

Comparison of Old and New Algorithms for s,t -Network Reliability

by

Simotwo Chepkirui Faith Zainabu



*Thesis presented in partial fulfilment of the requirements
for the degree of Master of Science in Mathematics in the
Faculty of Science at Stellenbosch University*

Supervisor: Prof. Marcel Wild

March 2020

Abstract

Comparison of Old and New Algorithms for s,t -Network Reliability

Simotwo Zainabu

*Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.*

Thesis: MSc

March 2020

Network reliability is the probability of an operative path connecting the source s with the terminal t . s, t -network reliability problems have been proven to be #P-complete. In this thesis we present some old techniques which have existed since the 1950's, as well as four new algorithms for calculating the network reliability. Because these algorithms are all coded in Mathematica as a common platform, they can be compared in a fair way.

We first consider the exhaustive enumeration method. Then we explain in detail the series-parallel reduction which is applied in the contraction-deletion algorithm. Let $ps[i]$ and $cs[i]$ be the number of cardinality i pathsets and cutsets respectively. It was long known that knowing these parameters yields the network reliability at once. Two algorithms of Wild (which more generally concern arbitrary set filters) can be used to calculate the numbers $ps[i]$ and $cs[i]$ more efficiently than previous approaches.

The comparison is based on CPU time where several random networks have been tested. The results are presented in the form of graphs and tables and we demonstrate some of the algorithms by examples.

Uittreksel

Vergelyking van oue en nuwe algoritmes vir netwerkbetroubaarheid

Simotwo Zainabu

Tesis: MSc

Maart 2020

Netwerkbetroubaarheid is die waarskynlikheid dat 'n operasionele pad die bron s met die terminale t verbind. In praktiese gevalle was die meeste probleme met netwerkbetroubaarheid $\#P$ -volledig. In hierdie tesis sal ons 'n aantal ou tegnieke wat sedert die vyftigerjare bestaan, sowel as nuwe algoritmes vir die berekening van netwerkbetroubaarheid in Mathematica kodeer. Op hierdie manier raak die algoritmes se werkverrigtings vergelykbaar.

Ons kyk eers na die uitputtende enumeratiewe metode. Vervolgens verduidelik ons die serie-parallele reduksie wat toegepas word in die stelling vir sametrekking-skrapping. Uiteindelik is ons ook in staat om vier nuwe metodes aan te bied.

Die vergelyking is gebaseer op die SVE-tyd wat deur netwerke benodig word. Die resultate word aangebied in die vorm van grafieke en tabelle en ons demonstreer enkele voorbeelde van die algoritmes.

Acknowledgements

I would like to express my very great appreciation to my supervisor Prof. Marcel Wild (Stellenbosch University) for his support during the development of this work and most importantly for making me discover the power of Mathematica. His guidance, support and friendliness helped me to get the best out of this thesis.

I would also like to thank the Mandela Rhodes Foundation for providing me with a full scholarship funding and connecting me to my mentor and friend Prof. Chet (Stellenbosch University) who inspired me throughout my stay at Stellenbosch.

My special thanks are extended to the teaching staff, non-teaching staff and fellow postgraduate students in the Mathematical Sciences Department who have made this journey fascinating, encouraged me constantly and created a conducive space during my research.

To my dear family, friends and office colleagues, my heart is full. I'm grateful for your emotional support and love that I have received from you throughout my studies.

Dedications

For my late mother Amina for investing and giving me the best foundation
for something she didn't have, education!

Contents

Abstract	i
Uittreksel	ii
Acknowledgements	iii
Dedications	iv
Contents	v
List of Figures	vii
List of Tables	x
Nomenclature	xi
1 Introduction	1
1.1 Layout of the thesis	2
1.2 Network model and basic concepts	2
2 Exhaustive state enumeration	4
2.1 Introduction	4
2.1.1 Equal edge reliabilities	5
2.1.2 Individual edge reliabilities	5
2.2 Conclusion	6
3 Series-parallel networks	7
3.1 Introduction	7
3.2 Series reduction	7
3.3 Parallel reduction	8

3.4	Illustration of the series-parallel reduction algorithm by example	9
3.5	Conclusion	12
4	Computing $nr(G)$ by contraction-deletion	13
4.1	Introduction	13
4.2	The Last-In-First-Out (LIFO) Technique	14
4.3	How the graphs G_i arise	15
4.4	Last but not least: The CDT of Moskowitz	26
5	Minpath and mincut methods	30
5.1	Introduction	30
5.2	Computing all minpaths of a network	30
5.3	Computing all mincuts of a network	32
5.4	Naive inclusion-exclusion on the minpaths	33
5.4.1	Method 1	33
5.4.2	Method 2	35
5.5	Naive inclusion-exclusion on the mincuts	35
5.6	Two methods to count a set filter	36
5.6.1	MethodA	37
5.6.2	MethodB	38
5.7	Calculating $nr(G)$ with Minpath-MethodA	38
5.8	Calculating $nr(G)$ with Minpath-MethodB	39
5.9	Calculating $nr(G)$ with Mincut-MethodA or Mincut-MethodB	43
6	Comparison of algorithms	44
6.1	Introduction	44
6.2	Computational results of the Mathematica algorithms	45
6.3	Algorithm recommendation	48
A	Parameters for random networks used to test FindPath command and SimpleMincut method	50
B	Mincuts and Minpaths	52
C	Random networks tested	53
	List of References	59

List of Figures

1.1	Our running example network G_0	3
2.1	Undirected bridge network.	5
3.1	Series reduction of a network.	8
3.2	8
3.3	Graphical illustration of a complete series-parallel reduction with ne=17 and nv=12.	9
4.1	The LIFO stack for processing G_0	15
4.2	Reduction of G_0	16
4.3	Reduction of G_2	17
4.4	Complete reduction of G_4 yielding $poly_4$	18
4.5	Reduction of G_3	19
4.6	Complete reduction of G_8 yielding $poly_8$	20
4.7	Reduction of G_1	21
4.8	Complete reduction of G_{12} yielding $poly_{12}$	22
4.9	Reduction of G_{11}	23
4.10	Reduction of G_{16}	24
4.11	Complete reduction of G_{18} yielding $poly_{18}$	25
4.12	Binary tree of the G_0 subtasks	26
5.1	Running example network G_0	31
5.2	CPU time for generating all minpaths using Mathematica's <i>FindPath</i> command, using a logarithmic scale.	32
5.3	CPU time for generating mincuts using SimpleMincut algorithm, again using a logarithmic scale.	33
5.4	34
6.1	Example of a 2-connected network.	44

6.2	Example of a tree-like network.	45
6.3	Graphical analysis of the CPU time (in seconds) against number of edges for 2-connected networks.	46
6.4	Graphical analysis of the CPU time (in seconds) against number of edges for tree-like networks.	47
C.1	$G=([5],[8])$	53
C.2	$G=([6],[9])$	53
C.3	$G=([7],[11])$	53
C.4	$G=([7],[13])$	53
C.5	$G=([8],[14])$	54
C.6	$G=([8],[15])$	54
C.7	$G=([9],[16])$	54
C.8	$G=([9],[18])$	54
C.9	$G=([10],[20])$	54
C.10	$G=([11],[22])$	54
C.11	$G=([11],[23])$	55
C.12	$G=([7],[25])$	55
C.13	$G=([12],[27])$	55
C.14	$G=([12],[28])$	55
C.15	$G=([13],[30])$	55
C.16	$G=([14],[32])$	55
C.17	$G=([16],[35])$	56
C.18	$G=([22],[45])$	56
C.19	$G=([9],[12])$	56
C.20	$G=([10],[16])$	56
C.21	$G=([13],[18])$	56
C.22	$G=([14],[20])$	56
C.23	$G=([15],[22])$	57
C.24	$G=([20],[28])$	57
C.25	$G=([22],[31])$	57
C.26	$G=([26],[35])$	57
C.27	$G=([30],[38])$	57
C.28	$G=([32],[43])$	57
C.29	$G=([35],[46])$	58
C.30	$G=([38],[49])$	58

LIST OF FIGURES

ix

C.31 $G=([42],[54])$	58
C.32 $G=([47],[60])$	58

List of Tables

2.1	The sixteen operational states in G	6
5.1	Descriptive facets-to-faces algorithm	37
5.2	Bitstrings corresponding to subsets of r_3	37
6.1	CPU time (in seconds) for the Mathematica programs on 2-connected networks	46
6.2	CPU time (in seconds) for the Mathematica programs on tree-like networks	47
6.3	CPU time (in seconds) for the Mathematica programs on series-parallel network, $n_e=17$	48
6.4	Method recommendation when calculating the network reliability	48
A.1	Parameters and CPU time of the random networks tested using FindPath command	50
A.2	Parameters and CPU time of the random networks tested using SimpleMincut algorithm	51
B.1	The transversal e-algorithm compact representation of the family of all cutsets X as disjoint union of thirty two 012e-rows r_1 to r_{32} .	52

Nomenclature

Combinatorics

$G = (V, E)$ A network with vertex set V and edge set E .

ne The number of edges of G .

nv The number of vertices of G .

s The source vertex of G .

t The terminal vertex of G .

$nr(G)$ The network reliability of G .

$nf(G)$ The network fallibility of G .

$cs[i]$ The number of cutsets of cardinality i .

$ps[i]$ The number of pathsets of cardinality i .

$p(e)$ The edge reliability of an edge e of G .

$G - e$ The network obtained by deleting edge e of G .

$G.e$ The network obtained by contracting edge e of G .

Chapter 1

Introduction

Reliability analysis is a complex combinatorial study which has received significant research due to its relevance in many practical areas such as data communication networks, computer networks, electrical power transmission and distributions systems, etc [1, 5].

The edges and vertices of a network can take either failed or operative states. Several techniques since the 1950's [2, 3] have been used to calculate the network reliability, i.e. the probability of an operative path connecting the source s with the terminal t . There are no known algorithms which ensure polynomial running time. More specifically, the problem is known to be #P-complete [1, 4, 6].

Several authors have proposed various algorithms coded in different programming languages leaving a quandary as to which is the most efficient. In order to achieve comparability we present a variety of old and new algorithms freshly programmed in Mathematica.

Let us briefly glimpse the broader picture. Apart from our specific network model (defined in Section 1.2) many other kinds of networks and various notions of network reliability can be considered. In real life the networks are much larger than the ones considered in this thesis, and therefore one resorts to merely *approximating* the network reliability.

1.1 Layout of the thesis

The sequel of Chapter 1 introduces basic theoretic concepts and assumptions required throughout this thesis. Herein, we define the network model to be used and other important definitions.

In Chapter 2 we explain the exhaustive state enumeration algorithm for computing the network reliability.

In Chapter 3 we concentrate on the special case of series-parallel networks which will show up repeatedly in chapter 4.

Chapter 4 is dedicated to the contraction-deletion method to compute the network reliability. We illustrate this method meticulously by many pictures.

In Chapter 5 we sketch two new algorithms that use the transversal e-algorithm [8] and the facets-to-faces algorithm [7] respectively to compute the network reliability.

In Chapter 6 we test 34 networks and provide computational results for the methods presented in Chapter 2, Chapter 4 and Chapter 5.

1.2 Network model and basic concepts

In the literature 'network' can mean a lot of things. Throughout the thesis we shall adopt the definition below which also allows for parallel edges but not loops.

Definition 1.2.1. A **network** $G = (V, E)$ is a finite graph where V is the set of n vertices and E is the set of m undirected edges. There are two distinct vertices, the source $s \in V$ and the terminal vertex $t \in V$. Parallel edges are allowed but not loops.

Definition 1.2.2. The **edge reliability** $p(e)$ of a particular edge e in a network G is the probability that it operates at any given moment.

Definition 1.2.3. The **network reliability** is the probability $nr = nr(G)$ that s and t are connected at any moment of inspection.

To fix ideas, suppose that when all edges are operating then travelling from s to t on the longest (circle-free) path takes less than a day. Then we postulate that in intervals of 24 hours all edges e simultaneously, but independently, change their state according to $p(e)$, and keep it for the next 24 hours.

Throughout this Master's Thesis, we shall consider the network G_0 (Figure 1.1) to illustrate our various methods to compute $nr(G)$. Thus $n_v=7$ and $n_e=11$. Recall that s and t are the source and terminal respectively.

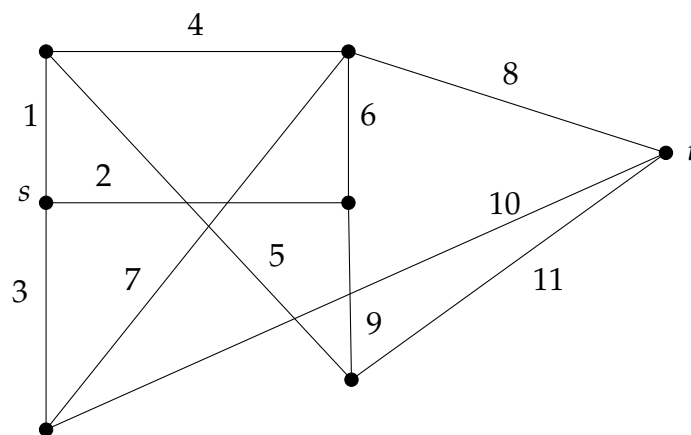


Figure 1.1: Our running example network G_0 .

Chapter 2

Exhaustive state enumeration

2.1 Introduction

The exhaustive state enumeration method is the most simple and direct method to compute the network reliability [1].

Definition 2.1.1. A **state** of a network $G = (V, E)$ is just a subset $X \subseteq E$ of edges. We call X operational if X contains a path from s to t .

Notice that X is operational if and only if deleting the edges in $E \setminus X$ yields a graph which (is not necessarily itself connected but) has s and t appearing in the same connected component. Using the Mathematica command `ConnectedComponents` this is easily decided. By generating all 2^{ne} states of $G = (V, E)$ and summing the probabilities coupled to the operational ones the network reliability is simply, but not efficiently, computed.

Example 2.1.2. We illustrate this method using Figure 2.1.

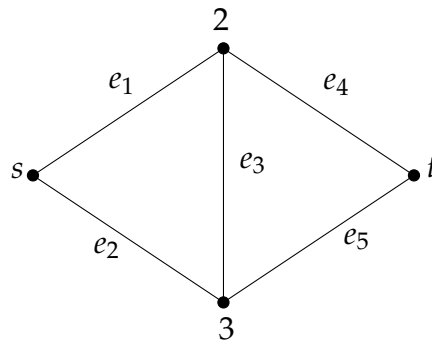


Figure 2.1: Undirected bridge network.

2.1.1 Equal edge reliabilities

Suppose first each edge e has the same edge reliability $p(e) =: p$. Then the probability that exactly the edges in a fixed state X are working is $p^m(1-p)^{ne-m}$ where $m = |X|$. Let a_m be the number of m -element states which are operational. It follows that

$$nr(G) = \sum_{m=1}^{ne} a_m p^m (1-p)^{ne-m}. \quad (2.1.1)$$

A noteworthy special case arises when $p = 0.5$. Then $p^m(1-p)^{ne-m} = (0.5)^m \cdot (0.5)^{ne-m} = (0.5)^{ne}$, and so

$$nr(G) = \sum_{m=1}^{ne} a_m (0.5)^{ne} = \frac{1}{2^{ne}} \sum_{m=1}^{ne} a_m = \frac{\text{number of operational states}}{\text{number of all states}}. \quad (2.1.2)$$

For the particular network G in Figure 2.1 one has 16 operational states (see Table 2.1), and so $nr(G) = \frac{16}{32} = \frac{1}{2}$.

2.1.2 Individual edge reliabilities

Suppose now we have individual edge reliabilities p_1, \dots, p_5 . Putting $q_i = 1 - p_i$ in Table 2.1 shows the 16 operational states of G in Figure 2.1 (encoded in the well known 0,1 manner) and the probability for each. The network reliability is then computed by adding up all the probabilities in the right hand column of Table 2.1.

Table 2.1: The sixteen operational states in G

$\{e_1, e_2, e_3, e_4, e_5\}$	probability	$\{e_1, e_2, e_3, e_4, e_5\}$	probability
$\{1, 1, 1, 1, 1\}$	$p_1 p_2 p_3 p_4 p_5$	$\{1, 0, 1, 1, 0\}$	$p_1 q_2 p_3 p_4 q_5$
$\{1, 1, 1, 0, 1\}$	$p_1 p_2 p_3 q_4 p_5$	$\{1, 0, 0, 1, 1\}$	$p_1 q_2 q_3 p_4 p_5$
$\{1, 1, 1, 1, 0\}$	$p_1 p_2 p_3 p_4 q_5$	$\{1, 0, 1, 0, 1\}$	$p_1 q_2 p_3 q_4 p_5$
$\{1, 1, 0, 1, 1\}$	$p_1 p_2 q_3 p_4 p_5$	$\{1, 0, 0, 1, 0\}$	$p_1 q_2 q_3 p_4 q_5$
$\{1, 0, 1, 1, 1\}$	$p_1 q_2 p_3 p_4 p_5$	$\{0, 1, 1, 0, 1\}$	$q_1 p_2 p_3 q_4 p_5$
$\{0, 1, 1, 1, 1\}$	$q_1 p_2 p_3 p_4 p_5$	$\{0, 1, 1, 1, 0\}$	$q_1 p_2 p_3 p_4 q_5$
$\{1, 1, 0, 1, 0\}$	$p_1 p_2 q_3 p_4 q_5$	$\{0, 1, 0, 1, 1\}$	$q_1 p_2 q_3 p_4 p_5$
$\{1, 1, 0, 0, 1\}$	$p_1 p_2 q_3 q_4 p_5$	$\{0, 1, 0, 0, 1\}$	$q_1 p_2 q_3 q_4 p_5$

2.2 Conclusion

The advantage of the exhaustive state enumeration is that it works for both equal and individual edge reliabilities. (We shall encounter methods which only work for equal edge reliabilities). The main disadvantage of exhaustive state enumeration is that its computational complexity is exponential in the number of edges.

Chapter 3

Series-parallel networks

3.1 Introduction

'Reduction' of a network G is any transformation method which allows the simplification of G while preserving $nr(G)$. Usually that involves reducing the number of edges, or vertices, or both [1]. In this chapter we present an algorithm that computes the network reliability of so-called series-parallel networks. For the latter a particularly smooth kind of reduction will do.

3.2 Series reduction

Definition 3.2.1. *Two edges $e_1 = (a, b)$ and $e_2 = (b, c)$ in a network G are in series if $b \notin \{s, t\}$ and b is only incident with edges e_1 and e_2 .*

The following is evident: if $p(e_1) = p_1$ and $p(e_2) = p_2$ and G^1 arises from G by substituting e_1, e_2 with $e = (a, c)$ having $p(e) := p_1 p_2$ then $nr(G^1) = nr(G)$.

Example 3.2.2. *Every edge in the graph on the left in (Figure 3.1) is of edge reliability p .*

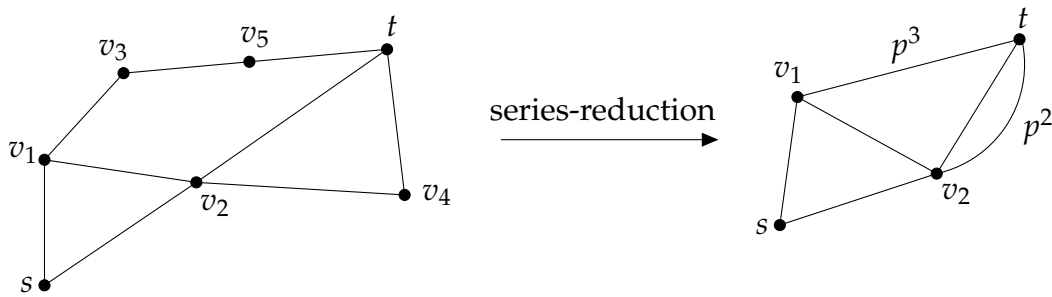


Figure 3.1: Series reduction of a network.

3.3 Parallel reduction

Definition 3.3.1. Two edges e_1 and e_2 in a network G are **in parallel** if they both connect the same end vertices.

Suppose that two parallel edges in a network G have reliability p and k respectively. Then the following possibilities can occur:

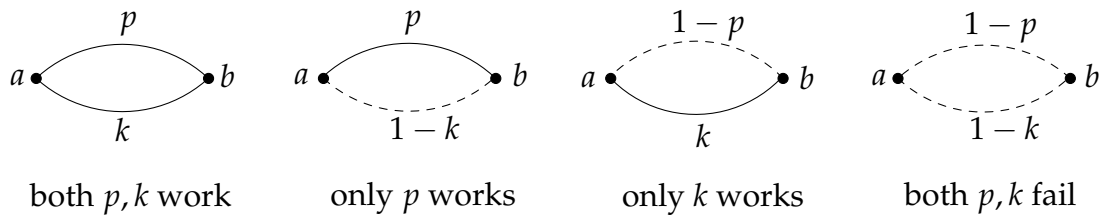


Figure 3.2

It follows that the probability that at least one edge works is

$$pk + p(1 - k) + k(1 - p) = p + k - pk.$$

If we thus replace the two edges by a single edge with edge reliability $p + k - pk$, the new network G^1 evidently satisfies $nr(G^1) = nr(G)$.

3.4 Illustration of the series-parallel reduction algorithm by example

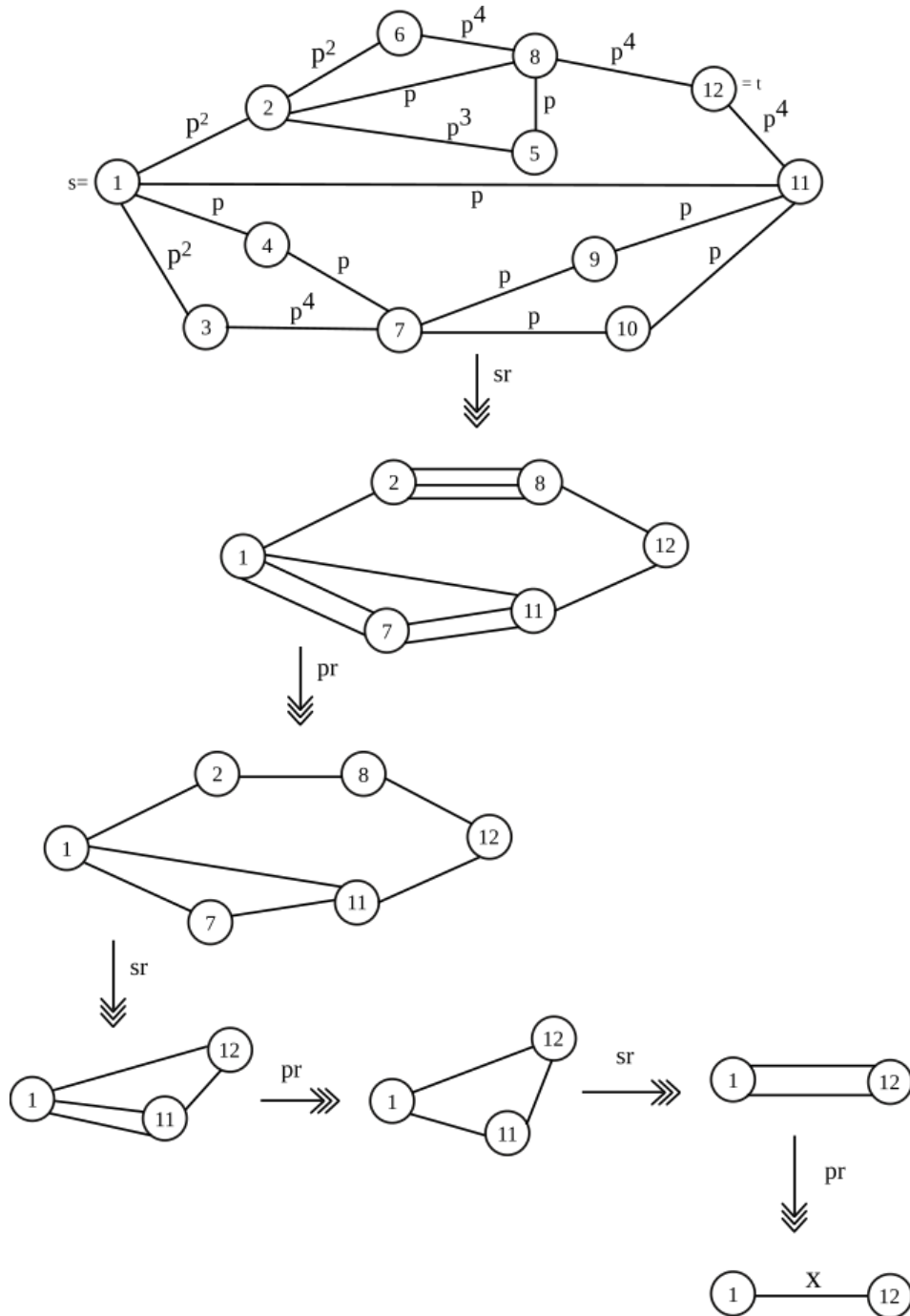


Figure 3.3: Graphical illustration of a complete series-parallel reduction with $ne=17$ and $nv=12$.

The series-parallel reduction algorithm applies series-reduction and parallel-reduction in turn until neither of them can be applied. Let us illustrate how the algorithm works on G in Figure 3.3. Suppose initially the edge reliabilities are these:

$$\left\{ \begin{array}{l} \{1,2,3,4,5,6,7,8,9,10,11,12\}, \\ \left(\begin{array}{ll} \{1,2\} & p^2 \\ \{1,3\} & p^2 \\ \{1,4\} & p \\ \{1,11\} & p \\ \{2,5\} & p^3 \\ \{2,6\} & p^2 \\ \{2,8\} & p \\ \{3,7\} & p^4 \\ \{4,7\} & p \\ \{5,8\} & p \\ \{6,8\} & p^4 \\ \{7,9\} & p \\ \{7,10\} & p \\ \{8,12\} & p^4 \\ \{9,11\} & p \\ \{10,11\} & p \\ \{11,12\} & p^4 \end{array} \right) \end{array} \right\}.$$

After series reduction:

$$\left\{ \begin{array}{l} \{1,2,7,8,11,12\}, \\ \left(\begin{array}{ll} \{1,2\} & p^2 \\ \{1,7\} & p^2 \\ \{1,7\} & p^6 \\ \{1,11\} & p \\ \{2,8\} & p \\ \{2,8\} & p^4 \\ \{2,8\} & p^6 \\ \{7,11\} & p^2 \\ \{7,11\} & p^2 \\ \{8,12\} & p^4 \\ \{11,12\} & p^4 \end{array} \right) \end{array} \right\}.$$

After parallel reduction:

$$\left\{ \{1, 2, 7, 8, 11, 12\}, \left(\begin{array}{cc} \{1, 2\} & p^2 \\ \{1, 7\} & p^2 + p^6 - p^8 \\ \{1, 11\} & p \\ \{2, 8\} & p + p^4 - p^5 + p^6 - p^7 - p^{10} + p^{11} \\ \{7, 11\} & 2p^2 - p^4 \\ \{8, 12\} & p^4 \\ \{11, 12\} & p^4 \end{array} \right) \right\}.$$

After series reduction:

$$\left\{ \{1, 11, 12\}, \left(\begin{array}{cc} \{1, 11\} & p \\ \{1, 11\} & 2p^4 - p^6 + 2p^8 - 3p^{10} + p^{12} \\ \{1, 12\} & p^7 + p^{10} - p^{11} + p^{12} - p^{13} - p^{16} + p^{17} \\ \{11, 12\} & p^4 \end{array} \right) \right\}.$$

After parallel reduction:

$$\left\{ \{1, 11, 12\}, \left(\begin{array}{cc} \{1, 11\} & p + 2p^4 - 2p^5 - p^6 + p^7 + 2p^8 - \\ & 2p^9 - 3p^{10} + 3p^{11} + p^{12} - p^{13} \\ \{1, 12\} & p^7 + p^{10} - p^{11} + p^{12} - p^{13} - p^{16} + p^{17} \\ \{11, 12\} & p^4 \end{array} \right) \right\}.$$

After series reduction:

$$\left\{ \{1, 12\}, \left(\begin{array}{cc} \{1, 12\} & p^7 + p^{10} - p^{11} + p^{12} - p^{13} - p^{16} + p^{17} \\ \{1, 12\} & p^5 + 2p^8 - 2p^9 - p^{10} + p^{11} + 2p^{12} \\ & -2p^{13} - 3p^{14} + 3p^{15} + p^{16} - p^{17} \end{array} \right) \right\}.$$

After parallel reduction:

$$\left\{ \{1, 12\}, \left(\begin{array}{cc} \{1, 12\} & p^5 + p^7 + 2p^8 - 2p^9 + 2p^{12} - 3p^{13} - 3p^{14} \\ & + 3p^{16} - 2p^{18} + 2p^{19} - p^{20} + 6p^{21} - 6p^{22} + \\ & p^{23} + 3p^{24} - 6p^{25} + 4p^{26} - 2p^{27} + 2p^{28} - \\ & 2p^{29} - 2p^{30} + 6p^{31} - 2p^{32} - 2p^{33} + p^{34} \end{array} \right) \right\}.$$

Thus the single edge between $\{1, 12\}$ with reliability x is obtained in Figure 3.3, where $x = p^5 + p^7 + 2p^8 - \dots - 2p^{33} + p^{34}$.

3.5 Conclusion

We illustrated how the series-parallel reduction Mathematica algorithm sometimes works to reduce a network G to a single edge. In this case its label x equals $nr(G)$. More precisely, "sometimes" happens if and only if G is a so-called series-parallel network. The latter can be neatly defined recursively, which we omit (see [1]). Even when G is not a series-parallel network, as in Chapter 4, sp -reduction will continue to play an important role.

Chapter 4

Computing $nr(G)$ by contraction-deletion

4.1 Introduction

For most networks, the parallel and series reductions as illustrated in Chapter 3 do not immediately apply. Instead, suppose a network G has an edge e incident with the vertices v_1, v_2 . Then G can be reduced to a smaller network $G - e$ by *deleting* the edge e (while keeping v_1, v_2). A smaller network $G.e$ can also be obtained by *contracting* the edge e . This means shrinking $\{v_1, v_2\}$ into a sole vertex. Concrete examples follow soon.

The following **contraction-deletion theorem** (CDT) can be traced back to Moskowitz [3]:

Theorem 4.1.1. *If p is the edge-reliability of edge e then*

$$nr(G) = p.nr(G.e) + (1 - p).nr(G - e). \quad (4.1.1)$$

The CDT gives rise to the **contraction-deletion algorithm** (CDA), yet in quite intricate ways. In a nutshell, the network G is decomposed into many smaller networks (Section 4.3). Then the CDT is applied repeatedly until one arrives at $nr(G)$ (Section 4.4). The book-keeping is done by a LIFO-stack (Section 4.2).

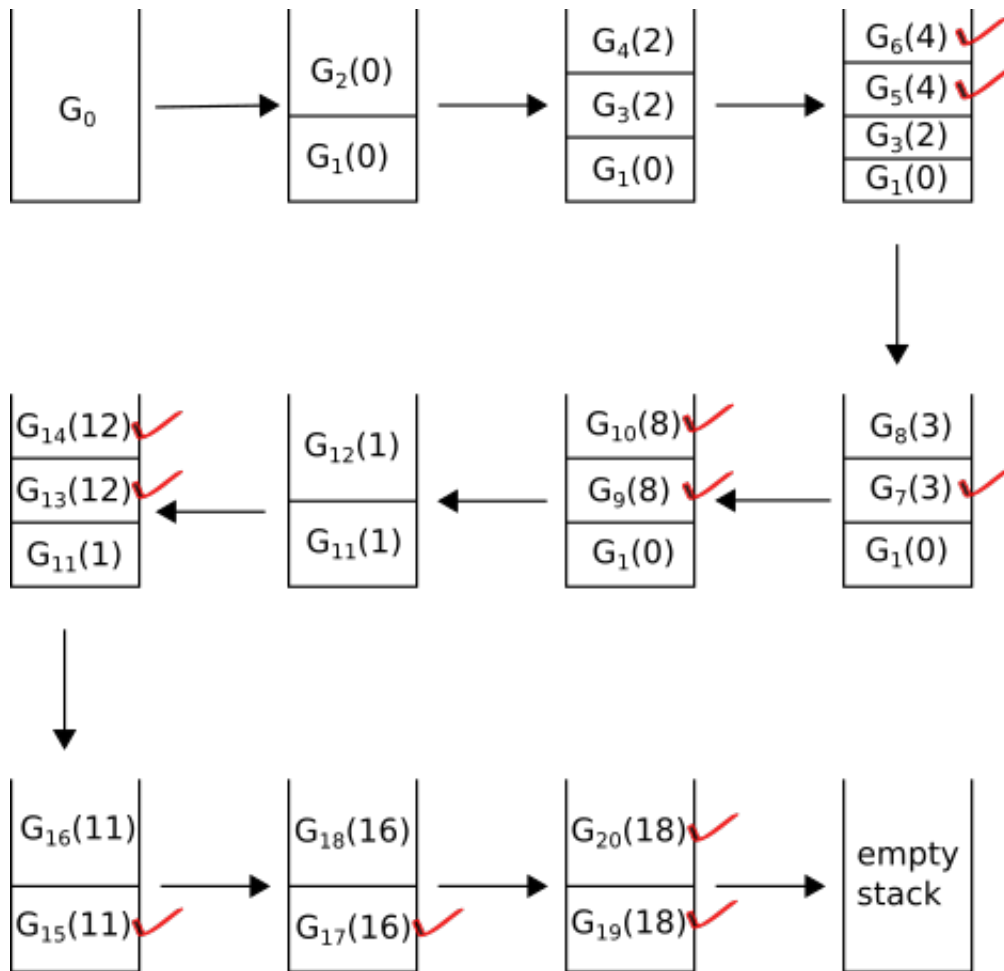
Before we go on, it should be mentioned that the efficiency of the CDA depends on the strategy of selecting edges e for factorisation. To keep things simple, we pick these edges e at random.

4.2 The Last-In-First-Out (LIFO) Technique

The network G_0 from Figure 1.1 will give rise (upon contraction respectively deletion) to two smaller networks G_1 and G_2 . Unfortunately they are too big to be evaluated right away. In such situations it is common practice to put one task aside (say G_1) and fully concentrate on the other task (thus G_2) and its potential subtasks. In turn G_2 gives rise to G_3 (which is put aside) and G_4 . Then G_4 gives rise to G_5 and G_6 , and so forth. More precisely, "so forth" means that

This is the so-called Last-In-First-Out (LIFO) technique. Figure 4.1 matches well the "physical" stack employed in our Mathematica implementation of the contraction-deletion algorithm. Let us explain the details of Figure 4.1. The numbering of the two subtasks of a task is such that the one with the smaller index arises from contraction, and the one with the larger index from deletion of an edge. For instance contracting a certain edge e of G_3 yields a network $\overline{G_7}$ which by series and parallel reduction simplifies to a network G_7 with the same reliability polynomial. On the other hand, deleting e gives rise to a network G_8 . The notation $G_7(3)$ and $G_8(3)$ in Figure 4.1 allows us to remember that both G_7 and G_8 arose from G_3 .

The eleven networks G_i ticked off in Figure 4.1 are the most simple kind as they consist of a single edge between s, t which is labelled by a polynomial that coincides with the reliability polynomial of G_i (as seen in Chapter 4).

Figure 4.1: The LIFO stack for processing G_0 .

4.3 How the graphs G_i arise

This section is almost entirely pictures! They will be self-explanatory, and make us understand how the LIFO-stack in Figure 4.1 came about. We mention that the edges chosen for deletion/contraction are picked at random. Choosing them in more thoughtful ways can well make a difference but we don't go into that.

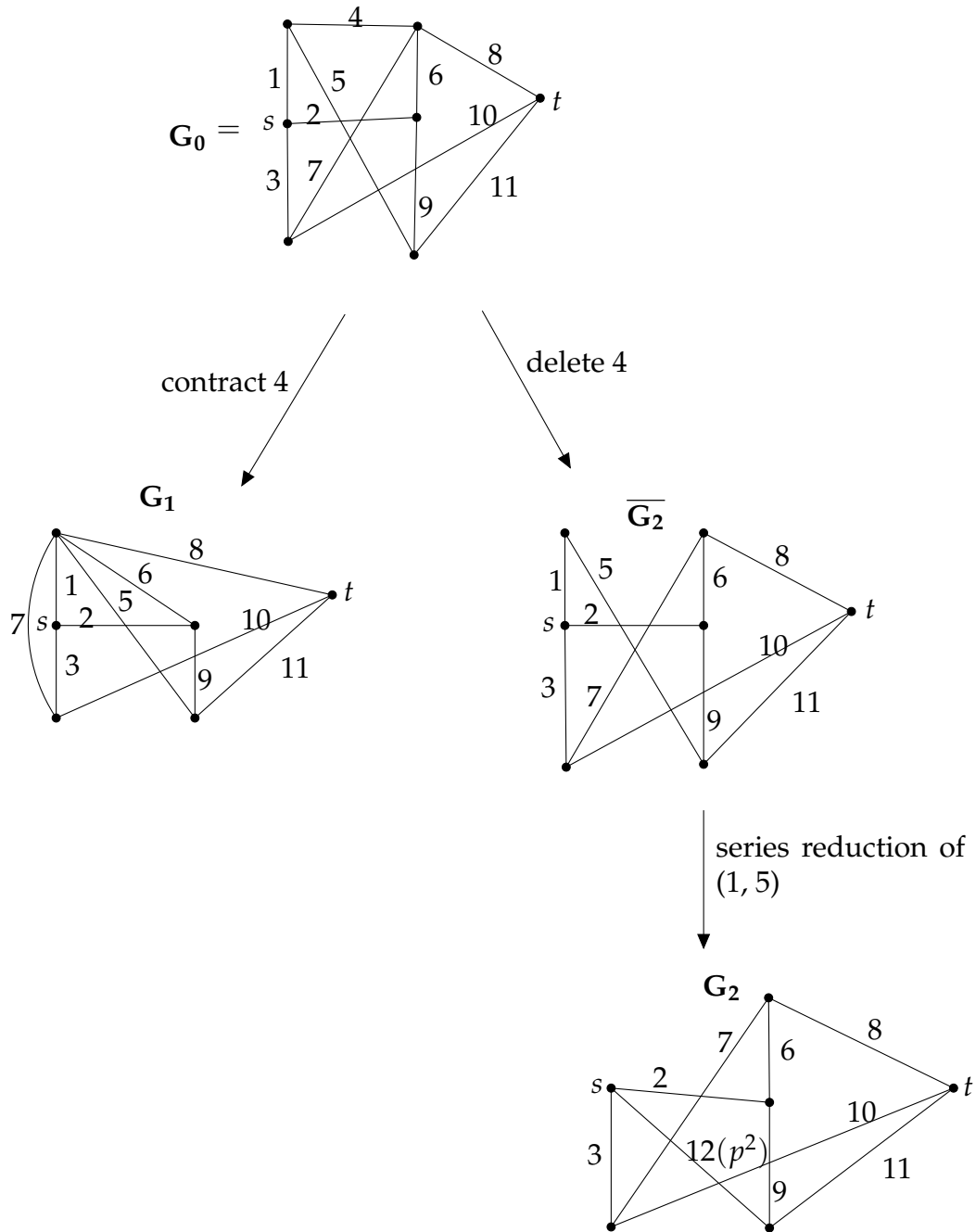


Figure 4.2: Reduction of G_0 .

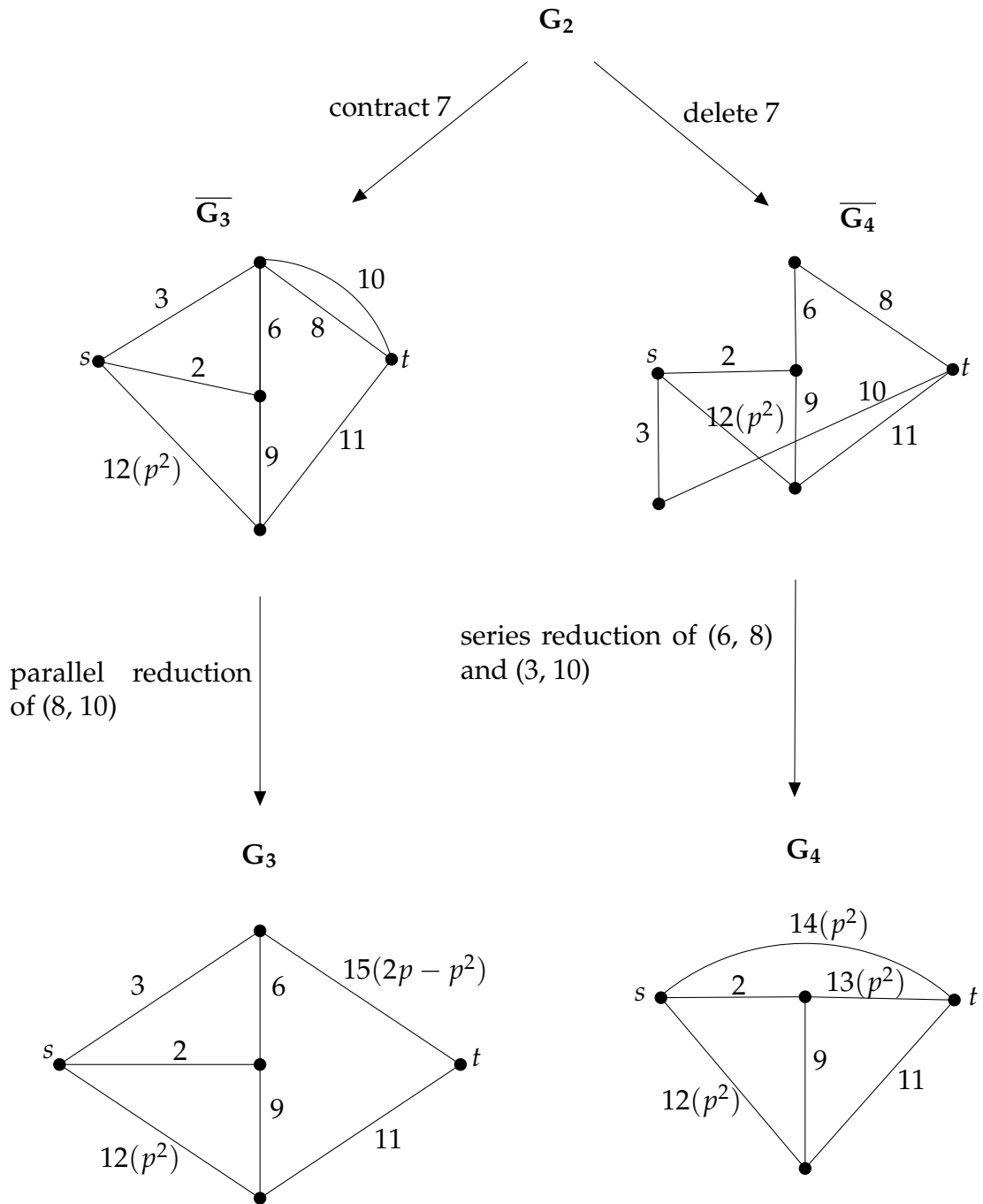


Figure 4.3: Reduction of G_2 .

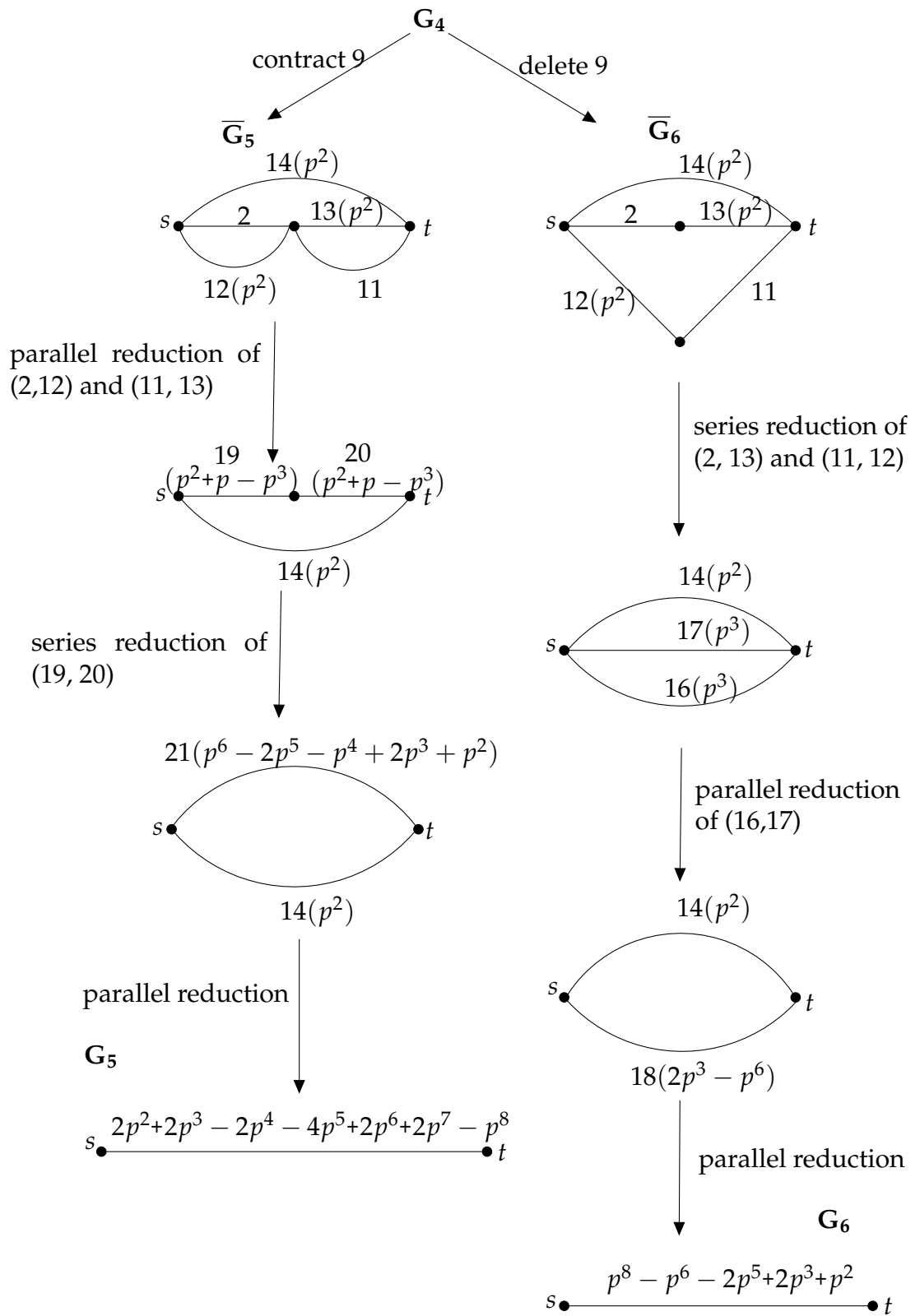


Figure 4.4: Complete reduction of G_4 yielding $poly_4$.

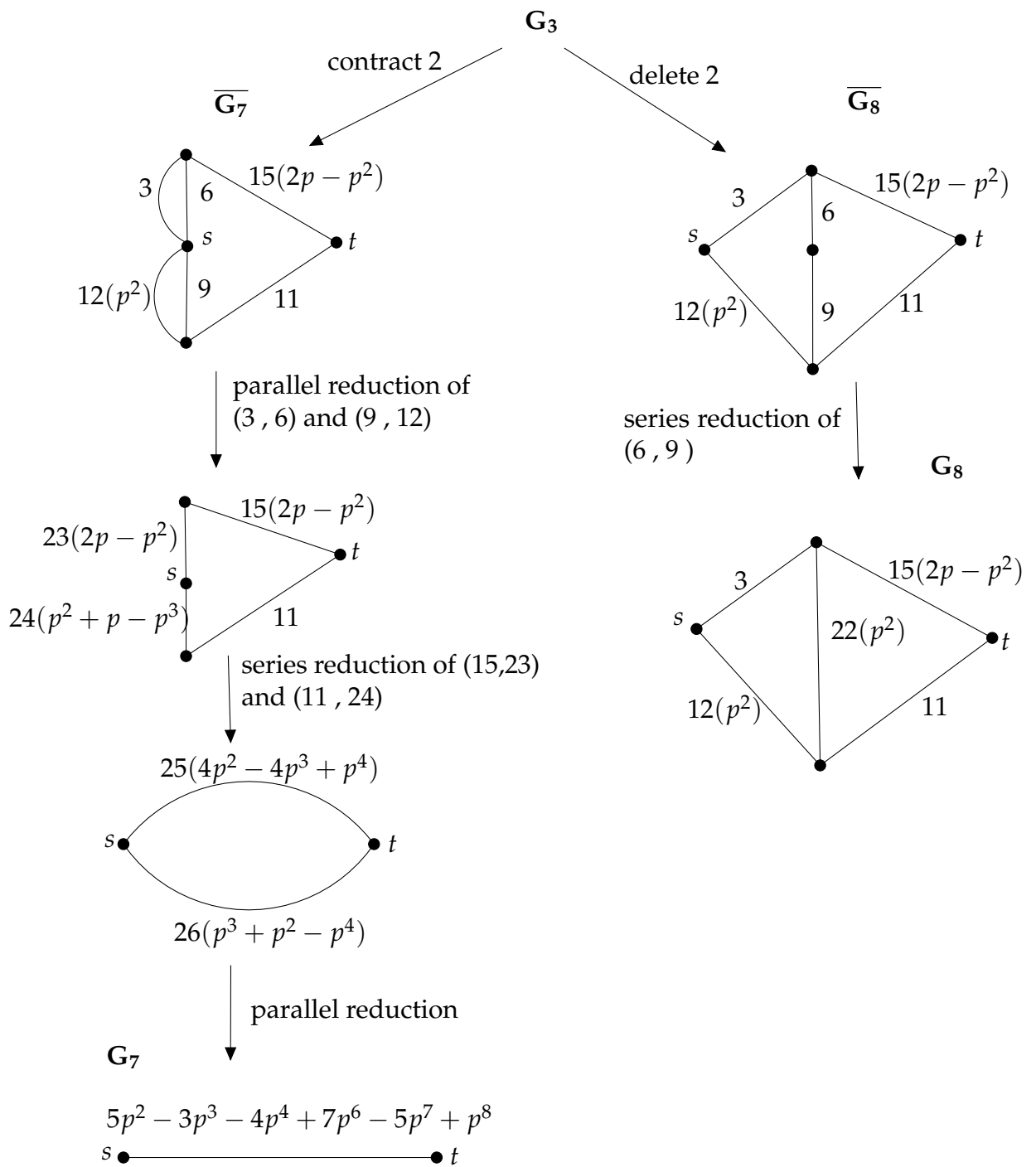


Figure 4.5: Reduction of G_3 .

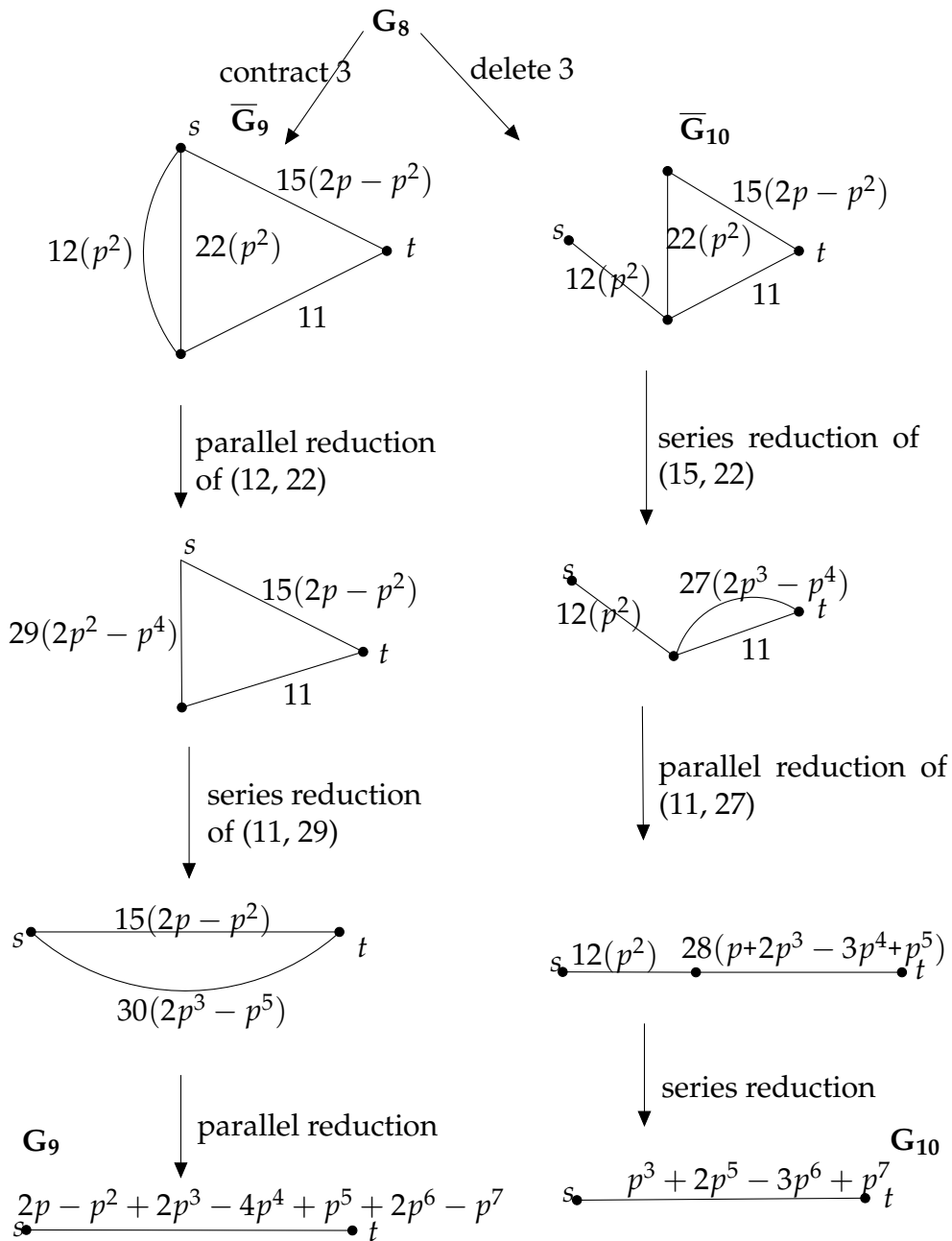


Figure 4.6: Complete reduction of G_8 yielding $poly_8$.

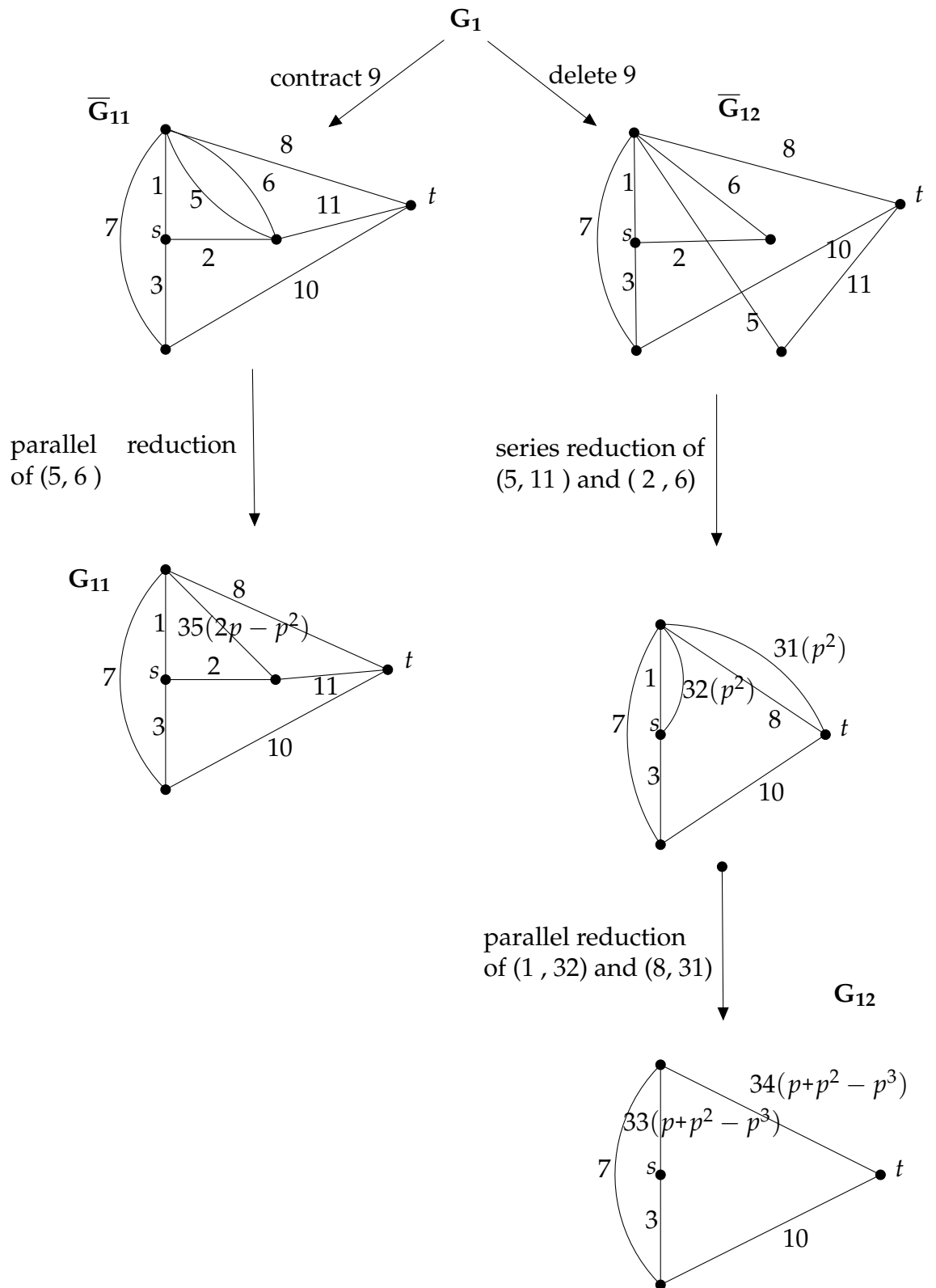


Figure 4.7: Reduction of G_1 .

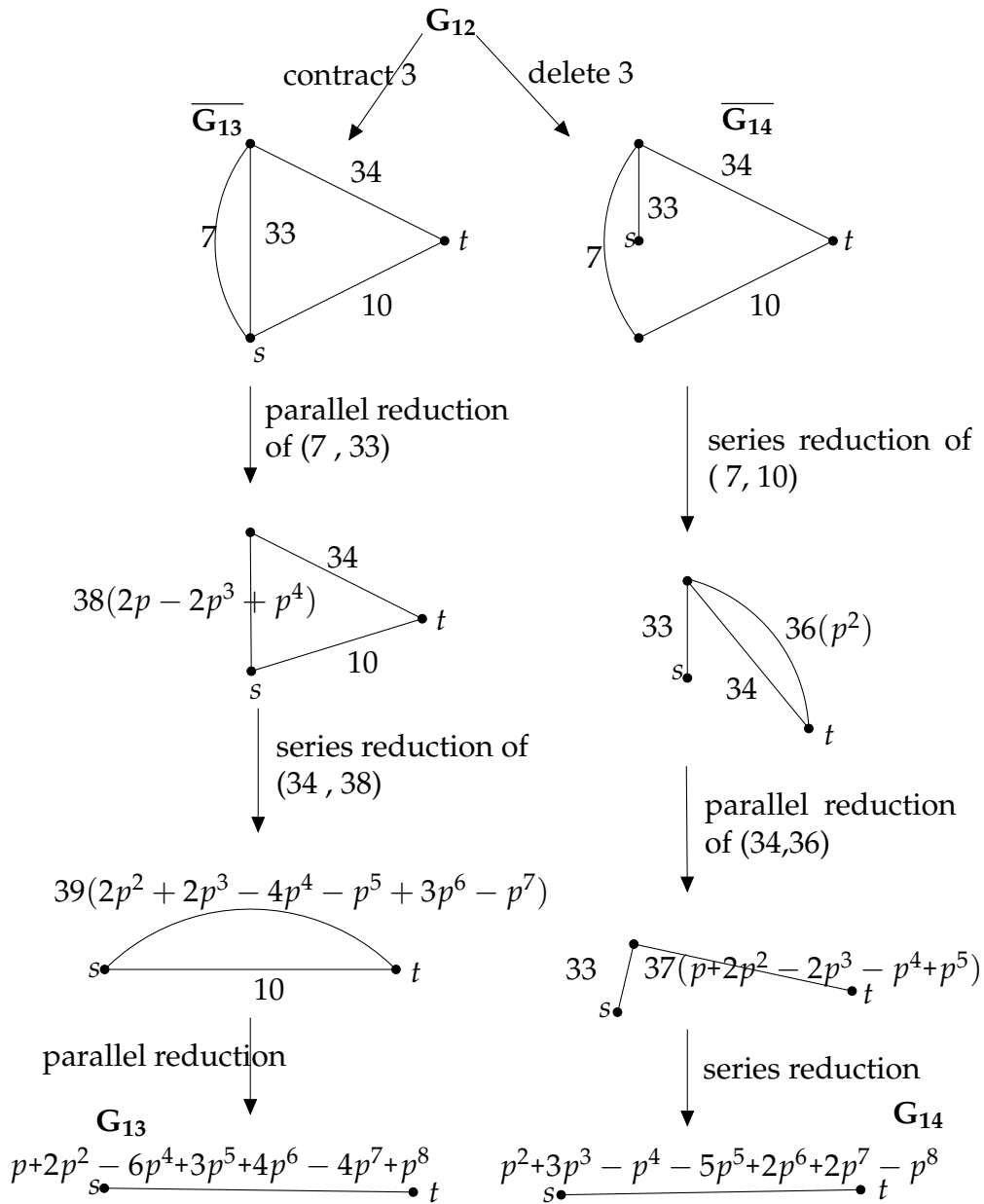


Figure 4.8: Complete reduction of G_{12} yielding $poly_{12}$.

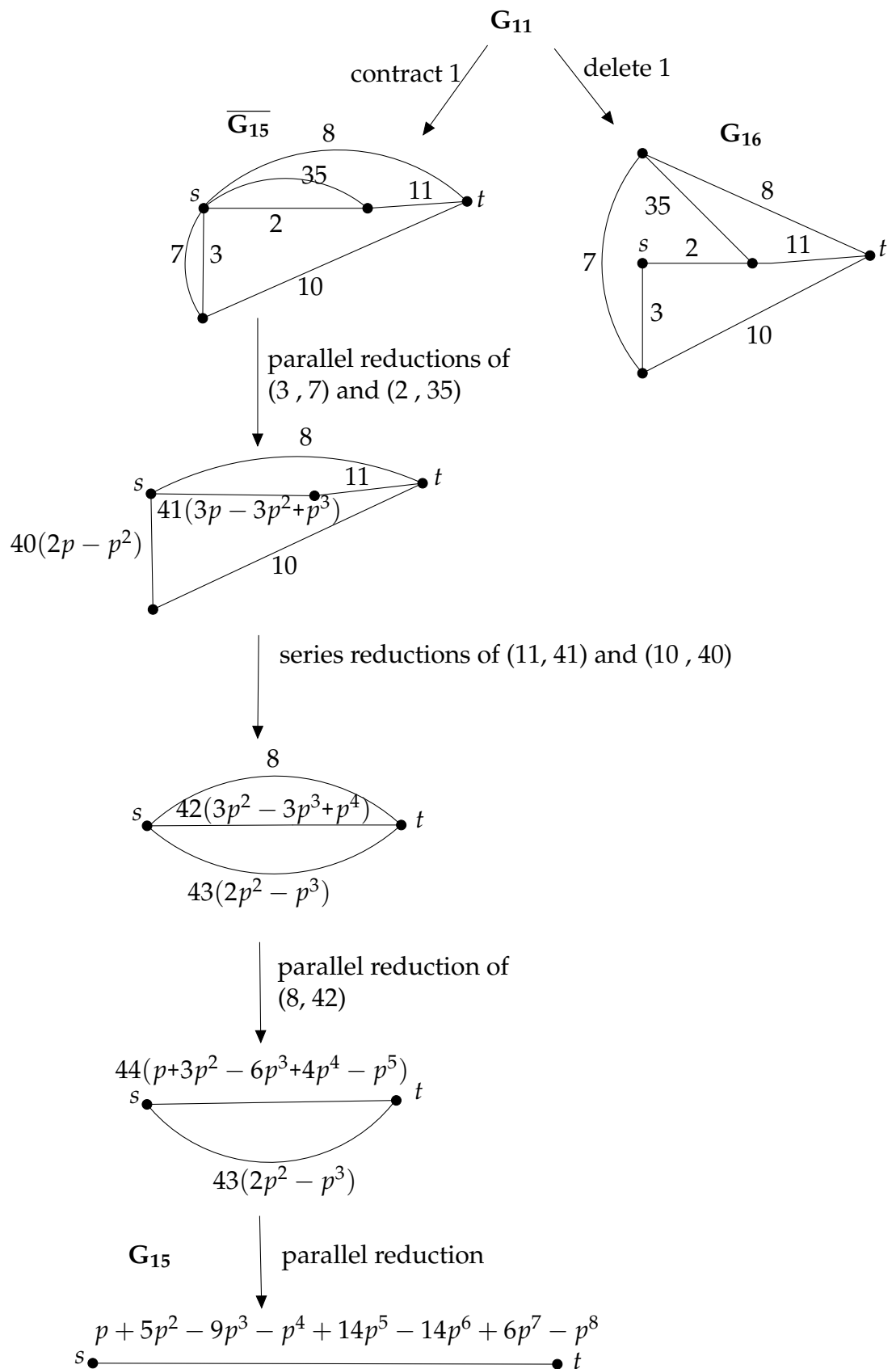


Figure 4.9: Reduction of G_{11} .

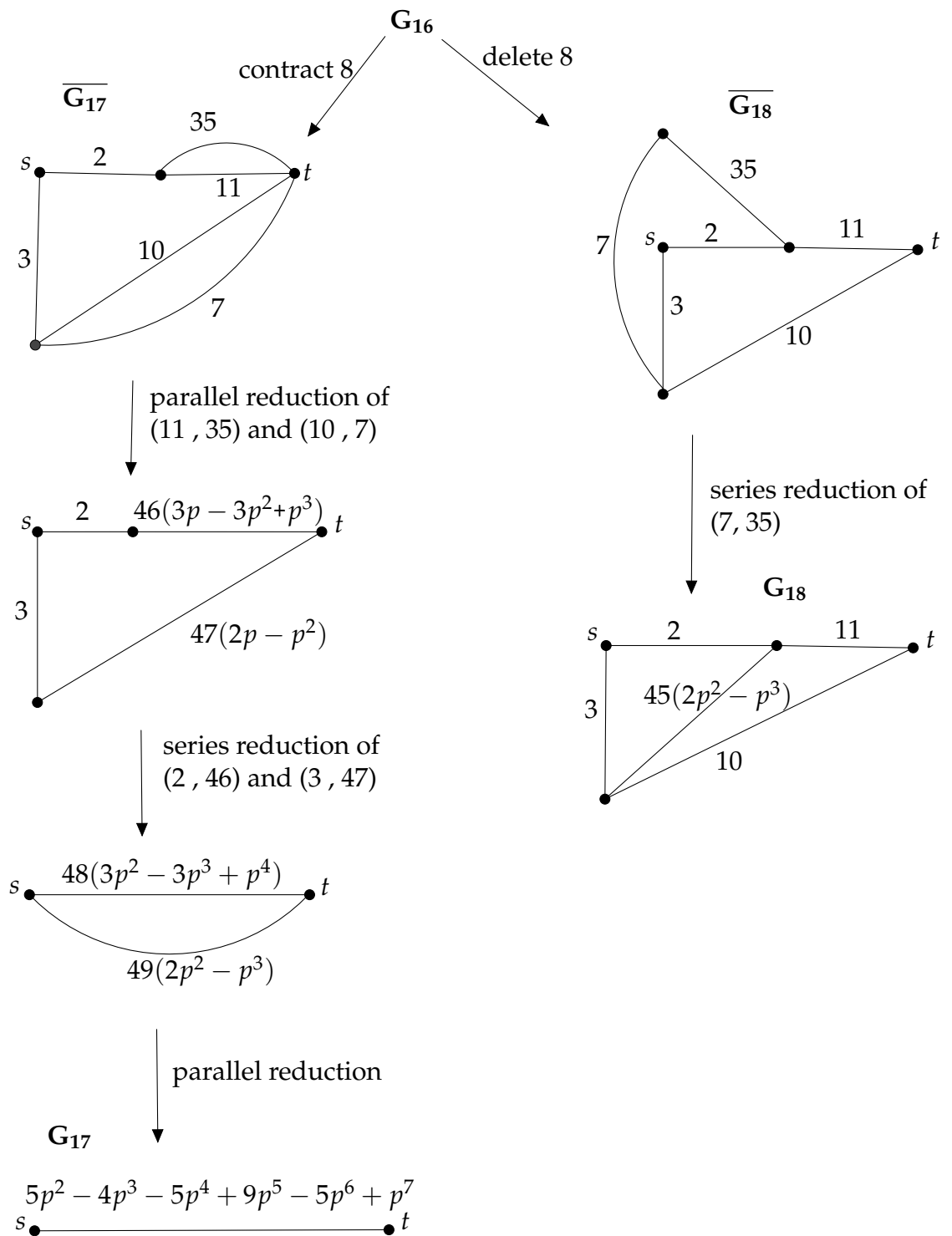


Figure 4.10: Reduction of G_{16} .

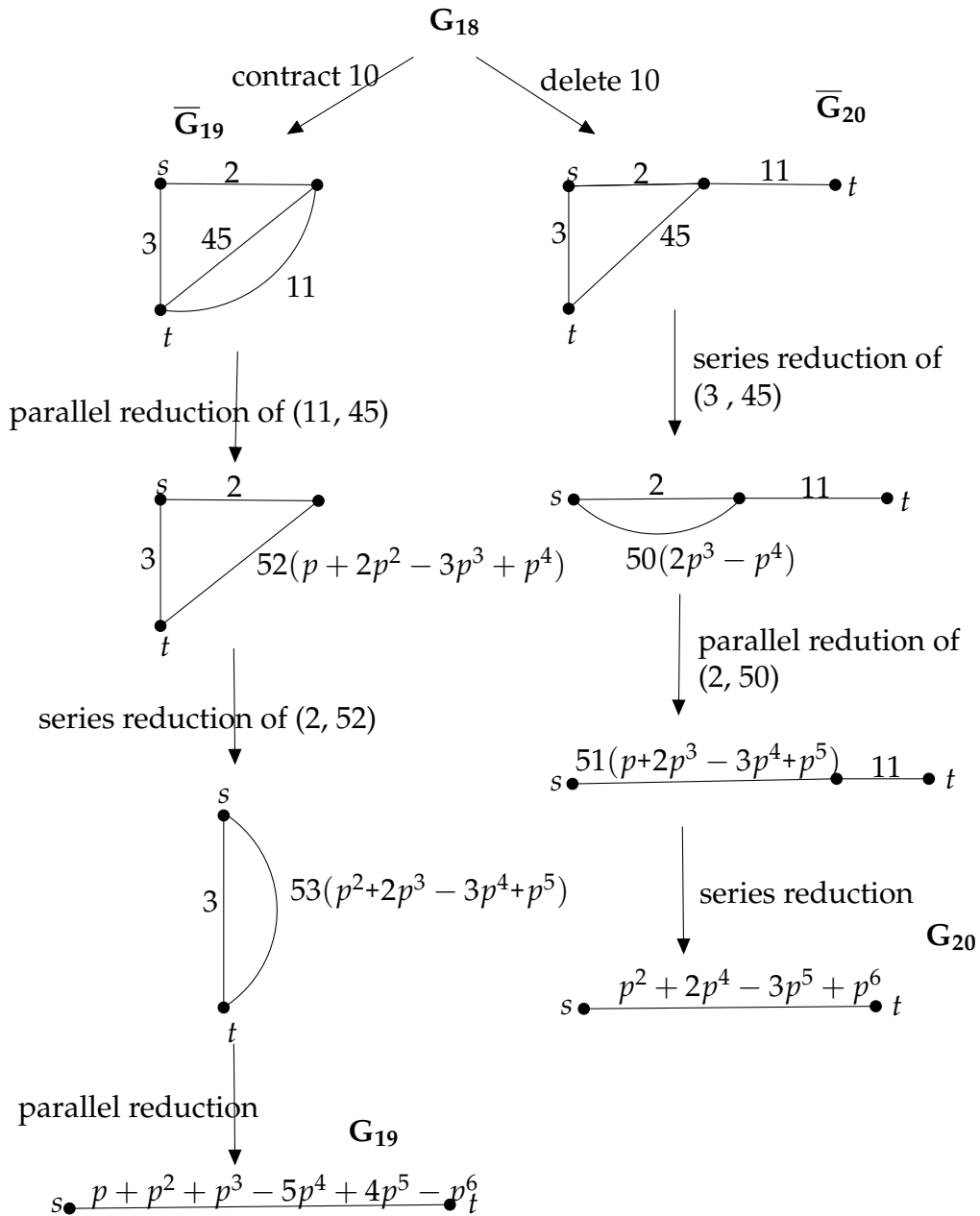


Figure 4.11: Complete reduction of G_{18} yielding $poly_{18}$.

4.4 Last but not least: The CDT of Moskowitz

In Section 4.3 we computed a lot of reliability polynomials. In order to combine them and calculate $nr(G_0)$ (using CDT repeatedly), we must recall the LIFO-stack in (Figure 4.1). In fact, it is more illuminating to render this stack (a well-known equivalent view) as a binary tree:

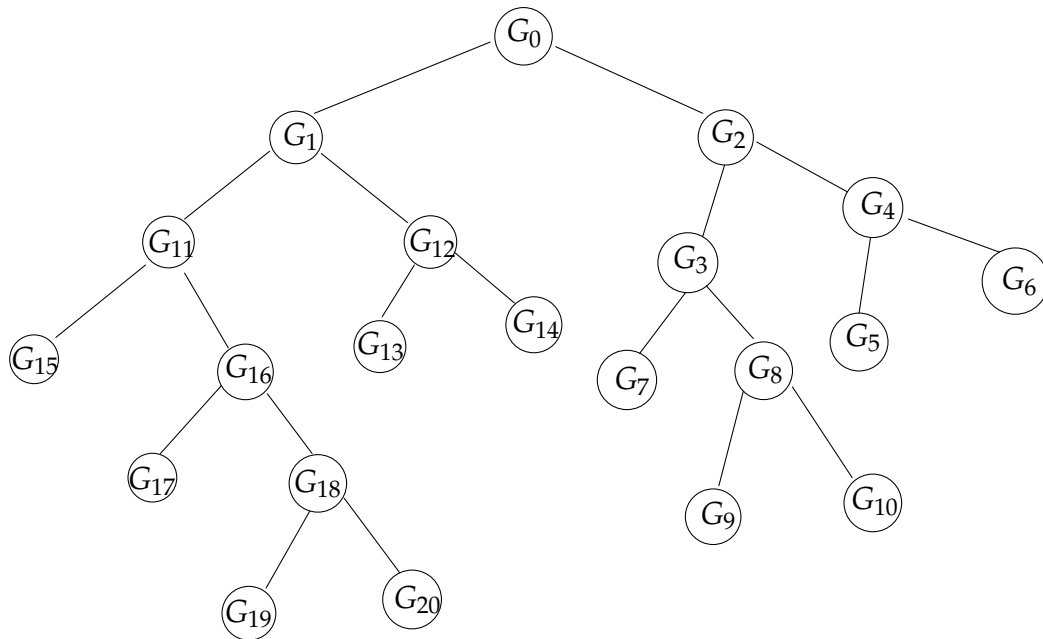


Figure 4.12: Binary tree of the G_0 subtasks

We shall work our way upwards, starting with the leaves G_5, G_6 (any other pair of leaves would also work) until we reach the root G_0 . The father G_4 of G_5, G_6 is completely reduced (Figure 4.4) to give $poly_5$ and $poly_6$. As argued earlier, since the series or parallel reduction does not change the reliability polynomial, the reliability polynomial $\overline{poly_5}$ of $\overline{G_5}$ equals the reliability polynomial $poly_5$ of G_5 . The latter is written in (Figure 4.4) on the sole remaining edge of G_5 . Similarly $\overline{poly_6} = poly_6$ and the latter is written on the

sole edge of G_6 . It thus follows from the contraction-deletion theorem that

$$\begin{aligned} poly_4 &= p.\overline{poly_5} + (1-p)\overline{poly_6}, \\ &= p.poly_5 + (1-p)poly_6, \\ &= p.(2p^2 + 2p^3 - 2p^4 - 4p^5 + 2p^6 + 2p^7 - p^8) + \\ &\quad (1-p)(p^2 + 2p^3 - 2p^5 - p^6 + p^8), \\ &= p^2 + 3p^3 - 4p^5 - 3p^6 + 3p^7 + 3p^8 - 2p^9. \end{aligned}$$

Calculating the reliability polynomial of G_8 (Figure 4.6) by applying the CDT to $poly_9$ and $poly_{10}$ gives:

$$\begin{aligned} poly_8 &= p.\overline{poly_9} + (1-p)\overline{poly_{10}}, \\ &= p.poly_9 + (1-p)poly_{10}, \\ &= p.(2p - p^2 + 2p^3 - 4p^4 + p^5 + 2p^6 - p^7) + (1-p)(p^3 + 2p^5 - 3p^6 + p^7), \\ &= 2p^2 + p^4 - 2p^5 - 4p^6 + 6p^7 - 2p^8. \end{aligned}$$

Similarly, we further calculate the reliability polynomial of network G_3 (Figure 4.5) by applying the CDT to $poly_7$ and $poly_8$:

$$\begin{aligned} poly_3 &= p.\overline{poly_7} + (1-p)\overline{poly_8}, \\ &= p.poly_7 + (1-p)poly_8 \\ &= p.(5p^2 - 3p^3 - 4p^4 + 7p^6 - 5p^7 + p^8) + \\ &\quad (1-p)(2p^2 + p^4 - 2p^5 - 4p^6 + 6p^7 - 2p^8), \\ &= 2p^2 + 3p^3 - 2p^4 - 7p^5 - 2p^6 + 17p^7 - 13p^8 + 3p^9. \end{aligned}$$

We calculate the reliability polynomial of G_2 (Figure 4.3) by applying the CDT to $poly_3$ and $poly_4$:

$$\begin{aligned} poly_2 &= p.\overline{poly_3} + (1-p)\overline{poly_4}, \\ &= p.poly_3 + (1-p)poly_4, \\ &= p.(2p^2 + 3p^3 - 2p^4 - 7p^5 - 2p^6 + 17p^7 - 13p^8 + 3p^9) + \\ &\quad (1-p)(p^2 + 3p^3 - 4p^5 - 3p^6 + 3p^7 + 3p^8 - 2p^9), \\ &= p^2 + 4p^3 - 6p^5 - 6p^6 + 4p^7 + 17p^8 - 18p^9 + 5p^{10}. \end{aligned}$$

Likewise (Figure 4.8):

$$\begin{aligned}
 poly_{12} &= p.\overline{poly_{13}} + (1-p)\overline{poly_{14}}, \\
 &= p.poly_{13} + (1-p)poly_{14}, \\
 &= p.(p + 2p^2 - 6p^4 + 3p^5 + 4p^6 - 4p^7 + p^8) + \\
 &\quad (1-p)(p^2 + 3p^3 - p^4 - 5p^5 + 2p^6 + 2p^7 - p^8), \\
 &= 2p^2 + 4p^3 - 4p^4 - 10p^5 + 10p^6 + 4p^7 - 7p^8 + 2p^9.
 \end{aligned}$$

We strive to combine $poly_{12}$ with $poly_{11}$ to get $poly_1$. The latter can then be combined with the already known $poly_2$ to finally get the reliability polynomial $poly_0$ of G_0 . To get $poly_{11}$, consider G_{18} which reduces to the networks G_{19} and G_{20} by respectively contracting or deleting *edge 10* (Figure 4.11). This yields $poly_{19}$ and $poly_{20}$ respectively. Hence:

$$\begin{aligned}
 poly_{18} &= p.\overline{poly_{19}} + (1-p)\overline{poly_{20}}, \\
 &= p.poly_{19} + (1-p)poly_{20}, \\
 &= p.(p + p^2 + p^3 - 5p^4 + 4p^5 - p^6) + (1-p)(p^2 + 2p^4 - 3p^5 + p^6), \\
 &= 2p^2 + 3p^4 - 10p^5 + 8p^6 - 2p^7.
 \end{aligned}$$

With the reliability polynomial $poly_{18}$ and $poly_{17}$, we calculate the reliability polynomial of G_{16} (Figure 4.10) as usual:

$$\begin{aligned}
 poly_{16} &= p.poly_{17} + (1-p)poly_{18}, \\
 &= p.(5p^2 - 4p^3 - 5p^4 + 9p^5 - 5p^6 + p^7) + \\
 &\quad (1-p)(2p^2 + 3p^4 - 10p^5 + 8p^6 - 2p^7), \\
 &= 2p^2 + 3p^3 - p^4 - 18p^5 + 27p^6 - 15p^7 + 3p^8.
 \end{aligned}$$

By applying the CDT to $poly_{16}$ and $poly_{15}$, we calculate the reliability polynomial of G_{11} (Figure 4.9):

$$\begin{aligned}
 poly_{11} &= p.poly_{15} + (1-p)poly_{16}, \\
 &= p.(p + 5p^2 - 9p^3 - p^4 + 14p^5 - 14p^6 + 6p^7 - p^8) + \\
 &\quad (1-p)(2p^2 + 3p^3 - p^4 - 18p^5 + 27p^6 - 15p^7 + 3p^8), \\
 &= 3p^2 + 6p^3 - 13p^4 - 18p^5 + 59p^6 - 56p^7 + 24p^8 - 4p^9.
 \end{aligned}$$

With the reliability polynomials $poly_{11}$ and $poly_{12}$, we calculate the reliability polynomial of G_1 (Figure 4.7):

$$\begin{aligned} poly_1 &= p.poly_{11} + (1-p)poly_{12}, \\ &= p.(3p^2 + 6p^3 - 13p^4 - 18p^5 + 59p^6 - 56p^7 + 24p^8 - 4p^9) + \\ &\quad (1-p)(2p^2 + 4p^3 - 4p^4 - 10p^5 + 10p^6 + 4p^7 - 7p^8 + 2p^9), \\ &= 2p^2 + 5p^3 - 2p^4 - 19p^5 + 2p^6 + 53p^7 - 67p^8 + 33p^9 - 6p^{10}. \end{aligned}$$

Finally we apply the CDT to $poly_1$ and $poly_2$ to calculate the reliability polynomial of G_0 (Figure 4.2) as follows:

$$\begin{aligned} poly_0 &= p.poly_1 + (1-p)poly_2, \\ &= p.(2p^2 + 5p^3 - 2p^4 - 19p^5 + 2p^6 + 53p^7 - 67p^8 + 33p^9 - 6p^{10}) + \\ &\quad (1-p)(p^2 + 4p^3 - 6p^5 - 6p^6 + 4p^7 + 17p^8 - 18p^9 + 5p^{10}), \\ &= p^2 + 5p^3 + p^4 - 8p^5 - 19p^6 + 12p^7 + 66p^8 - 102p^9 + 56p^{10} - 11p^{11}. \end{aligned}$$

Chapter 5

Minpath and mincut methods

5.1 Introduction

In Subsections 5.2 to 5.5 we present algorithms to compute all minpaths or all mincuts of a network G , and apply (naive) inclusion-exclusion in both cases to calculate $nr(G)$. Subsection 5.6 presents two crucial algorithms of Wild [7, 8] that concern arbitrary set filters. These algorithms will enable us in Subsections 5.7 to 5.9 to process all minpaths (or all mincuts) in more efficient ways to get $nr(G)$.

Definition 5.1.1. A **minpath** is any minimal edge-configuration $M \subseteq E$ that connects s and t in a network G .

One concludes that M is a minpath if and only if M is a path without cycles. It also follows that a state X is operational (Definition 2.1.1) if and only if it contains a minpath.

Definition 5.1.2. A **pathset** is a set that contains at least one minpath.

Definition 5.1.3. A **cutset** of a network $G = (V, E)$ is a set $X \subseteq E$ that intersects all minpaths. X is referred to as a **mincut** (minimal cutset) if each proper subset is no longer a cutset.

5.2 Computing all minpaths of a network

Generating all minpaths in a network has been widely studied [1]. In this section we simply compute minpaths using the hardwired Mathematica command `FindPath` that generates all minpaths between s and t in G .

Example 5.2.1. Our running example network G_0 (Figure 5.1) has these 16 min-paths:

$\{3, 10\}$, $\{3, 7, 8\}$, $\{2, 9, 11\}$, $\{2, 6, 8\}$, $\{1, 5, 11\}$, $\{1, 4, 8\}$, $\{2, 6, 7, 10\}$,
 $\{1, 4, 7, 10\}$, $\{3, 7, 6, 9, 11\}$, $\{3, 7, 4, 5, 11\}$, $\{2, 9, 5, 4, 8\}$, $\{2, 6, 4, 5, 11\}$,
 $\{1, 5, 9, 6, 8\}$, $\{1, 4, 6, 9, 11\}$, $\{2, 9, 5, 4, 7, 10\}$, $\{1, 5, 9, 6, 7, 10\}$.

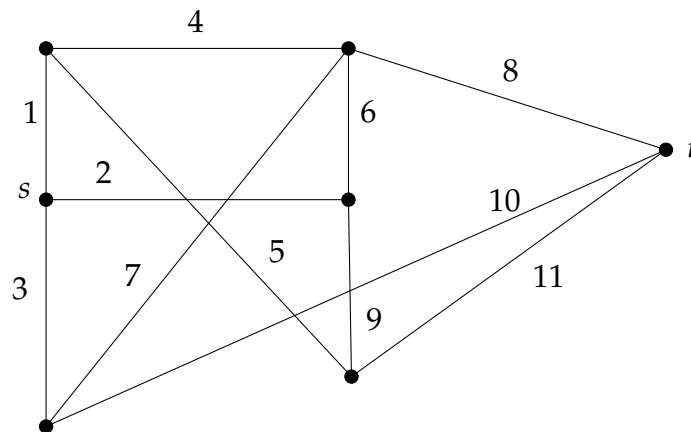


Figure 5.1: Running example network G_0 .

We have tested ten random examples in Figure 5.2 to compare the CPU time. For instance, it takes approximately 21.92 seconds to generate all 17'020'105 minpaths in some random graph with 49 vertices and 86 edges. (See the other parameters in Appendix A.1).

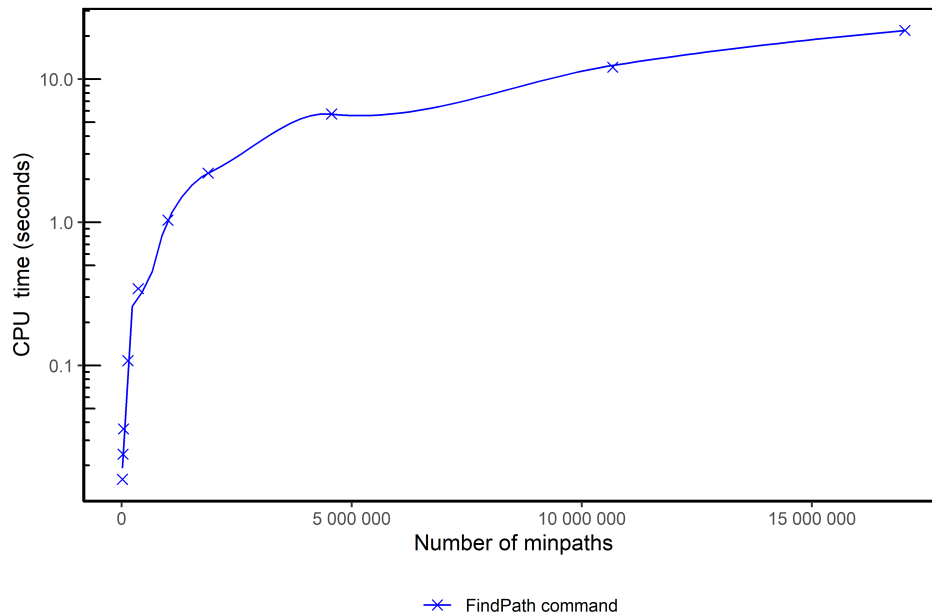


Figure 5.2: CPU time for generating all minpaths using Mathematica's *FindPath* command, using a logarithmic scale.

5.3 Computing all mincuts of a network

The mincuts are exactly the minimal transversals of the minpaths. Hence any algorithm for finding all minimal transversals of a set system will do. Many algorithms for this important problem have been proposed. It is however more efficient to generate the mincuts "directly", thus leaving aside the minpaths.

Again, several algorithms for this purpose exist. The simplest one works as follows. If $X \subseteq E$ is any mincut then the graph with edge set $E \setminus X$ has exactly two connected components S and T , where $s \in S$ and $t \in T$. Conversely, if removal of an edge set X yields such a graph then X must have been a mincut. Hence one can find a mincut by processing as follows all subsets of vertices $S \subseteq V$ that contain s . If both the sub-graph induced by S and the sub-graph induced by $T := V \setminus S$ are connected, one gets a mincut, otherwise not. This algorithm, implemented in Mathematica will be called *SimpleMincut*.

We have tested eight random examples in Figure 5.3 and we e.g. observe that it takes 14.372 seconds to generate all 17'390 mincuts in a random graph with 19 vertices and 41 edges. (See the other parameters in Appendix A.2).

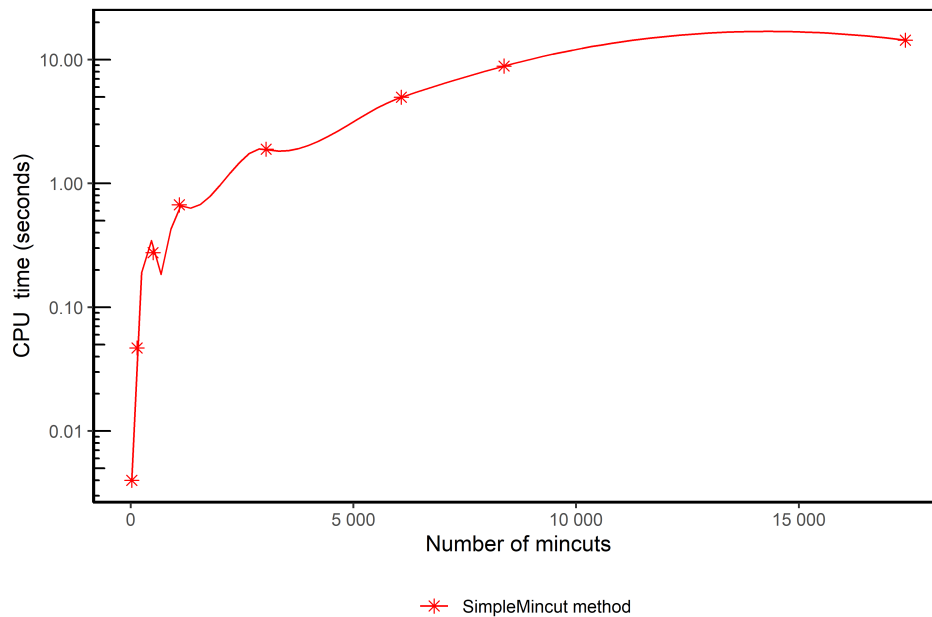


Figure 5.3: CPU time for generating mincuts using SimpleMincut algorithm, again using a logarithmic scale.

5.4 Naive inclusion-exclusion on the minpaths

We illustrate two ways in which inclusion-exclusion can be used to calculate the network reliability using minpaths.

5.4.1 Method 1

Suppose that each edge has edge reliability $p = 0.5$, and hence failure probability $q = 0.5$. Then at each fixed moment for any two edge-sets (=states) $X, Y \subseteq E$ it is equally likely that exactly the edges in X (respectively the edges in Y) are the working edges. For instance, if $X = \{1, 2, 3, 4\}$ and $Y = \{2, 5, 6\}$ then the corresponding probabilities are: $Pr(X) = p^4 q^2 = (0.5)^6$

and $Pr(Y) = p^3q^3 = (0.5)^6$. Thus if we put

$N :=$ number of pathsets of G

then

$$nr(G) = \frac{N}{2^{ne}} \quad (5.4.1)$$

Example 5.4.1. Consider this network G :

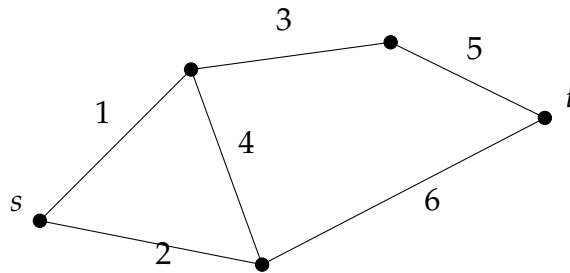


Figure 5.4

One verifies ad hoc that there are exactly four minpaths mp_1 to mp_4 . In order to calculate N , we first define four properties of subsets $X \subseteq E = [6]$:

property a_1 : X contains $mp_1 = \{2, 6\}$,

property a_2 : X contains $mp_2 = \{1, 4, 6\}$,

property a_3 : X contains $mp_3 = \{1, 3, 5\}$,

property a_4 : X contains $mp_4 = \{2, 4, 3, 5\}$.

If say $N(a_2a_4)$ is the number of X satisfying both a_2 and a_4 then the principle of inclusion-exclusion gives

$$\begin{aligned} N &= N(a_1 \text{ or } a_2 \text{ or } a_3 \text{ or } a_4), \\ &= N(a_1) + N(a_2) + N(a_3) + N(a_4) - N(a_1a_2) - N(a_1a_3) - N(a_1a_4) \\ &\quad - N(a_2a_3) - N(a_2a_4) - N(a_3a_4) + N(a_1a_2a_3) + N(a_1a_2a_4) + N(a_1a_3a_4) \\ &\quad + N(a_2a_3a_4) - N(a_1a_2a_3a_4), \\ &= 2^4 + 2^3 + 2^3 + 2^2 - 2^2 - 2^1 - 2^1 - 2^1 - 2^0 - 2^1 + 2^0 + 2^0 + 2^0 + 2^0 - 2^0 = 26. \end{aligned}$$

Therefore by (5.4.1) the network reliability is

$$nr(G) = \frac{26}{64} = 0.40625.$$

5.4.2 Method 2

More generally we can apply the *probability version* of the principle of inclusion-exclusion, which applies no matter what the reliabilities (=probabilities) p_1 to p_6 of the individual edges 1 to 6 are. For instance, putting $q_i = 1 - p_i$, the probability that at any given moment the set of working edges X equals $\{2, 6\}$ is $p_2 p_6 q_1 q_3 q_4 q_5$. If X must only *contain* $\{2, 6\}$ the probability is simply $p_2 p_6$ (since the behaviour of e_1, e_3, e_4, e_5 is irrelevant). It follows that

$$\begin{aligned} nr(G) &:= Pr(\text{the set } X \text{ of working edges connects } s \text{ and } t) \\ &= Pr(mp_1 \subseteq X \text{ or } mp_2 \subseteq X \text{ or } mp_3 \subseteq X \text{ or } mp_4 \subseteq X), \\ &= Pr(mp_1 \subseteq X) + \dots - Pr(mp_1 \cup mp_2 \cup mp_3 \cup mp_4 \subseteq X). \end{aligned}$$

When we apply inclusion-exclusion to the expression above we get:

$$\begin{aligned} &= p_2 p_6 + p_1 p_4 p_6 + p_1 p_3 p_5 + p_2 p_3 p_4 p_5 - p_1 p_2 p_4 p_6 - p_1 p_2 p_3 p_5 p_6 - p_2 p_3 p_4 p_5 p_6 - \\ & p_1 p_3 p_4 p_5 p_6 - p_1 p_2 p_3 p_4 p_5 p_6 - p_1 p_2 p_3 p_4 p_5 + p_1 p_2 p_3 p_4 p_5 p_6 + p_1 p_2 p_3 p_4 p_5 p_6 + \\ & p_1 p_2 p_3 p_4 p_5 p_6 + p_1 p_2 p_3 p_4 p_5 p_6 - p_1 p_2 p_3 p_4 p_5 p_6. \end{aligned}$$

If all edges have the same edge reliability $p_i = p$ then the reliability polynomial becomes

$$RelPol(p) = p^2 + 2p^3 - 4p^5 + 2p^6.$$

In particular $p = \frac{1}{2}$ yields as before

$$RelPol\left(\frac{1}{2}\right) = \frac{16 + 16 - 8 + 2}{64} = \frac{26}{64} = 0.40625.$$

5.5 Naive inclusion-exclusion on the mincuts

Let $nf(G) := 1 - nr(G)$ be the *network fallibility*. Let Y be the set of failing edges of G (for a brief period of time). Then s, t are disconnected in $G(E \setminus Y)$ iff Y contains a cutset, hence iff Y contains a mincut. Our network G (Figure 5.4) has the following 6 mincuts:

$$\begin{aligned} mc_1 &= \{1, 2\}, mc_2 = \{3, 6\}, mc_3 = \{5, 6\}, \\ mc_4 &= \{1, 4, 6\}, mc_5 = \{2, 3, 4\}, mc_6 = \{2, 4, 5\}. \end{aligned}$$

It follows that

$$\begin{aligned}
nf(G) &= Pr(\text{the set } Y \text{ of failing edges contains a mincut}), \\
&= Pr(mc_1 \subseteq Y \text{ or } mc_2 \subseteq Y \text{ or } \dots \text{ or } mc_6 \subseteq Y), \\
&= Pr(mc_1 \subseteq Y) + \dots + Pr(mc_6 \subseteq Y) - Pr(mc_1 \cup mc_2 \subseteq Y) - \dots \\
&\quad - Pr(mc_5 \cup mc_6 \subseteq Y) + Pr(mc_1 \cup mc_2 \cup mc_3 \subseteq Y) + \dots \\
&\quad - Pr(mc_1 \cup mc_2 \cup mc_3 \cup mc_4 \cup mc_5 \cup mc_6 \subseteq Y), \\
&= q_1q_2 + q_3q_6 + \dots + q_2q_4q_5 - q_1q_2q_3q_6 \dots - q_1q_2q_3q_4q_5q_6,
\end{aligned}$$

and therefore

$$\begin{aligned}
nr(G) &= 1 - nf(G) = 1 - (1 - p_1)(1 - p_2) - (1 - p_3)(1 - p_6) - \dots \\
&= p_2p_6 + p_1p_4p_6 + \dots - p_1p_2p_3p_4p_5p_6,
\end{aligned}$$

which matches network reliability in Section 5.4. If there are fewer mincuts than minpaths this algorithm beats its companion in Section 5.4.2.

5.6 Two methods to count a set filter

Definition 5.6.1. A **set ideal** (also known as *simplicial complex*) is a family \mathcal{S} of sets such that from $A \in \mathcal{S}$ and $B \subseteq A$ follows $B \in \mathcal{S}$.

Definition 5.6.2. A **transversal** of a set system $\mathcal{H} := \{Y_1, Y_2, \dots, Y_t\}$ is any set X that satisfies $X \cap Y_i \neq \emptyset$ for all $1 \leq i \leq t$.

Whenever \mathcal{S} is finite, the only case of interest to us, then $\mathcal{S} \subseteq \mathcal{P}(E)$ for some finite (base) set E . Moreover, there are inclusion-maximal members $F_1, F_2, \dots, F_h \in \mathcal{S}$, called the *facets* of \mathcal{S} . If we write $X \downarrow$ for $\mathcal{P}(X)$ then $\mathcal{S} = F_1 \downarrow \cup F_2 \downarrow \cup \dots \cup F_h \downarrow$. Trouble is, except for trivial cases, this union is not disjoint.

Example 5.6.3. If say $h = 4$ and $F_1 = \{1, 3, 4, 7\}$, $F_2 = \{3, 5, 7\}$, $F_3 = \{2, 3, 4, 6, 8\}$ and $F_4 = \{1, 6, 8\}$, then brute-force calculation shows that $\mathcal{S} := F_1 \downarrow \cup F_2 \downarrow \cup F_3 \downarrow \cup F_4 \downarrow$ has 51 members (also called *faces*). We can obtain the 51 faces using Wild's [7] facets-to-faces algorithm that represents \mathcal{S} in a more compact way, as a disjoint union

$$\mathcal{S} = r_1 \uplus \dots \uplus r_5.$$

Table 5.1: Descriptive facets-to-faces algorithm

	1	2	3	4	5	6	7	8	
$r_1 =$	1	0	e	e	0	0	e	0	→ 7
$r_2 =$	0	0	2	1	0	0	1	0	→ 2
$r_3 =$	0	0	2	0	e	0	e	0	→ 6
$r_4 =$	0	e	e	e	0	2	0	2	→ 28
$r_5 =$	2	0	0	0	0	2	0	2	→ 8
									= 51

The compressed output (012e-rows) in Table 5.1 consists of either 0's , 1's , 2's or e-bubbles. The 2's (also called the "don't care symbols") in each set are translated into either 0's or 1's while the set of e's means "at least one 1 here". (One can have several e-wildcards, provided that they are distinguished by subscripts). For instance r_3 in Table 5.1 allows to code the following subsets.

Table 5.2: Bitstrings corresponding to subsets of r_3

1	2	3	4	5	6	7	8	
0	0	2	0	e	0	e	0	:=
0	0	0	0	1	0	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	1	0	0	1	
0	0	1	0	1	0	0	0	
0	0	1	0	0	0	0	1	
0	0	1	0	1	0	0	1	

5.6.1 MethodA

A set filter is a family \mathcal{F} of sets such that from $A \in \mathcal{F}$ and $B \supseteq A$ follows $B \in \mathcal{F}$. Fix a finite base set E . For any $X \subseteq E$ put

$$X \uparrow := \{Y : X \subseteq Y \subseteq E\}. \tag{5.6.1}$$

If $\mathcal{F} \subseteq \mathcal{P}(E)$ is a set filter then its inclusion-minimal members $H_1, \dots, H_h \in \mathcal{F}$ are the generators of \mathcal{F} . Obviously $\mathcal{F} = H_1 \uparrow \cup \dots \cup H_h \uparrow$. It is not surprising that the facets-to-faces algorithm readily adapts to make such unions disjoint.

Example 5.6.4. Let $E = \{1, \dots, 8\}$ and $H_1 := \{2, 5, 6, 8\}$, $H_2 := \{1, 2, 4, 6, 8\}$, $H_3 := \{1, 5, 7\}$, $H_4 := \{2, 3, 4, 5, 7\}$. If $\mathcal{F} := H_1 \uparrow \cup \dots \cup H_4 \uparrow$ and $F_i := H_i^c (= E \setminus H_i)$ then for all $A \subseteq E$ it holds that

$$A \supseteq H_1 \text{ or } \cdots \text{ or } A \supseteq H_4 \iff A^c \subseteq F_1 \text{ or } \cdots \text{ or } A^c \subseteq F_4.$$

Now the sets F_1 to F_4 happen to be the facets of the set ideal \mathcal{S} in Example 5.6.3. Hence $A \in \mathcal{F} \iff A^c \in \mathcal{S}$. For instance, one verifies ad hoc that there are exactly 16 sets $B \in \mathcal{S}$ with $|B| = 3$ (i.e. $3 + 1 + 1 + 10 + 1$ in r_1, \dots, r_5 respectively). Their complements are exactly the 16 sets $A \in \mathcal{F}$ with $|A| = 5$.

By lack of imagination we call *Method A* the algorithm sketched above which, given the generators of a set filter \mathcal{F} , calculates for each $k = 1, 2, \dots$ the number of k -element sets in \mathcal{F} .

5.6.2 MethodB

If X is a transversal of \mathcal{H} then a fortiori each superset $X' \supseteq X$ is a transversal of \mathcal{H} . In other words, the family of all transversals of \mathcal{H} is a set filter. Actually, we will not be interested in all infinitely many transversals but only admit X 's with $X \subseteq E$.

Here E is a suitable finite set that in particular contains all sets Y_i . In this set-up the set \mathcal{F} of admissible transversals satisfies $\mathcal{F} \subseteq \mathcal{P}(E)$, akin to Section 5.6.1. However, now \mathcal{F} is not obtained by processing its generators H_i but rather the sets Y_i . Specifically, the transversal e-algorithm of [8], for systematic reasons we henceforth call it *MethodB*, manages to represent \mathcal{F} as a disjoint union of 012e-rows. (However, the technicalities of *MethodA* and *MethodB* differ quite a bit).

5.7 Calculating nr(G) with Minpath-MethodA

When $p_1 = p_2 = \dots = p_w =: p$, then the reliability polynomial $RelPol[p]$ can be calculated without the time-consuming inclusion-exclusion in Sections 5.4 and 5.5. Namely, let X be any one of the 2^{ne} edge subsets, say $|X| = i$. Then the probability that exactly the edges in X work and the others fail is $p^i(1-p)^{ne-i}$. This expression does not depend on the particular X , but only on the cardinality i of X . Consider all $\binom{ne}{i}$ possible i -element states X . Some of these X 's are pathsets while others are not. Let $ps[i]$ be the number of the former X 's. It then follows [1] that

$$RelPol[p] = \sum_{i=0}^{ne} ps[i] p^i (1-p)^{ne-i}. \quad (5.7.1)$$

Evidently $N = \sum_{i=0}^{ne} ps[i]$ is the number N of *all* pathsets (as defined in 5.4.1). As is to be expected one verifies that

$$RelPol\left[\frac{1}{2}\right] = \sum_{i=0}^{ne} ps[i] \left(\frac{1}{2}\right)^i \left(\frac{1}{2}\right)^{(ne-i)} = \sum_{i=0}^{ne} ps[i] \left(\frac{1}{2}\right)^{ne} = \left(\frac{1}{2}\right)^{ne} \sum_{i=0}^{ne} ps[i] = \frac{N}{2^{ne}}.$$

In view of (5.7.1) it is desirable to calculate the numbers $ps[i]$ efficiently. One way of doing so, call it *Minpath-MethodA*, is illustrated in the example below.

Example 5.7.1. Our running network G_0 in Figure 1.1 has 16 minpaths, which we view as the generators H_1 to H_{16} of $\mathcal{F} := H_1 \uparrow \cup \cdots \cup H_{16} \uparrow \subseteq \mathcal{P}[ne] = \mathcal{P}[16]$.

Hence \mathcal{F} is the set filter of all pathsets, and so applying MethodA from Section 5.6.1 yields the required numbers $ps[0], ps[1], \dots, ps[16]$.

5.8 Calculating nr(G) with Minpath-MethodB

According to Colbourn [1], similar to equation (5.7.1) one can argue that

$$RelPol[p] = 1 - \sum_{i=0}^{ne} cs[i] (1-p)^i p^{ne-i}, \quad (5.8.1)$$

where $cs[i]$ is the number of cutsets of cardinality i for any network G .

Let $Y_1, Y_2, \dots, Y_t \subseteq E$ be the minpaths of some network $G = (V, E)$. It is well known and easy to see that the induced set filter \mathcal{F} of transversals consists of all cutsets of G . Hence MethodB from Section 5.6.2 yields the numbers $cs[0], \dots, cs[ne]$ required in Formula 5.8.1. The overall algorithm, thus including the calculation of the minpaths, will be called *Minpath-MethodB*. In the example below we also replace the previous ad hoc argumentation by a systematic technique for cardinality-wise counting the transversals within a 012e-row.

Example 5.8.1. Consider our running example G_0 (Figure 1.1) and its 16 minpaths (5.2) computed using the Mathematica `FindPath` command. Feeding these

16 minpaths to MethodB yields the family \mathcal{F} of all cutsets X as a disjoint union of 32 012e-rows r_1 to r_{32} . (Check Appendix (B)). In other words

$$\mathcal{F} = r_1 \uplus r_2 \uplus \cdots \uplus r_{32}.$$

The cardinalities $|r_i|$ (the number of cutsets $X \in r_i$) are easily determined, check for instance $|r_{27}| = 9$, and say $\{2, 3, 5, 7, 8, 9\} \in r_{27}$ but $\{2, 3, 5, 6, 9\} \notin r_{27}$ and $|r_{32}| = 2^8 = 256$.

Generally, if

$$r = \underbrace{\{0, \dots, 0\}}_{\alpha \text{ many}}, \underbrace{\{1, \dots, 1\}}_{\beta \text{ many}}, \underbrace{\{2, \dots, 2\}}_{\gamma \text{ many}}, \underbrace{\{e_1, \dots, e_1\}}_{\varepsilon_1 \text{ many}}, \dots, \underbrace{\{e_t, \dots, e_t\}}_{\varepsilon_t \text{ many}} \quad (5.8.2)$$

then [7]

$$|r| = 2^\gamma \cdot (2^{\varepsilon_1} - 1) \cdot (2^{\varepsilon_2} - 1) \cdots (2^{\varepsilon_t} - 1). \quad (5.8.3)$$

Using the formula (5.8.3) one can calculate

$$|\mathcal{F}| = |r_1| + \cdots + |r_{32}|.$$

To calculate $RelPol[p]$, we need the numbers (5.8.1)

$$cs[k] := |\{X \in \mathcal{F} : |X| = k\}| \quad (0 \leq k \leq 11).$$

Evidently, if we define

$$card(r_i, k) := |\{X \in r_i : |X| = k\}|,$$

then we have

$$cs[k] = \sum_{i=1}^{32} card(r_i, k). \quad (5.8.4)$$

Fortunately one can obtain the numbers $card(r_i, k)$ as the coefficients of a suitable polynomial

$$poly(y) = poly(r, y)$$

which is dependent on the particular shape of the 012e-row r . Namely, if r is as in (Equation 5.8.2) then its associated polynomial is

$$poly(r, y) := y^\beta \cdot (y + 1)^\gamma \cdot [(y + 1)^{\varepsilon_1} - 1] \cdots [(y + 1)^{\varepsilon_t} - 1]. \quad (5.8.5)$$

Implementing (Equation 5.8.5) on r_1 to r_{32} as detailed in appendix (B), yields

1. $poly(r_1, y) = y^6$.
2. $poly(r_2, y) = y^5 + 2y^6 + y^7$.
3. $poly(r_3, y) = 2y^6 + y^7$.
4. $poly(r_4, y) = y^5 + 3y^6 + 3y^7 + y^8$.
5. $poly(r_5, y) = 2y^6 + y^7$.
6. $poly(r_6, y) = y^5 + 3y^6 + 3y^7 + y^8$.
7. $poly(r_7, y) = y^4 + 5y^5 + 10y^6 + 10y^7 + 5y^8 + y^9$.
8. $poly(r_8, y) = y^6$.
9. $poly(r_9, y) = y^5 + 2y^6 + y^7$.
10. $poly(r_{10}, y) = 2y^6 + y^7$.
11. $poly(r_{11}, y) = y^5 + 3y^6 + 3y^7 + y^8$.
12. $poly(r_{12}, y) = 2y^6 + y^7$.
13. $poly(r_{13}, y) = y^5 + 3y^6 + 3y^7 + y^8$.
14. $poly(r_{14}, y) = y^4 + 5y^5 + 10y^6 + 10y^7 + 5y^8 + y^9$.
15. $poly(r_{15}, y) = y^3 + 7y^4 + 21y^5 + 35y^6 + 35y^7 + 21y^8 + 7y^9 + y^{10}$.
16. $poly(r_{16}, y) = y^5 + 2y^6 + y^7$.
17. $poly(r_{17}, y) = y^4 + 4y^5 + 6y^6 + 4y^7 + y^8$.
18. $poly(r_{18}, y) = y^6$.
19. $poly(r_{19}, y) = 2y^5 + 5y^6 + 4y^7 + y^8$.
20. $poly(r_{20}, y) = 4y^6 + 4y^7 + y^8$.
21. $poly(r_{21}, y) = y^5 + 4y^6 + 6y^7 + 4y^8 + y^9$.
22. $poly(r_{22}, y) = 2y^5 + 5y^6 + 4y^7 + y^8$.
23. $poly(r_{23}, y) = y^4 + 5y^5 + 10y^6 + 10y^7 + 5y^8 + y^9$.

$$24. \text{poly}(r_{24}, y) = 2y^6 + y^7.$$

$$25. \text{poly}(r_{25}, y) = y^5 + 3y^6 + 3y^7 + y^8.$$

$$26. \text{poly}(r_{26}, y) = 2y^4 + 11y^5 + 25y^6 + 30y^7 + 20y^8 + 7y^9 + y^{10}.$$

$$27. \text{poly}(r_{27}, y) = 4y^6 + 4y^7 + y^8.$$

$$28. \text{poly}(r_{28}, y) = 2y^5 + 7y^6 + 9y^7 + 5y^8 + y^9.$$

$$29. \text{poly}(r_{29}, y) = 2y^6 + y^7.$$

$$30. \text{poly}(r_{30}, y) = 2y^5 + 7y^6 + 9y^7 + 5y^8 + y^9.$$

$$31. \text{poly}(r_{31}, y) = y^4 + 6y^5 + 15y^6 + 20y^7 + 15y^8 + 6y^9 + y^{10}.$$

$$32. \text{poly}(r_{32}, y) = y^3 + 8y^4 + 28y^5 + 56y^6 + 70y^7 + 56y^8 + 28y^9 + 8y^{10} + y^{11}.$$

Applying formula (5.8.4) one can check that indeed

$$\text{card}(r_{27}, 6) = 4$$

since the 6-element cutsets in r_{27} are

$$T \cup \{6, 7\}, T \cup \{6, 10\}, T \cup \{9, 7\}, T \cup \{9, 10\}, \text{ where } T = \{2, 3, 5, 8\}.$$

To compute the number $cs[k]$ of cutsets of cardinality k , we add the coefficients $a[i, k]$ of the y^k terms in the polynomials $\text{poly}(r_i, y)$.

As an illustration in (Equation 5.8.4) we begin by counting cutsets of cardinality 9. The non-zero coefficients $a[i, 9]$ are $a[7, 9]$, $a[14, 9]$, and so on. Adding them up yields $cs[9] = 1 + 1 + 7 + 1 + 1 + 7 + 1 + 1 + 6 + 28 = 54$.

For our running example (Figure 1.1), we have the following

1. $cs[0] = 0$ (cutsets of cardinality 0).
2. $cs[1] = 0$ (cutsets of cardinality 1).
3. $cs[2] = 0$ (cutsets of cardinality 2).
4. $cs[3] = 2$ (cutsets of cardinality 3).
5. $cs[4] = 22$ (cutsets of cardinality 4).

6. $cs[5] = 102$ (cutsets of cardinality 5).
7. $cs[6] = 239$ (cutsets of cardinality 6).
8. $cs[7] = 253$ (cutsets of cardinality 7).
9. $cs[8] = 151$ (cutsets of cardinality 8).
10. $cs[9] = 54$ (cutsets of cardinality 9).
11. $cs[10] = 11$ (cutsets of cardinality 10).
12. $cs[11] = 1$ (cutsets of cardinality 11).

Using the formula in Equation (5.8.1)) we obtain

$$RelPol[p] = 1 - ((54(1-p)^9 p^2) + \dots + ((1-p)^{11})).$$

The computed network reliability is hence

$$= p^2 + 5p^3 + p^4 - 8p^5 - 19p^6 + 12p^7 + 66p^8 - 102p^9 + 56p^{10} - 11p^{11}$$

which coincides with the result derived from contraction-deletion in Chapter 4.

5.9 Calculating $nr(G)$ with Mincut-MethodA or Mincut-MethodB

Recall how $RelPol[p]$ can be computed by virtue of (5.7.1) and (5.8.1). In Sections 5.7 and 5.8 we first calculated all minpaths (using Mathematica's FindPath command) and then employed MethodA and MethodB to evaluate (5.7.1) and (5.8.1) respectively.

Recall that MethodA and MethodB apply to *arbitrary* set filters. So Minpath-MethodA was just a shorthand for applying MethodA to the set filter generated by the minpaths (in order to evaluate (5.7.1)). Similarly we shall use the shorthand *Mincut-MethodA* for applying MethodA to the set filter generated by the mincuts (in order to evaluate (5.8.1)). Likewise the Minpath-MethodB (which evaluates (5.8.1)) translates to an obvious *Mincut-MethodB* which evaluates (5.7.1).

Chapter 6

Comparison of algorithms

6.1 Introduction

All the algorithms in the previous chapters have been programmed in high level Wolfram Mathematica 11.2 and our results are implemented on a Linux 16.04 operating system. The algorithms are compared according to the CPU time (in seconds). Altogether 34 random networks, see Appendix C, will be tested.

When a random graph is defined, the output can either be a *2-connected* network or a *tree-like* network. Recall that '2-connected' mean that any two vertices are part of at least one cycle.

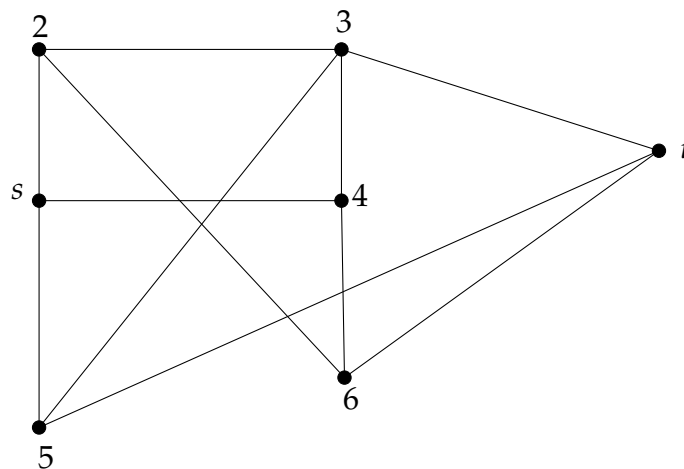


Figure 6.1: Example of a 2-connected network.

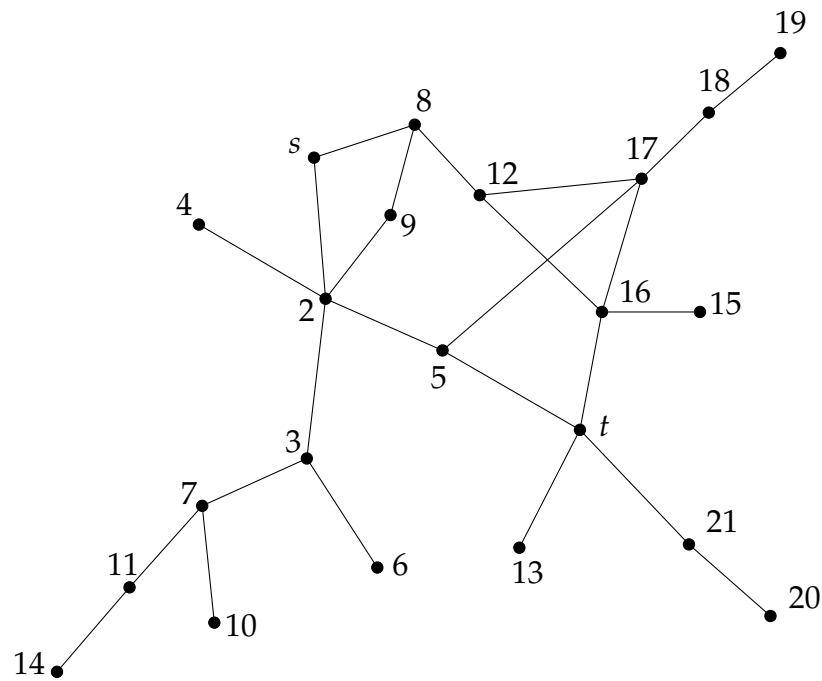


Figure 6.2: Example of a tree-like network.

6.2 Computational results of the Mathematica algorithms

Table 6.1 and 6.2 show the parameters of the networks analysed and the CPU times (indicated in the column of the method). *NF* means 'not feasible' in 24 hours. For instance, with respect to the number of edges exhaustive state enumeration is not doable in reasonable time if $ne > 20$. Similarly the naive inclusion-exclusion on minpaths is only feasible for at most 20 minpaths i.e. $nmp \leq 20$ which is very rare for random instances.

Table 6.1: CPU time (in seconds) for the Mathematica programs on 2-connected networks

ne	nv	mp_i	mc_i	State enumeration	Inclusion-exclusion	Minpath-MethodA	Minpath-MethodB	Mincut-MethodA	Mincut-MethodB	Contraction-deletion
8	5	8	6	0.024	0.008	0.004	0.004	0.004	0.004	0.008
9	6	13	9	0.044	0.356	0.004	0.008	0.004	0.004	0.008
11	7	16	23	0.172	3.468	0.012	0.016	0.016	0.032	0.016
13	7	26	22	0.788	5654.47	0.016	0.064	0.02	0.04	0.032
14	8	37	29	1.324	NF	0.036	0.172	0.036	0.092	0.044
15	8	38	32	3.204	NF	0.04	0.224	0.04	0.112	0.052
16	9	47	38	5.732	NF	0.068	0.448	0.052	0.224	0.08
18	9	59	44	27.192	NF	0.12	1.144	0.128	0.272	0.208
20	10	95	60	111.292	NF	0.39	6.625	0.236	1.406	0.64
22	11	197	166	500.532	NF	1.968	49.244	2.196	16.412	2.836
23	11	176	133	973.356	NF	1.46	40.54	1.004	8.188	1.776
25	11	408	225	4506.4	NF	11.692	766.732	8.112	49.696	12.32
27	12	642	245	NF	NF	26.46	2971.22	9.528	78.264	21.168
28	12	738	242	NF	NF	63.3125	3195.34	9.734	73.125	31.375
30	13	1637	402	NF	NF	249.144	NF	45.828	419.108	272.612
32	14	1397	759	NF	NF	340.578	NF	139.609	3614.28	363.18
35	16	2226	1450	NF	NF	1256.06	NF	574.332	NF	912.188
37	17	7552	1534	NF	NF	NF	NF	2250.13	NF	NF
45	22	27894	8924	NF	NF	NF	NF	23233.2	NF	NF

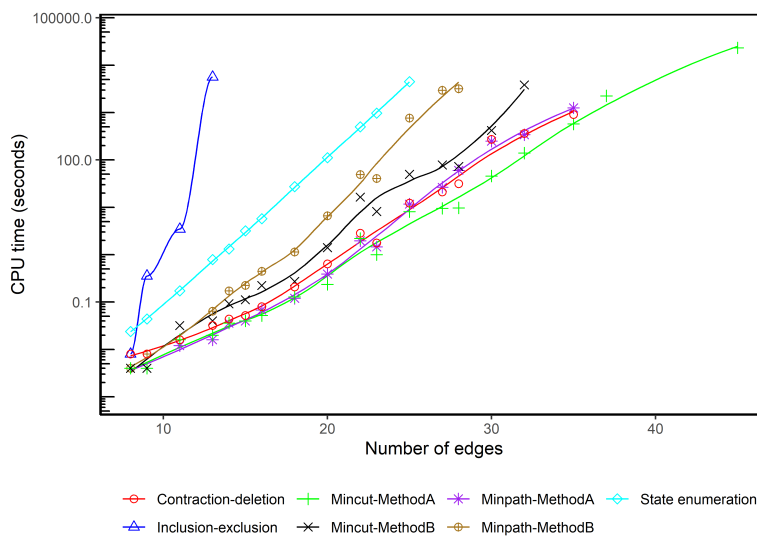


Figure 6.3: Graphical analysis of the CPU time (in seconds) against number of edges for 2-connected networks.

Table 6.2: CPU time (in seconds) for the Mathematica programs on tree-like networks

ne	nv	mp_i	mc_i	State enumeration	Inclusion-exclusion	Minpath-MethodA	Minpath-MethodB	Mincut-MethodA	Mincut-MethodB	Contraction-deletion
12	9	7	10	0.368	0.004	0.004	0.004	0.004	0.004	0.008
16	10	26	29	6.192	4474.06	0.024	0.072	0.036	0.076	0.032
18	13	32	23	20.568	NF	0.032	0.092	0.024	0.084	0.132
20	14	33	48	110.12	NF	0.044	0.08	0.096	0.332	0.52
22	15	70	93	365.032	NF	0.148	1.332	0.248	3.012	1.544
28	20	67	352	NF	NF	0.252	1.448	3.972	60.128	489.664
31	22	103	359	NF	NF	0.476	4.988	5.524	76.792	49983.6
35	26	142	249	NF	NF	0.888	11.872	2.648	25.66	44312.7
38	30	178	11395	NF	NF	6.776	57.052	NF	NF	NF
43	32	581	6970	NF	NF	43.316	1672.4	NF	NF	NF
46	35	573	3488	NF	NF	48.36	1634.06	1202.21	NF	NF
49	38	593	8765	NF	NF	43.748	1249.23	NF	NF	NF
54	42	868	20990	NF	NF	102.808	5203.04	NF	NF	NF
60	47	1356	> 150000	NF	NF	500.624	NF	NF	NF	NF

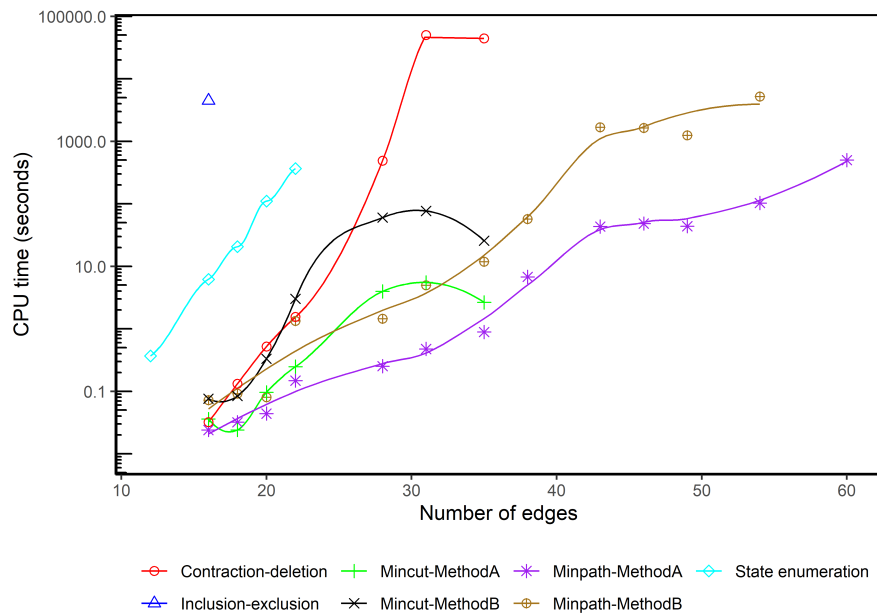


Figure 6.4: Graphical analysis of the CPU time (in seconds) against number of edges for tree-like networks.

We have tested the series-parallel network (Figure 3.3) from Chapter 3 to compare the results with other methods.

Table 6.3: CPU time (in seconds) for the Mathematica programs on series-parallel network, $ne=17$

ne	nv	mp_i	mc_i	State enumeration	Inclusion-exclusion	Minpath-MethodA	Minpath-MethodB	Mincut-MethodA	Mincut-MethodB	Contraction-deletion	s,p-reduction
17	12	8	54	11.672	0.024	0.012	0.004	0.084	0.032	0.024	0.004

6.3 Algorithm recommendation

Following the detailed comparison of the techniques corresponding to the CPU time we now provide Table 6.4 that shows our recommendation. The entries in the columns are in decreasing order of preference.

Table 6.4: Method recommendation when calculating the network reliability

2-connected networks	Tree-like networks	Series-parallel network
Mincut-MethodA	Minpath-MethodA	Minpath-MethodB / Series-parallel reduction
Contraction-deletion	Minpath-MethodB	Minpath-MethodA
Minpath-MethodA		

==== Appendices =====

Appendix A

Parameters for random networks used to test FindPath command and SimpleMincut method

Table A.1: Parameters and CPU time of the random networks tested using FindPath command

ne	nv	mp_i	FindPath command
38	17	30,727	0.024
40	20	17,901	0.016
45	20	138,307	0.108
46	25	40,409	0.036
57	29	362,849	0.344
58	30	1,007,504	1.036
63	30	1,880,656	2.208
71	39	4,562,374	5.70313
75	40	10,671,339	12.125
86	49	17,020,105	21.9219

Table A.2: Parameters and CPU time of the random networks tested using SimpleMincut algorithm

ne	nv	mc_i	SimpleMincut
11	7	23	0.004
20	10	140	0.046875
25	13	500	0.276
32	14	1,089	0.671875
35	17	6,074	4.96875
36	16	3,038	1.89063
36	18	8,380	8.875
41	19	17,390	14.372

Appendix B

Mincuts and Minpaths

Table B.1: The transversal e-algorithm compact representation of the family of all cutsets X as disjoint union of thirty two 012e-rows r_1 to r_{32} .

	1	2	3	4	5	6	7	8	9	10	11
r_1	0	0	0	1	1	1	1	0	1	1	0
r_2	1	0	0	2	2	1	1	0	1	1	0
r_3	1	0	0	0	e_1	1	1	0	e_1	1	1
r_4	2	0	0	1	2	1	1	0	2	1	1
r_5	0	1	0	1	0	e_1	1	0	e_1	1	1
r_6	0	1	0	1	1	2	1	0	2	1	2
r_7	1	1	0	2	2	2	1	0	2	1	2
r_8	1	1	0	1	0	1	0	1	0	1	0
r_9	1	1	0	2	0	2	1	1	0	1	0
r_{10}	e_1	1	0	e_1	1	0	1	1	0	1	0
r_{11}	2	1	0	2	1	1	2	1	0	1	0
r_{12}	1	e_1	0	0	0	e_1	1	1	1	1	0
r_{13}	1	2	0	1	0	2	2	1	1	1	0
r_{14}	2	2	0	2	1	2	2	1	1	1	0
r_{15}	2	2	0	2	2	2	2	1	2	1	1
r_{16}	0	0	1	1	1	1	2	0	1	2	0
r_{17}	1	0	1	2	2	1	2	0	1	2	0
r_{18}	0	0	1	1	1	1	0	1	1	0	0
r_{19}	0	0	1	2	1	2	e_1	1	1	e_1	0
r_{20}	1	0	1	e_2	e_2	0	e_1	1	1	e_1	0
r_{21}	1	0	1	2	2	1	2	1	1	2	0
r_{22}	1	0	1	0	e_2	1	e_1	0	e_2	e_1	1
r_{23}	2	0	1	1	2	1	2	0	2	2	1
r_{24}	1	0	1	0	e_1	1	0	1	e_1	0	1
r_{25}	2	0	1	1	2	1	0	1	2	0	1
r_{26}	2	0	1	2	2	2	e_1	1	2	e_1	1
r_{27}	0	1	1	0	1	e_2	e_1	1	e_2	e_1	0
r_{28}	0	1	1	0	2	2	e_1	1	2	e_1	1
r_{29}	0	1	1	1	0	0	e_1	1	0	e_1	1
r_{30}	0	1	1	1	0	e_1	2	2	e_1	2	1
r_{31}	0	1	1	1	1	2	2	2	2	2	2
r_{32}	1	1	1	2	2	2	2	2	2	2	2

Appendix C

Random networks tested

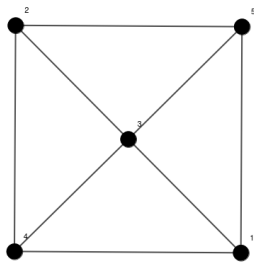


Figure C.1: $G=([5],[8])$.

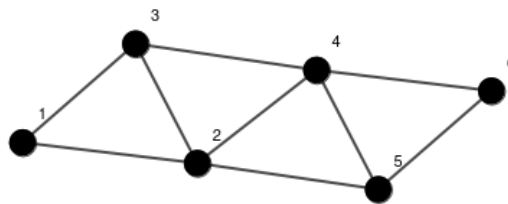


Figure C.2: $G=([6],[9])$.

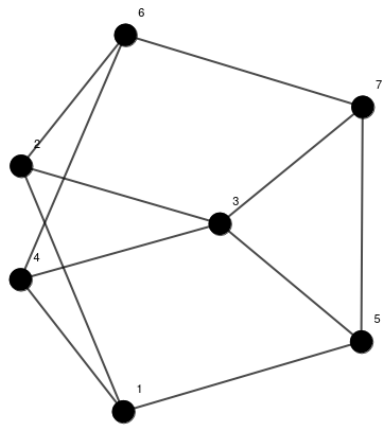


Figure C.3: $G=([7],[11])$.

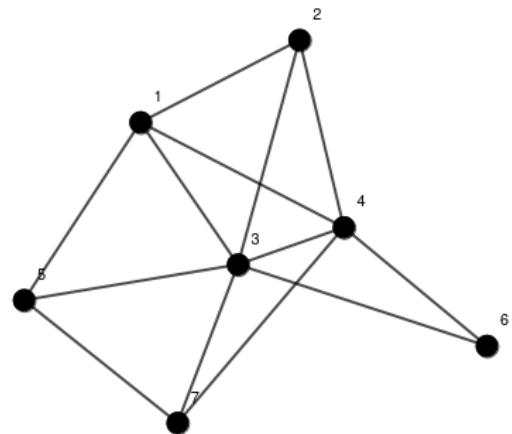


Figure C.4: $G=([7],[13])$.

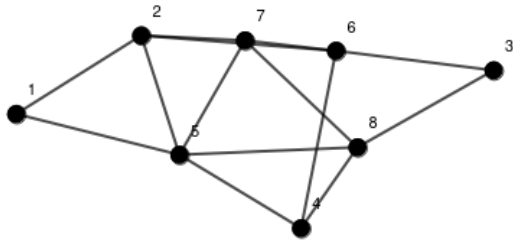


Figure C.5: $G=([8],[14])$.

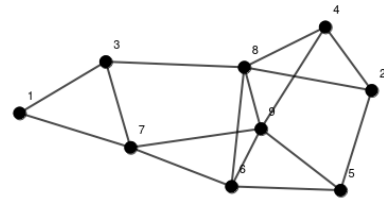


Figure C.6: $G=([8],[15])$.

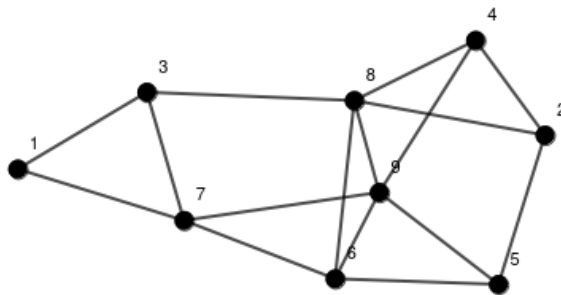


Figure C.7: $G=([9],[16])$.

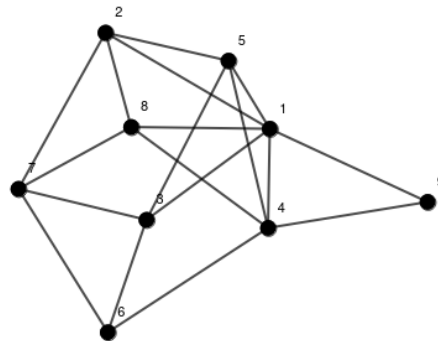


Figure C.8: $G=([9],[18])$.

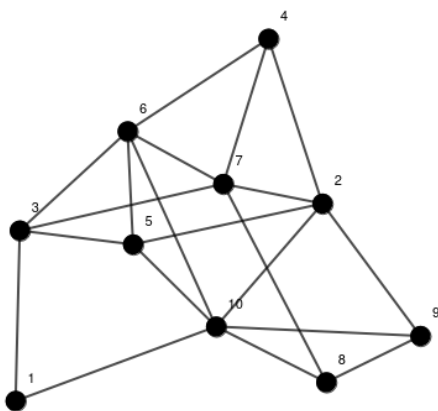


Figure C.9: $G=([10],[20])$.

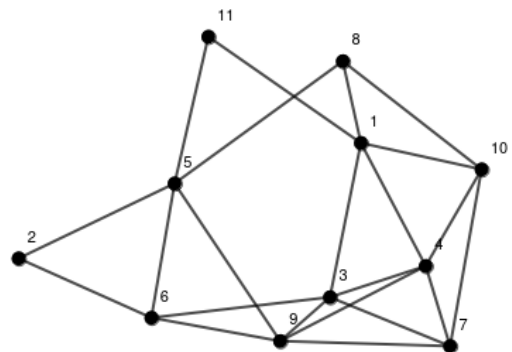


Figure C.10: $G=([11],[22])$.

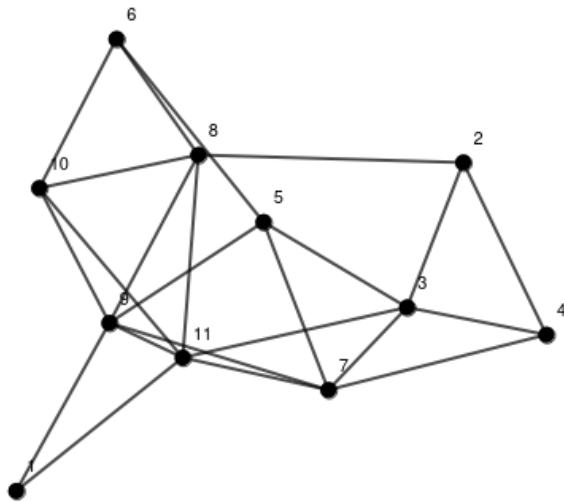


Figure C.11: $G=([11],[23])$.

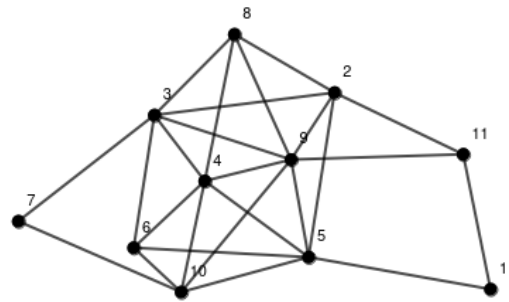


Figure C.12: $G=([7],[25])$.

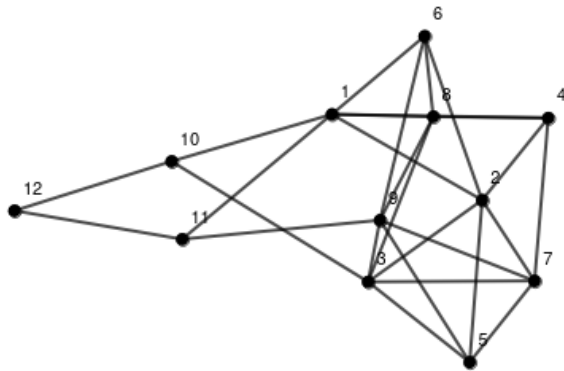


Figure C.13: $G=([12],[27])$.

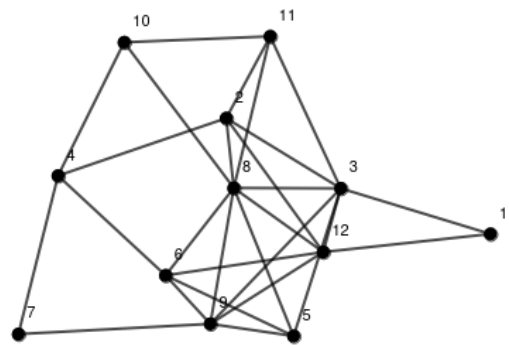


Figure C.14: $G=([12],[28])$.

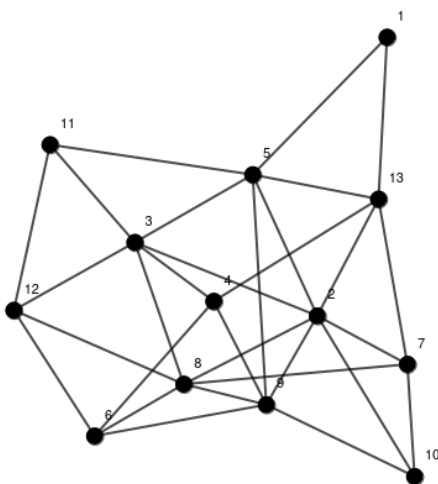


Figure C.15: $G=([13],[30])$.

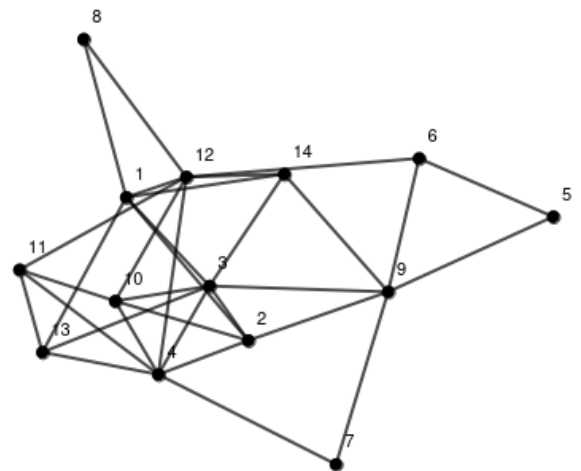


Figure C.16: $G=([14],[32])$.

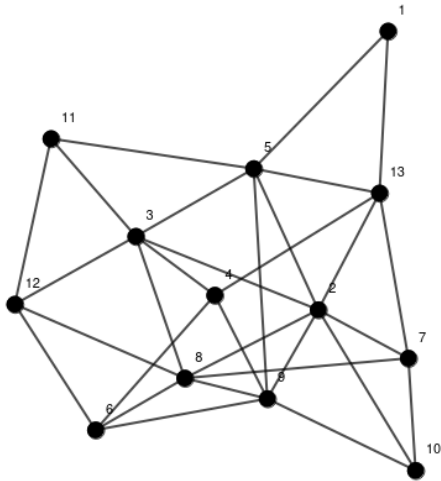


Figure C.17: $G=([16],[35])$.

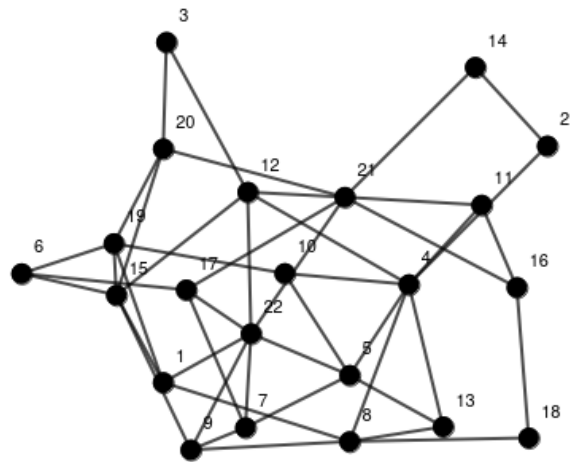


Figure C.18: $G=([22],[45])$.

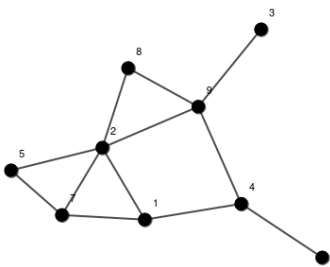


Figure C.19: $G=([9],[12])$.

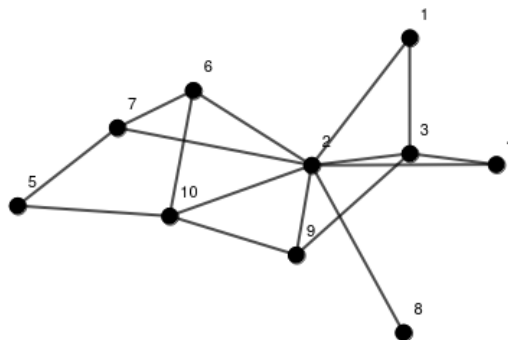


Figure C.20: $G=([10],[16])$.

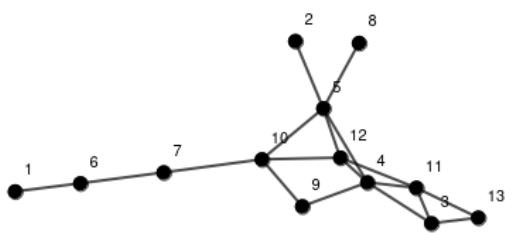


Figure C.21: $G=([13],[18])$.

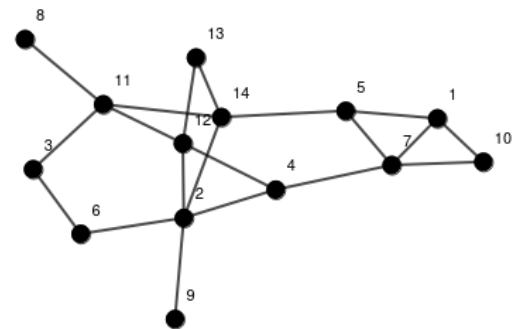


Figure C.22: $G=([14],[20])$.

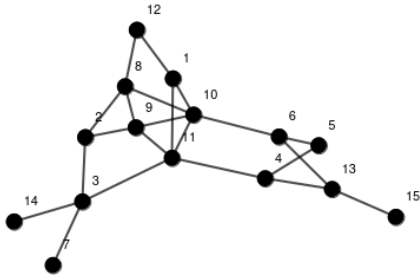


Figure C.23: $G=([15],[22])$.

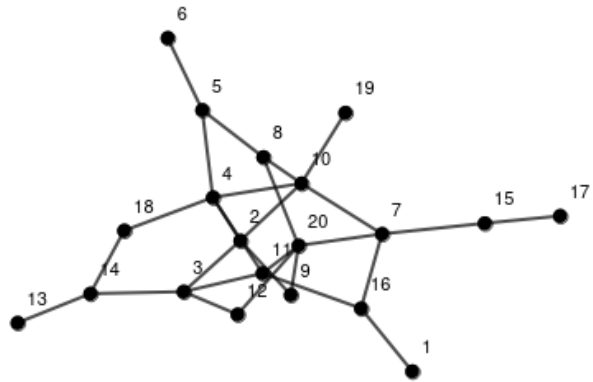


Figure C.24: $G=([20],[28])$.

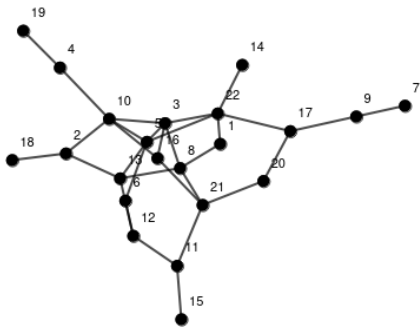


Figure C.25: $G=([22],[31])$.

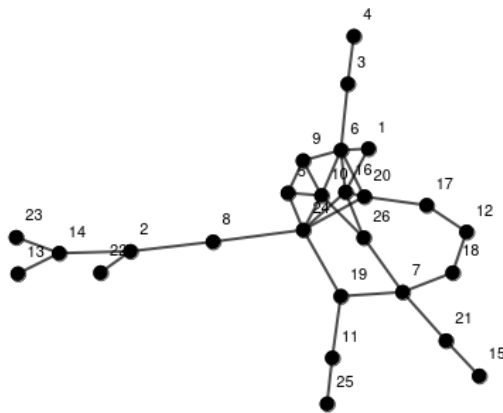


Figure C.26: $G=([26],[35])$.

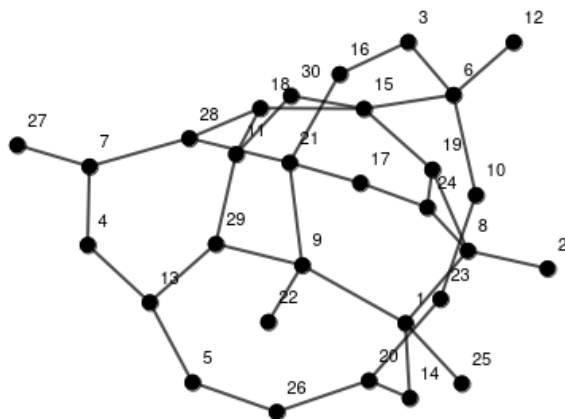


Figure C.27: $G=([30],[38])$.

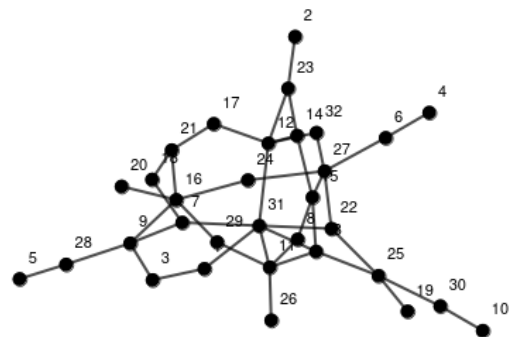


Figure C.28: $G=([32],[43])$.

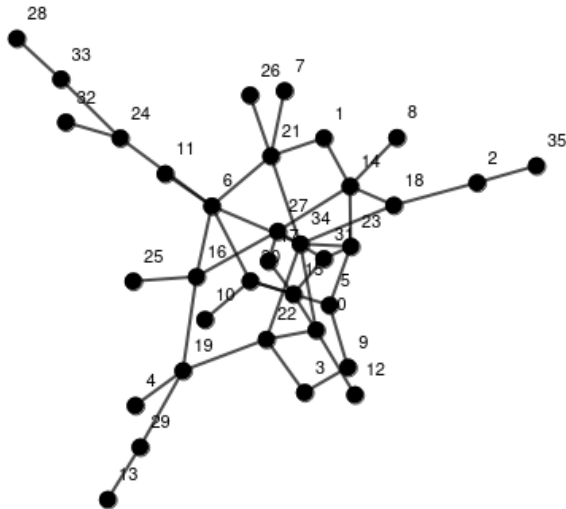


Figure C.29: $G=([35],[46])$.

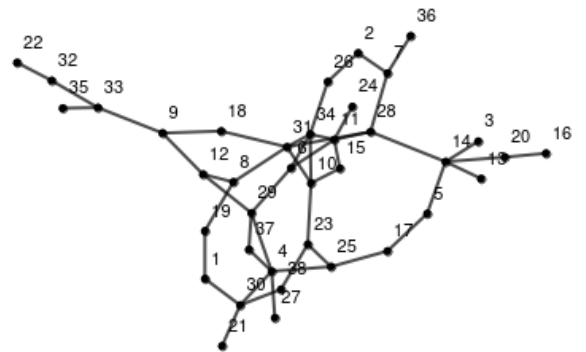


Figure C.30: $G=([38],[49])$.

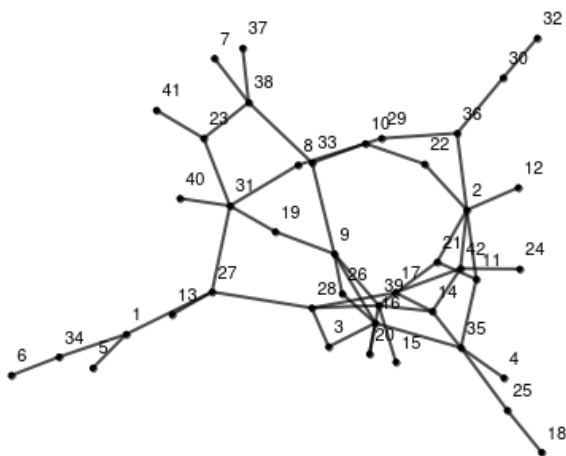


Figure C.31: $G=([42],[54])$.

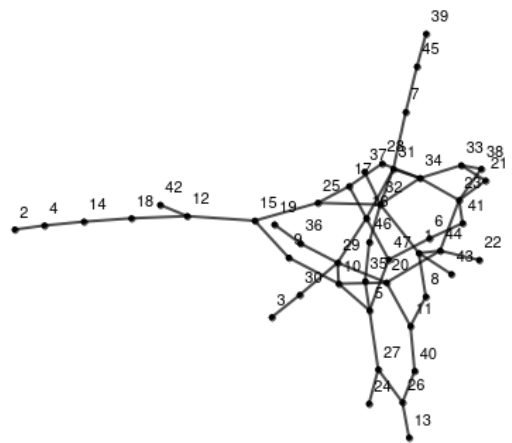


Figure C.32: $G=([47],[60])$.

List of References

- [1] C. J. Colbourn. *The combinatorics of network reliability, International Series of Monographs on Computer Science*, volume 198. The Clarendon Press Oxford University Press, New York, 1987.
- [2] E. F. Moore and C. E. Shannon. Reliable circuits using less reliable relays. *Journal of the Franklin Institute*, 262(3):191–208, 1956.
- [3] F. Moskowitz. The analysis of redundancy networks. *Transactions of the American institute of electrical engineers, part i: communication and electronics*, 77(5):627–632, 1958.
- [4] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.
- [5] D. R. Shier. *Network reliability and algebraic structures*. Clarendon Press, 1991.
- [6] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [7] M. Wild. Allsat compressed with wildcards: Partitionings and face-numbers of simplicial complexes. *arXiv:1812.02570*.
- [8] M. Wild. Counting or producing all fixed cardinality transversals. *Algorithmica*, 69(1):117–129, 2014.