0	0.119
3	$\{\{1,2,4\},\{3,5,6\},\{7,8,9\}\}$	0.679	2	0.269
4	$\{\{1,2,5\},\{2,3,8\},\{4,7,9\},\{5,6,8\}\}$	0.774	77 226	753.929
5	$\{\{1,2,4\},\{1,3,6\},\{2,3,5\},\{4,5,6\},\{7,8,9\}\}$	0.893	43 161	386.869
6	$\{\{1,2,3\},\{1,3,6\},\{2,6,8\},\{4,5,9\},\{4,7,8\},$ $\{5,7,9\}\}$	0.952	85 587	491.509
7	$\{\{1,2,8\},\{1,3,8\},\{1,5,8\},\{2,3,5\},\{2,4,7\},$ $\{4,6,9\},\{6,7,9\}\}$	1	0	0.069
	$\langle 9,3,4,2\rangle$			
1	$\{\{2,3,8\}\}$	0.405	0	0.439
2	$\{\{2,3,6\},\{5,7,8\}\}$	0.738	2 133	78.029
3	$\{\{1,2,3\},\{4,8,9\},\{5,6,7\}\}$	1	0	0.079
	$\langle 9,3,5,2\rangle$			
1	$\{\{2,5,9\}\}$	0.595	0	0.319
2	$\{\{1,6,8\},\{5,7,9\}\}$	0.929	16	1.989
3	$\{\{1,5,8\},\{2,3,4\},\{6,7,9\}\}$	1	0	0.089
	$\langle 9,3,6,2\rangle$			
1	$\{\{2,4,6\}\}$	0.774	0	0.139
2	$\{\{1,5,6\},\{2,3,4\}\}$	1	0	0.069
	$\langle 9,4,3,2\rangle$			
1	$\{\{3,4,8,9\}\}$	0.405	0	0.299
2	$\{\{1,2,3,8\},\{4,5,6,9\}\}$	0.810	0	0.249
3	$\{\{1,5,6,8\},\{2,3,4,9\},\{2,3,7,9\}\}$	0.952	438	10.139
4	$\{\{1,2,3,4\},\{1,2,3,8\},\{1,2,4,8\},\{5,6,7,9\}\}$	1	0	0.079
	$\langle 9,4,4,2\rangle$			
1	$\{\{3,5,6,9\}\}$	0.643	0	0.509
2	$\{\{1,3,5,7\},\{2,4,8,9\}\}$	1	0	0.129
	$\langle 9,4,4,3\rangle$			
1	$\{\{5,7,8,9\}\}$	0.167	0	0.319
2	$\{\{1,3,8,9\},\{3,4,6,7\}\}$	0.333	0	0.368

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
3	$\{\{1,3,5,7\},\{2,4,5,6\},\{6,7,8,9\}\}$	0.5	0	0.449
4	$\{\{1,2,3,9\},\{1,2,6,8\},\{3,4,5,6\},\{4,7,8,9\}\}$	0.635	325 782	9 819.239
5	$\{\{1,2,7,9\},\{1,6,7,8\},\{2,4,5,8\},\{3,4,6,9\},$ $\{3,5,7,8\}\}$	0.738	277 847	14 400
6	$\{\{1,2,3,6\},\{1,4,6,9\},\{2,4,6,7\},\{2,5,8,9\},$ $\{3,4,5,8\},\{3,7,8,9\}\}$	0.825	353 245	14 400
7	$\{\{1,3,6,7\},\{1,4,5,7\},\{1,4,6,8\},\{1,7,8,9\},$ $\{2,3,4,9\},\{2,3,5,8\},\{2,5,6,9\}\}$	0.913	428 601	14 400
8	$\{\{1,2,8,9\},\{1,3,4,9\},\{1,5,6,7\},\{2,3,6,7\},$ $\{2,4,5,6\},\{2,4,5,7\},\{3,5,8,9\},\{4,6,7,8\}\}$	0.960	657 794	144 00
9	$\{\{1,2,5,7\},\{1,3,4,7\},\{1,4,5,6\},\{1,6,8,9\},$ $\{2,3,4,9\},\{2,3,6,8\},\{2,4,7,8\},\{3,5,8,9\},$ $\{5,6,7,9\}\}$	1	156	1.559
$\langle 9,4,5,2\rangle$				
1	$\{\{2,4,6,8\}\}$	0.833	0	0.459
2	$\{\{1,2,3,6\},\{1,4,5,7\}\}$	1	0	0.139
$\langle 9,4,5,3\rangle$				
1	$\{\{4,5,6,7\}\}$	0.357	0	0.409
2	$\{\{2,4,8,9\},\{3,5,6,7\}\}$	0.714	0	0.469
3	$\{\{1,3,6,9\},\{2,5,7,8\},\{3,4,5,8\}\}$	0.857	54 186	1 144.269
4	$\{\{1,6,7,8\},\{2,3,7,8\},\{2,4,5,9\},\{3,4,5,9\}\}$	0.976	18 238	271.479
5	$\{\{1,2,3,4\},\{1,4,5,8\},\{2,3,5,6\},\{2,6,7,9\},$ $\{3,7,8,9\}\}$	1	3	0.229
$\langle 9,4,6,3\rangle$				
1	$\{\{1,3,4,6\}\}$	0.595	0	0.289
2	$\{\{1,3,7,9\},\{2,4,5,8\}\}$	1	0	0.159
$\langle 10,3,3,2\rangle$				
1	$\{\{4,5,8\}\}$	0.183	0	0.439
2	$\{\{1,5,9\},\{2,7,8\}\}$	0.367	0	0.389
3	$\{\{2,5,7\},\{3,4,8\},\{6,9,10\}\}$	0.55	0	0.439
4	$\{\{1,3,4\},\{2,4,8\},\{2,6,9\},\{5,7,10\}\}$	0.667	394 983	14 400
5	$\{\{1,5,6\},\{2,3,7\},\{2,8,9\},\{3,4,8\},\{7,9,10\}\}$	0.767	312 622	14 400
6	$\{\{1,3,5\},\{1,6,7\},\{2,9,10\},\{3,4,7\},\{4,5,6\},$ $\{8,9,10\}\}$	0.85	417 227	14 400
7	$\{\{1,2,3\},\{1,8,9\},\{2,4,5\},\{3,4,10\},\{5,6,10\},$ $\{5,7,9\},\{6,7,8\}\}$	0.917	799 315	14 400
8	$\{\{1,2,3\},\{1,3,9\},\{1,3,10\},\{2,9,10\},\{4,5,7\},$ $\{4,6,8\},\{5,6,8\},\{6,7,8\}\}$	1	0	0.339
$\langle 10,3,4,2\rangle$				
1	$\{\{3,5,6\}\}$	0.333	0	1.039
2	$\{\{1,3,7\},\{5,6,9\}\}$	0.624	8 142	1 585.749
3	$\{\{1,5,8\},\{2,7,9\},\{4,6,10\}\}$	0.871	151 803	14 400
4	$\{\{1,2,3\},\{4,6,9\},\{5,7,10\},\{5,8,10\}\}$	0.957	27 230	1 316.078
5	$\{\{1,2,6\},\{1,3,5\},\{3,8,10\},\{4,7,9\},\{5,8,10\}\}$	1	0	0.198
$\langle 10,3,5,2\rangle$				
1	$\{\{2,5,10\}\}$	0.5	0	0.968
2	$\{\{3,5,9\},\{4,7,10\}\}$	0.833	111 722	2 686.248

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
3	$\{\{1,2,3\},\{4,5,10\},\{6,7,9\}\}$	1	0	0.258
	$\langle 10,3,6,2\rangle$			
1	$\{\{1,3,6\}\}$	0.667	0	1.109
2	$\{\{1,5,6\},\{3,8,10\}\}$	0.957	26	6.249
3	$\{\{1,4,6\},\{2,5,9\},\{7,9,10\}\}$	1	0	0.209
	$\langle 10,3,7,2\rangle$			
1	$\{\{1,2,8\}\}$	0.817	0	0.389
2	$\{\{1,5,6\},\{3,4,10\}\}$	1	0	0.129
	$\langle 10,4,3,2\rangle$			
1	$\{\{1,4,6,9\}\}$	0.333	0	0.859
2	$\{\{1,4,6,8\},\{3,5,9,10\}\}$	0.667	0	1.149
3	$\{\{1,6,7,8\},\{2,3,4,9\},\{5,6,8,10\}\}$	0.867	59 099	2 260.909
4	$\{\{1,2,5,7\},\{3,4,6,8\},\{3,8,9,10\},\{4,6,9,10\}\}$	1	0	0.229
	$\langle 10,4,4,2\rangle$			
1	$\{\{2,3,4,10\}\}$	0.548	0	2.069
2	$\{\{1,3,7,9\},\{4,6,8,10\}\}$	0.924	36	133.439
3	$\{\{1,7,8,9\},\{2,4,7,10\},\{3,4,5,6\}\}$	1	0	0.489
	$\langle 10,4,4,3\rangle$			
1	$\{\{5,8,9,10\}\}$	0.119	0	1.249
2	$\{\{1,3,7,9\},\{5,8,9,10\}\}$	0.238	0	1.259
3	$\{\{1,2,7,9\},\{3,6,8,9\},\{4,7,8,10\}\}$	0.357	1	2.099
4	$\{\{1,3,6,9\},\{1,4,5,10\},\{2,3,8,10\},\{5,7,8,9\}\}$	0.476	1	2.029
5	$\{\{1,2,4,5\},\{1,3,6,8\},\{2,3,7,10\},\{4,6,9,10\},$ $\{5,7,8,9\}\}$	0.595	6	2.999
6	$\{\{1,2,3,10\},\{1,6,7,9\},\{2,4,7,8\},\{3,5,8,9\},$ $\{4,6,9,10\},\{5,6,7,10\}\}$	0.662	90 150	14 400
7	$\{\{1,2,3,7\},\{1,3,6,9\},\{1,4,8,10\},\{2,4,5,6\},$ $\{2,5,8,9\},\{4,5,7,9\},\{6,7,8,10\}\}$	0.743	69 869	14 400
8	$\{\{1,2,4,9\},\{1,2,6,8\},\{1,5,8,9\},\{2,3,5,7\},$ $\{2,3,8,10\},\{3,4,7,8\},\{4,5,6,10\},\{6,7,9,10\}\}$	0.810	57 677	14 400
9	$\{\{1,2,3,4\},\{1,3,8,9\},\{1,5,6,9\},\{1,5,7,10\},$ $\{2,3,9,10\},\{2,4,7,9\},\{2,6,7,8\},\{3,4,6,7\},$ $\{4,5,8,10\}\}$	0.862	81 900	14 400
10	$\{\{1,2,5,7\},\{1,3,5,6\},\{1,4,6,7\},\{1,8,9,10\},$ $\{2,3,4,8\},\{2,3,4,9\},\{2,6,9,10\},\{3,5,7,10\},$ $\{4,5,8,10\},\{6,7,8,9\}\}$	0.910	146 793	14 400
11	$\{\{1,2,4,6\},\{1,3,5,8\},\{1,3,7,9\},\{1,4,5,9\},$ $\{1,7,8,10\},\{2,3,4,10\},\{2,5,6,7\},\{2,5,9,10\},$ $\{3,6,8,9\},\{4,5,6,10\},\{4,7,8,9\}\}$	0.948	136 449	14 400
12	$\{\{1,2,5,7\},\{1,3,8,9\},\{1,4,6,7\},\{1,5,9,10\},$ $\{2,3,6,10\},\{2,4,6,9\},\{2,4,8,10\},\{2,5,7,9\},$ $\{3,4,5,8\},\{3,4,7,10\},\{3,5,6,8\},\{6,7,8,10\}\}$	0.971	210 837	14 400
13	$\{\{1,2,5,9\},\{1,2,7,8\},\{1,3,6,10\},\{1,4,5,6\},$ $\{1,4,8,9\},\{2,3,4,10\},\{2,5,6,8\},\{2,6,7,9\},$ $\{3,4,6,7\},\{3,5,7,9\},\{3,5,8,10\},\{4,5,7,10\},$ $\{6,8,9,10\}\}$	0.990	204 147	14 400
14	$\{\{1,2,3,8\},\{1,4,5,7\},\{1,4,6,10\},\{1,5,6,9\},$	1	543	32.199

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
	$\{1,5,7,10\},\{1,7,9,10\},\{2,3,4,9\},\{2,3,5,9\},$ $\{2,4,5,10\},\{2,6,7,8\},\{3,4,5,8\},\{3,6,7,10\},$ $\{4,7,8,9\},\{6,8,9,10\}\}$			
$\langle 10,4,5,2\rangle$				
1	$\{\{1,4,5,9\}\}$	0.738	0	2.009
2	$\{\{1,2,6,9\},\{2,3,5,9\}\}$	1	0	0.319
$\langle 10,4,5,3\rangle$				
1	$\{\{3,4,8,10\}\}$	0.262	0	5.579
2	$\{\{1,4,8,10\},\{2,3,5,7\}\}$	0.524	0	2.599
3	$\{\{1,2,6,9\},\{3,5,8,10\},\{4,5,7,9\}\}$	0.714	71 736	14 400
4	$\{\{1,5,7,9\},\{2,3,4,5\},\{3,6,7,8\},\{4,8,9,10\}\}$	0.833	47 849	14 400
5	$\{\{1,2,3,6\},\{1,7,9,10\},\{2,4,8,9\},\{3,5,8,10\},$ $\{4,5,6,7\}\}$	0.952	79 900	14 400
6	$\{\{1,2,4,7\},\{1,2,9,10\},\{3,4,5,10\},\{3,5,7,9\},$ $\{4,6,8,9\},\{6,7,8,10\}\}$	0.976	127 812	14 400
7	$\{\{1,2,7,8\},\{1,3,5,10\},\{1,4,6,9\},\{2,3,6,9\},$ $\{2,4,5,9\},\{4,7,8,10\},\{5,6,7,8\}\}$	1	18	3.240
$\langle 10,4,6,2\rangle$				
1	$\{\{1,4,6,8\}\}$	0.881	0	3.059
2	$\{\{1,2,3,5\},\{3,4,6,7\}\}$	1	0	0.639
$\langle 10,4,6,3\rangle$				
1	$\{\{2,3,5,6\}\}$	0.452	0	2.760
2	$\{\{1,4,7,8\},\{3,5,6,9\}\}$	0.829	77 522	10 260.720
3	$\{\{1,2,5,6\},\{2,3,5,7\},\{4,8,9,10\}\}$	0.971	40 478	2 844.430
4	$\{\{1,2,3,7\},\{1,2,5,6\},\{1,3,5,6\},\{4,8,9,10\}\}$	1	0	0.340
$\langle 10,4,7,3\rangle$				
1	$\{\{1,4,5,7\}\}$	0.667	0	1.069
2	$\{\{1,2,4,5\},\{3,7,8,10\}\}$	1	0	0.389
$\langle 10,5,3,2\rangle$				
1	$\{\{1,2,4,6,10\}\}$	0.5	0	1.169
2	$\{\{1,2,5,6,8\},\{3,4,7,9,10\}\}$	1	0	0.519
$\langle 10,5,4,2\rangle$				
1	$\{\{1,5,6,7,9\}\}$	0.738	0	3.389
2	$\{\{1,2,3,8,10\},\{2,4,5,7,9\}\}$	1	0	0.729
$\langle 10,5,4,3\rangle$				
1	$\{\{1,2,3,4,5\}\}$	0.262	0	2.359
2	$\{\{1,2,3,5,6\},\{1,4,8,9,10\}\}$	0.524	0	4.689
3	$\{\{1,2,4,5,10\},\{3,5,7,8,9\},\{4,6,8,9,10\}\}$	0.7	56 333	14 400
4	$\{\{1,4,7,8,10\},\{1,5,6,7,9\},\{2,3,4,6,9\},$ $\{2,3,5,8,10\}\}$	0.876	82 174	14 400
5	$\{\{1,3,4,5,7\},\{1,3,6,9,10\},\{2,3,7,8,10\},$ $\{2,4,6,8,9\},\{2,5,6,8,9\}\}$	0.924	104 717	14 400
6	$\{\{1,2,5,6,9\},\{1,3,4,7,8\},\{1,5,7,9,10\},$ $\{2,3,4,8,10\},\{2,5,6,7,9\},\{3,4,6,8,10\}\}$	0.986	130 990	14 400
7	$\{\{1,2,3,4,8\},\{1,4,5,9,10\},\{1,4,7,9,10\},$ $\{1,5,6,7,10\},\{2,3,5,7,8\},\{2,4,6,8,9\},$	1	3	1.099

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
	$\{3,4,6,9,10\}\}$			
	$\langle 10,5,5,2\rangle$			
1	$\{\{1,6,7,8,10\}\}$	0.897	0	5.419
2	$\{\{1,2,3,6,7\},\{2,4,5,7,10\}\}$	1	0	0.919
	$\langle 10,5,5,3\rangle$			
1	$\{\{2,3,4,5,9\}\}$	0.5	0	2.049
2	$\{\{1,2,3,6,7\},\{4,5,8,9,10\}\}$	1	0	0.799
	$\langle 10,5,5,4\rangle$			
1	$\{\{1,3,6,8,10\}\}$	0.103	0	1.999
2	$\{\{2,4,6,8,10\},\{3,4,6,7,9\}\}$	0.206	0	3.599
3	$\{\{1,2,3,8,9\},\{1,2,5,6,10\},\{4,6,7,9,10\}\}$	0.31	0	1.819
4	$\{\{1,2,4,8,9\},\{1,3,4,6,10\},\{2,5,6,7,10\},$ $\{3,5,7,8,9\}\}$	0.413	1	3.929
5	$\{\{1,2,5,8,9\},\{1,3,4,5,7\},\{1,6,7,8,10\},$ $\{2,3,6,7,9\},\{4,5,6,9,10\}\}$	0.516	1	5.989
6	$\{\{1,2,4,8,9\},\{1,3,7,8,10\},\{1,5,6,9,10\},$ $\{2,3,5,6,8\},\{2,4,5,7,10\},\{3,4,6,7,9\}\}$	0.619	0	5.999
7	$\{\{1,2,3,4,7\},\{1,4,5,8,9\},\{1,6,7,9,10\},$ $\{2,3,6,8,9\},\{2,4,5,6,10\},\{3,5,7,8,10\},$ $\{4,5,6,7,9\}\}$	0.675	49 712	14 400
8	$\{\{1,2,4,5,10\},\{1,2,6,8,9\},\{1,3,4,6,9\},$ $\{1,3,7,8,10\},\{2,3,4,5,8\},\{2,3,5,6,7\},$ $\{2,4,7,9,10\},\{5,6,8,9,10\}\}$	0.746	38 233	14 400
9	$\{\{1,2,7,8,9\},\{1,3,4,9,10\},\{1,3,6,7,8\},$ $\{1,4,5,8,10\},\{1,5,6,9,10\},\{2,3,5,7,10\},$ $\{2,3,5,8,9\},\{2,4,6,8,10\},\{4,5,6,7,9\}\}$	0.806	30 816	14 400
10	$\{\{1,2,4,5,7\},\{1,2,5,6,8\},\{1,3,4,8,9\},$ $\{1,3,5,9,10\},\{1,6,7,8,10\},\{2,3,4,8,10\},$ $\{2,3,6,7,9\},\{2,4,6,9,10\},\{3,4,5,6,7\},$ $\{5,7,8,9,10\}\}$	0.873	73 107	14 400
11	$\{\{1,2,3,5,8\},\{1,2,4,8,9\},\{1,2,6,7,9\},$ $\{1,3,5,7,9\},\{1,3,6,8,10\},\{1,4,5,7,10\},$ $\{2,3,4,5,6\},\{2,3,4,9,10\},\{2,5,7,8,10\},$ $\{3,4,6,7,8\},\{5,6,8,9,10\}\}$	0.897	77 725	14 400
12	$\{\{1,2,3,9,10\},\{1,2,4,7,8\},\{1,2,5,6,9\},$ $\{1,3,4,6,7\},\{1,4,5,8,10\},\{1,5,6,7,10\},$ $\{2,3,4,5,7\},\{2,3,6,8,10\},\{2,4,6,9,10\},$ $\{3,4,5,8,9\},\{3,6,7,8,9\},\{5,7,8,9,10\}\}$	0.929	85 107	14 400
13	$\{\{1,2,3,6,7\},\{1,2,4,5,9\},\{1,2,7,8,10\},$ $\{1,3,4,9,10\},\{1,4,6,7,8\},\{1,5,6,9,10\},$ $\{2,3,4,8,9\},\{2,3,5,7,10\},\{2,4,6,9,10\},$ $\{2,5,6,8,9\},\{3,4,5,6,7\},\{3,6,8,9,10\},$ $\{4,5,7,8,10\}\}$	0.96	88 888	14 400
14	$\{\{1,2,3,5,7\},\{1,2,3,6,8\},\{1,2,6,7,10\},$ $\{1,3,4,5,8\},\{1,4,5,7,10\},\{1,4,6,8,10\},$ $\{1,5,6,7,8\},\{2,3,4,9,10\},\{2,4,5,6,9\},$ $\{2,4,7,8,9\},\{2,5,8,9,10\},\{3,4,6,7,9\},$	1	6 486	741.550

Table C.1 – continued from previous page

ℓ	Playing set	Ψ_ℓ	Branches	Time (secs)
	$\{3,5,6,9,10\},\{3,7,8,9,10\}\}$			

APPENDIX D

Programming code

This appendix contains the C# programming code for the implementation of the exhaustive enumeration lottery tree presented in Chapter 5.

```
using System;
using System.Collections;
using System.Linq;
using System.Text;
using System.IO;

namespace permutations
{
    public class Node
    {
        public ArrayList children; //list of children in the node
        public ArrayList tempchildren; //2nd list of chuildren
        public int[] arrX; //Vector X representing an overlapping playing set structure
        public int sz;// size of the array
        public bool TorF; //binary value of the node
        public ArrayList MListT; //D_T
        public ArrayList MListF; //D_F
        public static int count = 0;
        public static Node New(int size)
        {
            return new Node(size);
        }
        public Node(int size)
        {
         //initialise the member variables of the Node
            count++;
            sz = size;
            arrX = new int[sz];
            children = new ArrayList();
            tempchildren = new ArrayList();
            MListT = new ArrayList();
            MListF = new ArrayList();
            TorF = false;
        }
        //add a child to the 1st list of children
        public void AddChild(Node child)
        {
            children.Add(child);
        }
         //add a child to the 2nd list of children
        public void AddTempChild(Node child)
        {
            if (child.getTorF() == true)
            {
                MListT.Add(M(child));
            }
            else
            {
```

```
            MListF.Add(M(child));
        }
        tempchildren.Add(child);
    }
    //set the value of vetor X
    public void setarrX(int[] newArray)
    {
        int i;
        for (i = 0; i < sz; i++)
        {
            arrX[i] = newArray[i];
        }
    }
    //set binary flag of node
    public void setTorF(bool boolval)
    {
        TorF = boolval;
    }
    //retrieve binary flag of node
    public bool getTorF()
    {
        return TorF;
    }
    //retrieve vector X
    public int[] getArrX()
    {
        return arrX;
    }
    //output vector X
    public void printarrX()
    {
        int i;
        Console.Write("(");
        for (i = 0; i < sz; i++)
        {
            if (i > 0)
            {
                Console.Write(", ");
            }
            Console.Write("{0}", arrX[i]);
        }
        Console.WriteLine(")");
    }
    //print the 2nd set of children
    public void printTempChildren()
    {
        int j;
        for (j = 0; j < tempchildren.Count; j++)
        {
            Console.Write("tempchild: ");
            Node tempNode = (Node)tempchildren[j];
            tempNode.printarrX();
            Console.WriteLine("{0}",tempNode.TorF);
        }
    }
    //print 1st set of children
    public void printChildren()
    {
        int j;
        for (j = 0; j < children.Count; j++)
        {
            Console.Write("child: ");
            Node tempNode = (Node)children[j];
            tempNode.printarrX();
            Console.WriteLine("{0}", tempNode.getPsi());
        }
    }
    //retrieve the number fo children
    public int getNumChildren()
    {
```

```
        return children.Count;
    }
    //retrieve the probability-of-win value associated with the node
    public double getPsi()
    {
        double numerator, denominator;
        numerator = 0;
        denominator = 0;
        for (int i = 0; i < MListT.Count; i++)
        {
            numerator += (double)MListT[i];
        }

        for (int i = 0; i < MListF.Count; i++)
        {
            denominator += (double)MListF[i];
        }
        if (numerator == 0)
        {
            return 0;
        }
        else
        {
            return numerator / (numerator + denominator);
        }
    }
    //the function M(X) defined in Ch5
    public double M(Node X)
    {
        int i;
        double product=1;
        for (i = 0; i < X.sz; i++)
        {
            product *=  Program.factorial(X.arrX[i]);
        }
        return (Program.factorial(Program.m)/ product);
    }
}

class Program
{
    public static int m;
    public static int n;
    public static int t;
    public static int k;
    public static int ell;
    public static int ell1;
    public static double[] probabilities;
    public static int[] numStructures;
    public static void Main(string[] args)
    {
     //initialise the parameters of the lottery <m,n,t,k>
        m = 5;
        n = 3;
        t = 3;
        k = 2;
        //The fixed playing set cardinality
        ell = 3;
        //The complete lottery number L_1(m,n,t,k)
        ell1 =3;
        //output the lottery parameters
        Console.WriteLine("<{0},{1},{2},{3}> l <= {4}", m, n, t, k, ell);
        probabilities = new double[ell]; //a list of the highest probability of win value at each level of the tree
        numStructures = new int[ell + 1];//a list of the value of the lottery charactersation number at each level of th
        for (int p = 0; p < probabilities.Length; p++)
        {
            probabilities[p] = 0;
            numStructures[p] = 0;
        }
        DateTime startTime = DateTime.Now; //initialise the start time of execution
```

```
    //initialise the root node of the lottery tree
    int[] arr = new int[2];
    Node rootNode = Node.New(2);
    arr[0] = m - n;
    arr[1] = n;
    rootNode.setarrX(arr);
    //Build the lottery tree
    LevelK(n, rootNode, 2, ell);
    //Record the execution time
    DateTime stopTime = DateTime.Now;
    TimeSpan duration = stopTime - startTime;
    //Output the results
    printTree(rootNode);
    Console.WriteLine("\n\n\n");
    for (int p = 0; p < numStructures.Length; p++)
    {
        Console.WriteLine("Nodes at level {0}: {1}",p+1, numStructures[p]);
    }
    for (int p = 0; p < probabilities.Length; p++)
    {
        if (p == 0)
        {
            Console.WriteLine("Tickets: {0}, Probability: {1}", p + 1, probabilities[p]);
        }
        if (p > 0 && probabilities[p - 1] != 1)
        {
            Console.WriteLine("Tickets: {0}, Probability: {1}", p + 1, probabilities[p]);
        }
    }
    Console.WriteLine("\n\n{0}", duration);
    Console.WriteLine("\n\nDone");
    Console.ReadLine();
}

// |arrY| = 2* |arrX|, |arrX| = sz
private class CalcV
{
    public int sz;
    public int[] arrY;
    public int[] arrX;
    private int iter;
    public CalcV(int sz, ref int[] arrY, ref int[] arrX)
    {
        this.sz = sz; this.arrY = arrY; this.arrX = arrX;
        this.iter = 0;
    }
    //Build all possible child vectors from the parent vextor arrX
    public bool get()
    {
        int iter2 = iter;
        for (int i = 1; i <= sz; i++)
        {
            int a = arrX[sz - i] + 1;
            arrY[2 * sz - i] = iter2 % a;
            iter2 /= a;
        }
        if (iter2 > 0)
        {
            return false;
        }
        for (int i = 0; i < sz; i++)
        {
            arrY[i] = arrX[i] - arrY[sz + i];
        }
        iter++;
        return true;
    }
}
static void LevelK(int n, Node parentNode, int sz, int max)
{
```

```
if (max == 0)
{
    return;
}
int sz2 = sz*2;
int[] arrX = parentNode.getArrX();
int[] arrY = new int[sz2];
CalcV calcv = new CalcV(sz, ref arrY, ref arrX);
while (calcv.get()) //calculate the values for array Y from the values in array X.
{
    int sum = 0;
    for (int i = sz; i < sz2; i++)
    {
        sum += arrY[i];
    }
    if (sum == n && arrY[sz * 2 - 1] <= n) //add a new child
    {
        Node newChild = Node.New(sz2);
        newChild.setarrX(arrY);
        int totest=-1;
        if (arrY.Length >= 8)
        {
            totest= getL2Parent(arrY);
        }
        bool boolval = false;
        bool isValid = true;
        int l;
        int kSum;
        kSum = 0;
        int cntr;
        //explore every element in the new vector and sum all its elements where the
        //extra ticket overlaps one of the other tickets. If the sum>=k, then
        //the extra ticket has more than k numbers in common with one of the tickets
        //in the current playing set, and it is a valid overlapping of a winning ticket
        //with the playing set.
        for (l = sz + 1; (l < sz2); l++)
        {
            if(isValidBinary(l))
            {
                kSum = 0;
                for (cntr = l; cntr < sz2; cntr++)
                {
                    if ((int)(l & cntr) == l)
                    {
                        kSum = kSum + newChild.arrX[cntr];
                    }
                }
                if (kSum >= k) //the child vector passes the domination test
                {
                    boolval = true;
                }
                if (kSum >= n)  //the new ticket in the child vector overlaps exactly
                {        //with one of the original tickets, meaning the child
                 //vector does not need to be added to the tree
                    isValid = false;
                }
                //check for redundant structures which may be omitted from the tree
                if ((totest >= 0) && isValid)
                {
                    if (kSum < totest)
                    {
                        isValid = false;
                    }
                }
            }
        }
        //set binary flag of node
        if (boolval == true)
        {
            newChild.setTorF(true);
```

```
                }
                //add new child to 2nd list of children
                if (sum == t)
                {
                    parentNode.AddTempChild(newChild);
                }
                if (arrY.Length <= Math.Pow(2, ell))
                {
                 //apply pruning rules
                    if (isValid && !toPrune(arrY))
                    {
                        if (arrY.Length <= 4)
                        {
                            parentNode.AddChild(newChild);
                            LevelK(n, newChild, sz2, max - 1);
                        }
                        //ensure vector is not a duplicate before adding to the tree
                        else if (!isDuplicate(arrY))
                        {
                            parentNode.AddChild(newChild);
                            LevelK(n, newChild, sz2, max - 1);
                        }
                        if ((parentNode.getPsi() == 1) || (arrY.Length >= Math.Pow(ell, 2)))
                        {
                            max = 0;
                        }
                    }
                }
            }
            if (sum == t && arrY[sz * 2 - 1] <= t  && n != t) //overlap a winning ticket of size t.
            {
                Node newChild = Node.New(sz2);
                newChild.setarrX(arrY);
                bool boolval = false;
                int l;
                int kSum;
                kSum = 0;
                int cntr;
                for (l = sz + 1; (l < sz2) && (boolval == false); l++)
                {
                    if (isValidBinary(l))
                    {
                        kSum = 0;
                        for (cntr = l; cntr < sz2; cntr++)
                        {
                            if ((int)(l & cntr) == l)
                            {
                                kSum = kSum + newChild.arrX[cntr];
                            }
                        }
                        if (kSum >= k) //the domination test is passed
                        {
                            boolval = true;
                        }
                    }
                }
                //set the binary value of the node
                if (boolval == true)
                {
                    newChild.setTorF(true);
                }
                parentNode.AddTempChild(newChild);
            }
        }
    }
//compute the number of elements in compartment x11 of
//the parent node at level 2 of the tree
    static int getL2Parent(int[] vector)
    {
        ArrayList arr1 = new ArrayList();
```

```
            ArrayList arr2 = new ArrayList();
            foreach (int j in vector)
            {
                arr1.Add(j);
            }
            while(true)
            {
                for (int i = 0; i < arr1.Count/2; i++)
                {
                    arr2.Add((int)arr1[i] + (int)arr1[i + arr1.Count/2]);
                }
                if (arr2.Count == 4)
                {
                    return (int)arr2[3];
                }
                else
                {
                    ArrayList tmp = arr1;
                    arr1 = arr2;
                    arr2 = tmp;
                    arr2.Clear();
                }
            }
        }
//test if an entry in a child vector needs to be considered
//used in LevelK(..)
        public static bool isValidBinary(int myNum)
        {
            char[] toTest = ToBinary(myNum);
            int sum = 0;
            for (int i = 0; i < toTest.Length; i++)
            {
                if (toTest[i] == '1')
                {
                    sum++;
                }
            }
            if (sum == 2)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        static void printVector(int[] vector)
        {
            Console.Write("(");
            for (int cnt = 0; cnt < vector.Length; cnt++)
            {
                if (cnt > 0)
                {
                    Console.Write(", ");
                }
                Console.Write("{0}", vector[cnt]);
            }
            Console.WriteLine(")");
        }
        static void printTree(Node root)
        {
            int i;
            if (root.getPsi() != 0)
            {
                if (root.getPsi() > probabilities[(int)(Math.Log(root.arrX.Length) / Math.Log(2)-1)])
                {
                    probabilities[(int)(Math.Log(root.arrX.Length) / Math.Log(2)-1)] = root.getPsi();
                }
            }
            numStructures[(int)(Math.Log(root.arrX.Length) / Math.Log(2) - 1)]++;
```

```
    for (i = 0; i < root.getNumChildren(); i++)
    {
        printTree((Node)root.children[i]);
    }
}
static void printBest(Node root)
{
    int i;
    if (root.getPsi() == probabilities[ell-1])
    {
        printVector(root.arrX);
        root.printTempChildren();
    }
    for (i = 0; i < root.getNumChildren(); i++)
    {
        printBest((Node)root.children[i]);
    }
}
//return the value of n!
public static double factorial(int n)
{
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}
//a binomial coefficient function
public static double choose(int n, int k)
{
    return factorial(n) / (factorial(k) * factorial(n - k));
}
//convert an integer to a binary array
public static char[] ToBinary(int myInt)
{
    char[] BinaryArray;
    int mybase = 2;
    string binary = Convert.ToString(myInt, mybase);
    BinaryArray = binary.ToCharArray();
    return BinaryArray;
}
//apply the pruning rules
public static bool toPrune(int[] vector)
{
    if (vector.Length <= 4 || k==1)
    {
        return false;
    }
    int sum;
    if (vector[0] >= (ell1 - (int)(Math.Log(vector.Length) / Math.Log(2)) + 1) * t)
    {
        return true;
    }
    sum = 0;
    for (int i = 0; i < vector.Length; i++)
    {
        for (int j = 0; j < Math.Log(vector.Length) / Math.Log(2); j++)
        {
            if (i == Math.Pow(2, j))
            {
                sum += Math.Min(vector[i], k - 1);
            }
        }
    }
    if ((sum >= (ell1 - (int)(Math.Log(vector.Length) / Math.Log(2)) + 1) * t))
    {
        return true;
    }
    if (vector.Length == Math.Pow(2, ell1))
    {
        sum = 0;
        for (int i = 0; i < vector.Length; i++)
        {
```

```
                for (int j = 0; j < Math.Log(vector.Length) / Math.Log(2); j++)
                {
                    if (i == Math.Pow(2, j))
                    {
                        sum += Math.Min(vector[i], k - 1);
                    }
                }
            }
            if (sum >= t)
            {
                return true;
            }
        }
        if (vector.Length == Math.Pow(2, ell1 - 1))
        {
            if (vector[0] >= n + 1)
            {
                return true;
            }
            sum = 0;
            for (int i = 0; i < vector.Length; i++)
            {
                for (int j = 0; j < Math.Log(vector.Length) / Math.Log(2); j++)
                {
                    if (i == Math.Pow(2, j))
                    {
                        sum += Math.Min(vector[i], k - 1);
                    }
                }
            }
            if (sum >= n + t)
            {
                return true;
            }
        }
        return false;
    }
//check if the child vector is a duplicate
    public static bool isDuplicate(int[] vector)
    {
        int[] vecX = vector;
        int[] tempvecX = new int[vector.Length];
        for (int l = 0; l < vector.Length; l++)
        {
            tempvecX[l] = vecX[l];
        }
        int i, j, k, size;
        String myString;
        Permutation p4;
        char[] charArrTemp, charArr;
        int numTickets = (int)(Math.Log(vector.Length)/Math.Log(2));
        int numCombos = (int)factorial(numTickets);
        for (i = 0; i < numCombos; i++)
        {
            p4 = new Permutation(numTickets, i);
            for (j = 0; j < vector.Length; j++)
            {
                charArr = new char[numTickets];
                charArrTemp = ToBinary(j);
                String charArrTempStr = new String(charArrTemp);
                size = charArrTempStr.Length;
                for (k = numTickets - 1; k >= 0; k--)
                {
                    if (size <= 0)
                    {
                        charArr[k] = '0';
                    }
                    else
                    {
                        charArr[k] = charArrTemp[size - 1];
```

```
                }
                size--;
            }
            myString = new String(p4.ApplyTo(charArr));
            vecX[j] = tempvecX[Convert.ToInt32(myString, 2)];
        }
        int[] vec1 = buildVector(vector);
        int[] vec2 = buildVector(tempvecX);
        //returns TRUE if arr1 < arr2, lexicographically
        //public static bool isLess(int[] arr1, int[] arr2)
        if (isLess(vec2, vec1))
        {
            return true;
        }
    }
    return false;
}
//Create ancestor vector
//This function looks at the second half of input parameter "vector"
//and builds a string containing its second half elements, and it's
//parent's second half elements, and it's parent's parent's second half
//elements etc...
public static int[] buildVector(int[] vector)
{
    int returnVecSize = 0;
    for(int i = (vector.Length) / 2; i > 0; i /= 2)
    {
        returnVecSize += i;
    }
    int[] vectorToReturn = new int[returnVecSize];
    ArrayList tempVector = new ArrayList();
    ArrayList tempVector2 = new ArrayList();
    for (int i = 0; i < vector.Length; i++)
    {
        tempVector.Add(vector[i]);
    }
    int placeholder = returnVecSize-1;

    for (int j = 0; j <= (int)(Math.Log(vector.Length) / Math.Log(2) - 1); j++)
    {
        //fill values from 2nd half of tempVector into vectorToReturn, from the BACK
        for (int k = tempVector.Count - 1; k >= tempVector.Count / 2; k--)
        {
            vectorToReturn[placeholder] = (int)tempVector[k];
            placeholder--;
        }
        //change tempVector to contain elements of it's parentNode
        if (tempVector2.Count > 0)
        {
            tempVector2.Clear();
        }
        for (int cnt = 0; cnt < tempVector.Count/2; cnt++)
        {
            tempVector2.Add((int)tempVector[cnt]+(int)tempVector[cnt+tempVector.Count/2]);
        }
        if (tempVector.Count > 0)
        {
            tempVector.Clear();
        }
        for (int cnt = 0; cnt < tempVector2.Count; cnt++)
        {
            tempVector.Add((int)tempVector2[cnt]);
        }
    }
    return vectorToReturn;
}

public static void printArrayList(ArrayList AL)
{
    Console.Write("(");
```

```
            for(int i=0;i<AL.Count;i++)
            {
                if (i > 0)
                {
                    Console.Write(", ");
                }
                Console.Write("{0}", (int)AL[i]);
            }
            Console.WriteLine(")");
        }

        //returns TRUE if arr1 < arr2, lexicographically
        public static bool isLess(int[] arr1, int[] arr2)
        {
            int i;
            for (i = 0; i < arr1.Length; i++)
            {
                if (arr1[i] < arr2[i])
                {
                    return true;
                }
                else if (arr1[i] > arr2[i])
                {
                    return false;
                }
            }
            return false;
        }

        //returns TRUE if arr1 = arr2
        public static bool isEqual(int[] arr1, int[] arr2)
        {
            int i;
            for (i = 0; i < arr1.Length; i++)
            {
                if (arr1[i] != arr2[i])
                    return false;
            }
            return true;
        }
    }

//An efficient permuation class obtained online
    class Permutation
    {
        private int[] data = null;
        private int order = 0;
        public Permutation(int n, int k)
        {
            this.data = new int[n];
            this.order = this.data.Length;
            // Step #1 - Find factoradic of k
            int[] factoradic = new int[n];
            for (int j = 1; j <= n; ++j)
            {
                factoradic[n - j] = k % j;
                k /= j;
            }
            // Step #2 - Convert factoradic to permuatation
            int[] temp = new int[n];
            for (int i = 0; i < n; ++i)
            {
                temp[i] = ++factoradic[i];
            }
            this.data[n - 1] = 1;  // right-most element is set to 1.
            for (int i = n - 2; i >= 0; --i)
            {
                this.data[i] = temp[i];
                for (int j = i + 1; j < n; ++j)
                {
```

```
                if (this.data[j] >= this.data[i])
                    ++this.data[j];
            }
        }
        for (int i = 0; i < n; ++i)  // put in 0-based form
        {
            --this.data[i];
        }
    }  // Permutation(n,k)
    public override string ToString()
    {
        System.Text.StringBuilder sb = new System.Text.StringBuilder();
        sb.Append("(");
        for (int i = 0; i < this.order; ++i)
        {
            sb.Append((this.data[i]+1).ToString() + "");
        }
        sb.Append(")");
        return sb.ToString();
    }
    public char[] ApplyTo(char[] arr)
    {
        if (arr.Length != this.order)
            return null;
        char[] result = new char[arr.Length];
        for (int i = 0; i < result.Length; i++)
        {
            result[i] = arr[this.data[i]];
        }
        return result;
    }
  }
}
```

APPENDIX E

Values of $P(N, \lambda)$ for small lotteries

TABLE E.1: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 6, 3, 3, 2\rangle$, $\langle 7, 3, 3, 2\rangle$, $\langle 7, 3, 4, 2\rangle$, $\langle 8, 3, 3, 2\rangle$ or $\langle 8, 3, 4, 2\rangle$ for $N = m,\ 10 \times m$, or $100 \times m$.*

λ	$\langle 6,3,3,2\rangle$	$\langle 7,3,3,2\rangle$	$\langle 7,3,4,2\rangle$	$\langle 8,3,3,2\rangle$	$\langle 8,3,4,2\rangle$
			$N = m$		
0	1.563×10^{-2}	3.877×10^{-2}	9.753×10^{-4}	6.776×10^{-2}	3.906×10^{-3}
1	9.375×10^{-2}	1.604×10^{-1}	1.155×10^{-2}	2.168×10^{-1}	3.125×10^{-2}
2	2.344×10^{-1}	2.843×10^{-1}	5.866×10^{-2}	3.036×10^{-1}	1.094×10^{-1}
3	3.125×10^{-1}	2.800×10^{-1}	1.654×10^{-1}	2.429×10^{-1}	2.188×10^{-1}
4	2.344×10^{-1}	1.654×10^{-1}	2.800×10^{-1}	1.214×10^{-1}	2.734×10^{-1}
5	9.375×10^{-2}	5.866×10^{-2}	2.843×10^{-1}	3.886×10^{-2}	2.188×10^{-1}
6	1.563×10^{-2}	1.155×10^{-2}	1.604×10^{-1}	7.771×10^{-3}	1.094×10^{-1}
7	–	9.753×10^{-4}	3.877×10^{-2}	8.881×10^{-4}	3.125×10^{-2}
8	–	–	–	4.441×10^{-5}	3.906×10^{-3}
			$N = 10 \times m$		
0	8.674×10^{-19}	7.671×10^{-15}	7.785×10^{-31}	2.041×10^{-12}	8.272×10^{-25}
10	6.539×10^{-8}	1.579×10^{-5}	5.950×10^{-17}	3.523×10^{-4}	1.362×10^{-12}
20	3.636×10^{-3}	3.345×10^{-2}	4.678×10^{-9}	7.932×10^{-2}	2.924×10^{-6}
30	1.026×10^{-1}	5.937×10^{-2}	3.081×10^{-4}	2.087×10^{-2}	7.338×10^{-3}
40	3.636×10^{-3}	3.081×10^{-4}	5.937×10^{-2}	2.652×10^{-5}	8.893×10^{-2}
50	6.539×10^{-8}	4.678×10^{-9}	3.345×10^{-2}	2.295×10^{-10}	7.338×10^{-3}
60	8.674×10^{-19}	5.950×10^{-17}	1.579×10^{-5}	9.589×10^{-18}	2.924×10^{-6}
70	–	7.785×10^{-31}	7.671×10^{-15}	4.683×10^{-28}	1.362×10^{-12}
80	–	–	–	2.982×10^{-44}	8.272×10^{-25}
			$N = 100 \times m$		
0	2.410×10^{-181}	7.051×10^{-142}	8.179×10^{-302}	1.252×10^{-117}	1.500×10^{-241}
100	2.678×10^{-65}	2.052×10^{-41}	1.182×10^{-155}	6.862×10^{-28}	5.116×10^{-112}
200	6.039×10^{-17}	3.575×10^{-7}	1.022×10^{-75}	2.497×10^{-3}	1.159×10^{-47}
300	3.256×10^{-2}	2.491×10^{-4}	3.536×10^{-27}	1.073×10^{-8}	3.097×10^{-13}
400	6.039×10^{-17}	3.536×10^{-27}	2.491×10^{-4}	1.570×10^{-37}	2.820×10^{-2}
500	2.678×10^{-65}	1.022×10^{-75}	3.575×10^{-7}	2.770×10^{-88}	3.097×10^{-13}
600	2.410×10^{-181}	1.182×10^{-155}	2.052×10^{-41}	0	1.159×10^{-47}
700	–	8.179×10^{-302}	7.051×10^{-142}	0	5.116×10^{-112}
800	–	–	–	0	1.500×10^{-241}

TABLE E.2: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 8, 3, 5, 2\rangle$, $\langle 8, 4, 3, 2\rangle$, $\langle 8, 4, 4, 2\rangle$, $\langle 8, 4, 4, 3\rangle$ or $\langle 9, 3, 3, 2\rangle$ for $N = m$, $10 \times m$, or $100 \times m$.*

λ	$\langle 8, 3, 5, 2\rangle$	$\langle 8, 4, 3, 2\rangle$	$\langle 8, 4, 4, 2\rangle$	$\langle 8, 4, 4, 3\rangle$	$\langle 9, 3, 3, 2\rangle$
			$N = m$		
0	4.441×10^{-5}	3.689×10^{-3}	1.210×10^{-5}	1.080×10^{-1}	9.947×10^{-2}
1	8.881×10^{-4}	2.993×10^{-2}	3.018×10^{-4}	2.771×10^{-1}	2.617×10^{-1}
2	7.771×10^{-3}	1.063×10^{-1}	3.293×10^{-3}	3.111×10^{-1}	3.060×10^{-1}
3	3.886×10^{-2}	2.156×10^{-1}	2.053×10^{-2}	1.996×10^{-1}	2.087×10^{-1}
4	1.214×10^{-1}	2.734×10^{-1}	8.002×10^{-2}	8.002×10^{-2}	9.150×10^{-2}
5	2.429×10^{-1}	2.219×10^{-1}	1.996×10^{-1}	2.053×10^{-2}	2.675×10^{-2}
6	3.036×10^{-1}	1.125×10^{-1}	3.111×10^{-1}	3.293×10^{-3}	5.212×10^{-3}
7	2.168×10^{-1}	3.261×10^{-2}	2.771×10^{-1}	3.018×10^{-4}	6.530×10^{-4}
8	6.776×10^{-2}	4.135×10^{-3}	1.080×10^{-1}	1.210×10^{-5}	4.772×10^{-5}
9	–	–	–	–	1.550×10^{-6}
			$N = 10 \times m$		
0	2.982×10^{-44}	4.662×10^{-25}	6.731×10^{-50}	2.159×10^{-10}	9.486×10^{-11}
10	4.683×10^{-28}	8.854×10^{-13}	9.614×10^{-33}	4.097×10^{-3}	2.471×10^{-3}
20	9.589×10^{-18}	2.193×10^{-6}	1.791×10^{-21}	1.014×10^{-1}	1.003×10^{-1}
30	2.295×10^{-10}	6.348×10^{-3}	3.898×10^{-13}	2.934×10^{-3}	6.031×10^{-3}
40	2.652×10^{-5}	8.875×10^{-2}	4.098×10^{-7}	4.098×10^{-7}	2.443×10^{-6}
50	2.087×10^{-2}	8.448×10^{-3}	2.934×10^{-3}	3.898×10^{-13}	1.112×10^{-11}
60	7.932×10^{-2}	3.884×10^{-6}	1.014×10^{-1}	1.791×10^{-21}	5.696×10^{-19}
70	3.523×10^{-4}	2.086×10^{-12}	4.097×10^{-3}	9.614×10^{-33}	1.965×10^{-28}
80	2.041×10^{-12}	1.462×10^{-24}	2.159×10^{-10}	6.731×10^{-50}	1.004×10^{-40}
90	–	–	–	–	7.993×10^{-59}
			$N = 100 \times m$		
0	$0.000 \times 10^{+00}$	4.846×10^{-244}	$0.000 \times 10^{+00}$	2.199×10^{-97}	5.897×10^{-101}
10	$0.000 \times 10^{+00}$	1.563×10^{-221}	$0.000 \times 10^{+00}$	7.089×10^{-80}	2.454×10^{-83}
100	$0.000 \times 10^{+00}$	6.898×10^{-114}	$0.000 \times 10^{+00}$	3.108×10^{-17}	2.124×10^{-19}
200	$0.000 \times 10^{+00}$	6.519×10^{-49}	$0.000 \times 10^{+00}$	2.916×10^{-2}	3.070×10^{-2}
300	2.770×10^{-88}	7.271×10^{-14}	5.542×10^{-116}	3.229×10^{-17}	5.809×10^{-14}
400	1.570×10^{-37}	2.763×10^{-2}	1.218×10^{-55}	1.218×10^{-55}	1.105×10^{-47}
500	1.073×10^{-8}	1.266×10^{-12}	3.229×10^{-17}	5.542×10^{-116}	0
600	2.497×10^{-3}	1.977×10^{-46}	2.916×10^{-2}	0	0
700	6.862×10^{-28}	3.642×10^{-110}	3.108×10^{-17}	0	0
800	1.252×10^{-117}	4.455×10^{-239}	2.199×10^{-97}	0	0
900	–	–	–	–	0

TABLE E.3: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 9, 3, 4, 2\rangle$, $\langle 9, 3, 5, 2\rangle$, $\langle 9, 3, 6, 2\rangle$, $\langle 9, 4, 3, 2\rangle$ or $\langle 9, 4, 4, 2\rangle$ for $N = m$, $10 \times m$, or $100 \times m$.*

λ	$\langle 9, 3, 4, 2\rangle$	$\langle 9, 3, 5, 2\rangle$	$\langle 9, 3, 6, 2\rangle$	$\langle 9, 4, 3, 2\rangle$	$\langle 9, 4, 4, 2\rangle$
$N = m$					
0	9.380×10^{-3}	2.916×10^{-4}	1.550×10^{-6}	9.103×10^{-3}	9.453×10^{-5}
1	5.741×10^{-2}	3.859×10^{-3}	4.772×10^{-5}	5.617×10^{-2}	1.531×10^{-3}
2	1.561×10^{-1}	2.270×10^{-2}	6.530×10^{-4}	1.540×10^{-1}	1.103×10^{-2}
3	2.478×10^{-1}	7.790×10^{-2}	5.212×10^{-3}	2.464×10^{-1}	4.631×10^{-2}
4	2.527×10^{-1}	1.718×10^{-1}	2.675×10^{-2}	2.534×10^{-1}	1.250×10^{-1}
5	1.718×10^{-1}	2.527×10^{-1}	9.150×10^{-2}	1.738×10^{-1}	2.251×10^{-1}
6	7.790×10^{-2}	2.478×10^{-1}	2.087×10^{-1}	7.942×10^{-2}	2.701×10^{-1}
7	2.270×10^{-2}	1.561×10^{-1}	3.060×10^{-1}	2.334×10^{-2}	2.083×10^{-1}
8	3.859×10^{-3}	5.741×10^{-2}	2.617×10^{-1}	4.000×10^{-3}	9.376×10^{-2}
9	2.916×10^{-4}	9.380×10^{-3}	9.947×10^{-2}	3.047×10^{-4}	1.875×10^{-2}
$N = 10 \times m$					
0	5.274×10^{-21}	4.446×10^{-36}	7.993×10^{-59}	3.905×10^{-21}	5.699×10^{-41}
10	6.378×10^{-10}	1.203×10^{-21}	1.004×10^{-40}	5.128×10^{-10}	1.164×10^{-25}
20	1.202×10^{-4}	5.072×10^{-13}	1.965×10^{-28}	1.049×10^{-4}	3.704×10^{-16}
30	3.354×10^{-2}	3.168×10^{-7}	5.696×10^{-19}	3.179×10^{-2}	1.746×10^{-9}
40	6.306×10^{-2}	1.333×10^{-3}	1.112×10^{-11}	6.489×10^{-2}	5.545×10^{-5}
50	1.333×10^{-3}	6.306×10^{-2}	2.443×10^{-6}	1.489×10^{-3}	1.980×10^{-2}
60	3.168×10^{-7}	3.354×10^{-2}	6.031×10^{-3}	3.843×10^{-7}	7.947×10^{-2}
70	5.072×10^{-13}	1.202×10^{-4}	1.003×10^{-1}	6.681×10^{-13}	2.149×10^{-3}
80	1.203×10^{-21}	6.378×10^{-10}	2.471×10^{-3}	1.721×10^{-21}	8.610×10^{-8}
90	4.446×10^{-36}	5.274×10^{-21}	9.486×10^{-11}	6.904×10^{-36}	5.374×10^{-18}
$N = 100 \times m$					
0	1.666×10^{-203}	0	0	8.253×10^{-205}	0
10	3.218×10^{-182}	0	0	1.731×10^{-183}	0
100	2.786×10^{-85}	0	0	3.142×10^{-86}	0
200	1.870×10^{-31}	3.360×10^{-115}	0	4.803×10^{-32}	0
300	1.643×10^{-6}	9.298×10^{-57}	0	9.611×10^{-7}	2.404×10^{-79}
400	1.452×10^{-3}	2.587×10^{-20}	0	1.933×10^{-3}	4.013×10^{-34}
500	2.587×10^{-20}	1.452×10^{-3}	1.105×10^{-47}	7.845×10^{-20}	1.351×10^{-8}
600	9.298×10^{-57}	1.643×10^{-6}	5.809×10^{-14}	6.421×10^{-56}	9.176×10^{-3}
700	3.360×10^{-115}	1.870×10^{-31}	3.070×10^{-2}	5.283×10^{-114}	6.265×10^{-19}
800	0	2.786×10^{-85}	2.124×10^{-19}	0	5.600×10^{-64}
900	0	1.666×10^{-203}	5.897×10^{-101}	0	2.009×10^{-173}

TABLE E.4: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 9, 4, 4, 3\rangle$, $\langle 9, 4, 5, 2\rangle$, $\langle 9, 4, 5, 3\rangle$, $\langle 9, 4, 6, 3\rangle$ or $\langle 10, 3, 3, 2\rangle$ for $N = m$, $10 \times m$, or $100 \times m$.*

λ	$\langle 9,4,4,3\rangle$	$\langle 9,4,5,2\rangle$	$\langle 9,4,5,3\rangle$	$\langle 9,4,6,3\rangle$	$\langle 10,3,3,2\rangle$
			$N = m$		
0	1.938×10^{-1}	9.923×10^{-8}	1.875×10^{-2}	2.916×10^{-4}	1.320×10^{-1}
1	3.489×10^{-1}	4.465×10^{-6}	9.376×10^{-2}	3.859×10^{-3}	2.962×10^{-1}
2	2.791×10^{-1}	8.931×10^{-5}	2.083×10^{-1}	2.270×10^{-2}	2.993×10^{-1}
3	1.302×10^{-1}	1.042×10^{-3}	2.701×10^{-1}	7.790×10^{-2}	1.792×10^{-1}
4	3.907×10^{-2}	7.814×10^{-3}	2.251×10^{-1}	1.718×10^{-1}	7.038×10^{-2}
5	7.814×10^{-3}	3.907×10^{-2}	1.250×10^{-1}	2.527×10^{-1}	1.896×10^{-2}
6	1.042×10^{-3}	1.302×10^{-1}	4.631×10^{-2}	2.478×10^{-1}	3.547×10^{-3}
7	8.931×10^{-5}	2.791×10^{-1}	1.103×10^{-2}	1.561×10^{-1}	4.550×10^{-4}
8	4.465×10^{-6}	3.489×10^{-1}	1.531×10^{-3}	5.741×10^{-2}	3.830×10^{-5}
9	9.923×10^{-8}	1.938×10^{-1}	9.453×10^{-5}	9.380×10^{-3}	1.911×10^{-6}
10	–	–	–	–	4.290×10^{-8}
			$N = 10 \times m$		
0	7.476×10^{-8}	9.255×10^{-71}	5.374×10^{-18}	4.446×10^{-36}	1.601×10^{-9}
10	4.380×10^{-2}	5.171×10^{-51}	8.610×10^{-8}	1.203×10^{-21}	9.011×10^{-3}
20	3.997×10^{-2}	4.500×10^{-37}	2.149×10^{-3}	5.072×10^{-13}	9.069×10^{-2}
30	5.404×10^{-5}	5.802×10^{-26}	7.947×10^{-2}	3.168×10^{-7}	1.616×10^{-3}
40	4.922×10^{-10}	5.040×10^{-17}	1.980×10^{-2}	1.333×10^{-3}	2.458×10^{-7}
50	5.040×10^{-17}	4.922×10^{-10}	5.545×10^{-5}	6.306×10^{-2}	5.864×10^{-13}
60	5.802×10^{-26}	5.404×10^{-5}	1.746×10^{-9}	3.354×10^{-2}	2.597×10^{-20}
70	4.500×10^{-37}	3.997×10^{-2}	3.704×10^{-16}	1.202×10^{-4}	1.804×10^{-29}
80	5.171×10^{-51}	4.380×10^{-2}	1.164×10^{-25}	6.378×10^{-10}	1.070×10^{-40}
90	9.255×10^{-71}	7.476×10^{-8}	5.699×10^{-41}	5.274×10^{-21}	1.123×10^{-54}
100	–	–	–	–	2.109×10^{-74}
			$N = 100 \times m$		
0	5.456×10^{-72}	0	2.009×10^{-173}	0	1.109×10^{-88}
10	5.106×10^{-56}	0	5.142×10^{-153}	0	9.494×10^{-72}
100	6.490×10^{-7}	0	5.600×10^{-64}	0	9.325×10^{-14}
200	3.099×10^{-6}	0	6.265×10^{-19}	3.360×10^{-115}	1.273×10^{-2}
300	1.938×10^{-34}	0	9.176×10^{-3}	9.298×10^{-57}	1.376×10^{-19}
400	0	0	1.351×10^{-8}	2.587×10^{-20}	1.657×10^{-57}
500	0	0	4.013×10^{-34}	1.452×10^{-3}	0
600	0	1.938×10^{-34}	2.404×10^{-79}	1.643×10^{-6}	0
700	0	3.099×10^{-6}	0	1.870×10^{-31}	0
800	0	6.490×10^{-7}	0	2.786×10^{-85}	0
900	0	5.456×10^{-72}	0	1.666×10^{-203}	0
1000	–	–	–	–	0

TABLE E.5: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 10, 3, 4, 2\rangle$, $\langle 10, 3, 5, 2\rangle$, $\langle 10, 3, 6, 2\rangle$, $\langle 10, 3, 6, 7, 2\rangle$ or $\langle 10, 4, 3, 2\rangle$ for $N = m,\ 10 \times m$, or $100 \times m$.*

λ	$\langle 10, 3, 4, 2\rangle$	$\langle 10, 3, 5, 2\rangle$	$\langle 10, 3, 6, 2\rangle$	$\langle 10, 3, 7, 2\rangle$	$\langle 10, 4, 3, 2\rangle$
$N = m$					
0	1.734×10^{-2}	9.766×10^{-4}	1.694×10^{-5}	4.290×10^{-8}	1.703×10^{-2}
1	8.671×10^{-2}	9.766×10^{-3}	3.387×10^{-4}	1.911×10^{-6}	8.563×10^{-2}
2	1.951×10^{-1}	4.395×10^{-2}	3.048×10^{-3}	3.830×10^{-5}	1.937×10^{-1}
3	2.601×10^{-1}	1.172×10^{-1}	1.626×10^{-2}	4.550×10^{-4}	2.597×10^{-1}
4	2.276×10^{-1}	2.051×10^{-1}	5.690×10^{-2}	3.547×10^{-3}	2.284×10^{-1}
5	1.366×10^{-1}	2.461×10^{-1}	1.366×10^{-1}	1.896×10^{-2}	1.378×10^{-1}
6	5.690×10^{-2}	2.051×10^{-1}	2.276×10^{-1}	7.038×10^{-2}	5.772×10^{-2}
7	1.626×10^{-2}	1.172×10^{-1}	2.601×10^{-1}	1.792×10^{-1}	1.658×10^{-2}
8	3.048×10^{-3}	4.395×10^{-2}	1.951×10^{-1}	2.993×10^{-1}	3.125×10^{-3}
9	3.387×10^{-4}	9.766×10^{-3}	8.671×10^{-2}	2.962×10^{-1}	3.491×10^{-4}
10	1.694×10^{-5}	9.766×10^{-4}	1.734×10^{-2}	1.320×10^{-1}	1.755×10^{-5}
$N = 10 \times m$					
0	2.460×10^{-18}	7.889×10^{-31}	1.940×10^{-48}	2.109×10^{-74}	2.057×10^{-18}
10	4.158×10^{-8}	1.366×10^{-17}	3.439×10^{-32}	1.123×10^{-54}	3.669×10^{-8}
20	1.257×10^{-3}	4.228×10^{-10}	1.091×10^{-21}	1.070×10^{-40}	1.170×10^{-3}
30	6.728×10^{-2}	2.317×10^{-5}	6.119×10^{-14}	1.804×10^{-29}	6.607×10^{-2}
40	3.075×10^{-2}	1.084×10^{-2}	2.933×10^{-8}	2.597×10^{-20}	3.186×10^{-2}
50	2.204×10^{-4}	7.959×10^{-2}	2.204×10^{-4}	5.864×10^{-13}	2.409×10^{-4}
60	2.933×10^{-8}	1.084×10^{-2}	3.075×10^{-2}	2.458×10^{-7}	3.381×10^{-8}
70	6.119×10^{-14}	2.317×10^{-5}	6.728×10^{-2}	1.616×10^{-3}	7.444×10^{-14}
80	1.091×10^{-21}	4.228×10^{-10}	1.257×10^{-3}	9.069×10^{-2}	1.399×10^{-21}
90	3.439×10^{-32}	1.366×10^{-17}	4.158×10^{-8}	9.011×10^{-3}	4.657×10^{-32}
100	1.940×10^{-48}	7.889×10^{-31}	2.460×10^{-18}	1.601×10^{-9}	2.771×10^{-48}
$N = 100 \times m$					
0	8.105×10^{-177}	9.333×10^{-302}	0	0	1.357×10^{-177}
10	2.085×10^{-156}	2.458×10^{-278}	0	0	3.682×10^{-157}
100	4.082×10^{-67}	5.959×10^{-162}	0	0	1.167×10^{-67}
200	3.337×10^{-21}	6.176×10^{-86}	0	0	1.630×10^{-21}
300	2.160×10^{-3}	5.066×10^{-38}	0	0	1.801×10^{-3}
400	1.558×10^{-6}	4.634×10^{-11}	9.698×10^{-67}	0	2.220×10^{-6}
500	6.692×10^{-28}	2.523×10^{-2}	6.692×10^{-28}	0	1.628×10^{-27}
600	9.698×10^{-67}	4.634×10^{-11}	1.558×10^{-6}	1.657×10^{-57}	4.029×10^{-66}
700	0	5.066×10^{-38}	2.160×10^{-3}	1.376×10^{-19}	0
800	0	6.176×10^{-86}	3.337×10^{-21}	1.273×10^{-2}	0
900	0	5.959×10^{-162}	4.082×10^{-67}	9.325×10^{-14}	0
1000	0	9.333×10^{-302}	8.105×10^{-177}	1.109×10^{-88}	0

TABLE E.6: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 10, 4, 4, 2\rangle$, $\langle 10, 4, 4, 3\rangle$, $\langle 10, 4, 5, 2\rangle$, $\langle 10, 4, 5, 3\rangle$ or $\langle 10, 4, 6, 2\rangle$ for $N = m$, $10 \times m$, or $100 \times m$.*

λ	$\langle 10, 4, 4, 2\rangle$	$\langle 10, 4, 4, 3\rangle$	$\langle 10, 4, 5, 2\rangle$	$\langle 10, 4, 5, 3\rangle$	$\langle 10, 4, 6, 2\rangle$
			$N = m$		
0	3.590×10^{-4}	2.815×10^{-1}	1.519×10^{-6}	4.799×10^{-2}	5.718×10^{-10}
1	4.345×10^{-3}	3.804×10^{-1}	4.280×10^{-5}	1.703×10^{-1}	4.231×10^{-8}
2	2.367×10^{-2}	2.314×10^{-1}	5.427×10^{-4}	2.719×10^{-1}	1.409×10^{-6}
3	7.641×10^{-2}	8.337×10^{-2}	4.079×10^{-3}	2.573×10^{-1}	2.780×10^{-5}
4	1.619×10^{-1}	1.972×10^{-2}	2.012×10^{-2}	1.598×10^{-1}	3.600×10^{-4}
5	2.351×10^{-1}	3.197×10^{-3}	6.803×10^{-2}	6.803×10^{-2}	3.197×10^{-3}
6	2.372×10^{-1}	3.600×10^{-4}	1.598×10^{-1}	2.012×10^{-2}	1.972×10^{-2}
7	1.641×10^{-1}	2.780×10^{-5}	2.573×10^{-1}	4.079×10^{-3}	8.337×10^{-2}
8	7.448×10^{-2}	1.409×10^{-6}	2.719×10^{-1}	5.427×10^{-4}	2.314×10^{-1}
9	2.004×10^{-2}	4.231×10^{-8}	1.703×10^{-1}	4.280×10^{-5}	3.804×10^{-1}
10	2.425×10^{-3}	5.718×10^{-10}	4.799×10^{-2}	1.519×10^{-6}	2.815×10^{-1}
			$N = 10 \times m$		
0	3.552×10^{-35}	3.128×10^{-6}	6.521×10^{-59}	6.475×10^{-14}	3.733×10^{-93}
10	4.154×10^{-21}	1.100×10^{-1}	3.567×10^{-41}	3.547×10^{-5}	3.182×10^{-71}
20	8.691×10^{-13}	6.914×10^{-3}	3.490×10^{-29}	3.475×10^{-2}	4.851×10^{-55}
30	3.218×10^{-7}	7.695×10^{-7}	6.044×10^{-20}	6.027×10^{-2}	1.309×10^{-41}
40	1.018×10^{-3}	7.314×10^{-13}	8.939×10^{-13}	8.926×10^{-4}	3.017×10^{-30}
50	5.047×10^{-2}	1.090×10^{-20}	2.073×10^{-7}	2.073×10^{-7}	1.090×10^{-20}
60	4.646×10^{-2}	3.017×10^{-30}	8.926×10^{-4}	8.939×10^{-13}	7.314×10^{-13}
70	6.708×10^{-4}	1.309×10^{-41}	6.027×10^{-2}	6.044×10^{-20}	7.695×10^{-7}
80	8.270×10^{-8}	4.851×10^{-55}	3.475×10^{-2}	3.490×10^{-29}	6.914×10^{-3}
90	1.805×10^{-14}	3.182×10^{-71}	3.547×10^{-5}	3.567×10^{-41}	1.100×10^{-1}
100	7.045×10^{-27}	3.733×10^{-93}	6.475×10^{-14}	6.521×10^{-59}	3.128×10^{-6}
			$N = 100 \times m$		
0	0	8.963×10^{-56}	0	1.295×10^{-132}	0
100	0	6.830×10^{-3}	0	8.330×10^{-38}	0
200	8.314×10^{-113}	8.448×10^{-14}	0	8.695×10^{-7}	0
300	1.353×10^{-56}	8.271×10^{-53}	0	7.184×10^{-4}	0
400	2.454×10^{-21}	0	0	6.618×10^{-22}	0
500	2.650×10^{-4}	0	3.628×10^{-58}	3.628×10^{-58}	0
600	9.656×10^{-5}	0	6.618×10^{-22}	0	0
700	2.094×10^{-23}	0	7.184×10^{-4}	0	8.271×10^{-53}
800	5.063×10^{-63}	0	8.695×10^{-7}	0	8.448×10^{-14}
900	9.689×10^{-131}	0	8.330×10^{-38}	0	6.830×10^{-3}
1000	3.010×10^{-262}	0	1.295×10^{-132}	0	8.963×10^{-56}

TABLE E.7: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 10, 4, 6, 3\rangle$, $\langle 10, 4, 7, 2\rangle$, $\langle 10, 5, 3, 2\rangle$, $\langle 10, 5, 4, 2\rangle$ or $\langle 10, 5, 4, 3\rangle$ for $N = m,\ 10 \times m$, or $100 \times m$.*

λ	$\langle 10, 4, 6, 3\rangle$	$\langle 10, 4, 7, 2\rangle$	$\langle 10, 5, 3, 2\rangle$	$\langle 10, 5, 4, 2\rangle$	$\langle 10, 5, 4, 3\rangle$
			$N = m$		
0	2.425×10^{-3}	1.694×10^{-5}	8.457×10^{-4}	1.473×10^{-6}	4.747×10^{-2}
1	2.004×10^{-2}	3.387×10^{-4}	8.702×10^{-3}	4.169×10^{-5}	1.691×10^{-1}
2	7.448×10^{-2}	3.048×10^{-3}	4.029×10^{-2}	5.309×10^{-4}	2.712×10^{-1}
3	1.641×10^{-1}	1.626×10^{-2}	1.106×10^{-1}	4.006×10^{-3}	2.577×10^{-1}
4	2.372×10^{-1}	5.690×10^{-2}	1.991×10^{-1}	1.984×10^{-2}	1.607×10^{-1}
5	2.351×10^{-1}	1.366×10^{-1}	2.458×10^{-1}	6.736×10^{-2}	6.869×10^{-2}
6	1.619×10^{-1}	2.276×10^{-1}	2.108×10^{-1}	1.589×10^{-1}	2.040×10^{-2}
7	7.641×10^{-2}	2.601×10^{-1}	1.240×10^{-1}	2.569×10^{-1}	4.153×10^{-3}
8	2.367×10^{-2}	1.951×10^{-1}	4.783×10^{-2}	2.726×10^{-1}	5.548×10^{-4}
9	4.345×10^{-3}	8.671×10^{-2}	1.094×10^{-2}	1.714×10^{-1}	4.393×10^{-5}
10	3.590×10^{-4}	1.734×10^{-2}	1.125×10^{-3}	4.851×10^{-2}	1.565×10^{-6}
			$N = 10 \times m$		
0	7.045×10^{-27}	1.940×10^{-48}	1.871×10^{-31}	4.814×10^{-59}	5.815×10^{-14}
10	1.805×10^{-14}	3.439×10^{-32}	4.310×10^{-18}	2.744×10^{-41}	3.319×10^{-5}
20	8.270×10^{-8}	1.091×10^{-21}	1.776×10^{-10}	2.797×10^{-29}	3.388×10^{-2}
30	6.708×10^{-4}	6.119×10^{-14}	1.295×10^{-5}	5.048×10^{-20}	6.121×10^{-2}
40	4.646×10^{-2}	2.933×10^{-8}	8.066×10^{-3}	7.778×10^{-13}	9.445×10^{-4}
50	5.047×10^{-2}	2.204×10^{-4}	7.878×10^{-2}	1.880×10^{-7}	2.286×10^{-7}
60	1.018×10^{-3}	3.075×10^{-2}	1.428×10^{-2}	8.432×10^{-4}	1.027×10^{-12}
70	3.218×10^{-7}	6.728×10^{-2}	4.062×10^{-5}	5.933×10^{-2}	7.233×10^{-20}
80	8.691×10^{-13}	1.257×10^{-3}	9.863×10^{-10}	3.564×10^{-2}	4.352×10^{-29}
90	4.154×10^{-21}	4.158×10^{-8}	4.239×10^{-17}	3.790×10^{-5}	4.634×10^{-41}
100	3.552×10^{-35}	2.460×10^{-18}	3.259×10^{-30}	7.210×10^{-14}	8.826×10^{-59}
			$N = 100 \times m$		
0	3.010×10^{-262}	0	5.261×10^{-308}	0	4.417×10^{-133}
10	1.173×10^{-239}	0	1.844×10^{-284}	0	3.836×10^{-114}
100	9.689×10^{-131}	0	5.850×10^{-167}	0	4.281×10^{-38}
200	5.063×10^{-63}	0	1.056×10^{-89}	0	6.734×10^{-7}
300	2.094×10^{-23}	0	1.508×10^{-40}	0	8.385×10^{-4}
400	9.656×10^{-5}	9.698×10^{-67}	2.403×10^{-12}	0	1.164×10^{-21}
500	2.650×10^{-4}	6.692×10^{-28}	2.278×10^{-2}	1.362×10^{-58}	9.619×10^{-58}
600	2.454×10^{-21}	1.558×10^{-6}	7.287×10^{-10}	3.746×10^{-22}	0
700	1.353×10^{-56}	2.160×10^{-3}	1.387×10^{-35}	6.133×10^{-4}	0
800	8.314×10^{-113}	3.337×10^{-21}	2.945×10^{-82}	1.120×10^{-6}	0
900	0	4.082×10^{-67}	4.949×10^{-157}	1.618×10^{-37}	0
1000	0	8.105×10^{-177}	1.350×10^{-295}	3.794×10^{-132}	0

TABLE E.8: *The probability, $P(N, \lambda)$, of λ participants winning a k-prize in the lottery $\langle 10, 5, 5, 2\rangle$, $\langle 10, 5, 5, 3\rangle$ or $\langle 10, 5, 5, 4\rangle$ for $N = m$, $10 \times m$, or $100 \times m$.*

λ	$\langle 10, 5, 5, 2\rangle$	$\langle 10, 5, 5, 3\rangle$	$\langle 10, 5, 5, 4\rangle$
		$N = m$	
0	1.367×10^{-10}	9.766×10^{-4}	3.366×10^{-1}
1	1.188×10^{-8}	9.766×10^{-3}	3.872×10^{-1}
2	4.647×10^{-7}	4.395×10^{-2}	2.005×10^{-1}
3	1.077×10^{-5}	1.172×10^{-1}	6.150×10^{-2}
4	1.639×10^{-4}	2.051×10^{-1}	1.238×10^{-2}
5	1.709×10^{-3}	2.461×10^{-1}	1.709×10^{-3}
6	1.238×10^{-2}	2.051×10^{-1}	1.639×10^{-4}
7	6.150×10^{-2}	1.172×10^{-1}	1.077×10^{-5}
8	2.005×10^{-1}	4.395×10^{-2}	4.647×10^{-7}
9	3.872×10^{-1}	9.766×10^{-3}	1.188×10^{-8}
10	3.366×10^{-1}	9.766×10^{-4}	1.367×10^{-10}
		$N = 10 \times m$	
0	2.277×10^{-99}	7.889×10^{-31}	1.865×10^{-5}
10	9.704×10^{-77}	1.366×10^{-17}	1.311×10^{-1}
20	7.398×10^{-60}	4.228×10^{-10}	1.649×10^{-3}
30	9.983×10^{-46}	2.317×10^{-5}	3.670×10^{-8}
40	1.150×10^{-33}	1.084×10^{-2}	6.975×10^{-15}
50	2.079×10^{-23}	7.959×10^{-2}	2.079×10^{-23}
60	6.975×10^{-15}	1.084×10^{-2}	1.150×10^{-33}
70	3.670×10^{-8}	2.317×10^{-5}	9.983×10^{-46}
80	1.649×10^{-3}	4.228×10^{-10}	7.398×10^{-60}
90	1.311×10^{-1}	1.366×10^{-17}	9.704×10^{-77}
100	1.865×10^{-5}	7.889×10^{-31}	2.277×10^{-99}
		$N = 100 \times m$	
0	0	9.333×10^{-302}	5.104×10^{-48}
10	0	2.458×10^{-278}	5.460×10^{-34}
100	0	5.959×10^{-162}	3.977×10^{-2}
200	0	6.176×10^{-86}	5.030×10^{-20}
300	0	5.066×10^{-38}	5.035×10^{-66}
400	0	4.634×10^{-11}	0
500	0	2.523×10^{-2}	0
600	0	4.634×10^{-11}	0
700	5.035×10^{-66}	5.066×10^{-38}	0
800	5.030×10^{-20}	6.176×10^{-86}	0
900	3.977×10^{-2}	5.959×10^{-162}	0
1000	5.104×10^{-48}	9.333×10^{-302}	0