



**UNIVERSITY
OF LONDON**

Operations research and combinatorial optimisation

R.W. Whitty

CO3352

2010

Undergraduate study in
Computing and related programmes

This guide was prepared for the University of London by:

Robin Whitty BSc PhD CMath FIMA, Faculty of Business, London South Bank University.

The guide was produced by:

Sarah Rauchas, Department of Computing, Goldsmiths, University of London.

This is one of a series of subject guides published by the University. We regret that due to pressure of work the authors are unable to enter into any correspondence relating to, or arising from, the guide. If you have any comments on this subject guide, favourable or unfavourable, please use the form at the back of this guide.

In this and other publications you may see references to the 'University of London International Programmes', which was the name of the University's flexible and distance learning arm until 2018. It is now known simply as the 'University of London', which better reflects the academic award our students are working towards. The change in name will be incorporated into our materials as they are revised.

University of London
Publications Office
32 Russell Square
London WC1B 5DN
United Kingdom
london.ac.uk

Published by: University of London

© University of London 2010

The University of London asserts copyright over all material in this subject guide except where otherwise indicated. All rights reserved. No part of this work may be reproduced in any form, or by any means, without permission in writing from the publisher. We make every effort to respect copyright. If you think we have inadvertently used your copyright material, please let us know.

Contents

Preface	iii
Aims and objectives of this unit	v
Advice on study/reading	vii
Assessment of this unit	ix
1 Matroids	1
1.1 Introduction	1
1.2 Matroids as algebraic objects	1
1.2.1 Linear algebra	2
1.2.2 Matroids defined by axioms	5
1.3 Matroids and the greedy algorithm	7
1.3.1 Graph theory	8
1.3.2 Minimum-weight spanning trees	9
1.3.3 Minimum-cost matching	12
1.3.4 The greedy algorithm	14
2 Representations of matroids	19
2.1 Introduction	19
2.2 Matrix algebra	20
2.2.1 Matrices and submatrices	20
2.2.2 The determinant	20
2.2.3 Multiplication and the Binet-Cauchy Theorem	24
2.3 Representations	28
2.3.1 How to represent a matroid	28
2.3.2 Partition matroids	30
2.3.3 Graphic matroids	32
2.3.4 Cographic matroids and duality	35
2.4 Transversal matroids	40
2.4.1 The Mirsky-Perfect representation	40
2.4.2 The Piff-Welsh Theorem	42
2.4.3 The matching matroid	49
3 Matroid intersection	53
3.1 Introduction	53
3.2 The intersection of two matroids	53
3.3 Applications of matroid intersection	57
3.3.1 Bipartite matching	57
3.3.2 Graceful trees	59
3.3.3 Branchings	61
3.3.4 Edge-disjoint spanning trees	64
3.3.5 Simultaneous transversals	67
3.4 Beyond matroids	69
3.4.1 Good characterisations and hard problems	69
3.4.2 Three-matroid intersection	71
4 Linear and integer linear programming	75
4.1 Introduction	75
4.2 Complexity of algorithms and problems	75
4.2.1 Algorithmic complexity	75

4.2.2	Computational complexity	78
4.2.3	Does $\mathbf{P} = \mathbf{NP}$?	82
4.3	Polyhedral optimisation	83
4.3.1	Linear programming	83
4.3.2	Convexity	86
4.3.3	Duality	88
4.3.4	Integer linear programming	91
4.3.5	Totally unimodular matrices	95
4.4	Matroids again	99
4.4.1	The Birkhoff polytope	99
4.4.2	Matroid polytopes	102
A	Further reading	103
A.1	Books	103
A.2	Journals	104
A.3	Websites	105
B	Specimen examination	107
B.1	Examination paper	107
B.2	Model answers	111
C	Solutions to exercises	115
C.1	Solutions for Chapter 1	115
C.2	Solutions for Chapter 2	122
C.3	Solutions for Chapter 3	131
C.4	Solutions for Chapter 4	137
	Index	145

Preface

Operations, or Operational, Research (OR) has been defined* as ‘The Science of Better’. For the end-user it offers a collection of methods for getting things done more quickly, more cheaply or to a higher standard of quality. It is the job of the management scientist to make sure that these methods are practical and relevant. At the other end of the spectrum, the mathematician works to refine them using the latest advances in abstract theory. The role of the computing professional is to maintain the link between the two; to over-simplify, they are there to extract the *algorithms* from the *theorems*.

This study guide will give a computer scientist’s view of the mathematics on offer to management scientists. Largely, the body of mathematics we will be concerned with is ‘combinatorial optimisation’ which deals with maximising or minimising over a finite set of objects. Combinatorics is a subject which over the last fifty years has advanced enormously in terms of acquiring deep and coherent abstract foundations. The computer scientist wants to take advantage of the depth and coherence while avoiding the abstraction! At least, things must be kept as concrete as possible and theorems that cannot be turned into algorithms, or proofs of correctness of algorithms, will not be the first focus of attention.

In the light of modern theory it is arguable, and we do argue it, that OR begins in the branch of combinatorics called matroid theory. Matroid theory can be viewed as a theory of easy algorithms; when the problem at hand is to maximise or minimise something, the first question is: can we optimise by just always taking the largest or smallest at hand? There are surprisingly many optimisation problems for which the answer to this question is ‘Yes’: they can be solved or partially solved by this so-called ‘greedy’ approach. This first approach to optimisation is so fruitful in fact that the bulk of this study guide is devoted to matroid theory!

Chapter 1 provides the basics; it culminates in the greedy algorithm and the theorem that says that it works. A typical problem addressed in Chapter 1 is the Job Assignment problem: it may be partially solved using a so-called transversal matroid. **Chapter 2** is effectively devoted to computer representations of matroids and culminates in the Piff-Welsh theorem which says that transversal matroids have representations. We may then implement our partial solution to Job Assignment by recasting the theorem as an algorithm — an illustration of our computer science perspective since, although Piff and Welsh gave a constructive proof of their theorem, this requires some care to recast in algorithmic form. **Chapter 3** extends our capabilities by showing that, even when a problem resists the greedy approach, pairs of matroids may be intersected together to solve it; by the end of this chapter we will completely solve Job Assignment as well as many other not-so-easy problems. We end this chapter by opening up a discussion on the nature of ‘hard’ optimisation problems, some of which can be expressed, but apparently not efficiently solved, in matroid terms.

Chapter 4 then develops the theory of computation complexity and NP-completeness. This is an essential part of the technical equipment of anybody working in the field of optimisation; in this guide it gives us the context for our treatment of that cornerstone of OR, linear programming and the theory of convex polyhedra. We are particularly interested to distinguish those OR problems which may be posed as *integer* linear programmes, or ILPs, and it soon becomes clear that computational complexity plays a key role in this distinction. As a coda to Chapter 4 we discover that optimisation using matroids corresponds to dealing with convex polyhedra whose vertices are integer-valued; such polyhedra provide a special, efficiently solvable sub-class of the ILPs.

Our journey ends at this point. We stop short of many classical OR problems which are much harder to solve than those within reach of matroid theory or even linear programming. In the language of computational complexity, we have stayed within the realm of polynomial-time problems, leaving untouched those which are currently (and perhaps inherently) exponential-time in their complexity. These

*‘The Science of Better’ is a tag-line (and service mark) adopted by the Institute for Operations Research and the Management Sciences (INFORMS). They even provide a separate promotional website www.scienceofbetter.org. You will find this and many other links in Appendix A: Further Reading.

apparently harder problems will not solve themselves and the OR profession has equipped itself with a large collection of techniques for approximating solutions and for solving special cases; the notorious Travelling Salesman Problem, or TSP, is a case in point and this and some similar problems are explored on the CD-ROM which accompanies this study guide. The responses of OR to such problems are often highly inventive and effective but they are necessarily ad hoc: they do not offer such a coherent picture of optimisation as matroid theory or convexity theory. In starting from an in-depth study of matroids we hope to offer a secure basis for further study or professional practice.

There is a pedagogic advantage in exploring, in depth, one particular approach to optimisation: the prerequisites may be kept to a minimum and there is a reasonable chance of keeping the exploration self-contained. Matroid theory, and particularly our account of it, is based in linear algebra. Some of this, matrix algebra for example, will have been met in earlier units but we review almost all of what we need as we go along. All that is taken for granted is some background in set theory, including the idea of a function; some elementary combinatorics, including the factorial and binomial notations; the idea of a polynomial in one or several variables; and some basic 2- and 3-dimensional coordinate geometry. You will also benefit from having a little of what is usually referred to as ‘mathematical maturity’ — being used to the idea that mathematics is made up of axioms and theorems and proofs; and being used to attempting mathematically formulated exercises.

The exercises are an important part of this guide. They are placed throughout the text, rather than being collected together at the ends of chapters. They are intended to reinforce the topics as they are introduced and they are mostly deliberately easy or at least routine. *Do not ignore them!* If you can do them they will build up your confidence and embed the material in your memory; if you cannot do them then this is a warning sign — you have misunderstood something or there are some gaps in your background which need attention if you are to succeed with this unit. The appendix called *Further Reading* will suggest books and Internet sites which can help you.

The answers to all exercises are given in an appendix. It cannot be stressed too highly that an exercise will not fully reward you unless you try it before checking the solution! Conversely, it is usually a good idea to look at the solution even if you found the exercise easy — it may be that you get some tips on the best way to present a solution which will be useful for you in the examination.

An orthographical note is in order: anglicised spellings are used throughout this guide, thus: analyse, optimise, initialise, anglicise. This will cause no confusion but should be remembered if the words are sought in the indexes of textbooks where they may be spelt with a ‘z’: analyze, optimize, etc. Internet searching, also, may well yield more results if American spellings are used. Typing the words “four colour theorem” (in quotation marks) into Google produced, at the time of writing, about ten thousand hits; using the spelling “color” produced over twenty-one thousand.

The preparation of this guide has benefited from the help of many people. I should particularly mention Paddy Murphy who did a marvellous job painstakingly checking the mathematics and, especially, putting himself in the place of those students who will eventually be reading this. Some of the material was presented to Carrie Rutherford’s Maths Study Group at London South Bank University and I am grateful for the feedback I received from members of the group. Graeme Eaton of the Open University did a thorough job of reading a draft of this guide; Christina Webster in the Publications Office at The University of London International Programmes provided first-rate technical editing; and Christina Zarb at St Martin’s Institute of Information Technology road-tested many of the exercises. Finally, without the expert advice and monumental patience of Sarah Rauchas and Margaret Stern at Goldsmiths University of London this guide certainly would have never seen the light of day.

Aims and objectives of this unit

The unit aims to offer a modern and computationally-oriented introduction to discrete optimisation. The theory of matroids will be covered in detail as providing a deep and coherent approach to the principles of optimisation. The more advanced topic of matroid intersection will be given a novel treatment using symbolic computation which focuses on the underlying concepts while maintaining a strong link to computing science. This will lead on to a consideration of algorithmic and computational complexity and to the theory of linear and integer linear programming.

Learning Objectives By the end of the unit, the relevant reading and activities, you should be able to:

1. Understand the role of matroids in optimisation in terms of:
 - (a) their definition in terms of linear algebra and axioms
 - (b) their relationship with the greedy algorithm
 - (c) the optimisation problems which they can and cannot solve.
2. Derive matroid representations to support a computational approach to using the various kinds of matroid (partition, graphic, transversal etc).
3. Understand how matroid intersection can be used to (a) formulate and (b) solve more complex problems in optimisation and be able to apply symbolic matrix arithmetic to derive solutions for small problem instances.
4. Appreciate how the theory of computational complexity gives a potential distinction between easy and hard problems; be able to demonstrate simple polynomial reductions in the theory of NP-completeness, including vertex cover and integer linear programming.
5. Formulate problems as linear programmes and solve small problems geometrically. Understand the distinction between linear and integer linear programming.

Advice on study/reading

Subject to what was said above on pre-requisites for this unit and to your having completed some earlier compulsory units on your degree course, this guide aims to be self-contained. The exercises were also referred to earlier: they are your best learning resource! In short, if you have read carefully most of what is in this study guide and attempted carefully most of the exercises then you should feel confident that you have achieved the learning objectives set out in the previous subsection.

Appendix A: Further Reading, offers a large selection of books, journals and websites to which you may refer in order to broaden your understanding of the topics covered in this study guide; the accompanying CD-ROM supplies some software and other resources which will help you explore in more technical detail and on a larger scale, a selection of techniques from OR. The more use you make of this material the better: it will both improve your overall achievement in the unit and provide a context in which to exploit what you learn from the unit.

A typical level-three unit taught at a college of the University of London might involve three one-hour lectures plus some optional one-hour tutorials, so that a successful student would probably study a minimum of forty hours over a ten-week period. You may take this as a rough guide to how much time you will need to spend studying this unit. Of course, extra hours of self-study are very much recommended: if you are going to a tutorial to discuss some exercises or coursework then it pays to have tried them in advance!

Assessment of this unit

This subject guide is quite long! It provides quite a lot of background material on graph theory, linear algebra and coordinate geometry; it also covers some topics more widely than could easily be taught or assimilated within one term's worth of study.

The final examination for this unit contributes a maximum of 80% to the total mark for the unit. A guide to the scope and level of difficulty of the final examination for this unit is the Specimen examination paper given in Appendix B. A more detailed list of what needs to be revised for the examination is given below. You should study:

Chapter 1: Matroids

- 1.2.2 Matroids defined by axioms
- 1.3.2 Minimum-weight spanning trees
- 1.3.3 Minimum-cost matching
- 1.3.4 The greedy algorithm

Chapter 2: Representations of Matroids

- 2.3.2 Partition matroids
- 2.3.3 Graphic matroids
- 2.3.4 Cographic matroids and duality
- 2.4.1 The Mirsky-Perfect representation

Chapter 3: Matroid Intersection

- 3.3.1 Bipartite matching
- 3.3.3 Branchings
- 3.3.4 Edge-disjoint spanning trees
- 3.3.5 Simultaneous transversals

Chapter 4: Linear and Integer Programming

- 4.2.1 Algorithmic complexity
- 4.2.2 Computational complexity
- 4.3.1 Linear programming
- 4.3.2 Convexity
- 4.3.3 Duality
- 4.3.4 Integer linear programming

Bear in mind that, even if a section in this guide is not examinable you should still try to read and understand it. Sometimes it will contain material which you need in order to understand later sections which are examinable. For example, the idea of 'good characterisations', given in section 3.4.1 of Chapter 3 is used in the definition of the class **NP**, in section 4.2.2 in Chapter 4.

An additional reason for not ignoring the more advanced topics omitted from the above list is that they will form the basis for two courseworks, which contribute a maximum of 20% to the total mark for this unit. You may therefore wish to treat the relevant sections of the study guide as self-study material, to be referred to as necessary for the courseworks when they are set.

Neither the examination paper nor the courseworks will ask questions directly about the background mathematical material covered in this study guide (on matrices, vectors, graphs, etc). Nevertheless, familiarity with these topics will be assumed. **It is essential that you become thoroughly familiar with Sections 1.2.1, 1.3.1 and all of 2.2, even though they do not appear on the above list.**

Chapter 1

Matroids

1.1 Introduction

Exercise 1* *Go to the index at the back of this guide and look up ‘OR problem’: you will find more than a dozen entries. Turn to each and spend a minute thinking about how you might solve it.*

The problems on which you have just reflected are representative of a subclass of those addressed by the models and techniques of Operations Research (OR). You will not have noticed the notorious Travelling Salesman Problem (TSP) among them, nor problems to do job shop scheduling nor timetabling (other than simple room allocation); such mountains in the optimisation world tend to be completely surmounted only by exhaustive search, at least with our current understanding, or must be circumvented using sophisticated approximation methods. The subclass of OR problems we have chosen, on the other hand, are relatively easy to solve. They are the foothills around a well-established base camp and it is this territory which we have chosen to explore in this study guide. It is a territory which may be more or less completely mapped out in terms of a beautiful and far-reaching branch of combinatorics called matroid theory.

The idea of a matroid (pronounced ‘*may*-troid’) is largely due to a single American mathematician, Hassler Whitney, in the mid-1930s[†]. He belonged to the era in which modern abstract algebra was constructed by giants like Emil Artin, Garrett Birkhoff, Elie Cartan, Emmy Noether and Saunders MacLane; Birkhoff and MacLane also had a hand in inventing matroid theory, as did Noether’s disciple Bartel Leendert van der Waerden. So matroids are primarily algebraic objects and as such have since become an established field of study for mathematicians, offering a generalising framework for questions in graph theory and finite geometry, coding theory, statics, electrical network theory and even, recently, mathematical physics.

In the 1960s matroids were identified as having a second role in capturing one of the simplest ways to solve an optimisation problem: via the so-called ‘greedy’ approach. It was found, notably by Jack Edmonds, who was then at the National Bureau of Standards and by Eugene Lawler at the University of Michigan that even non-greedy problems could often be dealt with efficiently in terms of matroid ‘intersection’. It is remarkable just how many optimisation problems can be formulated and solved in these terms. The roots of matroid theory are firmly in linear algebra and geometry and these are also the foundation on which convexity theory is based. By the time we have fully explored matroid theory we shall find we already have the background we need to venture further.

We shall first meet matroids, then, as arising out of the idea of linear independence in vector spaces. We shall then start again, as it were, and arrive at matroids by way of certain optimisation problems on graphs (networks). In Chapter 3 we shall return to linear algebra when we discuss the intersection of matroids; indeed, we shall eventually restrict attention to matroids which are ‘representable’ as systems of vectors since this allows a particularly simple account of matroid intersection.

1.2 Matroids as algebraic objects

Modern algebra is based on proposing collections of abstract axioms and studying the properties of objects which satisfy these axioms. Our first approach to matroids will be in terms of axioms which generalise the

*Perhaps this is the first ever study guide to open with an exercise! Incidentally, this is the first and last exercise which does not have an answer in appendix C.

[†]In a striking example of simultaneous co-discovery, the foundations of matroid theory must be attributed also to Takeo Nakasawa, a young Japanese mathematician who joined the settlement of Manchuria in the 1930s and tragically lost his life after World War II, following the Russian annexation of the province. An account is given in *A Lost Mathematician, Takeo Nakasawa: The Forgotten Father of Matroid Theory*, edited by Nishimura, H. and Kuroda, S., see Appendix A: Further Reading.

idea of linear independence of vectors. It will not appear at first to be much related to optimisation. However, it is this route to matroid theory which introduces some basic linear algebra and motivates many of the terms and themes we will use later on.

1.2.1 Linear algebra

We shall be much concerned with \mathbb{R}^n , the n -dimensional vector space of real vectors. For now, we can safely regard this as just the collection of n -tuples of the form (x_1, x_2, \dots, x_n) . Thus $(105, 70, -3.5)$ is a triple in \mathbb{R}^3 , as might represent, say, distance, speed and acceleration; $(1.10, 2.72, 3.14, 0.00, 0.00)$ is a 5-tuple in \mathbb{R}^5 ; as a special case, (1.14) is a 1-tuple in $\mathbb{R}^1 = \mathbb{R}$ which would normally be written without the parentheses $()$, as 1.14.

Exercise 2 Is \mathbb{R}^3 a subset of \mathbb{R}^5 ? If not why not? Is \mathbb{R} a subset of \mathbb{R}^5 ?

To say that \mathbb{R}^n is a *vector space* means, among other things,

1. we have the *zero vector* $(0, 0, \dots, 0)$, which we often just represent by 0;
2. we can add vectors together componentwise, for example,
 $(1, 2, 3) + (0, -3, 1) = (1 + 0, 2 - 3, 3 + 1) = (1, -1, 4)$;
3. we can multiply vectors by single numbers (referred to as *scalars*), for example,
 $-0.5 \times (1, -1, 4) = (-0.5 \times 1, -0.5 \times -1, -0.5 \times 4) = (-0.5, 0.5, -2)$; and
4. any vector may be obtained by adding together scalar multiples of the n *unit vectors*, the i -th unit vector having a 1 in position i and zeros everywhere else.

Exercise 3 Calculate the following and say which vector space you are working in:

- (a) $(1, -2) + (0, 3)$;
- (b) $-2 \times (1, 1, 1, 1, 1, 1) - 2 \times (1, -1, 0, -1, 1, 0)$;
- (c) $3.142 - 2.718$.
- (d) 3.142×2.718 .

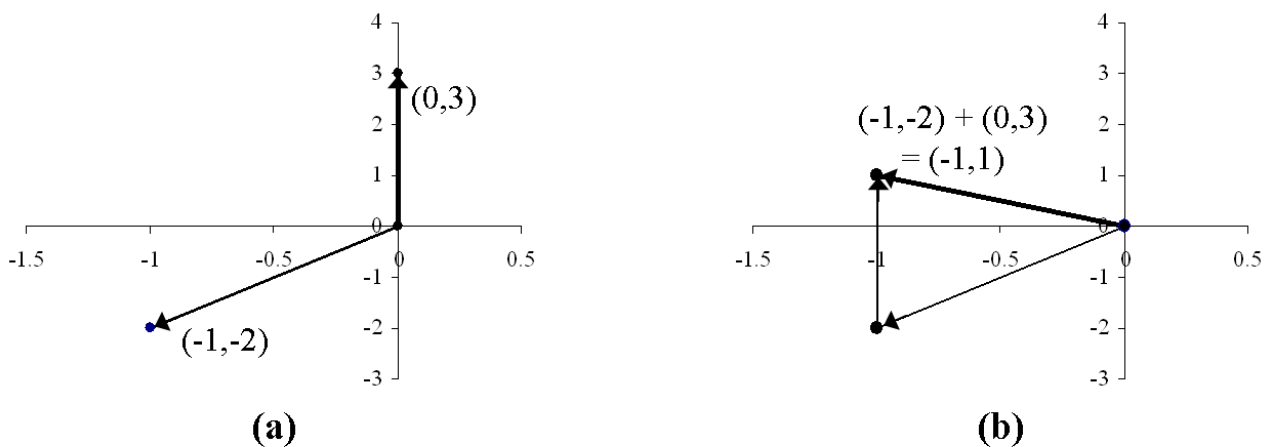


Figure 1.1: vector addition in two dimensions

It does no harm, in two dimensions, to think of a vector (x, y) as being an arrow pointing to the point (x, y) from the origin $(0, 0)$ as illustrated in Fig. 1.1(a). Then the scalar multiplication of (x, y) by λ just scales the length of the arrow by λ . Adding a vector (x', y') can be depicted as 'sliding' this vector along the line of the first vector, from the origin, while maintaining its angle and direction, and then 'completing the triangle', as shown in Fig. 1.1(b).

We will say that a vector v is a *linear combination* of vectors u_1, \dots, u_k if v can be expressed as a weighted sum of the u_i : $v = a_1 u_1 + \dots + a_k u_k$, the a_i being real numbers. This idea motivates a central concept in vector space theory, that of *linear independence*:

Definition 4 A collection of vectors v_1, v_2, \dots, v_k is called *linearly dependent* if there are scalars a_1, a_2, \dots, a_k , not all zero, such that $a_1 v_1 + a_2 v_2 + \dots + a_k v_k = 0$ the zero vector. If no such scalars can be found then the vectors are called *linearly independent*.

Note that the zero vector on its own forms a linearly dependent set.

Exercise 5 Which of the following are linear combinations of $(1, 1, 1)$ and $(-1, 0, 0)$?

- (a) $22/7 \times (1, 1, 1) - \pi \times (-1, 0, 0)$;
- (b) $0 \times (1, 1, 1) + 0 \times (-1, 0, 0)$;
- (c) $4 \times (1, 1, 1)$;
- (d) $4 \times (1, 1, 1) + 5 \times (-1, 0, 0)$;
- (e) $x^2 \times (1, 1, 1) - y^2 \times (-1, 0, 0)$, for $x, y \in \mathbb{R}$;
- (f) $x \times (1, 1, 1)^2 + y \times (-1, 0, 0)^2$, for $x, y \in \mathbb{R}$;
- (g) $\sqrt{x} \times (1, 1, 1) - \sqrt{y} \times (-1, 0, 0)$, for $x, y \in \mathbb{R}$;

Example 6 Are the vectors $(-1, -2)$ and $(0, 3)$ linearly dependent in \mathbb{R}^2 ?

Solution If v_1 and v_2 are any two vectors then they are linearly dependent if and only if one is a scalar multiple of the other, that is, they are *collinear*. This is not the case for $(-1, -2)$ and $(0, 3)$, as is clear from Fig. 1.1(a) (the arrows point in different directions). To give a formal proof based on definition 4: if $(-1, -2)$ and $(0, 3)$ were linearly dependent then we could find scalars a_1 and a_2 , with $a_1(-1, -2) + a_2(0, 3) = (0, 0)$, with at least one of a_1 and a_2 non-zero. Assume that $a_2 \neq 0$. Then, rearranging,

$$(0, 3) = \lambda(1, 2), \text{ where } \lambda = \frac{a_1}{a_2}.$$

Comparing the two sides of this equation componentwise, we require

$$0 = \lambda \times 1, \quad 3 = \lambda \times 2,$$

which are incompatible. So no such a_1, a_2 can exist.

Exercise 7 Show that assuming that $a_1 \neq 0$ in Example 6 also leads to an incompatibility and confirms that the vectors are linearly independent.

Example 8 Are the three vectors $(1, -1)$ and $(0, 3)$ and $(2, -4)$ linearly dependent?

Solution This time, we seek a_1, a_2 and a_3 so that

$$a_1(1, -1) + a_2(0, 3) + a_3(2, -4) = 0.$$

If we arrange our three vectors in a 3×2 array or *matrix*, then we can answer the question schematically:

$$\begin{matrix} R1 \\ R2 \\ R3 \end{matrix} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -4 \end{pmatrix} \xrightarrow{R3-2 \times R1} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 0 & -2 \end{pmatrix} \xrightarrow{R3+\frac{2}{3} \times R2} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 0 & 0 \end{pmatrix}.$$

Here, the calculations over the \rightarrow symbols are so-called *elementary row operations*: we add to any row (vector) some linear combination of all the other rows (vectors). So $R3 - 2 \times R1$ means “add to Row 3 the combination given by $-2 \times$ Row 1 plus $0 \times$ Row 2”. If elementary row operations can produce a zero vector then we have proved linear dependence. In fact, these elementary row operations provide the values

of a_1, a_2 and a_3 that we require: we multiplied Row 1 by -2 and Row 2 by $2/3$ and added the results to $1 \times$ Row 3:

$$-2 \times (1, -1) + \frac{2}{3}(0, 3) + 1 \times (2, -4) = (0, 0).$$

Exercise 9 Depict the linear combination in the solution of Example 8 graphically, as in Fig. 1.1(b).

In Example 8 we used a procedure for showing linear dependence among vectors: we applied elementary row operations to set every entry below the first in column 1 to zero, and then to set every entry below the second in column 2 to zero. We did this by subtracting off multiples of the i -th row under the assumption that its own i -th entry was non-zero. This non-zero i -th entry in row i is often referred to as a *pivot element*. We can always reorder our rows — they are just vectors, which do not come in any particular order — so unless some column is entirely composed of zeros we can always arrange for the i -th entry in the corresponding i -th column to be non-zero. For example, in

$$\begin{array}{l} \text{Row 2} \\ \text{Row 3} \end{array} \left(\begin{array}{ccc} a_{11} & a_{12} & \cdots \\ 0 & 0 & \cdots \\ 0 & a_{32} & \cdots \\ 0 & 0 & \cdots \\ 0 & a_{52} & \cdots \\ 0 & a_{62} & \cdots \end{array} \right),$$

assuming that entry a_{32} is non-zero, we can interchange Row 2 and Row 3 to give Row 2 a non-zero pivot element. Now subtract suitable multiples of the new Row 2 from all rows below it: the first column will remain zero, the second column will now be zero from the 3rd row downwards. These ideas lead us to a very important definition and theorem:

Definition 10 Any matrix whose elements satisfy the following two rules:

1. in each row, every element below the leftmost non-zero element is zero; and
2. every zero row has only zero rows below it

is said to be in row-echelon form.

Theorem 11 (Gaussian Elimination) Any matrix may be reduced to row-echelon form by elementary row operations, that is by repeatedly adding to any row some linear combination of all other rows.

Exercise 12 Reduce the matrix

$$\left(\begin{array}{ccc} 2 & 3 & 5 \\ 7 & 11 & 13 \\ 17 & 19 & 23 \end{array} \right)$$

to row-echelon form.

Example 8 illustrated two important applications of Theorem 11:

Corollary 13 In \mathbb{R}^n , no linearly independent set of vectors can have cardinality greater than n .

Corollary 14 The number of linearly independent row vectors in a matrix X is equal to the number of non-zero rows when X is reduced to row echelon form. This number is called the row rank of X . The transpose of X , denoted X^T , is the matrix whose rows are the columns of X . The column rank of X is the row rank of X^T . For any matrix the row rank and column rank have the same value:

$$\text{row rank } X = \text{column rank } X = \text{row rank } X^T,$$

and this number is called simply the rank of the matrix X .

Thus far, we have just been accumulating background information on linear independence. It is now time to see how this lead Whitney to the notion of a matroid.

1.2.2 Matroids defined by axioms

Let A be a finite subset of the vectors of \mathbb{R}^n . Let \mathcal{I} be the collection whose members are all those subsets of A which consist of linearly independent vectors. Then the pair $M = (A, \mathcal{I})$ is an example of what we shall call a *matroid*. A is called the *ground set* of M . The members of \mathcal{I} are called, naturally enough, the *independent sets* of M . A maximal member of \mathcal{I} , that is, a set $S \in \mathcal{I}$ such that $S \cup \{v\}$ is not in \mathcal{I} for any v in A , is called a *basis* of M .

Example 15 Let $A \subset \mathbb{R}^4$ be the set of vectors $\{(1, 2, 3, 4), (1, -1, 1, -1), (-2, 2, -2, 2), (6, 0, 10, 4)\}$. Find the collection \mathcal{I} for which (A, \mathcal{I}) is a matroid.

Solution For convenience, we will refer to the four vectors as v_1, v_2, v_3 , and v_4 , respectively. We observe at once that v_2 and v_3 are collinear (a line in four dimensions!) so they can certainly not belong together in an independent set in \mathcal{I} . So $\{v_1, v_2, v_3\}$, for instance, cannot be in \mathcal{I} . We now apply elementary row operations to the set of vectors:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & -1 & 1 & -1 \\ -2 & 2 & -2 & 2 \\ 6 & 0 & 10 & 4 \end{pmatrix} \xrightarrow[R2-R1]{R3+2 \times R2, R4-6 \times R2} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -3 & -2 & -5 \\ 0 & 0 & 0 & 0 \\ 0 & 6 & 4 & 10 \end{pmatrix}.$$

Except for the first column we have not achieved row-echelon form: there is a zero row above some non-zero entries; even if we interchanged rows three and four there would still be a non-zero entry below the 2nd (pivot) entry in row 2. Nevertheless we have done enough for our analysis. Firstly, notice that Row 2 and Row 4 have become collinear. Does this mean that v_2 and v_4 cannot belong together in a set of \mathcal{I} ? No! Because our elementary row operations used Row 1 to produce the new version of Row 2. It does mean, however, that v_2, v_4 and v_1 cannot appear together. Since v_2 and v_3 must also avoid each other, this means that $\{v_1, v_2\}$ is a basis. In fact, we have eliminated all possible subsets of A except:

$$\{\emptyset, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_1, v_3, v_4\}\}.$$

Finally, we decide that the last of these, $\{v_1, v_3, v_4\}$, must also be removed: v_2 and v_3 are collinear, so if v_1, v_2 and v_4 are linearly dependent, so must v_1, v_3 and v_4 be. Thinking of it another way, we could have set Row 2 of our array to zero just as easily as Row 3.

In conclusion, the independent sets of our matroid are: all subsets of A of size at most two, except $\{v_2, v_3\}$.

Exercise 16 Apply elementary row operations to the array

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

to reduce two of the four rows to zero. What does this suggest about the matroid whose independent sets are linearly independent subsets of $A = \{(1, 2, 3, 4), (5, 6, 7, 8), (9, 10, 11, 12), (13, 14, 15, 16)\}$?

Example 15 and Exercise 16 give an indication of what is, for us, the key property of any matroid: *all its bases, that is its maximal independent sets, have the same size*. That is, for matroids,

maximal is the same as maximum.

Now we see the relevance to OR where our primary concern is to maximise things: in matroids, we just add things until we have reached maximality and cannot add anything more; then we know we have also maximised — we have added as much as it is possible to add.

Hassler Whitney's great insight was to realise that, while this key property can be proved for vectors in a vector space it could be asserted as a *defining* property for a collection of subsets of *any* set. This need not necessarily have been a useful way to generalise but we shall see that indeed it was.

That maximal = maximum in vector spaces is an old result of linear algebra. It has many equivalent formulations and we will use one which will be found to fit well with the idea of a matroid:

Theorem 17 (Steinitz Exchange Lemma) *Suppose that V is a set of vectors containing a linearly independent subset $S = \{v_1, \dots, v_s\}$. Suppose that $T = \{u_1, \dots, u_t\}$ is another linearly independent subset of V with $s < t$. Then there is some $u_i \in T$, $1 \leq i \leq t$, $u_i \notin S$, for which $S \cup \{u_i\}$ is linearly independent.*

Proof. We shall prove this for vector spaces over the field of real numbers \mathbb{R} , just to avoid having to think about other fields. Suppose that no such u_i existed. Then every $u_i \in T$ must be linearly dependent on the elements of S . But then we can write $T = \{a_{11}v_1 + \dots + a_{1s}v_s, a_{21}v_1 + \dots + a_{2s}v_s, \dots, a_{t1}v_1 + \dots + a_{ts}v_s\}$, with the $a_{ij} \in \mathbb{R}$. Now the a_{ij} may be taken to be the entries in t vectors: $(a_{11}, \dots, a_{1s}), \dots, (a_{t1}, \dots, a_{ts})$ but since $s < t$ these vectors cannot be linearly independent, by Corollary 13. It follows that the u_i cannot be linearly independent either (see Exercise 19), which is a contradiction.

Corollary 18 *Every basis in a vector space, that is, every maximal linearly independent set, has the same size.*

Exercise 19 Linearity of vector arithmetic is defined to mean that, for vectors v_1 and v_2 and scalars a and b ,

1. $a(v_1 + v_2) = av_1 + av_2$;
2. $a(bv_1) = (ab)v_1$.

Deduce that if $b_1(a_{11}, a_{12}) + b_2(a_{21}, a_{22}) + b_3(a_{31}, a_{32}) = 0$, for $a_{ij}, b_i \in \mathbb{R}$, $1 \leq i \leq 3$, $1 \leq j \leq 2$, with not all the b_i being zero, then the vectors $a_{11}v_1 + a_{12}v_2$, $a_{21}v_1 + a_{22}v_2$ and $a_{31}v_1 + a_{32}v_2$ are not linearly independent, for any choice of vectors v_1 and v_2 .

Exercise 20 Prove Corollary 18.

Whitney turned Theorem 17, together with two intuitive facts about linear independence into *axioms* for a mathematical structure:

Definition 21 A matroid is any collection of subsets \mathcal{I} , the independent sets, of a finite set A (the ground set) satisfying the following axioms:

1. the empty set \emptyset is independent;
2. the collection of independent sets is subset-closed: if X is an independent set and $Y \subseteq X$ then Y is also an independent set; and
3. (the exchange — or sometimes the ‘augmentation’ — axiom) if X and Y are independent with $|X| < |Y|$ then there is some element $y \in Y$ such that y is not in X and $X \cup \{y\}$ is independent.

(Note that (1) follows from (2) automatically but is usually given emphasis as a separate axiom).

Example 22 Suppose that $A = \{\text{red, blue, green, purple}\}$ and that each colour on its own constitutes an independent set of a matroid M defined on A . Suppose that M has eight independent sets in all. Use the matroid axioms to determine the structure of M .

Solution The empty set \emptyset must always be independent, so we actually have only three other independent sets to find. We can reason as follows:

1. Suppose some independent set contains three elements. It has three subsets of 2 elements which, by axiom (2) of definition 21, are also independent, giving a total of nine independent sets. This is too many, so no independent set has three elements and we must find three 2-element independent sets.
2. Now suppose two of these are disjoint; say, $\{\text{red, blue}\}$ and $\{\text{green, purple}\}$ are independent. By applying definition 21, axiom (3) to the pair $X = \{\text{green}\}$ and $Y = \{\text{red, blue}\}$ we see that either $\{\text{red, green}\}$ or $\{\text{blue, green}\}$ must also be independent. But if we now choose $X = \{\text{purple}\}$ the same argument makes one of $\{\text{red, purple}\}$ and $\{\text{blue, purple}\}$ independent. This gives four 2-element sets in all which is too many. So no pair of independent sets of size two can be disjoint.

3. Suppose, then, that two of the independent sets are {red, blue} and {red, green}. Apply definition 21, axiom (3) to the pair $X = \{\text{purple}\}$ and $Y = \{\text{red, blue}\}$: we must have {purple, red} independent, since {purple, blue} would be disjoint from {red, green}.

We conclude that the structure of the matroid is that its independent sets are \emptyset , all singleton sets and three 2-elements independent sets which intersect in a common element and whose union is A .

Notice that we did not completely determine the matroid in this example: there were four different possible matroids depending on which colour was found in all the size 2 independent sets. However, we had discovered the *structure* of the matroid in the sense that all four possibilities are identical except for a relabelling of the ground set. Two matroids whose independent sets become identical under a relabelling of the ground set are called *isomorphic*.

Exercise 23 List the eight independent sets of two different matroids which would fit the specification of Example 22 and specify the relabelling which shows they are isomorphic.

Exercise 24 Let $A = \{1, 2, 4, 6\}$ be the ground set of a matroid whose independent sets are precisely* the subsets of A whose elements are mutually coprime, that is have greatest common divisor 1. List the independent sets.

Corollary 18 of Theorem 17 now reappears as a fundamental theorem of matroid theory:

Theorem 25 If $M = (A, \mathcal{I})$ is a matroid then every maximal member of \mathcal{I} has the same size.

Given a subset X of the ground set A of a matroid M , by analogy with Corollary 14, the size of a maximal independent set contained in X is called the *rank* of X , written $\text{rank}(X)$. It is well-defined by Theorem 25. The rank of A itself is then the size of any basis for the matroid and is called the rank of the matroid, $\text{rank}(M)$.

Exercise 26 Define a matroid M whose ground set A is the set of columns of the matrix

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & -1 & 1 & -1 & 1 \end{pmatrix}$$

and whose independent sets are the linearly independent subsets of columns, including the empty set. What is $\text{rank}(M)$? What are the ranks of the subsets of A (specified by column numbers) $\{1, 2, 3\}$, $\{2, 3\}$ and $\{4, 5\}$ and which of these subsets is a basis for M ?

Note an important subtlety in the definition of the matroid in Exercise 26: the ground set A had six elements but two, columns 4 and 6, were identical as vectors. It is best to think always of A as a set of possibly identical but *differently labelled* objects. We shall meet this subtlety again almost at once in the next section.

1.3 Matroids and the greedy algorithm

We will appear to completely change direction now, looking at a classical optimisation problem: that of finding a minimum-weight spanning tree in a graph or network. It will be seen that, in fact, constructing optimal solutions which are of maximal size can be done so that all such solutions are guaranteed also to have *maximum* size. Problems for which this is possible are said to be solved ‘greedily’; in fact, you have met this idea before in the level 2 unit *Software Engineering, Algorithm Design and Analysis*, in connection with Huffman coding. We shall see that the optimal solutions found by the greedy approach can, for many problems, be identified with the minimum-weight bases of an associated matroid. Before we return to matroids by this route we again need some background theory.

*The word ‘precisely’ in mathematics usually means “these and no others” or “then and only then”, as in “the prime numbers less than 10 are precisely 2, 3, 5 and 7,” or “a prime number is even precisely if it is two.”

1.3.1 Graph theory

The unit *Software Engineering, Algorithm Design and Analysis* dealt with graph theory from the point of view of abstract data types and graphs were also covered in the level 1 unit *Mathematics for Computing*, so we will be rather informal and recall the main definitions using an example. Consider Figure 1.2.

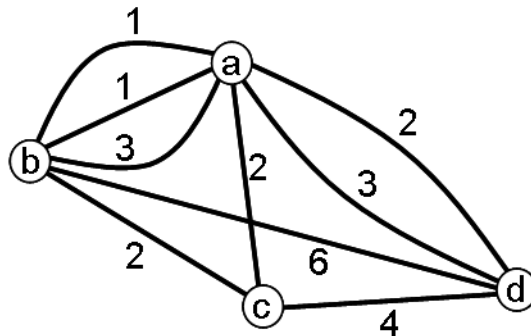


Figure 1.2: a graph on four vertices.

In the language of *graph theory* we refer to this object as a *graph*, consisting of four points called *vertices* and nine point-to-point connections called *edges*. Vertices thus connected are said to be *adjacent* to each other and to be *incident* with the edges connecting them. The letters on the vertices in Figure 1.2 are labels used for ease of reference; the numbers on the edges are *weights* over which we will want to optimise in some way. In general, we will denote the weight of edge e by $w(e)$, with w being some function from the edge set to the real numbers \mathbb{R} . We can refer to edges in terms of the vertices which they interconnect but we must be careful when we have *multiple edges*: in Figure 1.2, $ab = ba$ denotes a set of three edges joining a and b , so $w(ab)$ is not well-defined; $w(ac) = w(ca)$ is unambiguous and has value 2. (Remember the subtlety in Exercise 26 — here we have a set of edges which are all different but among which some appear identical when represented by their end vertices.)

It is essential to realise that only the edge weights and interconnections are of interest to us: the actual way in which they are drawn in Figure 1.2 is not important. So the following table is an equally valid and informative representation. A drawing is often a helpful visualisation, however.

b	c	d	
1,1,3	2	2,3	a
	2	6	b
		4	c

Graphs can be used to formulate and solve a great variety of problems, many of which are to do with connectivity. A *walk* in a graph is any sequence of edges u_1v_1, \dots, u_tv_t which are *consecutive* in the sense that $u_{i+1} = v_i$, for $i = 1, \dots, t-1$; this walk has length t and can be represented (modulo choice of multiple edges) by the vertex sequence $u_1, u_2, \dots, u_t, v_t$; if all the u_i are distinct, the walk is called a *path*. The graph is *connected* if there is a walk connecting any two vertices. The graph of Figure 1.2 is connected; in fact, every vertex can be reached from any other via a walk of length 1 — a single edge. Of course, any vertex v is connected to itself by the *trivial walk** having zero edges. A walk of any length from v to itself is said to be *closed*; if it repeats no edges and no vertices other than v it is a *cycle* (so a cycle is a closed path). A cycle of length 1, which is an edge joining a vertex to itself, is called a *self-loop*.

Example 27 How many closed walks of length 2 begin and end at vertex a in Figure 1.2? How many of these are cycles?

*The word ‘trivial’ in mathematics means ‘empty’ or ‘of size zero’. It may also mean ‘self-evident’ as in the anecdote of the mathematics professor who says to his class “The proof of what I have just written on the blackboard is trivial.” He then stares at what he has written with a puzzled frown. Then he begins pacing up and down, knitting his brows in thought. Finally after a quarter of an hour his face clears: “Yes!” he says, “it *is* trivial!” And he placidly continues his lecture.

Solution There are 14 closed walks of length 2 from a back to itself: going via b gives three choices of edge out and three choices back again, giving a total of nine walks; similarly going via d gives four walks and via c , one walk. In a cycle no edge may be repeated, so there is no cycle via vertex c and only $3 \times 2 + 2 \times 1 = 8$ via b and d .

We mentioned that we would be looking for something called a minimum-weight spanning tree. A *tree* is a graph which has no non-trivial cycles* and which is connected; if it fails to be connected then it must consist of a number of trees, and is accordingly referred to as a *forest*. Given a graph we may look for a collection of edges and vertices (a subgraph) which together form a tree; if all vertices are included then it is said to be *spanning*. We will be trying to minimise the weight of such a tree, the weight being the sum of its edge weights.

Let us summarise the main definitions:

Definition 28 A graph $G = (V, E)$ with vertex set V and edge set E is

1. said to be connected if any vertex in V may reach any other vertex along some walk, i.e., sequence of consecutive edges in E ;
2. called a forest if it does not contain any non-trivial cycles;
3. called a tree if it is a connected forest.

A collection H of weighted edges in a graph G , together with a subset V' of the vertex set V of G is

4. defined to have weight equal to the sum of the weights of the edges in H ;
5. called a subgraph of G if no edge of G appears more than once in H , and V' contains all vertices incident with edges in H ;
6. called a spanning subgraph of G if it is a subgraph and contains every vertex of G ;
7. called a spanning tree of G if it is a spanning subgraph which is a tree.

Exercise 29 In the graph of Figure 1.2 list all closed walks and cycles beginning and ending at vertex c which have length at most 3 and give their weights.

Exercise 30 In the graph of Figure 1.2 find all spanning forests of weight 4 which are not spanning trees.

1.3.2 Minimum-weight spanning trees

We can be precise now about what is meant by a minimum-weight spanning tree for a graph G : it is a collection of distinct edges of G which form a spanning tree and whose weights have the smallest possible sum over all such collections. We are going to define a matroid whose ground set is the edge set of G and whose independent sets are the forests in G . If we ‘greedily’ add the cheapest possible edges while building such independent sets then we will find that we always arrive at a basis — a spanning tree. Now instead of the axiomatic view of matroids we have an OR view: matroids are systems of subsets over a weighted set such that bases of minimum weight may be constructed greedily.

Before we specify the greedy algorithm and our matroid formally we will take some time to motivate the minimum-weight spanning tree problem from an OR standpoint, and we shall also digress to introduce another OR problem, maximum matching, which *cannot* be solved greedily.

Firstly, here are two problems which might lead an operations researcher to want to find a minimum-weight spanning tree:

Peer-to-peer Updating: a number of nodes in a network need to update each other regularly. A set of connections is to be nominated by which this may be done as cheaply as possible;

*The ‘trivial’ cycle, in line with the footnote on the previous page, is one which has zero edges—it just ‘stays where it is’.

Justified Product Elimination: a set of products is to be given a partial ranking via a number of comparisons, each involving a pair of products, as cheaply as possible, in order to justify the elimination of some ‘worst’ product.

These are rather vague problem specifications! They might be approached in countless ways and it is certainly one of the hardest parts of the operations researcher’s job to recast them as well-defined, mathematical problems whose solution will be both feasible and adequate for the needs of the client or end-user. In this case the mathematical reformulation will be:

MIN-WEIGHT SPANNING TREE

Input:	an edge-weighted graph the edge weights being in \mathbb{R}
Output:	a minimum-weight spanning tree

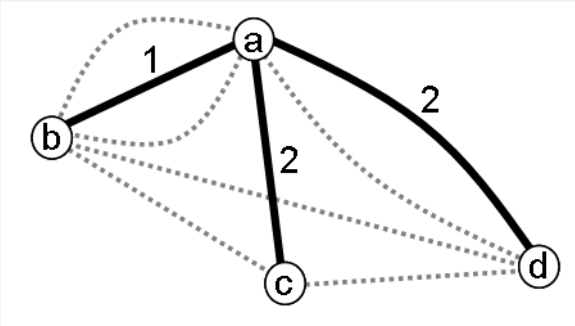
This grid will serve as a pattern for us in this study guide and for the most part it will be our starting point, since our focus is on mathematical techniques. In the present case, however, we will discuss in some detail how it comes about.

Let us take **Peer-to-peer Updating** first. Suppose the graph we met in Figure 1.2 in the previous subsection models the network, with the edge weights being the cost of individual connections (perhaps as offered by network service providers). We need a set B of edges along which any vertex may update any other, with the sum of the edge weights being as small as possible. So B must be a spanning, connected subgraph. Moreover, B should contain no cycles, for if $u_1u_2, u_2u_3, u_3u_4, \dots, u_tu_1$ were such a cycle then we could remove, say, u_2u_3 , breaking the cycle but maintaining the connection between u_2 and u_3 via the path $u_2u_1u_tu_{t-1} \dots u_4u_3$. Then $B \setminus \{u_2u_3\}$ is a cheaper spanning connected subgraph, contradicting the assertion that B had minimum weight.

Now any spanning tree in an edge-weighted graph is a candidate solution for peer-to-peer updating in that graph. Optimal solutions are precisely those spanning trees whose edge weights sum to the minimum possible.

Example 31 Solve the peer-to-peer updating problem for the graph of Figure 1.2.

Solution We proceed in the most natural way, by first choosing one of the ab edges of weight 1 — clearly it would be counterproductive to choose the weight 3 edge — and then adding in vertex c using a weight 2 edge, either choice looking equally suitable. We remark at this point that adding *both* edges ac and bc would create a cycle abc ; while this might increase the *reliability* of our updating network, our only concern is to minimise *cost*: any two edges of this cycle are sufficient to connect a, b and c , and we have already achieved this with a cost of $1 + 2 = 3$. Finally, we add vertex d to our subset of edges using edge ad , increasing the total cost to 5:



Exercise 32 There are four solutions to the peer-to-peer updating problem for Figure 1.2; find the other three.

Exercise 33 Find a minimum weight spanning tree in the graph shown in Figure 1.3.

Let us now turn to the other problem: **justified product elimination**. This time the problem does not directly involve a graph but we can construct one: we will use a vertex to represent each product and an edge joining vertices u and v will represent a comparison between products u and v , with the weight of the

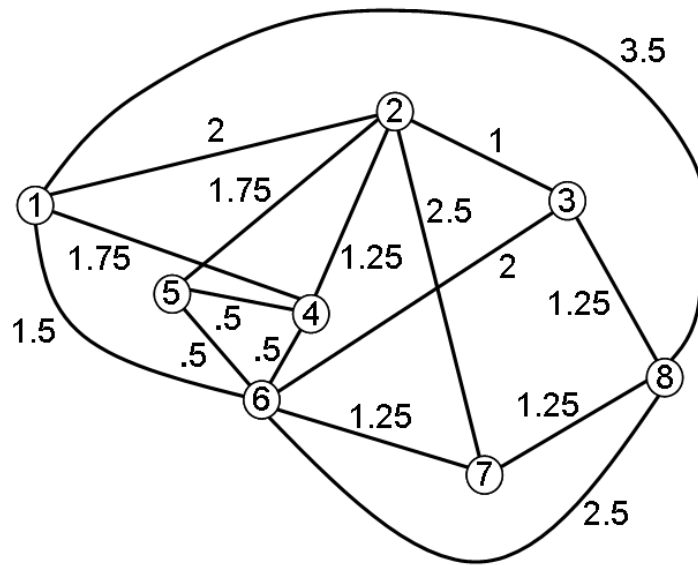


Figure 1.3: graph for Exercise 33.

edge being the cost of making the comparison (this might be the cost of a proposed customer survey, for example). We shall see that a minimum weight spanning tree is one way of ranking our products. Suppose Figure 1.2 now represents our comparison costings. We will compare a and b at cost 1; as before, a natural continuation is to compare c to either a or b at cost 2. Suppose we compare a to c . Figure 1.4 now shows two possible outcomes. Next to each subgraph is a diagram showing the ranking of a, b and c , so far as we

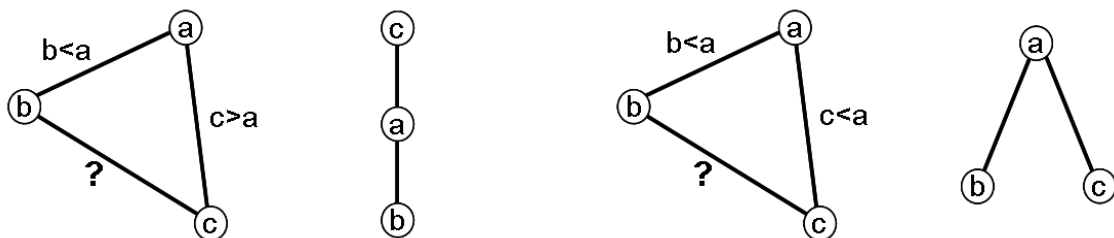


Figure 1.4: comparisons in the graph of Figure 1.2, with corresponding partial orders.

have discovered, the higher ranked products being placed above the lower ranked ones. The diagram on the left suggests that $b < c$ but we do not know this because we have not made the comparison. And we do not want to! We have been asked to rank the products, at minimal cost, to enable elimination. If we created a cycle by comparing b to c we would either (1) confirm that $b < c$ but at extra cost, or (2) get the result $c < b$, contradicting the previous ranking and undermining our justification for elimination of b . (Of course, this latter result suggests that further expenditure on product comparison might be important but you will rarely be thanked for giving your client this advice!)

The ranking on the right of Figure 1.4 indicates that we might eliminate either b or c (we have not justified choosing either one over the other). It is usually legitimate in management terms to toss a coin to make the final elimination of just one product.

Exercise 34 For each of your solutions to Exercise 32, add inequalities to each spanning tree edge (as in Figure 1.4) which (a) avoid the necessity of coin tossing, and (b) require coin tossing, in order to make a final choice of product to eliminate. Suggest another approach to solving the justified product elimination problem.

1.3.3 Minimum-cost matching

We have now gained a little experience in solving things ‘greedily’: we have been building our spanning trees guided only by the cost of the edges. We are going to describe the greedy algorithm more formally in subsection 1.3.4; before that it seems appropriate to introduce a problem which *cannot* be solved greedily and which will be of interest later.

In Oliver Stone’s 1987 film *Wall Street* the financier Gordon Gekko famously declares “Greed is right. Greed works.” For optimisation problems in general, the greedy approach is *not* sound. An alternative to the minimum spanning tree approach suggested in the solution to Exercise 34 was to compare products in pairs at minimum cost and then eliminate one of the losers at random. Let us try to apply this approach greedily to the graph of Figure 1.3. We might reasonably start by pairing vertices 4 and 6; the next cheapest pairing is then vertices 2 and 3 and then 7 with 8. This gives us the three edges shown in Figure 1.5(a).

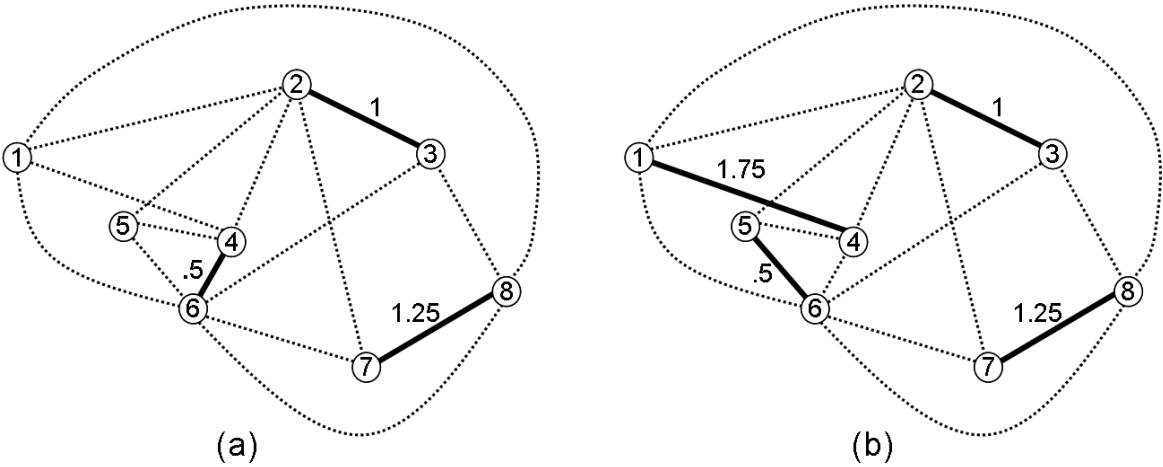


Figure 1.5: two matchings in the graph of Figure 1.3.

A collection of edges in a graph, such as the one highlighted in the figure, with no two edges sharing any vertex is called a *matching*. The matching we have constructed (greedily) is *maximal* because, even though vertices 1 and 5 are unpaired, no edge joins them (in our product elimination scenario, no comparison of these products has been proposed). But it is not *maximum*; indeed, in Figure 1.5(b) we see that, by choosing to pair vertices 5 and 6 first, instead of 4 and 6, we can successfully pair up all the vertices. Such a matching is said to be *perfect*.

We suggested earlier that the link between the greedy approach and matroids was even stronger: matroids are subset systems in which you can minimise cost while successfully constructing a maximum independent set. Although Figure 1.5(b) offers a perfect matching, it fails this minimisation criterion. Pairing vertices 4 and 5 first leads to a perfect matching of lower cost than Figure 1.5(b), as displayed in Figure 1.6.

So for the minimum-weight maximal matching problem the greedy approach fails completely: it fails to guarantee a maximal matching and if, by luck, one is found it fails to guarantee it will be minimum weight. Let us nevertheless formally identify this problem for later reference:

MIN-WEIGHT MAXIMAL MATCHING	
Input:	graph G with edge set E , function $w : E \rightarrow \mathbb{R}$
Output:	a maximal matching of G of minimum weight

We shall see in Chapter 3 that matroids can help with this problem to some extent: the idea of a ‘matroid intersection’ can at least find maximum matchings in a restricted case: that of *bipartite* graphs, those which have no odd-length cycles. So for bipartite graphs this will at least avoid the maximal \neq maximum failure of Figure 1.5(a), even though it will not avoid the non-minimum-cost failure of Figure 1.5(b). At

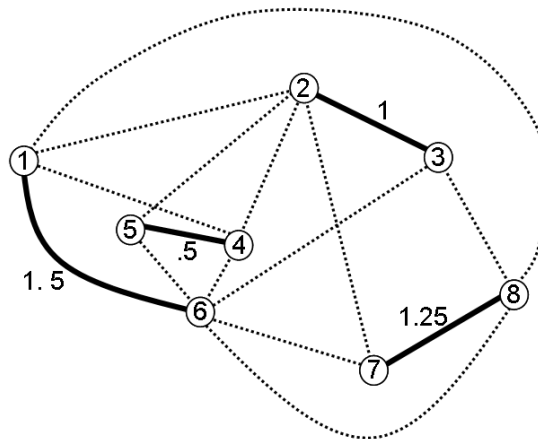


Figure 1.6: a minimum-cost perfect matching in the graph of Figure 1.3.

the end of Chapter 2 we shall even see that the greedy algorithm does ‘solve’ the general matching problem, but in a way which seems to highlight the limitations of the greedy approach!

Exercise 35 Show that the greedy approach will give a perfect matching in the graph of Figure 1.2 but will fail to give a minimum-cost perfect matching.

Before moving on, we will summarise, a little more formally, the new ideas from graph theory which we have just met.

Definition 36 In a graph G , the neighbourhood set of a vertex v is the subset of those vertices, excluding v , which are adjacent to v . The degree of v is the cardinality of its neighbourhood set + twice the number of self-loops at v . A subgraph of G in which every vertex has degree 1 consists of independent edges and is called a matching. A spanning matching is called perfect.

The following exercises use the graph of Figure 1.3 but without the edge weights. We reproduce the unweighted version in Figure 1.7 for convenience.

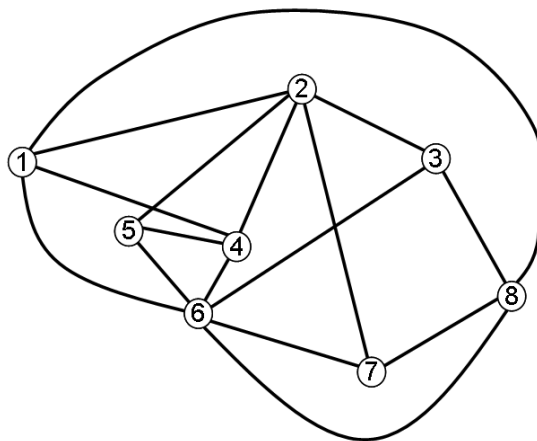


Figure 1.7: unweighted version of Figure 1.3.

Exercise 37 The **Hand-shaking Lemma** asserts that in any graph G there is an even number of vertices of odd degree. Confirm that this holds for the graph of Figure 1.7. If an additional edge is added between vertices 3 and 7, what happens to the number of vertices of odd degree?

Exercise 38 Euler's Theorem* asserts that in any graph G there is a walk which traverses every edge exactly once (an Euler trail) if and only there are at most two vertices of odd degree. Convince yourself that no such walk exists in the graph of Figure 1.7 but confirm that such a walk can be found if an additional edge is added between vertices 3 and 7. (Hint: begin your walk at a vertex of odd degree).

Exercise 39 (Harder) Suppose a graph G has n vertices and its vertex degrees, in ascending order, are d_1, d_2, \dots, d_n . **Chvátal's Theorem** asserts that if, for each $i < n/2$, either $d_i > i$ or $d_{n-i} \geq n - i$, then G has a Hamilton cycle: a cycle which passes each vertex exactly once. Show that:

1. the degrees of the graph of Figure 1.7 fail to satisfy this condition;
2. the condition becomes satisfied if an additional edge is added between vertices 3 and 7;
3. the condition is sufficient but not necessary since, in fact, a Hamilton cycle can be found in the graph of Figure 1.7; and hence
4. the graph has two edge-disjoint perfect matchings (i.e. two disjoint sets of edges each of which is a perfect matching).

Exercise 40 (Tricky!) Show that the graph of Figure 1.7 does not have three edge-disjoint matchings.

1.3.4 The greedy algorithm

We have just seen several problems whose solutions seem unavoidably to require *backtracking*. We get so far towards constructing an Euler trail or a Hamilton cycle or a perfect matching and then we are blocked. For instance, Figure 1.8 shows a well-known little 'brain-teaser': draw the envelope without taking your pen off the paper and without drawing any part twice. Having gained experience with Euler's Theorem

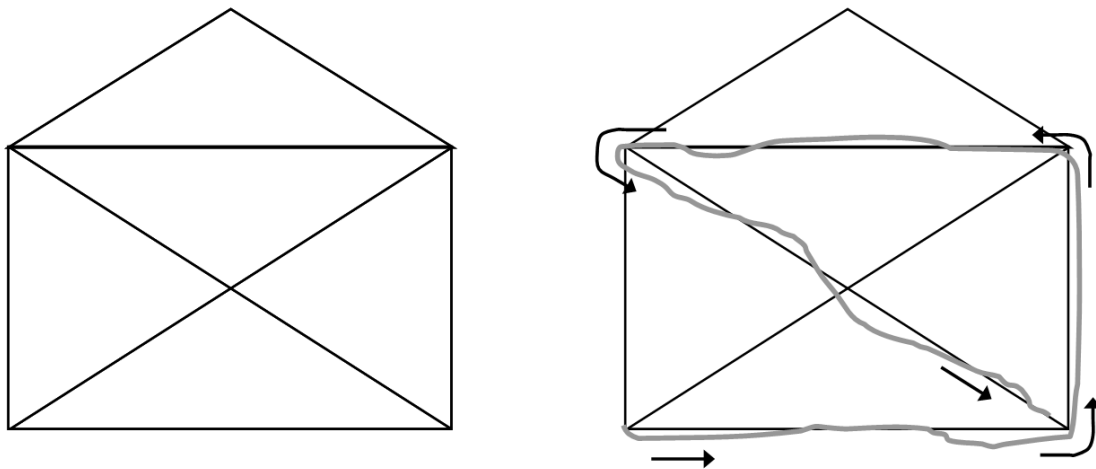


Figure 1.8: find an Euler trail in this 'graph'.

(Exercise 38) we know to begin and end our drawing at the two 'vertices' of degree 3 (the bottom corners of the envelope). But we still cannot draw 'greedily': the route taken on the right of Figure 1.8 fails to complete the drawing since we are blocked at the bottom right-hand corner; we must backtrack and choose a different direction from the top left-hand corner.

Constructions which can be represented in terms of the independent sets of some matroid, on the other hand, can always proceed without backtracking. What is more, we can optimise as we go. Let us make this formal. Here is the general problem:

*Named after the Swiss mathematician Leonhard Euler (1707–1783); his name is pronounced 'Oil-er'.

MIN-WEIGHT MAXIMAL SUBSET

Input:	finite set A , function $w : A \rightarrow \mathbb{R}$, collection of subsets \mathcal{A} of A
Output:	a maximal subset belonging to \mathcal{A} of minimum weight

Here is the algorithm we wish to apply to it*:

Algorithm 41 (The Greedy Algorithm)

INPUT: instance of MIN-WEIGHT MAXIMAL SUBSET problem
OUTPUT: optimal solution for this problem instance

```

1   $X := A; Y := \emptyset;$ 
2  while some  $x \in X$  has  $Y \cup \{x\} \in \mathcal{A}$  do
3      choose such an  $x$  of least weight;
4       $X := X \setminus \{x\};$ 
5       $Y := Y \cup \{x\}$ 
6  od
7  return  $Y$ 

```

And here are the conditions under which it works:

Theorem 42 Suppose that the collection of subsets \mathcal{A} of A satisfies the first two matroid axioms of definition 21, i.e. \mathcal{A} is subset-closed: $X \in \mathcal{A}$ and $X' \subset X$ implies $X' \in \mathcal{A}$. Then Algorithm 41 correctly solves MIN-WEIGHT MAXIMAL SUBSET for all choices of weight function w if and only if \mathcal{A} is the collection of independent sets of a matroid, i.e. \mathcal{A} also satisfies the exchange axiom of definition 21.

Exercise 43 Let $A = \{a, b, c, d\}$ and let \mathcal{A} be a collection of subsets $\{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{c, d\}\}$. Find a weight function for A which causes Algorithm 41 to fail on \mathcal{A} . Deduce that \mathcal{A} is not a matroid and explain how the matroid axioms (definition 21) are violated by \mathcal{A} .

The Greedy Algorithm is usually formulated in terms of a seemingly simpler problem:

MAXIMUM-WEIGHT SUBSET

Input:	finite set A , function $w : A \rightarrow \mathbb{R}$, collection of subsets \mathcal{A} of A
Output:	a subset belonging to \mathcal{A} of maximum weight

This is actually equivalent to the MIN-WEIGHT MAXIMAL SUBSET. Maximising and minimising amount to the same thing since, if A is a set of weighted objects then the maximum-weight subset among a collection of subsets of A is the same as the minimum-weight subset when, for each x in A , $w(x)$ is replaced by $-w(x)$. And as long as we are dealing with matroids, maximal is the same as maximum size, so that a subset of maximum weight will also be maximal and vice versa.

The remainder of this subsection will be spent on defining two classes of matroids and showing how the greedy algorithm applies to them. Firstly, we revisit the familiar example of the spanning tree.

Definition 44 Let G be a graph with edge set E . Let \mathcal{A} be the collection containing precisely those subsets of E which contain no cycles. Then the pair (E, \mathcal{A}) forms a matroid, called the cycle matroid of G , denoted $M(G)$.

The name *cycle matroid* derives historically from the definition of this matroid in terms of *dependent sets*: those which contain cycles. We remark that, although a graph G will usually be defined in terms of its vertices and edges, the cycle matroid $M(G)$ makes no reference to vertices. The cycle matroid cannot, therefore, completely specify a graph and, indeed, two different graphs may have the same cycle matroid; see Figure 1.9. An important implication of this is that the cycle matroid is often less helpful with optimisation problems in which vertices play an essential role.

Exercise 45 Suppose a graph G has vertex set $V = \{x, y, z\}$ and edge set $E = \{xy, xz, yz\}$. Write down the collection \mathcal{I} of independent sets of the cycle matroid $M(G)$.

*Algorithms are presented in a pseudocode which it is hoped will be self-explanatory. In particular, notice that the scope of an **if** statement is delimited by a matching **fi** while **do** loops end with a matching **od** (the convention used in MAPLE, for example).

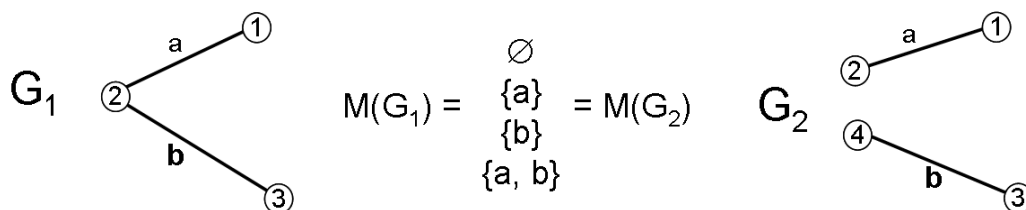


Figure 1.9: two different graphs having the same cycle matroid.

Example 46 Let $A = \{a, b, c, d, e\}$. Let a weight function w be defined by $w(a) = 3$, $w(b) = 2$, $w(c) = 1$, $w(d) = 4$, $w(e) = 5$. Apply the greedy algorithm to the collection \mathcal{A} which consists of all subsets of A except $\{a, b\}$ and $\{c, d, e\}$.

Solution Algorithm 41 proceeds as follows:

$X = \{a, b, c, d, e\}$, $Y = \emptyset$
 neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{c\} = \{c\}$ (so $\{c\}$ is independent)
 $X = \{a, b, d, e\}$, $Y = \{c\}$, total weight=1
 neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{b\}$
 $X = \{a, d, e\}$, $Y = \{b, c\}$, total weight=3
 neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{d\}$
 $X = \{a, e\}$, $Y = \{b, c, d\}$, total weight=7
 either $\{a, b\}$ or $\{c, d, e\}$ is contained in $Y \cup \{x\}$ for all $x \in \{a, e\}$

Output: $\{b, c, d\}$ with weight 7

Although Example 46 was presented purely in terms of a collection of subsets, it was a minimum-weight spanning tree problem in disguise:

Exercise 47 Show how the application of algorithm 41 in Example 46 corresponds to selecting a sequence of edges in the graph of Figure 1.10.

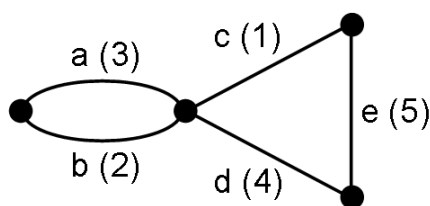


Figure 1.10: graph for Exercise 47 (edge weights in parenthesis).

Matroids, such as the one in Example 46, which are isomorphic to the cycle matroid of some graph are called *graphic*.

Exercise 48 In Exercise 24 in subsection 1.2.2 we defined a matroid on the ground set $A = \{1, 2, 4, 6\}$ by declaring any subset of coprime elements of A to be independent. Show that this matroid is graphic.

Exercise 49 Let $A = \{a, b, c\}$ be the ground set of a matroid M whose only independent sets are \emptyset and $\{a\}$. Find two different graphs whose cycle matroids are isomorphic to M . (Hint: use self-loops — edges joining a vertex to itself).

We shall now see that by choosing a different type of matroid the greedy algorithm can partially solve a very different optimisation problem. Let us do what we did in subsection 1.3.2 and motivate this using two management problems which might confront the OR professional:

Job Assignment: a number of jobs J_1, \dots, J_m need doing and we have a number of workers W_1, \dots, W_n each charging different rates and each offering to do any one of a subset of the jobs. Maximise the number of jobs done while minimising the cost;

Committee Selection: an organisation sells m products P_1, P_2, \dots, P_m . A certain committee needs to include a product representative for each. A pool of potential members, M_1, M_2, \dots, M_n , has been identified. We want to maximise the number of committee members who are familiar with each product.

We shall see that both problems can be *partially* solved in terms of searching for a cheapest *transversal* in a collection \mathcal{A} of subsets of a set A , that is, a subset T of A which has a distinct element from each member of \mathcal{A} . We shall define transversals more formally in a moment (definition 52). First here is an example:

Example 50 Suppose that there are three workers W_1, W_2 , and W_3 and two jobs J_1 and J_2 . Suppose W_1 offers to do either job while W_2 is only prepared to do J_2 and W_3 is only prepared to do job J_1 . Find an assignment of workers which gets both jobs done. If W_1 charges \$2 to do any job; W_2 charges \$3 and W_3 only \$1, what is the cheapest such assignment?

Solution We may list the ways each job can get done as subsets of the set $A = \{W_1, W_2, W_3\}$ of workers: $\mathcal{A} = \{\{W_1, W_3\}, \{W_1, W_2\}\}$. The first set specifies the workers who will do job J_1 ; the second set specifies workers who will do job J_2 . Now any transversal for \mathcal{A} will consist of two different workers, one who will do job J_1 and one who will do job J_2 . One such transversal is $\{W_1, W_2\}$; another is $\{W_1, W_3\}$ and a third is $\{W_2, W_3\}$. Since W_2 is expensive we choose the second of these three options. So far, this is an *incomplete* solution because we have yet to actually assign jobs to the workers in our chosen transversal $\{W_1, W_3\}$. In fact, there is only one option for us: since W_3 is only offering to do job J_1 , W_1 must take J_2 .

A transversal may also be referred to as a *system of distinct representatives*. This is a more descriptive name: we may imagine a collection of groups of people each nominating one of their members to represent them; a person may belong to more than one group but they may only accept one nomination, otherwise their loyalties will be divided.

Exercise 51 Suppose there are six workers, ranked in order of increasing cost: W_1, W_2, W_3, W_4, W_5 and W_6 . Suppose there are four jobs J_1, J_2, J_3 and J_4 and the workers offer themselves for these jobs as shown in the following table:

	W_1	W_2	W_3	W_4	W_5	W_6
Jobs	1,3	1,2	2	3	1,3,4	1,4

Construct the subset collection whose **four** members are the sets of workers who will offer to do each of the four jobs. Find the cheapest complete transversal of this collection and the most expensive.

Let us put our discussion on a more formal footing:

Definition 52 Let A be a finite, non-empty set and \mathcal{A} a collection, A_1, \dots, A_s , of non-empty subsets of A , with $s \geq 1$. A partial transversal for \mathcal{A} is a selection of t distinct elements of A from t distinct members of \mathcal{A} , where $t \leq s$. If $t = s$ then T is a complete transversal (or simply transversal) for \mathcal{A} . If no (partial) transversal of cardinality $t' > t$ contains T then T is called a maximal transversal. If the elements of A are weighted over \mathbb{R} then the weight of a (partial) transversal T is the sum of the weights of the elements of T .

It is very important to remember that the actual subsets of the collection \mathcal{A} in this definition may be identical even though they are distinct as members of the collection. This is the case in the following exercise:

Exercise 53 Suppose $A = \{a, b, c, d\}$ and \mathcal{A} is the collection of four (non-distinct) subsets: $A_1 = \{a, c\}$, $A_2 = \{a, b, d\}$, $A_3 = \{a, c\}$, $A_4 = \{c\}$. Find a maximal transversal for \mathcal{A} . If the elements of A are weighted such that $w(a) > w(b) > w(c) > w(d)$ find a minimum-weight maximal transversal.

We shall now find that the following problem:

MIN-WEIGHT MAXIMAL TRANSVERSAL

Input:	a collection of subsets of a weighted ground set, weights in \mathbb{R}
Output:	a minimum-weight maximal transversal for the subsets

is solved by the greedy algorithm.

Theorem 54 *Let A be a finite, non-empty set and let \mathcal{A} be a collection of non-empty subsets of A . Then A , together with the collection of all partial transversals of \mathcal{A} , forms a matroid, called the transversal matroid of \mathcal{A} and denoted $M(\mathcal{A})$.*

Since MIN-WEIGHT MAXIMAL TRANSVERSAL is a specific instance of the MIN-WEIGHT MAXIMAL MEMBER problem and Theorem 54 guarantees that it satisfies the conditions of Theorem 42 we have:

Corollary 55 *The greedy algorithm solves the MIN-WEIGHT MAXIMAL TRANSVERSAL problem.*

We said earlier that searching for a cheapest transversal could *partially* solve the Job Assignment Problem. Let us return to Exercise 51 to expand on this point, depicting the data from that exercise as a graph: Figure 1.11(a).

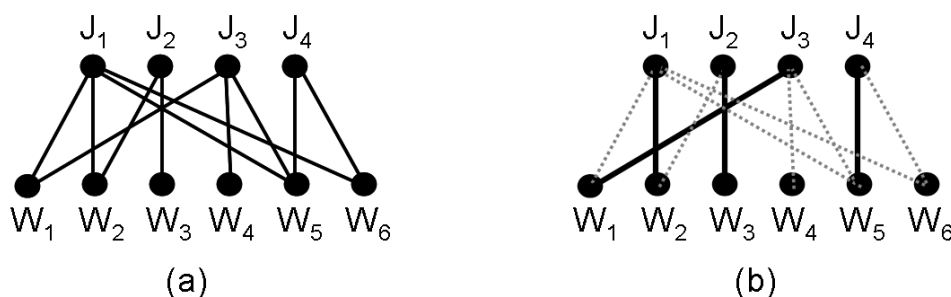


Figure 1.11: graphical representation of Exercise 51.

The collection of four subsets of $\{W_1, W_2, W_3, W_4, W_5, W_6\}$ for which transversals are assignments of workers to jobs is precisely the collection of neighbourhood sets of J_1, J_2, J_3 and J_4 : i.e. $\{W_1, W_2, W_5, W_6\}$, $\{W_2, W_3\}$, $\{W_1, W_4, W_5\}$ and $\{W_5, W_6\}$. The solution to Exercise 51 offered the set $\{W_1, W_2, W_3, W_5\}$ as a minimum-weight transversal and this (which happens to be the unique optimal solution) would be found by the greedy algorithm. However, this is *not* a solution to the Job Assignment problem because we have now to select the job which each worker will do: we have chosen the best workers but we do not know why! An actual assignment matching our transversal is shown in Figure 1.11(b) and harks back directly to subsection 1.3.3: it is a maximal matching. As far as our problem is concerned it is automatically minimum-weight, since our greedily constructed transversal achieves that part of the solution; but finding the edges in Figure 1.11(b) is *not* automatic and, as seen in subsection 1.3.3, it cannot be done greedily. Even if we know that only edges joining to W_1, W_2, W_3 and W_5 need be considered, a greedy approach may still fail: we will give J_1 to W_1 , then J_2 to W_2 ; then we are forced to give J_3 to W_5 , after which we are stuck!

In fact, the matching problem here is more easily solved than our example in subsection 1.3.3 because it involves a *bipartite graph*; we remarked at the time that in such graphs, precisely those having no odd-length cycles, maximal matchings may be found by ‘matroid intersection’. This will be our concern in the next chapter.

Exercise 56 *The Committee Selection problem: suppose an organization sells products P_1, P_2, P_3 and P_4 . Sales representatives Abdul, Cheung, Dominique and Enrique have experience with product P_1 ; Dominique, Enrique and Franz with P_2 ; Abdul and Baldish with P_3 ; and Cheung and Dominique with P_4 . Draw a bipartite graph representing this information. Form a committee of four so that each product is represented by as many people as possible with experience of that product.*

Chapter 2

Representations of matroids

2.1 Introduction

One of the ultimate aims of our matroid approach is to explain how certain problems in OR may be formulated in terms of pairs of matroids: we shall deal with this in the next chapter. Before that, there is an important issue in computer science to deal with: how do we implement our matroids for the purposes of algorithm design? There are two viewpoints which we could adopt, the first is a theoretical one, the second is more practical:

Oracle-based: we assume that a matroid M over the ground set A is defined by an oracle which answers Yes or No to the question: is set $X \subset A$ independent?

Representational: we assume that the ground set A of a matroid M may be encoded as a collection of vectors, so that $X \subset A$ is independent if and only if the corresponding vectors are linearly independent.

Let us make this more concrete by means of an example: at the end of the last chapter we saw that the Greedy Algorithm solved MIN-WEIGHT MAXIMAL TRANSVERSAL. We found a transversal $\{W_1, W_2, W_3, W_5\}$ for the subsets of workers who could do each of four different jobs: $\{W_1, W_2, W_5, W_6\}$ for job J_1 , $\{W_2, W_3\}$ for J_2 , $\{W_1, W_4, W_5\}$ for J_3 and $\{W_5, W_6\}$ for J_4 . We remarked that we knew that the transversal guaranteed an allocation of workers to jobs even though we did not know which worker would do which job. How was this possible? It was possible because the Greedy Algorithm contained the instruction

2 **while** some $x \in X$ has $Y \cup \{x\} \in \mathcal{A}$ **do**

and we deliberately avoided the issue, while we were busy with the basic definitions, of how this instruction was to be implemented. Thus, given that $\{W_1, W_2\}$ is a partial transversal, how do we test that $\{W_1, W_2\} \cup \{W_3\}$ again belongs to the set of partial transversals? The most obvious way is to make sure we can construct an assignment of these three workers to three jobs — we are actually solving the whole problem of which MIN-WEIGHT MAXIMAL TRANSVERSAL was a part!

Now the oracle approach to implementing matroids is satisfactory for mathematicians: they are asking questions about the class of problems which may *in theory* be solved by matroids. The computer scientist, on the other hand, wishes to know what problems can be solved by matroids *in practice*. The representational approach is much more satisfactory from this latter point of view and gives a sensible ‘separation of concerns’: we represent our workers by vectors and $\{W_1, W_2\} \cup \{W_3\}$ is accepted in step (2) of the Greedy Algorithm if the vector for W_3 is linearly independent from the vectors for W_1 and W_2 . We need not worry about what the W_i are or what we will do with our final collection of vectors because the Greedy Algorithm is now only concerned with checking linear independence.

It will be a considerable achievement, then, to confirm that we can find vector representations for many of the most useful matroids. This is what this chapter sets out to do. We shall appreciate the advantage of the representational approach even more in the next chapter when it will allow matroid intersection to be presented very elegantly; meanwhile, a practical convenience is that it motivates the introduction of some more ideas from linear algebra, particularly matrix algebra, which will be useful when we deal with linear programming and convexity theory later in this guide.

We will look at matrix algebra first after which we will be ready to discuss representations, firstly of partition matroids and then of matroids on graphs. This will prove an appropriate moment to introduce an idea which permeates all of OR — that of duality. The representation of transversal matroids is slightly more complicated and will be given a section to itself. Finally we will return to the oracle versus representation discussion by looking at another new matroid, the *matching matroid*. This matroid, as its

name suggests, can solve the problem of finding matchings in general graphs. Unfortunately, although it is representable in theory, constructing such representations appears to be harder than the matching problem itself: we appear to be forced to regard it as being defined by an oracle.

2.2 Matrix algebra

We used matrices extensively in the first part of Chapter 1 but really only as a way of organising collections of vectors. In the present chapter we shall require a little more: the ideas of matrix multiplication and the determinant function. You have met these in the level 1 unit *Mathematics for Computing* but it will do no harm to review the material here. We shall discuss the determinant function first, then multiplication; finally we shall describe an important combination of the two: the Binet-Cauchy Theorem.

2.2.1 Matrices and submatrices

A matrix X having m rows and n columns is said to have *size* $m \times n$ (said “ m by n ”). It consists of mn *entries*, with the notation x_{ij} to denote the entry in row i and column j being generally accepted.*

If an $m \times n$ matrix X has rows indexed by $\{1, \dots, m\}$ and columns by $\{1, \dots, n\}$ then a *submatrix* of X is obtained by taking those elements found in some subset of these rows and in some subset of these columns. More precisely, if $U \subset \{1, \dots, m\}$ and $V \subset \{1, \dots, n\}$ then we denote by $X[U|V]$ the $|U| \times |V|$ matrix with elements x_{ij} , $i \in U$, $j \in V$.

Example 57 If X is the 2×4 matrix:

$$X = \begin{pmatrix} 1 & 3 & -3 & -1 \\ 0 & 4 & -2 & 5 \end{pmatrix},$$

specify the matrices

$$(a) X[\{1, 2\} | \{1, 2\}] \quad (b) X[\{1, 2\} | \{1, 3, 4\}] \quad (c) X[\{2\} | \{1, 4\}].$$

Solution

1. $X[\{1, 2\} | \{1, 2\}] = \begin{pmatrix} 1 & 3 \\ 0 & 4 \end{pmatrix};$
2. $X[\{1, 2\} | \{1, 3, 4\}] = \begin{pmatrix} 1 & -3 & -1 \\ 0 & -2 & 5 \end{pmatrix};$
3. $X[\{2\} | \{1, 4\}] = (0, 5).$

2.2.2 The determinant

An $n \times n$ matrix is said to be *square of order* n . For square matrices we have:

Definition 58 Let X be a square matrix of order n which is in row echelon form. Then the determinant of X , denoted $\det X$, is defined to be the product of its diagonal elements: $\det X = x_{11} \times x_{22} \times \dots \times x_{nn}$. Let X' be any matrix which may be reduced to X by elementary row operations. Then the determinant of X' is given by $\det X' = \det X$.

*The notation is attributed to Leibniz by Charles Dodgson (the author of *Alice in Wonderland*) who regarded it as a “fatal objection” that space was wasted writing ‘ x ’. He tried to promote instead (*Elementary Treatise on Determinants*, 1867) the notation $i \setminus j$.

Example 59 Calculate the following determinants:

$$(a) \det \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} \quad (b) \det \begin{pmatrix} 4 & 2 & -3 \\ 3 & 1 & 0 \\ 1 & 1 & 2 \end{pmatrix} \quad (c) \det \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 5 & 3 & 4 \end{pmatrix}.$$

Solution None of the matrices is already in row echelon form, so they must take the role of X' in definition 58, and we will reduce them to row echelon form X :

$$\begin{aligned} (a) \quad \det X' &= \det \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} \xrightarrow[R3-R1-R2]{R2-3R1} \det \begin{pmatrix} 1 & 1 & 2 \\ 0 & -2 & -6 \\ 0 & 0 & -5 \end{pmatrix} = \det X = 1 \times -2 \times -5 = 10; \\ (b) \quad \det X' &= \det \begin{pmatrix} 4 & 2 & -3 \\ 3 & 1 & 0 \\ 1 & 1 & 2 \end{pmatrix} \xrightarrow[R3-R1+R2]{R2-\frac{3}{4}R1} \det \begin{pmatrix} 4 & 2 & -3 \\ 0 & -\frac{1}{2} & \frac{9}{4} \\ 0 & 0 & 5 \end{pmatrix} = \det X = 4 \times -\frac{1}{2} \times 5 = -10; \\ (c) \quad \det X' &= \det \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 5 & 3 & 4 \end{pmatrix} \xrightarrow[R3-2R1-R2]{R2-3R1} \det \begin{pmatrix} 1 & 1 & 2 \\ 0 & -2 & -6 \\ 0 & 0 & 0 \end{pmatrix} = \det X = 1 \times -2 \times 0 = 0. \end{aligned}$$

Example 59 illustrates two important properties of the determinant: firstly, if we interchange any two rows, as we did from (a) to (b), then the sign of the determinant is reversed. The second property, illustrated in (c), is important enough to single out, together with a few other basic properties, the first of which is implicit in our definition of the determinant:

Lemma 60 If X is a square matrix of order n then

- (a) the value of $\det X$ is unaffected by elementary row operations
- (b) X has rank n (i.e. has rows forming a linearly independent set of vectors) if and only if it has non-zero determinant;
- (c) $\det X^T = \det X$ (recall that X^T is the transpose of X , interchanging rows and columns);
- (d) $\det kX = k^n \det X$, where k is a scalar and the scalar product kX multiplies every entry of X by k ;
- (e) if the elements of one row of X are multiplied by k then the result has determinant kX .

When a square matrix of order n has non-zero determinant, as in (b), it is called *non-singular*. If the determinant is zero then the matrix is called *singular* or *rank-deficient*. So, for n column vectors of length n , taken as the columns of an $n \times n$ matrix, we have:

$$\begin{array}{ll} \text{linearly independent} & \longleftrightarrow \text{non-singular} \\ \text{linearly dependent} & \longleftrightarrow \text{singular} \end{array}$$

Non-singular matrices will soon be seen to play an important role in the representation of matroids.

You may have seen the determinant function defined in terms of the so-called Laplace expansion, in which it is calculated recursively by multiplying first-row elements by determinants of submatrices which omit the first row and corresponding column. Although this is computationally inefficient it is a simple way to evaluate determinants for 2×2 and 3×3 matrices, as illustrated in, respectively, Figure 2.1(a) and (b). The scheme in Figure 2.1(b), in which the first two columns of the 3×3 matrix are repeated, is known as *the Rule of Sarrus* after the French mathematician Pierre Frédéric Sarrus (1789–1861).

Exercise 61 Repeat the determinant calculations of Example 59 but using the Rule of Sarrus.

Closely allied to the Laplace expansion is a definition of the determinant in terms of permutations. There are many ways to represent permutations but it will be convenient for us to regard them as so-called

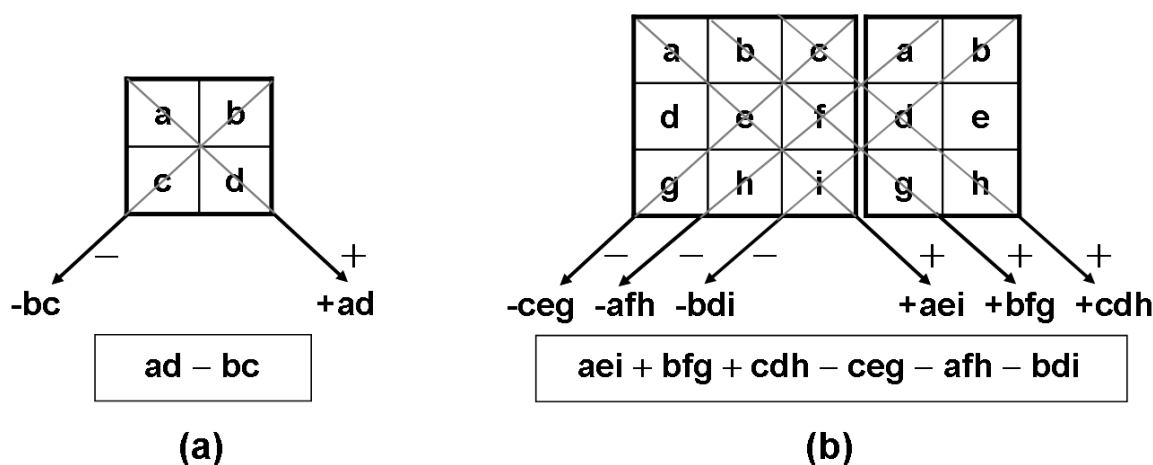


Figure 2.1: evaluation of (a) 2×2 and (b) 3×3 determinants.

permutation matrices, with a 1 in column j of row i precisely if the permutation replaces j with i , as in:

$$\begin{array}{ccc}
 \begin{array}{cc} 1 \rightarrow 2 & 1 \\ 2 \rightarrow 1 & 2 \end{array} \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, & \begin{array}{ccc} 1 \rightarrow 2 & 1 & 2 & 3 \\ 2 \rightarrow 3 & 2 & 0 & 1 & 0 \\ 3 \rightarrow 1 & 3 & 1 & 0 & 0 \end{array} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, & \begin{array}{ccccccc} 1 \rightarrow 7 & 1 \\ 2 \rightarrow 1 & 2 \\ 3 \rightarrow 2 & 3 \\ 4 \rightarrow 4 & 4 \\ 5 \rightarrow 3 & 5 \\ 6 \rightarrow 6 & 6 \\ 7 \rightarrow 5 & 7 \end{array} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & & & & & & 1 \\ 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \end{pmatrix}
 \end{array}$$

(the zeros are omitted in the third example for clarity).

In fact it will be useful to have this formalised with some definitions:

Definition 62 A permutation matrix is a square matrix in which every entry is 0 or 1 and in which every row and column has exactly one non-zero entry. If P is such a matrix of order n then it may be transformed to the identity matrix I_n which has 1's on the diagonal and zeros everywhere else by a sequence σ of interchanges of two rows. Then

1. the sign of P , denoted $\text{sgn}(P)$ is defined to be the value $+1$ if σ has an even number of interchanges and -1 otherwise, and this number is independent of the way σ is chosen;
2. if X is any $n \times n$ matrix then P corresponds to a subset of n entries of X (one for each non-zero entry in P). We denote by $\Pi(X, P)$ the product of these selected entries.

There are $n!$ different permutation matrices for a given positive integer n . If we sum over these we get:

Theorem 63 If X is an $n \times n$ matrix then

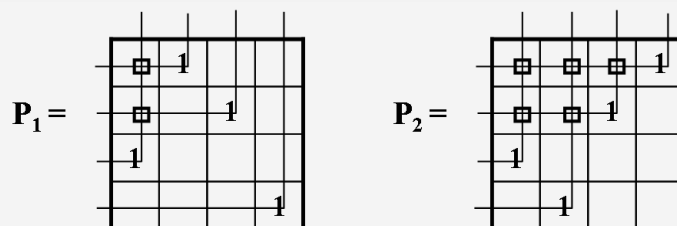
$$\det X = \sum_P \text{sgn}(P) \Pi(X, P).$$

In fact $\text{sgn}(P)$ for a permutation matrix P is just the value of $\det P$ but we prefer to avoid circularity by using the sgn function in Theorem 63. There is a convenient way to perform the calculation of $\text{sgn}(P)$ for small matrices: we draw a vertical line down each column until we reach the unique non-zero element and from this entry draw a horizontal line to the left-hand end of the corresponding row; now if the number of times these lines cross is even then $\text{sgn}(P) = 1$ and if it is odd then $\text{sgn}(P) = -1$.

Example 64 Evaluate the signs of the permutation matrices P_1 and P_2 below. Hence calculate the determinant of X :

$$P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad P_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ -6 & 0 & 0 & 0 \\ 0 & 3 & 0 & -1 \end{pmatrix}.$$

Solution Using the crossing lines trick we see that $\text{sgn}(P_1) = +1$ and $\text{sgn}(P_2) = -1$:



Moreover, we see that P_1 and P_2 are the only possible permutation matrices which can have all four non-zero entries matching non-zero entries of X . Since $\Pi(X, P) = 0$ for all other choices of P we get

$$\begin{aligned} \det X &= (+1) \times \Pi(X, P_1) + (-1) \times \Pi(X, P_2) \\ &= (+1) \times 4 \times 1 \times -6 \times -1 + (-1) \times 2 \times 1 \times -6 \times 3 = 60. \end{aligned}$$

Exercise 65 List the six permutation matrices of order 3 and evaluate their signs by counting crossing lines intersections. Hence use Theorem 63 to confirm that

$$\det \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

has the value given in Figure 2.1.

Every permutation in the summation of Theorem 63 includes a single entry in each row of matrix X . If we regard each entry value as a weighting for the permutations which use that entry then we can divide these weightings up between two matrices, a trick which is often useful:

Corollary 66 Suppose X is an $n \times n$ matrix whose i -th row is the vector $v = (x_{i1}, \dots, x_{in})$ and suppose that v is expressed as a sum of two vectors: $v = v_1 + v_2$. For $j = 1, 2$, let X_j denote X with first row replaced with v_j . Then

$$\det X = \det X_1 + \det X_2.$$

Exercise 67 By Corollary 66 we can write

$$\det \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \det \begin{pmatrix} 4 & 5 & 6 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \det \begin{pmatrix} -3 & -3 & -3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

Explain how this immediately tells us that the left-hand determinant is zero.

There is a special case of Corollary 66 which is especially useful when calculating determinants by hand. It uses the notation $X[U | V]$ which we introduced in section 2.2.1. Recall, for example,

$$\text{if } X = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} \text{ then } X[\{2, 3\} | \{2, 4\}] = \begin{pmatrix} f & h \\ j & l \end{pmatrix}.$$

Corollary 68 Suppose X is an $n \times n$ matrix whose i -th row has a single non-zero entry, say, $x_{ij} = \alpha$. Then

$$\det X = (-1)^{i+j} \alpha \det X[\{1, \dots, i-1, i+1, \dots, n\} | \{1, \dots, j-1, j+1, \dots, n\}].$$

Example 69 Calculate the determinant of

$$X = \begin{pmatrix} 3 & 1 & 7 & 0 \\ 0 & 1 & 5 & 3 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix}.$$

Solution Applying Corollary 68:

$$\begin{aligned} \det X &= (-1)^{4+4} \times -4 \times \det \begin{pmatrix} 3 & 1 & 7 \\ 0 & 1 & 5 \\ 0 & 2 & 0 \end{pmatrix} \\ &= -4 \times (-1)^{3+2} \times 2 \times \det \begin{pmatrix} 3 & 7 \\ 0 & 5 \end{pmatrix} = -4 \times -2 \times 3 \times 5 = 120. \end{aligned}$$

Exercise 70 Use Corollary 68 to calculate

$$\det \begin{pmatrix} 3 & 0 & 0 & 0 \\ 117 & 2 & 5 & 0 \\ -192 & -69 & 813 & 7 \\ -357 & 3 & 2 & 0 \end{pmatrix}.$$

The determinant plays a central role in any application of matrices: if a matrix X is viewed as a transformation of n -dimensional space then $\det X$ measures how much this transformation scales volume; in multi-dimensional differential calculus a certain determinant generalises the one-dimensional notion of ‘gradient’; other determinants yield polynomials whose solutions tell us about everything from the stability of control systems to the measurement of quantum states; we shall see shortly that the determinant can even count spanning trees in a graph!

2.2.3 Multiplication and the Binet-Cauchy Theorem

The other fundamental operations of matrix algebra are addition, multiplication and inversion*. Addition is only defined for matrices of the same size and then it behaves as one would expect — corresponding elements are added together to give the matrix sum. Multiplication, on the other hand, extends the idea of the linear combination, and this turns out to be tremendously useful; so useful, in fact, that we have already been doing it from the beginning of Chapter 1! Suppose v_1, v_2, \dots, v_m are m vectors in \mathbb{R}^n . As we saw in section 1.2.1 of Chapter 1, a linear combination of them is a weighted sum: $a_1 v_1 + a_2 v_2 + \dots + a_m v_m$, where the weights are scalars in \mathbb{R} , and the result is a new vector in \mathbb{R}^n . Suppose we collect the scalars a_i in another vector $u = (a_1, a_2, \dots, a_m)$ and collect the vectors in an $m \times n$ matrix X whose rows are the v_i . Then the weighted sum becomes *by definition* the multiplication of the vector u with the matrix X :

$$uX = u \times X = (a_1, a_2, \dots, a_m) \times \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} = a_1 v_1 + a_2 v_2 + \dots + a_m v_m.$$

*Matrix inversion will not be needed in this study guide and is omitted.

This makes it appear as if the multiplication of matrices by vectors is just a short-hand notation. It is perfectly correct to think in this way—but as with so much in mathematical notation it happens to suggest many useful and powerful ways to formulate and solve problems.*

Example 71 Calculate the following matrix products:

$$(a) \left(-2, \frac{2}{3}, 1\right) \times \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -4 \end{pmatrix} \quad (b) (-1, -1, 1) \times \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} \quad (c) \begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix}.$$

Solution (a) is just the matrix multiplication form of the linear combination in Example 8 (Chapter 1, section 1.2.1) which summed to the zero vector:

$$\left(-2, \frac{2}{3}, 1\right) \times \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -4 \end{pmatrix} = -2 \times (1, -1) + \frac{2}{3} \times (0, 3) + 1 \times (2, -4) = (0, 0);$$

(b) is the matrix multiplication form of the linear combination which gave the reduction of row 3 in Example 59(a) (i.e. $R_3 - R_1 - R_2$):

$$(-1, -1, 1) \times \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} = -1 \times (1, 1, 2) + -1 \times (3, 1, 0) + 1 \times (4, 2, -3) = (0, 0, -5);$$

(c) is the matrix multiplication form of the *three* linear combinations which gave the *three* row reductions in Example 59(a) (i.e. R_1 , $R_2 - 3R_1$, and $R_3 - R_1 - R_2$). For the three linear combinations we have three weight vectors arranged as the rows of a 3×3 matrix. We get three linear combination vectors, also forming the rows of a 3×3 matrix — the row echelon form:

$$\begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} = \begin{pmatrix} 1 \times (1, 1, 2) + 0 \times (3, 1, 0) + 0 \times (4, 2, -3) \\ -3 \times (1, 1, 2) + 1 \times (3, 1, 0) + 0 \times (4, 2, -3) \\ -1 \times (1, 1, 2) + -1 \times (3, 1, 0) + 1 \times (4, 2, -3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 \\ 0 & -2 & -6 \\ 0 & 0 & -5 \end{pmatrix}.$$

This illustrates the important fact that performing elementary row operations can always be represented as multiplying by a suitable matrix.

Exercise 72 Calculate the following matrix products:

$$(a) \begin{pmatrix} 1 & -1 & 0 \\ 3 & 2 & -4 \end{pmatrix} \times \begin{pmatrix} -2 \\ 2/3 \\ 1 \end{pmatrix} \quad (b) \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 2 \\ 0 & -2 & -6 \\ 0 & 0 & -5 \end{pmatrix} \quad (c) \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}.$$

To recap: a matrix product XY consists of weighted sums of the row vectors comprising Y , with each row of X giving a different weighted sum. It is clear that the length of each row of X must be identical to the number of rows of Y : if X has size $m \times n$ and Y has size $m' \times n'$ then $n = m'$. Then XY consists of m weighted sums, each of length n' , so XY has size $m \times n'$. Now, if it should happen that $m = n'$, then XY is a square matrix and we can ask what is the value of its determinant. This is the question answered by the Binet-Cauchy Theorem, discovered independently in 1812 by Jacques Binet (1786–1856) and Augustin Cauchy (1789–1857):

Theorem 73 (The Binet-Cauchy Theorem) Let X be an $m \times n$ matrix and Y an $n \times m$ matrix. If $U \subseteq \{1, \dots, n\}$ then let $X[-|U]$ denote the submatrix of X consisting of all rows and those columns indexed by U ; similarly let $Y[U|-]$ denote the submatrix of Y consisting of those rows indexed by U and all

*Another example is the way in which x^n , as a short-hand notation for $x \times \dots \times x$, leads a whole chain of ideas, culminating in Euler's miraculous identity $e^{\pi i} + 1 = 0$.

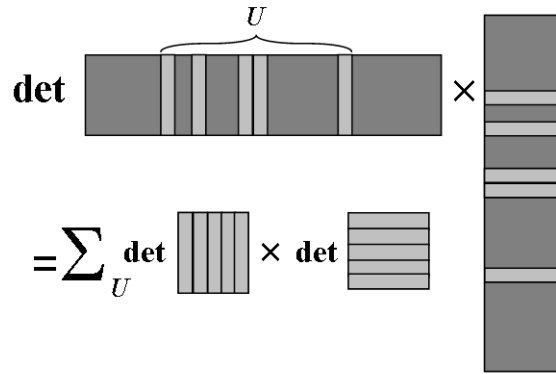


Figure 2.2: the Binet-Cauchy Theorem.

columns. Then

$$\det XY = \sum_{\substack{U \subseteq \{1, \dots, n\} \\ |U|=m}} \det X[-|U] \det Y[U|-].$$

The Binet-Cauchy Theorem is depicted in stylised form in Figure 2.2 and an explicit calculation for $m = 2$ and $n = 3$ would be:

$$\begin{aligned} \det \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix} \times \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{pmatrix} &= \sum_{i,j = \substack{1,2 \\ 1,3 \\ 2,3}} \det \begin{pmatrix} a_i & a_j \\ b_i & b_j \end{pmatrix} \det \begin{pmatrix} x_i & y_i \\ x_j & y_j \end{pmatrix} \\ &= \det \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \det \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix} + \det \begin{pmatrix} a_1 & a_3 \\ b_1 & b_3 \end{pmatrix} \det \begin{pmatrix} x_1 & y_1 \\ x_3 & y_3 \end{pmatrix} + \det \begin{pmatrix} a_2 & a_3 \\ b_2 & b_3 \end{pmatrix} \det \begin{pmatrix} x_2 & y_2 \\ x_3 & y_3 \end{pmatrix}. \end{aligned}$$

Corollary 74 If X and Y are two square matrices then

$$\det XY = \det X \det Y.$$

Example 75 Evaluate the following determinants directly and by applying the Binet-Cauchy Theorem:

$$\text{(a)} \det \left((1, 3, -1) \times \begin{pmatrix} 5 \\ 0 \\ -4 \end{pmatrix} \right) \quad \text{(b)} \det \begin{pmatrix} 2 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 6 & 3 \\ 5 & 2 \end{pmatrix}.$$

Solution (a) The product is a weighted sum of row vectors of length 1, or 1×1 matrices: $1.(5) + 3.(0) + -1.(-4) = (9)$ which is a single number — a 1×1 matrix whose determinant is just this same single number: $\det(9) = 9$. The Binet-Cauchy Theorem evaluates this as a sum of three products of determinants:

$$\det(1) \times \det(5) + \det(3) \times \det(0) + \det(-1) \times \det(-4) = 1 \times 5 + 3 \times 0 + -1 \times -4 = 9,$$

which is only different in notation from the original weighted sum.

(b) Performing the matrix product and applying the rule given in Figure 2.1(a):

$$\det \begin{pmatrix} 2 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 6 & 3 \\ 5 & 2 \end{pmatrix} = \det \begin{pmatrix} 45 & 19 \\ 116 & 49 \end{pmatrix} = 45 \times 49 - 19 \times 116 = 1.$$

The Binet-Cauchy Theorem equates this to a sum of three products of determinants corresponding to the three possible 2×2 submatrices that may be extracted from a 3×2 matrix (see the example just before Corollary 74):

$$\begin{aligned} & \det \begin{pmatrix} 2 & 3 \\ 7 & 9 \end{pmatrix} \times \det \begin{pmatrix} 1 & 0 \\ 6 & 3 \end{pmatrix} + \det \begin{pmatrix} 2 & 5 \\ 7 & 11 \end{pmatrix} \times \det \begin{pmatrix} 1 & 0 \\ 5 & 2 \end{pmatrix} + \det \begin{pmatrix} 3 & 5 \\ 9 & 11 \end{pmatrix} \times \det \begin{pmatrix} 6 & 3 \\ 5 & 2 \end{pmatrix} \\ &= (18 - 21) \times (3 - 0) + (22 - 35) \times (2 - 0) + (33 - 45) \times (12 - 15) = 1. \end{aligned}$$

Exercise 76 Confirm that the Binet-Cauchy Theorem holds for the matrix product XX^T , where

$$X = \begin{pmatrix} 0 & 1 & -1 & 0 \\ 2 & 0 & 0 & 2 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

The application of the Binet-Cauchy Theorem to a matrix and its transpose, as seen in Exercise 76, turns out to involve a sum of squares of determinants, so that it does not matter whether each determinant was positive or negative. This idea will reappear when we study representations of graphic matroids in the next section, as a key element in using determinants to enumerate spanning trees. Before we move to our consideration of matroid representations, though, we mention one more aspect of matrix multiplication, which will eventually reappear in Chapter 4.

Suppose A is a matrix of size $m \times n$ and suppose b is a column vector of size m . Then we can look for a vector x of size n such that the weighted sum of the columns of A , when the weights are given by x , gives exactly the vector b . This is shown schematically in Figure 2.3(a).

We began this section by pre-multiplying matrices by vectors, giving linear combinations of their rows. This is the right moment to remark that post-multiplying is just as valid but gives a linear combination of columns instead. Thus the weighted sum in Figure 2.3(a) is equivalent to the matrix multiplication shown in Figure 2.3(b). Another way of looking at this is to say that the equation $Ax = b$ represents m simultaneous equations in n variables, the variables being x_1, \dots, x_n .

Example 77 Represent the following simultaneous equations as a matrix equation:

$$\begin{aligned} p - q + 2r &= -2 \\ -2p + q - 7r &= 4 \end{aligned}$$

and hence find all solutions.

Solution The vector that will act as the column weights in Figure 2.3(a) is $x = (p, q, r)^T$; the multiples of each variable that appear in the two equations will form the corresponding column of the matrix A in Figure 2.3(b):

$$A = \begin{pmatrix} 1 & -1 & 2 \\ -2 & 1 & -7 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -2 \\ 4 \end{pmatrix}.$$

Meanwhile the vector b will just be the right-hand sides of the equations: $b = (-2, 4)^T$. We can solve the equations $Ax = b$ by elementary row operations which must be applied equally to A and b :

$$\begin{pmatrix} 1 & -1 & 2 \\ -2 & 1 & -7 \end{pmatrix} \xrightarrow{R2+2R1} \begin{pmatrix} 1 & -1 & 2 \\ 0 & -1 & -3 \end{pmatrix} \quad \begin{pmatrix} -2 \\ 4 \end{pmatrix} \xrightarrow{R2+2R1} \begin{pmatrix} -2 \\ 0 \end{pmatrix}.$$

Once we have reduced A to row-echelon form we translate back into equations and apply *back substitution*:

$$\begin{aligned} p - q + 2r &= -2 \\ -q - 3r &= 0 \end{aligned} \Rightarrow q = -3r \Rightarrow p = -2 - 2r - 3r = -2 - 5r.$$

So solutions are all vectors of the form $(-2 - 5r, -3r, r)$.

Exercise 78 A company manufactures bicycles at three different factories called P , Q and R . Factory P operates at a profit of \$10 per unit and Q at a profit of \$12 per unit but R operates at a loss of \$2 per unit. In the month up to the tax year end the company estimates that it will sell 1000 bicycles but its profits must not exceed \$9000 if it is to avoid tax penalties. By modelling this operation as a pair of simultaneous equations, give suggested production targets for each factory.

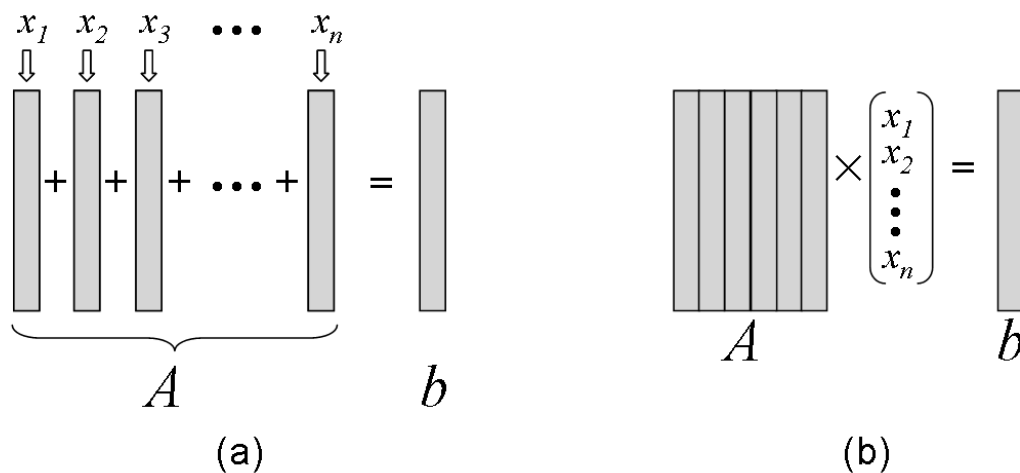


Figure 2.3: obtaining vector b via (a) a linear combination of columns of A ; and (b) by post-multiplying A by a column vector.

2.3 Representations

We now have covered all the linear algebra theory we shall need in the remainder of this study guide. The remainder of this chapter is devoted to depolying this theory to representing matroids using column vectors (arranged, as we shall see, into matrices). We shall outline the main idea first and, by way of an example, introduce a new type of matroid, the uniform matroid. Then we will progress to cover all the matroids which we have already seen applied to OR problems in Chapter 1.

2.3.1 How to represent a matroid

The key step in Algorithm 41, the Greedy Algorithm, is to ask if adding to an independent set some additional member of the ground set preserves independence. Although our examples so far have necessarily dealt with only small numbers of independent sets, it is easy to imagine cases where the answer to this question is not at all obvious. For instance, for a forest F in a graph with 1000 edges, the question, “does adding edge e to F form a cycle?” suggests at least some kind of forward search, perhaps setting flags on visited edges etc. A representation of the matroid turns this into a question about linear independence of vectors: we replace the 1000 edges with 1000 vectors, not necessarily all different. And we choose the vectors carefully so that:

1. F is represented by linearly independent vectors; and
2. e is represented by a vector which is linearly independent of the vectors representing F if and only if edge e does not form a cycle with forest F .

Why can we not insist that all our representing vectors be different? Because not every two-element subset needs to be independent in a matroid. If $\{a, b\}$ is not an independent set then the vectors we choose to represent a and b —let us call these vectors $\phi(a)$ and $\phi(b)$ —must be linearly dependent. Over \mathbb{R} this means that $\phi(a)$ must be a scalar multiple of $\phi(b)$; over the binary field $\{0, 1\}$ they must be *identical*. So-called *binary matroids* are not widely used in OR but are very important in other applications, for instance in designing error-correcting codes. So it is convenient to assume that we may be sometimes representing different matroid elements using the same vector: the map from ground set to vectors is not *injective*. We avoid confusion by making the vectors into the columns of a matrix: there is no difficulty in a matrix having as many identical columns as we like (recall Exercise 26 in Chapter 1, section 1.2.2).

The approach of replacing the ground set of a matroid with columns of a matrix naturally begs the question: can this always be done? The answer is no, there are some matroids which cannot be represented by matrices, even if we allow fields \mathbb{F} other than the real numbers. When it *is* possible, we say that the matroid is *representable*. We shall see that graphic and transversal matroids are representable and this is sufficiently generous for our needs.

Definition 79 Let M be a matroid with ground set A and independent sets \mathcal{I} and let \mathbb{F} be a field. M is said to be representable over the field \mathbb{F} if there is some function ϕ from A to \mathbb{F}^k , for some $k > 0$, such that $X \subset A$ is independent if and only if $\phi(X)$ is linearly independent over \mathbb{F} , where $\phi(X) = \{\phi(x) \mid x \in X\}$.

Notice that definition 79 did not refer to matrices. This is because the definition allows the function ϕ to be non-injective and this is enough to capture the idea of representation; as we said, using matrices is just a convenient way of recording multiple copies of the same representing vector.

To illustrate this idea we will introduce a new type of matroid. Suppose we have a set A and we define a matroid by saying that *all* subsets of A , up to and including a certain size, are independent. This is called a *uniform matroid*; if $|A| = n$ and all subsets of size at most k , and no others, are independent, then it is denoted by $U_{k,n}$. To represent such a matroid we need a collection of n vectors among which any k are linearly independent but any collection of $k + 1$ are linearly dependent.

Let us take the example of $U_{3,5}$: the ground set A has five elements and a subset of A is independent if and only if it has three or fewer elements.

Example 80 What are the independent sets of $U_{3,5}$ when the ground set is $A = \{a, b, c, d, e\}$?

Solution The maximal sets—that is, the bases—are all the subsets of size 3. There are $\binom{5}{3} = 10$ of these:

$$\begin{array}{ccccccc} \{a, b, c\} & & & & & & \\ \{a, b, d\} & \{a, c, d\} & & \{b, c, d\} & & & \\ \{a, b, e\} & \{a, c, e\} & \{a, d, e\} & \{b, c, e\} & \{b, d, e\} & \{c, d, e\} & \end{array} .$$

Any subset of any of these sets must automatically be independent—this gives a further $\binom{5}{2} + \binom{5}{1} + \binom{5}{0} = 10 + 5 + 1 = 16$ independent sets. So $U_{3,5}$ has 26 independent sets altogether.

Exercise 81 If a graph has four edges, a, b, c and d , what is the greatest number of spanning trees it can have? What are the bases of $U_{2,4}$, defined over the ground set $A = \{a, b, c, d\}$? Deduce that $U_{2,4}$ is not graphic (refer back to Example 46 and Exercise 47 in section 1.3.4 of Chapter 1 if you like).

How might we represent a uniform matroid as a collection of vectors? For $U_{3,5}$, for instance, we have a ground set of five elements, so this will be represented by five vectors. They will, in fact, all be different since all pairs must be independent (since any subset of size 3 or less is independent). Our challenge is to find five vectors such that:

1. any three vectors are linearly independent; but
2. no four vectors are linearly independent.

We can achieve (2) easily by using vectors of length 3: by Corollary 14, at the end of section 1.2.1 in Chapter 1, the vectors will form the columns of a matrix which can have row rank at most 3 and therefore column rank at most 3. The question is, can we make sure, not only that the column rank is actually 3, but that this holds for any subset of three columns? This is the question: is $U_{3,5}$ representable over \mathbb{R} ?

The answer is Yes: $U_{k,n}$ is representable over \mathbb{R} for all k and n . We shall see why at the end of this chapter with the help of transversal matroid representations. For now we shall just show it is true for $U_{3,5}$ by giving a representation: you can check that any set of three columns in the following matrix:

$$R(U_{3,5}) = \begin{array}{ccccc} & a & b & c & d & e \\ \begin{pmatrix} -3 & -2 & 1 & 0 & 0 \\ 4 & -3 & 0 & 1 & 0 \\ -1 & 2 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

are linearly independent. Notice that we have included column headings on the matrix to indicate that is is representing the matroid $U_{3,5}$ over $A = \{a, b, c, d, e\}$; these headings are part of the representation but not part of the matrix.

2.3.2 Partition matroids

For our representation of $U_{3,5}$ we used different column vectors for each element of the ground set. In fact, that has to be true for any uniform matroid $U_{k,n}$ when $k \geq 2$. We will now look at another new type of matroid, for which almost the reverse is true: very many of the vectors will usually be identical. The trick of arranging the vectors as columns of a matrix problem means we can refer to different columns in the matrix even if they are the same vector (recall Exercise 26 in Chapter 1, section 1.2.2). Here is a very simple example of these ideas in action:

Example 82 A graph G with vertex set V and n edges is said to be *graceful* if there is some injection γ from V to the set $\{1, \dots, n+1\}$ such that every edge, if labelled with the difference of its incident vertices, is labelled differently, which is to say that the set $\{|\gamma(u) - \gamma(v)| \mid u, v \in V, uv \text{ an edge of } G\}$ is equal to $\{1, \dots, n\}$. Such an injection is said to *gracefully label the graph*. Show that the set of gracefully labelled graphs on up to n edges can be thought of as a matroid and that this matroid is representable over \mathbb{R}^n .

Solution We will use the *complete graph* K_{n+1} , that is the graph on $n+1$ vertices any two of which are adjacent. We will label the vertices $1, \dots, n+1$ and the ground set of our matroid will be the set A of *labelled edges*, the label of edge ij being $|i - j|$. Now a subset X of A will be called independent if no two edges in X have the same label. Figure 2.4 illustrates this for $n = 4$: the vertex and edge-labelled K_5 is on the left; three maximal independent sets are shown on the right. Notice that this is clearly not the same as the cycle matroid of K_5 , since only one of the maximal independent sets shown (that is, the third one) is a spanning tree.

In fact, what we have constructed is a special type of transversal matroid, called a *partition matroid*. Consider the sets of edges all having the same label. Since no edge can belong to two such sets, and every edge must belong in one of them, this is a partition of the edges. Moreover, any partial transversal of these sets is a graceful graph.

Now, how might we represent this special transversal matroid by the columns of a matrix? Since members of the ground set are independent of each other precisely if they belong to different partition sets we need only map to as many different column vectors as there are different partition sets. For our example, with four different edge labels, the vectors can just be the *standard basis* of unit vectors in \mathbb{R}^4 : $(1, 0, 0, 0)^T$, $(0, 1, 0, 0)^T$, $(0, 0, 1, 0)^T$, $(0, 0, 0, 1)^T$ (recall from Chapter 1 that the i -th unit vector is 1 in position i and zero elsewhere). Since there are 10 edges in total, our matrix representation of this partition matroid could be:

$$\begin{array}{cccccccccc} & 12 & 23 & 34 & 45 & 13 & 24 & 35 & 14 & 25 & 15 \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

(as before, the column labels are included for convenience and are not part of the actual matrix).

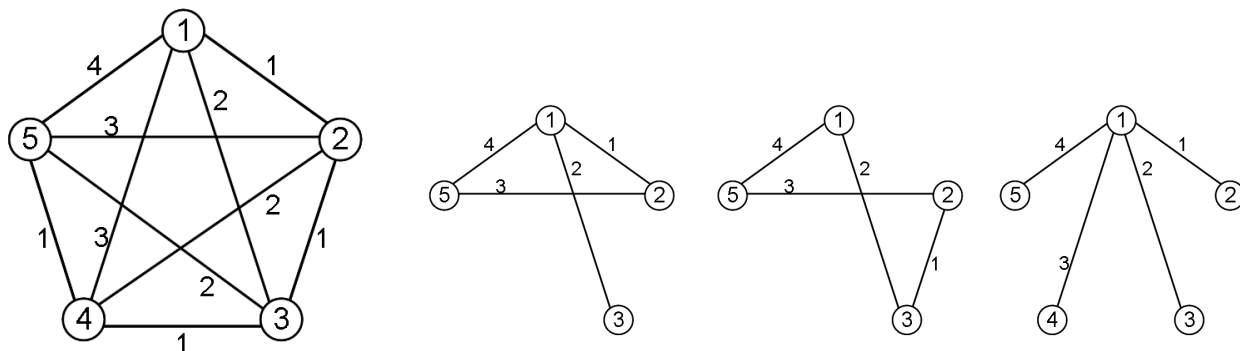


Figure 2.4: labelled K_5 whose edges are the ground set for the matroid in Example 82, and three maximal independent sets in this matroid.

Exercise 83 Write out the sets of column vectors corresponding to each of the maximal independent sets in Figure 2.4 and confirm that they are linearly independent.

We highlight the definition of this new matroid we have just represented:

Definition 84 Given a set A and a collection of subsets \mathcal{A} which partitions A , the partition matroid $M(\mathcal{A}(S))$ is the matroid whose independent sets are just those subsets of A having no two elements in the same member of \mathcal{A} .

We have seen in Example 82 that a partition matroid is just represented by an incidence matrix which has a 1 in row i and column j exactly when the i -th set contains element j of the ground set.

Before we move on to look at representing cycle matroids of graphs it is worth giving another example of a partition matroid because they occur so frequently in matroid applications.

Suppose we have a *bipartite graph*: we have briefly described these before as graphs with no odd-length cycles. An alternative description is to say that a bipartite graph is a graph G whose vertex set V can be partitioned into two sets S and T so that every edge of G joins some vertex of S to some vertex of T . (This is not the partition for our matroid but we take the opportunity to reiterate that, as in Example 82, a *set partition* is a separation into disjoint parts, with every set element appearing in exactly one part.) We take the edge set of G to be our ground set and let $\mathcal{A}(S)$ be the collection of those subsets of edges which are incident with the same vertex in S . Now $M(\mathcal{A}(S))$ is the *partition matroid* whose independent sets are exactly those subsets of E having no two edges in the same $\mathcal{A}(S)$ set. We may define $M(\mathcal{A}(T))$ similarly.

Example 85 For the bipartite graph shown in Figure 2.5 give a matrix representation for the matroid $M(\mathcal{A}(S))$. How many independent sets are there in the matroid?

Solution The partition sets of $\mathcal{A}(S)$ are $\{a, c\}$, $\{b\}$, $\{d, e\}$, $\{f\}$. The independent sets of the matroid $M(\mathcal{A}(S))$ are all partial transversals of this collection. Since the edge subsets in $\mathcal{A}(S)$ are disjoint, no pair of elements in the same subset can appear in a partial transversal, so as in the case of the graceful graph matroid (Example 82) we give the same vector to all edges in the same $\mathcal{A}(S)$ subset:

$$\begin{array}{c} a \quad b \quad c \quad d \quad e \quad f \\ \begin{array}{l} \{a, c\} \\ \{b\} \\ \{d, e\} \\ \{f\} \end{array} \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

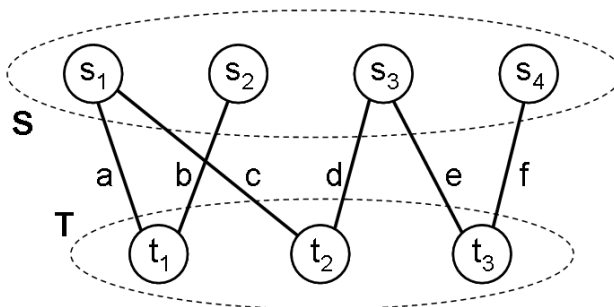


Figure 2.5: bipartite graph with vertex partition $S \cup T$ (see Example 85).

If we build a partial transversal we have 3 choices for the first $\mathcal{A}(S)$ subset: choose a , choose c or choose neither. Reasoning similarly for the other subsets gives us a total of $3 \times 2 \times 3 \times 2 = 36$ independent sets.

Exercise 86 Repeat Example 85 for the subset collection $\mathcal{A}(T)$ for Figure 2.5.

2.3.3 Graphic matroids

Examples 82 and 85 dealt with the representation of partition matroids; these are a special case of transversal matroids, whose representation in general involve considerably more work. We shall defer this case until we have looked at cycle matroids of graphs. In both cases the representations are based on an ‘incidence matrix’ although in the case of transversal matroids the representation is not a direct one.

Definition 87 Let G be a graph with edges e_1, \dots, e_n and vertices v_1, \dots, v_m . The incidence matrix of G , denoted $B(G)$, is the $m \times n$ matrix in which

1. there are m rows, corresponding to v_1, \dots, v_m ;
2. there are n columns, corresponding to e_1, \dots, e_n ;
3. the i -th column, if e_i is an edge incident with distinct vertices v_{i_1} and v_{i_2} , contains a 1 in position i_1 , a -1 in position i_2 and zeros everywhere else;
4. the i -th column, if e_i is a self-loop, is the zero vector.

Notice that in definition 87 we referred to *the* incidence matrix of a graph although, strictly speaking, the matrix depended on the ordering of the edges and vertices. We shall usually assume that our graphs come with a given labelling, so that this will cause no ambiguity. In the context of computer programming, for example, the graph might be given as an input file which automatically places an ordering on the vertices and edges. You may recall meeting the incidence matrix in this context in the unit *Software Engineering, Algorithm Design and Analysis*, although there, every non-zero entry was a +1 (this loses no information about the graph but is not suitable as a matroid representation, as we shall see in Exercise 94). In Chapter 4 we shall find ourselves in a situation where we attach significance to the + and - signs in the incidence matrix; for now they have no significance and we can choose which comes first in any column.

Exercise 88 Write out the incidence matrix for the complete graph K_5 , as given in Figure 2.4.

The following lemma shows that $B(G)$ is just what we need for matroid representation:

Lemma 89 If G is a graph and $B(G)$ its incidence matrix then a subset of the columns of $B(G)$ is linearly independent if and only if the corresponding edges of G contain no cycle.

Theorem 90 If G is a graph and $M(G)$ its cycle matroid then $M(G)$ is represented by the matrix $B(G)$.

Example 91 Of the three subsets of edges shown on the right in Figure 2.4, only the third is independent in the cycle matroid of K_5 . Confirm that the subset $\{12, 13, 25, 15\}$ is dependent by transposing the relevant submatrix of the incidence matrix in Exercise 88 and performing elementary row operations.

Solution Transposing the submatrix formed by columns 12, 13, 25 and 15 of the incidence matrix of K_5 and applying elementary row operations gives:

$$\begin{array}{l} 12 \\ 13 \\ 25 \\ 15 \end{array} \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix} \xrightarrow{R4-(R1+R3)} \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

showing that the column rank of the original submatrix is 3, so this is not a linearly independent set of columns. Indeed, the reduction of R_4 to zero using R_1 and R_3 shows that the corresponding three edges of the graph are dependent, i.e. 12, 25 and 15 form a cycle in K_5 , as can be confirmed in Figure 2.4.

Exercise 92 Repeat the analysis of Example 91 for the submatrix of the incidence matrix in Exercise 88 corresponding to edge subset $\{23, 13, 25, 15\}$.

Exercise 93 Show that a suitable reordering of the vertices of K_5 produces a row echelon form for the submatrix of the incidence matrix in Exercise 88 corresponding to edge subset $\{12, 13, 14, 15\}$. Deduce that these edges form a spanning tree of K_5 .

Exercise 94 Suppose G consists of a cycle on three edges: ab , ac and bc . Let $B'(G)$ be the simplified incidence matrix obtained by replacing -1 's by $+1$'s:

$$B'(G) = \begin{array}{c} a \\ b \\ c \end{array} \begin{array}{ccc} ab & ac & bc \\ \left(\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right) \end{array}.$$

Why does $B'(G)$ fail to represent the cycle matroid of G ?

We may observe that the rank of a cycle matroid for a graph G with n vertices may be at most $n - 1$ since this will be number of edges in any spanning tree. It follows from Lemma 89 that the matrix $B(G)$ can have rank at most $n - 1$. So we should expect to be able to represent the cycle matroid of G with a matrix consisting of column vectors over \mathbb{R}^{n-1} . We now see that this is indeed the case.

Exercise 95 Find a square submatrix of the following matrix (i.e. comprising a subset of four columns) which has non-zero determinant.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}.$$

If the matrix consists of the first four rows of the incidence matrix of a 5-vertex graph G what must the final row be? Now draw the graph G and confirm that your submatrix corresponds to a spanning tree.

Exercise 95 shows a graph G being represented by its incidence matrix $B(G)$ with one row removed. We observe that this row-deleted incidence matrix also represents the cycle matroid of the graph. For if we add any $n - 1$ rows of $B(G)$ then the 1 and -1 in each column will give zero except in those columns whose 1 or -1 lies in the removed row. Thus the removed row is a linear combination of the remaining rows, and a subset of columns is linearly independent in the row-deleted matrix if and only if it is linearly independent in $B(G)$.

Let us denote by $B_i(G)$ the incidence matrix of G with row i deleted. Since $B_i(G)$ represents the cycle matroid of G , any set U of $n - 1$ columns is linearly independent if and only if these columns correspond to the edges of a spanning tree. $B_i(G)$ has $n - 1$ rows, so the submatrix of $B_i(G)$ consisting of the columns

of U , which we denote $B_i(G)[-|U|]$, actually forms a square $(n-1) \times (n-1)$ matrix. By Lemma 60(b), the columns of the submatrix $B_i(G)[-|U|]$ are linearly independent if and only if $\det B_i(G)[-|U|] \neq 0$. In fact, we can say more:

Lemma 96 *Let G be a graph on n vertices and let X be any square submatrix of $B(G)$, its incidence matrix. Then $\det(X) \in \{-1, 0, 1\}$ and if X is of order $n-1$ then $\det X = \pm 1$ if and only if the edges of G corresponding to the $n-1$ columns of X in $B(G)$ form a spanning tree.*

Having submatrices with determinant values limited to 0 and ± 1 will turn out to be important property in the theory of linear programming, so it is worth mentioning now that matrices with this property have a special name:

Definition 97 *A matrix all of whose square submatrices have determinant lying in the set $\{-1, 0, 1\}$ is called unimodular.*

Exercise 98 *Determine whether the matrix*

$$\begin{pmatrix} 1 & 1 & x \\ 0 & 1 & -1 \end{pmatrix}$$

is unimodular for

- (a) $x = -1$, (b) $x = 0$, (c) $x = 1$, (d) $x = -2$.

We promised earlier that the Binet-Cauchy Theorem would enable us to count spanning trees in a graph and our cycle matroid representation gives us exactly that:

Theorem 99 (The Matrix Tree Theorem) *Let G be a graph with n vertices. Then for any i , $1 \leq i \leq n$,*

$$\text{number of spanning trees of } G = \det B_i(G)B_i^T(G),$$

(where $B_i^T(G)$ is just a simplified way of writing $(B_i(G))^T$)

Proof. By the Binet-Cauchy Theorem,

$$\begin{aligned} \det B_i(G)B_i^T(G) &= \sum_{\substack{U \subseteq \{1, \dots, n\} \\ |U|=n-1}} \det B_i(G)[-|U|] \det B_i^T(G)[|U|-] \\ &= \sum_{\substack{U \subseteq \{1, \dots, n\} \\ |U|=n-1}} (\det B_i(G)[-|U|])^2 && \text{by Lemma 60(c)} \\ &= \sum_{\substack{U \subseteq \{1, \dots, n\} \\ U=\text{spanning tree}}} (\pm 1)^2 && \text{by Lemma 96} \\ &= \text{number of spanning trees of } G. \end{aligned}$$

Example 100 *Use the Matrix Tree Theorem to determine the number of spanning trees in the graph G in Figure 2.6*

Solution The incidence matrix of G is

$$B(G) = \begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \end{matrix}.$$

Let us first calculate the product $B(G)B^T(G)$:

$$B(G)B^T(G) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 4 & -2 & -1 \\ 0 & -2 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix} = L(G).$$

Now instead of replacing $B(G)$ by $B_i(G)$ in this product we can delete row and column i from the product $L(G)$. In fact the Matrix Tree Theorem is often stated in terms of the matrix $L(G)$, which is called the *Laplacian* of G . It can be constructed directly from G : the diagonal entries, 1, 4, 3 and 2, are precisely the degrees of the vertices of G ; and when $i \neq j$, the ij -th entry is precisely the number of edges joining vertices i and j .

Let us delete the last row and column of $L(G)$. Then we have:

$$\det \begin{pmatrix} 1 & -1 & 0 \\ -1 & 4 & -2 \\ 0 & -2 & 3 \end{pmatrix} = 5,$$

so there are five spanning trees in G .

Exercise 101 Confirm that deleting row and column 1 of the matrix $L(G)$ in Example 100 also gives a matrix whose determinant has value 5. List the five spanning trees of G .

If counting spanning trees seems rather far from the concerns of OR, we may mention that in experimental design an important optimality criterion is that a design should minimise the confidence regions for estimators of variables that are of interest. Such optimisation is achieved precisely by minimising the number of spanning trees in an associated ‘concurrency graph’.

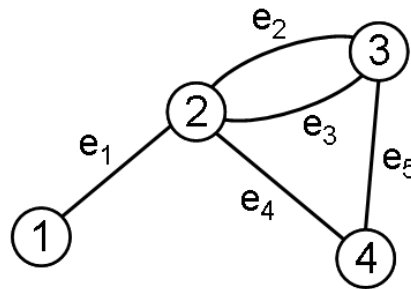


Figure 2.6: graph for Example 100.

2.3.4 Cographic matroids and duality

We digress slightly at this point to introduce a new matroid defined on graphs: the *cocycle matroid*. As before let us motivate it via some optimisation problems.

Network Pruning: links in a computer network which seem to be seldom or never used are to be discarded, subject to the requirement that connectivity of the network is preserved;

Framework Rigidity: a rectangular framework of rods is to be constructed using as few girders as possible subject to the framework being rigid.

As different as these problems seem, they can both be solved as instances of:

MIN-WEIGHT MAXIMAL NON-CUT

Input: an edge-weighted graph the edge weights being in \mathbb{R}

Output: a minimum-weight maximal edge subset whose deletion leaves the graph connected

The application to Network Pruning is immediate, the application to Framework Rigidity is less so; indeed it needs a more precise definition before we can propose a solution*. We shall take our framework to be a rectangular grid of square cells, whose height is m cells and whose length is n cells. Each cell may or may not be made rigid using a diagonal strut. An example is shown in Figure 2.7(a).

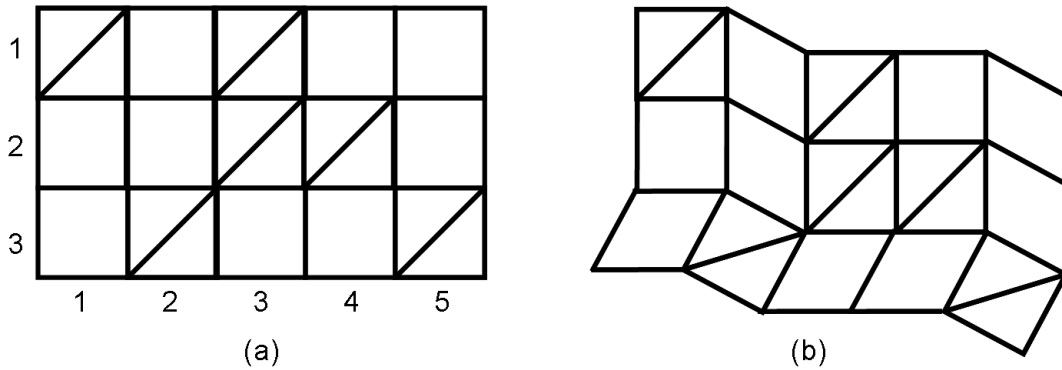


Figure 2.7: (a) a 3×5 rectangular grid; and (b) the same grid deformed.

We assume that the joints in our frameworks are rivets which cannot prevent movement in the plane, so that a cell with no diagonal strut can, in principle, deform into a rhombus. Clearly if every cell contains a diagonal strut then the whole framework must be rigid; however, if we economise on struts in some cells then the rigidity of the whole framework may be compromised. Thus Figure 2.7(a) may be deformed as shown in Figure 2.7(b) potentially, if the framework is part of a building say, with disastrous results.

The link to MIN-WEIGHT MAXIMAL NON-CUT comes from a lovely 1977 result of Ethan D. Bolker and Henry Crapo:

Theorem 102 Suppose an $m \times n$ rectangular grid L is represented by a bipartite graph $G(L)$ as follows:

1. The vertex set of G is $S \cup T$ where $S = \{u_1, \dots, u_m\}$ and $T = \{v_1, \dots, v_n\}$;
2. $u_i v_j$ is an edge of G precisely if there is a diagonal strut in the cell found in the i -th row and the j -th column of the grid.

Then the grid L is rigid if and only if the graph $G(L)$ is connected.

If L is the grid in Figure 2.7(a), for example, the $G(L)$ is the bipartite graph shown in Figure 2.8. This graph is not connected since vertices u_3, v_2 and v_5 form a connected subgraph (called a *component*) which has no edges to the remainder of the graph. Adding a single edge from u_3 to any vertex in $\{v_1, v_3, v_4\}$ would create a connected graph, and this corresponds to adding a strut in any additional cell on the bottom row of the grid.

We now see how MIN-WEIGHT MAXIMAL NON-CUT solves the Framework Rigidity problem: for an $m \times n$ grid we take a *complete bipartite graph* $K_{m,n}$, that is, a bipartite graph on vertex sets S of size m and T of size n with every vertex in S adjacent to every vertex in T . If every edge has weight 1, then any solution to MIN-WEIGHT MAXIMAL NON-CUT will be a set of edges corresponding to a set of diagonal struts which may be omitted from the grid without compromising its rigidity.

*We shall solve framework rigidity with the Greedy Algorithm; we should mention though, that there are much deeper applications of matroid theory in structural rigidity: see the chapter by Walter Whiteley in Neil White's *Matroid Applications*, for example (see Appendix A: Further Reading).

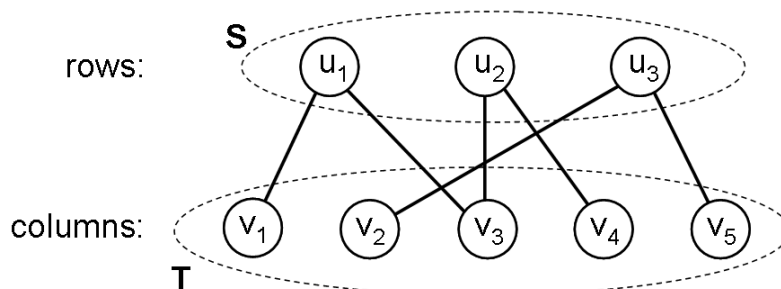


Figure 2.8: bipartite graph corresponding to the grid in Figure 2.7(a).

The point of our discussion, as you might expect, is that the Greedy Algorithm, Algorithm 41, solves MIN-WEIGHT MAXIMAL NON-CUT:

Theorem 103 Let G be a graph with edge-set E . Let \mathcal{A} be the collection of all subsets of E whose deletion leaves the graph connected. Then \mathcal{A} is the collection of independent sets of a matroid on ground set E called the cocycle matroid of G and denoted $M^*(G)$.

A matroid which is isomorphic to the cocycle matroid of some graph is called *cographic*.

Exercise 104 Suppose that the grid in Figure 2.7(a) represents a vertical structure which is to be made rigid. It is preferred where possible to use diagonal struts in the lower rows of cells in order to lower the centre of gravity of the structure, and struts are weighted to reflect this. Draw the weighted complete bipartite graph $K_{3,5}$ which represents the situation where every cell has a diagonal strut. What minimum-weight non-cut edge deletion will the Greedy Algorithm identify in this graph and what reconfiguration of struts will this determine for the original grid?

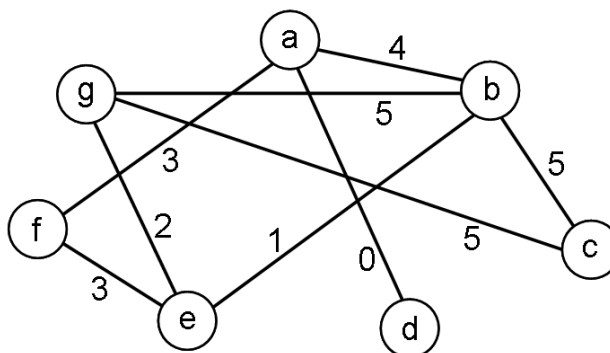


Figure 2.9: network graph for Exercise 105.

Exercise 105 Suppose the graph in Figure 2.9 represents a computer network with the edge weights representing the number of times each link has been used in the last seven days. Which subset of edges will the Greedy Algorithm select to solve MIN-WEIGHT MAXIMAL NON-CUT, and hence the Network Pruning problem, for this graph?

You may have noticed, in your solution to Exercise 105, that the Greedy Algorithm terminated when every remaining edge was required to maintain connectedness of the network, and that this happened precisely when the remaining edges formed a spanning tree. Thus finding a maximal non-cutset of edges was essentially the same as finding a maximal forest. This illustrates a very important concept in matroid theory and in OR generally - that of duality.

Definition 106 Let M be a matroid with ground set A and let \mathcal{B} be the collection of bases of M . Then $\{A \setminus B \mid B \in \mathcal{B}\}$ is again the collection of bases of a matroid, called the dual matroid and denoted M^* .

Notice that if we know the bases of a matroid then we know all its independent sets since they obey the set inclusion property (definition 21(2)).

Exercise 107 Let $A = \{a, b, c, d, e\}$ and let \mathcal{A} be the partition $\{\{a, b\}, \{c, d, e\}\}$. Write down the six bases of the partition matroid $M(\mathcal{A})$. Hence find $M^*(\mathcal{A})$. Explain how the graphs G_1 and G_2 in Figure 2.10 show that $M(\mathcal{A})$ and $M^*(\mathcal{A})$ are both graphic and also both cographic matroids.

Finally, we come to the representation of dual matroids and of cographic matroids in particular. Suppose that M is a representable matroid over ground set A of cardinality n . Let the rank of M , that is, the size of a basis, be r . Then there is some $r \times n$ matrix R_M with columns indexed by the elements of A and with subsets of columns being linearly independent if and only if the corresponding subset of A is independent in M . If M is a graphic matroid then R_M will be $B_t(G)$; if M is a partition matroid then R_M will be the matrix whose columns are unit vectors.

Let us put R_M into row echelon form. By dividing and reordering columns as necessary we can arrive at a new matrix, which still represents M but which has the form

$$\left(\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & X \\ 0 & 1 & \cdots & 0 & \\ \vdots & & \ddots & & \\ 0 & & & 1 & \end{array} \right) = [I_r | X],$$

where X is some $r \times (n - r)$ matrix. This is called a *standard representation* for M .

Exercise 108 Suppose that the following matrix:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

represents a matroid. By reordering the columns put it into standard form $[I_r | X]$. What is the value of r and what is the size of X ?

Now a representation for the dual matroid is given by:

Theorem 109 Let M be a matroid of rank r on ground set A of cardinality n , with $0 < r < n$. Suppose that a standard representation of M is the matrix $[I_r | X]$. Then the dual matroid M^* is represented by $[X^T | I_{n-r}]$, with each column representing the same element of A in both matrices.

We can use this to prove a basic fact about duality:

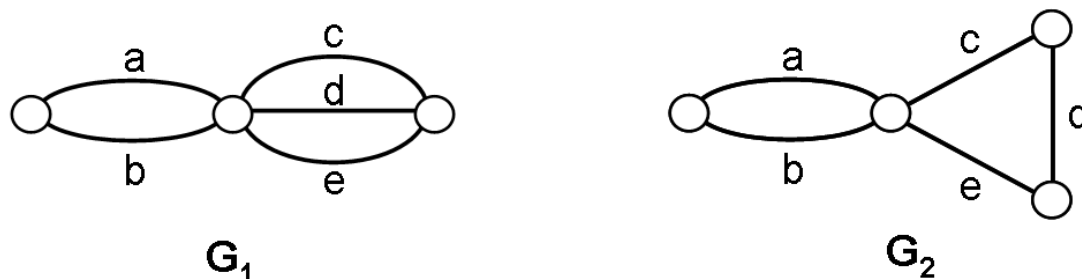


Figure 2.10: graphs G_1 and G_2 for Exercise 107.

Corollary 110 If M_1 is a matroid and M_2 is the dual of M_1 then $M_2^* = M_1$, that is to say, $(M_1^*)^* = M_1$

Proof. We shall prove the corollary for representable matroids, although it is true for any matroid. So let M_1 have standard representation $[I_r | X]$ where r is the rank of M_1 . By Theorem 109, M_2 has representation $[X^T | I_{n-r}]$, where n is the size of the common ground set of M_1 and M_2 . Since the ordering of columns in a representation does not matter we may replace this representation by $[I_{n-r} | X^T]$. Now applying Theorem 109, a representation for the dual of M_2 is $[X^{TT} | I_{n-(n-r)}] = [X | I_r]$ which is a representation for M_1 . So $M_2^* = M_1$, as required.

Exercise 111 Write down the dual representation for the standard representation you found in Exercise 108.



Figure 2.11: graphs for Example 112.

Example 112 Let $A = \{a, b, c, d, e\}$. Describe the partition matroid $M(\mathcal{A})$ which is represented by the matrix:

$$R_M = \begin{pmatrix} a & b & c & d & e \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 \end{pmatrix}.$$

Convert this to a standard representation of $M(\mathcal{A})$; hence find a representation R_{M^*} for $M^*(\mathcal{A})$. By appending an extra row to each representation, convert them to the incidence matrices of graphs G and G^* , respectively; how do these graphs compare to those you found in Exercise 107?

Solution The collection \mathcal{A} must be $\{\{a, b\}, \{c, d, e\}\}$ since the linearly independent subsets of columns of R_M correspond precisely to subsets of A having at most one element from each of $\{a, b\}$ and $\{c, d, e\}$. Now changing the order of the ground set elements gives:

$$\begin{pmatrix} a & c & b & d & e \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 1 \end{pmatrix},$$

which is a standard representation. Transposing the matrix consisting of its last three columns and appending I_3 gives:

$$R_{M^*} = \begin{pmatrix} a & c & b & d & e \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The incidence matrix of a graph has each column summing to zero, so an extra row may be added unambiguously to R_M and R_{M^*} and interpreted as graphs on three and four vertices respectively:

$$B(G) = \begin{pmatrix} a & b & c & d & e \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \end{pmatrix}$$

$$B(G^*) = \begin{pmatrix} a & c & b & d & e \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & -1 \end{pmatrix}$$

The graphs G and G^* , shown in Figure 2.11, are identical, up to relabelling of edges, to the graphs G_1 and G_2 in Figure 2.10. This suggests that if a matroid M and its dual matroid M^* are both graphic then they are both also cographic and the relevant graphs may be constructed from the appropriate dual representations.

Exercise 113 Repeat Example 112 for the same ground set $A = \{a, b, c, d, e\}$ but with representation matrix:

$$R_M = \begin{pmatrix} & a & b & c & d & e \\ \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} & -1 & 0 & 0 & 0 \\ & 0 & 0 & 1 & -1 & 0 \\ & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

(The graphs will be different this time.)

2.4 Transversal matroids

Our aim in this section is to represent transversal matroids over the real numbers in such a way that we can easily manipulate them on a computer. Graphic matroids were found to have an immediate representation in terms of graph structure; the representation of cographic matroids was then achieved indirectly using Theorem 109. Partition matroids, the special case of transversal matroids in which all sets are disjoint, was found to be even more simple.

Transversal matroids in general are harder; indeed, the fact that they are representable at all is quite an important result in matroid theory. Luckily, there is a natural stepping stone on the way to this goal, a representation which uses an incidence matrix similar to the incidence matrix of a graph. Rather than real numbers, it uses symbols, ‘indeterminates’, and this limits its usefulness from a computational point of view. In fact we shall make extensive use of indeterminates in Chapter 3 to give an elegant and self-contained description of matroid intersection; it would be much less elegant if the matroids we were to work with were already represented symbolically. So we finally present an algorithm for replacing the indeterminates one by one with real numbers. Thus we finally arrive at a representation we can be happy with.

2.4.1 The Mirsky-Perfect representation

Suppose we have a ground set $A = \{a, b, c\}$ and subset system $\mathcal{A} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. This simple example looks like a graph. Its cycle matroid would treat the sets in \mathcal{A} as the ground set (edges) and the incidence matrix of the graph would indicate that the three edges form a dependent set — a cycle:

$$\begin{array}{ccc} & ab & ac & bc \\ \begin{matrix} a \\ b \\ c \end{matrix} & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix} \end{array} \text{ which, transposed, becomes } \begin{array}{ccc} & a & b & c \\ \begin{matrix} ab \\ ac \\ bc \end{matrix} & \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}, \end{array}$$

both matrices having determinant zero. We took the transpose because it was suggestive of a representation of the transversal matroid $M(\mathcal{A})$: how convenient it would be if we could just generalise this idea to arbitrary collections of subsets to get a representation for transversal matroids in general! But there is immediately a problem: the set $\{a, b, c\}$ is actually a transversal for our system \mathcal{A} , so any representation needs to have *non-zero* determinant for the corresponding three columns. You may recall that in Exercise 94 an all +1’s incidence matrix failed to represent the cycle matroid precisely *because* it gave a non-zero determinant for the cycle of three edges; perhaps this simplified incidence matrix will work?

Exercise 114 Calculate the determinant for the following incidence matrix for a system of subsets of $\{a, b, c, d\}$:

$$\begin{array}{c} \{a, b\} \\ \{a, d\} \\ \{b, c\} \\ \{c, d\} \end{array} \begin{pmatrix} a & b & c & d \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Since $\{a, b, c, d\}$ is a transversal for the subset system $\mathcal{A} = \{\{a, b\}, \{a, d\}, \{b, c\}, \{c, d\}\}$, you will have found that your answer to Exercise 114 indicates that the all +1's incidence matrix is unsuitable for representing transversal matroids. There is a simple way of rescuing the situation invented by Leon Mirsky and Hazel Perfect in the 1960s: we will briefly describe it because it reinforces some ideas we have already met and introduces a new idea which will be essential for our description of matroid intersections. It will then be the basis of a somewhat simpler representation which is more suited to algorithm design.

Suppose we take the (transposed) all +1's incidence matrix but, instead of a +1 in position ij , we place an indeterminate t_{ij} . An indeterminate may be thought of as a variable but it is a variable which is not intended to take any value — it acts as a symbolic ‘placeholder’ to distinguish entry ij from other entries. Now we have:

Theorem 115 (Mirsky and Perfect) Let $A = \{a_1, \dots, a_n\}$ be a set and $\mathcal{A} = \{A_1, \dots, A_m\}$ a collection of subsets of A . Let $X_{\mathcal{A}}$ be the $m \times n$ matrix whose ij -th entry x_{ij} is defined by

$$x_{ij} = \begin{cases} t_{ij} & \text{if } A_i \text{ contains } a_j \\ 0 & \text{otherwise} \end{cases},$$

where the t_{ij} are indeterminates. Then $X_{\mathcal{A}}$ represents the transversal matroid $M(\mathcal{A})$.

Example 116 Construct the matrix $X_{\mathcal{A}}$ of the Mirsky-Perfect Theorem for the set system $\{\{a, b\}, \{a, d\}, \{b, c\}, \{c, d\}\}$ from Exercise 114 and show that it represents the transversal matroid $M(\mathcal{A})$.

Solution The matrix is

$$X_{\mathcal{A}} = \begin{array}{c} \{a, b\} \\ \{a, d\} \\ \{b, c\} \\ \{c, d\} \end{array} \begin{pmatrix} a & b & c & d \\ t_{11} & t_{12} & 0 & 0 \\ t_{21} & 0 & 0 & t_{24} \\ 0 & t_{32} & t_{33} & 0 \\ 0 & 0 & t_{43} & t_{44} \end{pmatrix}.$$

Any subset of $A = \{a, b, c, d\}$ will be a partial transversal for \mathcal{A} . Now if $\det X_{\mathcal{A}}$ is non-zero then any subset of the columns of the matrix must be linearly independent and this will confirm we are representing $M(\mathcal{A})$. Unfortunately, elementary row operations are very tricky to apply in the presence of indeterminates. For example, we might start by subtracting $(t_{21}/t_{11}) \times R1$ from $R2$ in $X_{\mathcal{A}}$. The first column will now be in row echelon form but the new pivot element in row 2 is $-t_{21}t_{12}/t_{11}$; and the more elementary row operations we apply the more complicated the remaining matrix elements will become — things quickly get unmanageable!

This is where Theorem 63 from subsection 2.2.2 comes in. Any 4×4 permutation matrix P whose non-zero entries match four non-zero entries in $X_{\mathcal{A}}$ will make a non-zero contribution to the summation which defines $\det X_{\mathcal{A}}$. And since any four such entries will yield a different set of indeterminates for different permutation matrices no terms in this summation can cancel. We need only exhibit one example of P to confirm that $\det X_{\mathcal{A}}$ is non-zero.

We will exhibit two, to illustrate what we have just said:

$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \longrightarrow +t_{11}t_{24}t_{32}t_{43} \qquad \begin{pmatrix} & 1 & & \\ 1 & & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \longrightarrow -t_{12}t_{21}t_{33}t_{44}.$$

You may like to check that the signs of these terms (i.e. the signs of the permutation matrices) are correct. Even though the signs are opposite it is clear that the terms do not cancel since different indeterminates appear in each.

Exercise 117 Construct the Mirsky-Perfect representation for the transversal matroid $M(\mathcal{A})$ with ground set $A = \{a, b, c, d\}$ and subset collection $\mathcal{A} = \{\{a, b, c\}, \{a, d\}, \{c\}\}$. Find the determinant of each 3×3 submatrix and confirm that the corresponding subsets of $\{a, b, c, d\}$ are transversals of \mathcal{A} precisely when the determinants are non-zero.

For Mirsky-Perfect representation matrices, summing over permutation matrices is no more computationally efficient than evaluating determinants by applying elementary row operations. However, permutation matrices are a natural choice for studying transversal matroids: notice that a 1 in row i and column j corresponds to representing set i from \mathcal{A} by element j from A . So one unique 1 in each row and column specifies a transversal *and* a matching of this transversal to subsets. Indeed, if we look at the first of the two terms calculated in Example 116: $+t_{11}t_{24}t_{32}t_{43}$, we can read off the corresponding transversal: a represents $\{a, b\}$, d represents $\{a, d\}$, b represents $\{b, c\}$ and c represents $\{c, d\}$.

Exercise 118 Return to Exercise 56 at the end of Chapter 1 and represent the collection of subsets using the matrix $X_{\mathcal{A}}$ of Theorem 115. Show that the columns corresponding to Abdul, Baldish, Cheung and Franz form a submatrix whose determinant evaluates to a single term, and that this single term specifies a transversal and matching for the product experience subsets.

2.4.2 The Piff-Welsh Theorem

While the Mirsky-Perfect representation is elegant and revealing it is disappointing to have to work with indeterminates instead of numerical quantities, not least because it puts out of reach much of computational linear algebra. As a matter of fact we can always replace indeterminates by so-called *algebraically independent* real numbers which behave analogously to indeterminates and confirm incidentally that all transversal matroids are representable over \mathbb{R} . But algebraically independent numbers are no easier to compute with than indeterminates and are moreover notoriously elusive (it is not even known whether π and e (the exponential, 2.71828...) are algebraically independent!) Luckily, Dominic Welsh at Oxford University and his DPhil student Mike Piff simplified matters dramatically in 1970 by proving that the indeterminates in the Mirsky-Perfect representation could be replaced by suitably chosen values (including zero) from any sufficiently large field:

Theorem 119 (The Piff-Welsh Theorem) *If M is a transversal matroid then M can be represented over any sufficiently large finite field and over any infinite field. In particular transversal matroids are representable over the field of rational numbers.*

Since we can represent transversal matroids over the rationals and since we can multiply vectors in a representation by scalars this means we can revert to the original incidence matrix representation that we abandoned after Exercise 114 but using different integer values. Discovering exactly *which* integer values is a somewhat laborious process but it is worth following in detail.

Piff and Welsh proved their theorem by showing that a representation for a larger (and less complicated) matroid could be reduced to a representation for a smaller (but more complicated) matroid by merging pairs of ground set elements, together with their representing vectors. We shall refer to this as the Piff-Welsh Merge Algorithm.

The effect of Piff-Welsh merging is most easily viewed as inserting values, one by one, into an incidence matrix. We start by splitting a given set system into a partition of its ground set augmented with new indeterminate elements t_{ij} . This is illustrated in Figure 2.12. As we saw in section 2.3.2, we can represent a partition matroid directly by its incidence matrix, so this is a larger but very uncomplicated matroid. The main work in constructing the Piff-Welsh representation involves progressively assigning values to the indeterminates t_{ij} . Assigning a value to t_{ij} has the effect of merging it with the i -th element of the original ground set; it is this merging which inserts the assigned value back into the original incidence matrix.

On the left of Figure 2.12, the set system is $A = \{a, c, e\}$, $B = \{a, b, c, d\}$ and $C = \{a, d\}$ and the Mirsky-Perfect representation of its transversal matroid is shown; on the right a has been split off from B and C as the indeterminates t_{21} and t_{31} . Similarly, c has been split off from B and d has been split off from C . The result is a partition of an augmented ground set: $\{a, b, c, d, e, t_{21}, t_{31}, t_{23}, t_{34}\}$.

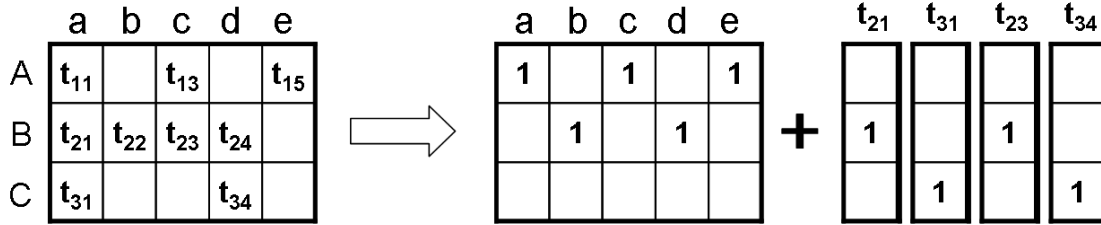


Figure 2.12: schematic illustration of splitting a set system into a partition: the relationship of the original sets A , B and C to the ground set $\{a, b, c, d, e\}$ is represented via the Mirsky-Perfect representation of its transversal matroid (left).

While the incidence matrix of a set partition represents the corresponding partition matroid, it is not true that the incidence matrix of a set system in general represents the corresponding transversal matroid. For example, if all the t_{ij} are set equal to 1 in the left-hand matrix in Figure 2.12 then the columns for a , d and e are not linearly independent even though $\{a, d, e\}$ is certainly a transversal of the set system. We shall show how the Piff-Welsh merge algorithm gives new values to the indeterminates t_{ij} by merging back all the indeterminate columns created in the split in Figure 2.12; some of these new values must clearly be other than +1.

Completing this example will take a couple of pages because the algorithm must be repeated for each indeterminate. Once we have seen it in action, the algorithm will be stated formally. Firstly, then, we set out the vectors from the right-hand side of Figure 2.12 in the form of a ‘tableau’. Our first insertion will insert the t_{21} entry into the a column, so these columns of the tableau are highlighted:

Iteration 1: indeterminate t_{21} :

a	b	c	d	e	t_{21}	t_{31}	t_{23}	t_{34}
1	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	1
L					R			

Beneath the highlighted columns we are going to store vectors called L and R . We shall also use L and R to refer to the columns (so, above, column a is the ‘ L ’ column and t_{21} is the ‘ R ’ column). We will set out a tableau like the one above for each t_{ij} that has to be inserted into columns a – e . Sometimes we can just use the value $t_{ij} = 1$ but not always: the purpose for the tableau is to identify combinations of columns which *prevent us* from making the assignment $t_{ij} = 1$. This information will be placed in the L and R vectors below the tableau. What are these combinations of columns? There are two key ingredients to getting the final value for each t_{ij} and this is the first of them:

Step I: Find a collection C of columns which combines with the L column to give a square non-singular matrix (call it X_L) and also combines with the R column to give a square non-singular matrix (call it X_R). Record $\det X_L$ in the L vector and record $(-1)^k \det X_R$ in the R vector, where k is number of columns in the collection C that appear in the tableau *between* the L column and the R column.*

We repeat this for all collections of columns which qualify by giving pairs of non-singular matrices. In the above tableau, there are, in fact, no possible combinations of columns which will give a non-singular

*We can avoid the need for this factor of $(-1)^k$ by moving both the L and R columns to one end of the tableau. We have chosen what seems more simple: always to keep the columns of the incidence matrix in their original ordering.

matrix both when combined with column a (the L column) and with column t_{21} (the R column); so the L and R vectors are blank. For example, columns b and t_{31} , when combined with columns a and t_{21} , give:

$$\det \begin{pmatrix} a & b & t_{31} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 1 \quad \text{and} \quad \det \begin{pmatrix} b & t_{21} & t_{31} \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 0, \quad \text{respectively,}$$

with only the first of the combinations being non-singular.

If no pairs of non-singular matrices can be formed then there is nothing to prevent us from setting $t_{ij} = 1$. So for our first insertion we have found we can place t_{21} back into column a of our matrix with value 1. Let us continue: we must now insert t_{31} into column a :

Iteration 2: indeterminate t_{31} :

a	b	c	d	e	t_{31}	t_{23}	t_{34}
1	0	1	0	1	0	0	0
1	1	0	1	0	0	1	0
0	0	0	0	0	1	0	1
L					R		

Notice that column t_{21} and column a have been merged: column t_{21} has disappeared and instead there is a new non-zero value in column a (the value being 1, because we found this was not prevented).

There is again no collection of two columns in the new tableau which we can combine with both the L column (i.e. a) and the R column (i.e. t_{31}) to get non-singular matrices. This means there is again nothing to prevent us inserting t_{31} into a using value 1. So now column t_{31} disappears and the final entry in column a becomes 1. This is beginning to look easy!

In fact, you will always find that the first insertion, at least, just gives an assignment $t_{ij} = 1$. We are now ready to make our third insertion, combining t_{23} into the c column. Now the c column is the one marked 'L' and this time we *can* find two columns that give non-singular matrices X_L with the L and X_R with the R column. They are marked below with an X:

Iteration 3: indeterminate t_{23} :

a	b	c	d	e	t_{23}	t_{34}
1	0	1	0	1	0	0
1	1	0	1	0	1	0
1	0	0	0	0	0	1
X		-1			1	X
		L			R	

Both combinations have non-zero determinant:

$$\det \begin{pmatrix} a & c & t_{34} \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = -1 \quad \text{and} \quad \det \begin{pmatrix} a & t_{23} & t_{34} \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = 1, \quad \text{respectively,}$$

and these values are recorded in the L and R vectors, respectively. Because neither of the columns a or t_{34} appear between the L and R columns, the actual value recorded in the R vector is $(-1)^0 \times 1 = 1$. This is easy to see from the tableau: no X symbols appear between the L and R columns.

They are the subject of the second key step in our algorithm:

Step II: Assign to t_{ij} any non-zero value such that the vector sum $L + t_{ij} \times R$ has no zero entries.

For the above tableau, we take our single-entry vectors $L = (-1)$ and $R = (1)$ and observe that $t_{23} = -1$ will give

$$L + t_{23} \times R = (-1) + -1 \times (1) = (-1) + (-1) = (-2),$$

a vector whose single entry is non-zero, as required. Note that $t_{23} = 0$ would also have given a non-zero sum but we insist on the t_{ij} being themselves non-zero. However, $t_{23} = 2$ would have been another valid choice.

We now come to our final insertion, as we place t_{34} into the d column of our representation. Notice that the c column has had the valid value $t_{23} = -1$ inserted in our final tableau:

Iteration 4: indeterminate t_{34} :

a	b	c	d	e	t_{34}
1	0	1	0	1	0
1	1	-1	1	0	0
1	0	0	0	0	1
×		×	1		-2
×			-1	×	1
			L		R

There are two pairs of columns giving non-singular matrices when combined with L and R : a and c , and a and e . The latter pair of matrices have determinants:

$$\det \begin{pmatrix} a & d & e \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = -1 \quad \text{and} \quad \det \begin{pmatrix} a & e & t_{34} \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = -1, \quad \text{respectively.}$$

The corresponding values in the L and R vectors are -1 and $(-1)^1 \times -1 = +1$, respectively, because column e appears between the L column (i.e. column d) and the R column (i.e. column t_{34}). This is easy to check: just notice that there is an X symbol appearing between the two highlighted columns in the second row of the L and R vectors.

If we make the assignment $t_{34} = -1$ then our L and R vectors will sum as follows:

$$L + t_{34}R = \begin{pmatrix} 1 \\ -1 \end{pmatrix} + (-1) \times \begin{pmatrix} -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -2 \end{pmatrix},$$

with no zero entries, as required. So this is a valid final insertion into our representation. All t_{ij} entries have now been inserted and the completed representation for the transversal matroid of the set system $A = \{a, c, e\}$, $B = \{a, b, c, d\}$ and $C = \{a, d\}$ is

$$\begin{matrix} & a & b & c & d & e \\ A & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix} \\ B & \begin{pmatrix} 1 & 1 & -1 & 1 & 0 \end{pmatrix} \\ C & \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \end{pmatrix} \end{matrix}$$

This has been a lot of work to turn two $+1$'s into -1 's! But there does not seem to be an easier way to construct representations of transversal matroids. Transversal matroids are subtle objects, as we have already seen, for example at the end of Chapter 1, and as we shall see again at the end of this chapter. The following exercise is quite long but it summarises quite a lot of material, so it is worth the work:

Exercise 120 A set system \mathcal{A} consists of three copies of the set $\{a, b, c, d\}$. A representation of the

transversal matroid $M(\mathcal{A})$ of \mathcal{A} has been partially constructed as shown in the tableau below:

a	b	c	d	t_{33}	t_{42}	t_{43}
1	1	1	1	0	0	0
1	-1	2	0	0	1	0
1	2	0	0	1	0	1
×	×	1		-2		
×		-2	×	1		
×		1		-1	×	
	×	-4	×	-1		
	×	2		-1	×	
			L	R		

The values of 1 and -1 in the second and fourth entries of the R vector are obtained from the determinants

$$\det \begin{pmatrix} a & d & t_{33} \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \det \begin{pmatrix} b & d & t_{33} \\ 1 & 1 & 0 \\ -1 & 0 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \text{ respectively.}$$

- What are the actual values of these determinants and why do they differ from the entries in the R vector?
- Why are there no X symbols in the t_{43} column of the tableau?
- To which indeterminate t_{ij} are we currently assigning a value?
- What value will the L and R vectors allow to be assigned to this indeterminate?
- Write down the next tableau, which will be used to insert t_{42} into the representation. There are six pairs of determinants to find and evaluate—this may take you rather a long time but the more you attempt the better.
- What value may be assigned to t_{42} as a result of this new tableau? (If you did not calculate all the determinants in part (e) you can still try to answer this by using the completed tableau as given in the solution in section C.2).

We now give a formal statement of the Piff-Welsh merge algorithm. The algorithm is presented in an abstract form without reference to tableaux or X symbols or factors of $(-1)^k$: all those belong to an implementation of the algorithm which makes it easy to apply by hand (see the footnote to Step I above).

Algorithm 121 (The Piff-Welsh Merge Algorithm)

INPUT: $m \times n$ matrix X

vectors v_L and v_R chosen from the columns of X

OUTPUT: real number t

- $C :=$ set of all pairs (X_L, X_R) of $m \times m$ non-singular matrices of the form $X_L = [v_L | Y]$ and $X_R = [v_R | Y]$, where Y is an $m \times (m-1)$ submatrix of X not including columns v_L or v_R ;
- $Z :=$ a matrix with two cols, initially having zero rows;
- for** each pair $(X_L, X_R) \in C$ **do**
- append the row vector $(\det X_L, \det X_R)$ to Z ;
- od**;
- return** a real number t such that $Z \times (1, t)^T$ has no zero elements

It is not hard to see that step 6 in Algorithm 121 is always possible, provided we have sufficient choices for t . This is certainly the case for \mathbb{R} (and indeed for sufficiently large finite fields). Moreover the choice of t ensures that the algorithm preserves independent sets. Suppose that the vectors in X are a representation and that we want to merge v_L and v_R to represent the same ground set element, and suppose X_L corresponds to a set of linearly independent columns (so it is a non-singular submatrix). Replace v_L in X_L with $v_L + tv_R$ to get X'_L . Then by Corollary 66, $\det X'_L = \det X_L + t \det X_R = (\det X_L, \det X_R) \times (1, t)^T$, which is non-zero by the choice of t in step 6. So SX'_L is again independent.* It can also be checked that dependent subsets of X are also preserved by Algorithm 121.

*In fact, we need X'_L also to be non-singular when either one of X_L and X_R is; the choice of t in step 6 of Algorithm 121 covers these cases as well.

Example 122 We revisited our committee representative problem in Exercise 118. The Mirsky-Perfect representation of the transversal matroid was:

$$X_{\mathcal{A}} = \begin{matrix} & A & B & C & D & E & F \\ \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix} & \begin{pmatrix} t_{11} & 0 & t_{13} & t_{14} & t_{15} & 0 \\ 0 & 0 & 0 & t_{24} & t_{25} & t_{26} \\ t_{31} & t_{32} & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{43} & t_{44} & 0 & 0 \end{pmatrix} \end{matrix}.$$

Find a representation matrix with integer entries.

Solution The initial tableau is:

Iteration 1: indeterminate t_{24} :

A	B	C	D	E	F	t_{24}	t_{25}	t_{31}	t_{43}	t_{44}
1	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	1
			L			R				

There are no combinations of three columns to give non-singular 4×4 matrices X_L with the L column and X_R with the R column. This is not so easy to check with a large matrix but here is useful tip:

A square submatrix in a tableau is non-singular if and only if some of its entries form a permutation matrix.

How can we form a permutation matrix incorporating column L (i.e. column D)? We must have a 1 in the second row but we can only do this by taking a column that is identical to the R column. This will make $\det X_R = 0$. Similarly to get a non-singular X_R we must take a column identical to column D so that X_L becomes singular. So (as usual at the first iteration), we can just take value 1 for our assignment: $t_{24} = 1$. Here is the second tableau:

Iteration 2: indeterminate t_{25} :

A	B	C	D	E	F	t_{25}	t_{31}	t_{43}	t_{44}
1	0	1	1	1	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	1
				\times				\times	\times
				L		R			

The new column D allows us to form lots of pairs of matrices containing a permutation matrix. They are all identical to the one shown by the \times symbols, which gives the pair of determinants recorded in the L and R vector. None of the \times symbols appear in between L and R so there is no need to adjust the sign in the R vector. An assignment of $t_{25} = -1$ gives a non-zero sum of L and R so this is the value which is inserted in column E .

The next two tableaux give no non-singular matrix pairs so we omit them*. This brings us to the final tableau (notice the inclusion of new values in columns E , A and C):

Iteration 5: indeterminate t_{44} :

A	B	C	D	E	F	t_{44}
1	0	1	1	1	0	0
0	0	0	1	-1	1	0
1	1	0	0	0	0	0
0	0	1	0	0	0	1
×	×	×	-2	×	×	1
×	×	×	1	×	×	-1
×	×	×	-2	×	×	1
×	×	×	1	×	×	-1
			L			R

Notice that, for every row of the L and R vectors, there is a single X between the vectors, so the signs recorded in the R vector are opposite to the actual determinants. The sum of the two vectors has two zero entries but if we assign $t_{44} = -1$ then $L + t_{44}R$ is zero-free, so this is the final value to insert in our representation:

$$\begin{array}{c} A \quad B \quad C \quad D \quad E \quad F \\ P_1 \quad \left(\begin{array}{cccccc} 1 & 0 & 1 & 1 & 1 & 0 \end{array} \right) \\ P_2 \quad \left(\begin{array}{cccccc} 0 & 0 & 0 & 1 & -1 & 1 \end{array} \right) \\ P_3 \quad \left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right) \\ P_4 \quad \left(\begin{array}{cccccc} 0 & 0 & 1 & -1 & 0 & 0 \end{array} \right) \end{array}.$$

Exercise 123 Write down the tableaux for iterations 3 and 4 which were omitted from Example 122.

Exercise 124 We saw in Exercise 114 that the transversal matroid for the set system with incidence matrix,

$$\begin{array}{c} a \quad b \quad c \quad d \\ \{a, b\} \quad \left(\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right) \\ \{a, d\} \\ \{b, c\} \\ \{c, d\} \end{array},$$

was not represented by this matrix since its four columns are linearly dependent. If we use the Piff-Welsh Merge Algorithm to construct a representation then the first three tableaux give no non-singular matrix pairs, so we arrive at the tableau:

a	b	c	d	t_{44}
1	1	0	0	0
1	0	0	1	0
0	1	1	0	0
0	0	1	0	1
			L	R

Complete this tableau by indicating the columns which produce a non-singular matrix pair with columns L and R and recording the determinants in the L and R vectors. What assignment to t_{44} is valid?

Right at the beginning of our discussion of representations, in section 2.3.1, we mentioned that studying representations of transversal matroids would also show us how to represent uniform matroids $U_{k,n}$ in which the independent sets were all subsets of size at most k of a ground set of size n . Indeed, this is a corollary to Theorem 119:

Corollary 125 Any uniform matroid may be represented by a matrix with integer entries.

Proof. Suppose the ground set of $U_{k,n}$ is $A = \{a_1, \dots, a_n\}$. Let \mathcal{A} be the set system which consists of k copies of A . Then any subset of size k of A is a transversal for \mathcal{A} and vice versa. So $M(\mathcal{A})$ is isomorphic to $U_{k,n}$. Since by Theorem 119 $M(\mathcal{A})$ is representable over the field of rational numbers, we may represent $U_{k,n}$ by a matrix whose entries are rationals. It suffices to multiply this matrix by the least common multiple of the denominators of its entries to get a representation using only integers.

Exercise 126 Suppose we wish to construct a representation for the matroid $U_{2,5}$. The proof of Corollary 125 suggests we equate $U_{2,5}$ with the transversal matroid for the set system consisting of two copies of $\{a, b, c, d, e\}$. This has incidence matrix:

$$\begin{array}{c} a \quad b \quad c \quad d \quad e \\ \{a, b, c, d, e\} \quad \left(\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} \right) \\ \{a, b, c, d, e\} \end{array}.$$

*Or rather, we delegate to the reader! See Exercise 123.

This is not a valid representation of $U_{2,5}$ since, for example, $\{a, b\}$ is independent in this matroid by definition but the corresponding columns in the incidence matrix are not linearly independent. Apply the Piff-Welsh Merge Algorithm to produce a valid representation. Hence give a representation for the dual matroid $U_{2,5}^*$.

The first tableau has been set up for you:

Iteration 1: indeterminate t_{21} :

a	b	c	d	e	t_{21}	t_{22}	t_{23}	t_{24}	t_{25}
1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1
L					R				

The solution reveals that $U_{2,5}^* = U_{3,5}$. In fact, it is true in general that $U_{k,n}^* = U_{n-k,n}$ (and, by Corollary 110, vice versa).

Exercise 127 Look back to Exercise 120: you may recognise this as being a partial construction of a representation of $U_{3,4}$. Using your solution to that exercise, perform the final Piff-Welsh merge iteration to complete the representation. Hence write down a representation for $U_{1,4}$.

2.4.3 The matching matroid

In section 1.3.3 of Chapter 1, we claimed that we could not construct a maximum matching in a graph by greedily choosing edges. It was certainly true that if we tried to copy what worked for building a spanning tree — adding edges greedily — then we found ourselves forced to backtrack. However, if we change the focus from edges to vertices we get a new matroid which can, in theory, help us find matchings:

Definition 128 Let G be a graph with vertex set V and edges set E . If E' is a subset of E then we say that all those vertices in V which are incident with some edge in E' are covered by E' . Now let X be a subset of V . Let \mathcal{A} be the family of subsets of X defined by $X' \subseteq X$ is in \mathcal{A} if and only if X' is covered by some matching in G . Then (X, \mathcal{A}) is a matroid, called the matching matroid of G with respect to X , which we will denote by $M_X(G)$.

A matroid M is called a *matching matroid* if there is some graph $G = (V, E)$ and some subset X of V , such that M is isomorphic to $M_X(G)$.

Example 129 Write down all the bases for the matching matroid of the graph G in Figure 2.13. Write down the bases of the transversal matroid whose ground set is the same as the vertex set of G , i.e. $\{a, b, c, d, e, f\}$ and whose independent sets are partial transversals of the subset collection $\mathcal{A} = \{\{a, b\}, \{a, c\}, \{d\}, \{c, e, f\}\}$. What do you notice?

Solution No perfect matching is possible in G . The maximal matchings are $\{ab, cd\}$, $\{ab, de\}$, $\{ab, df\}$, $\{ac, de\}$, $\{ac, df\}$, $\{bc, de\}$ and $\{bc, df\}$ and the vertices covered by the edges in each form a basis of the matching matroid $M_G(V)$. We may systematically generate the bases of the transversal matroid on \mathcal{A} by taking determinants of 4×4 submatrices in the Mirksy-Perfect representation:

$$\begin{array}{c} a \quad b \quad c \quad d \quad e \quad f \\ \begin{array}{l} \{a, b\} \\ \{a, c\} \\ \{d\} \\ \{c, e, f\} \end{array} \begin{pmatrix} t_{11} & t_{12} & 0 & 0 & 0 & 0 \\ t_{21} & 0 & t_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & t_{34} & 0 & 0 \\ 0 & 0 & t_{43} & 0 & t_{45} & t_{46} \end{pmatrix} \end{array}$$

Selecting the first four columns, for instance, gives

$$\det \begin{pmatrix} t_{11} & t_{12} & 0 & 0 \\ t_{21} & 0 & t_{23} & 0 \\ 0 & 0 & 0 & t_{34} \\ 0 & 0 & t_{43} & 0 \end{pmatrix} = t_{12}t_{21}t_{34}t_{43},$$

which we recall gives the transversal $b \rightarrow \{a, b\}$, $a \rightarrow \{a, c\}$, $d \rightarrow \{d\}$ and $c \rightarrow \{c, e, f\}$. Repeating this for all choices of four columns we observe finally that the bases of the transversal matroid are identical to those of the matching matroid.

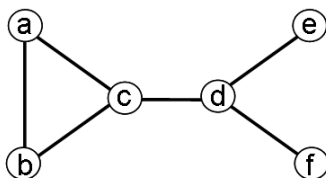


Figure 2.13: graph for Example 129.

Exercise 130 For the bipartite graph G in Figure 2.14 find all bases for the matching matroid $M_X(G)$. Referring back to Example 129, deduce that every transversal matroid is isomorphic to a matching matroid.

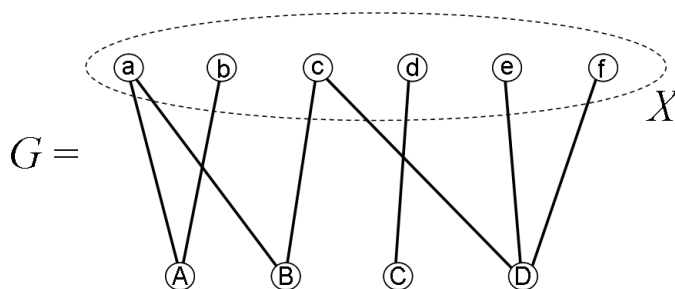


Figure 2.14: graph for Example 130.

Now, the approach of Chapter 1 was that, given a matroid, the Greedy Algorithm would produce a basis (in fact a basis of minimum weight, but we are not considering weights at present). According to this approach, we can apply the Greedy Algorithm to, say, the graph in Figure 1.7 at the end of section 1.3.3 and find a basis of size 8. This will tell us the graph has a perfect matching, although it will not tell us which edges form the matching. However, we can repeatedly delete edges from the graph and rerun the Greedy Algorithm. As soon as we get a basis of size less than 8 we know we have just deleted an edge, call it uv , which lies in the only remaining matching in our depleted graph. Delete u and v and all incident edges: any matching in the remaining graph must combine with edge uv to give a matching in G . Repeating this rather laborious but perfectly practical process on smaller and smaller graphs will eventually produce all the edges of a matching for the original graph.

Is the algorithm just described a good way to look for maximal matchings? This question takes us back to the opening of this chapter where we said that checking independence of a set X for a transversal matroid $M(\mathcal{A})$ amounted to looking for a matching of X to subsets in \mathcal{A} . We avoided this issue by instead taking a representation of the transversal matroid: now checking independence was separated off as a matter of applying linear algebra.

So the question is, are matching matroids representable? A deep theorem of Edmonds and Fulkerson says that every matching matroid is isomorphic to some transversal matroid, exactly as we found in Example 129. So, in theory, we can solve matching problems by solving the corresponding problem for an appropriate transversal matroid, represented over the real numbers. Unfortunately, the isomorphism linking the two classes of matroids appears to be far from easy to construct: at the current time

representing the matching matroid would seem to be more difficult than solving matching problems directly by graph theoretic means.

Exercise 131 *The Mirsky-Perfect representation for the transversal matroid in Example 129 is also a representation for the matching matroid for the graph in Figure 2.13. Confirm by inspection that all 4×4 submatrices which do not contain the fourth column must have determinant zero.*

Chapter 3

Matroid intersection

3.1 Introduction

We now come to the outer limits of matroid theory as an approach to OR. This chapter is devoted to the identification of optimisation problems which cannot be solved greedily but which can be solved by two greedy algorithms working in tandem. ‘Two heads are better than one’ the adage goes and this is certainly the case for problems which can be formulated in terms of the intersection of two matroids. Three heads, however, do not make things even better — we shall conclude this chapter by mentioning that some problems formulated as the intersection of *three* matroids appear to be intractable.

The main part of the chapter consists of examples of problems that are solved by matroid intersection. We shall see that the Binet-Cauchy theorem, together with the idea of using indeterminates, reformulates all such problems in terms of calculating determinants of matrix products.* This limits us to representable matroids but we have plenty of material from Chapter 2 to draw on!

3.2 The intersection of two matroids

We must first begin by saying what exactly *is* the intersection of matroids. We shall see that the Binet-Cauchy theorem and the idea of using indeterminates allows examples to be formulated very neatly and we shall move on to apply this idea to graphical and transversal problems. As we saw in the previous chapter, working with indeterminates poses its own problems computationally. Algorithms exist, notably the classic algorithm of Jack Edmonds, to solve matroid intersection problems efficiently. The approach taken here has the merit of solving small problems very directly and using only theory which we have already developed in Chapters 1 and 2.

Suppose we have two matroids on the same ground set. Since the empty set is independent in all matroids we know that they will share at least some of their independent sets. Their *intersection* is the totality of their shared independent sets:

Definition 132 *If $M_1 = (A, \mathcal{I}_1)$ and $M_2 = (A, \mathcal{I}_2)$ are matroids on ground set A then the intersection of M_1 and M_2 , denoted $M_1 \cap M_2$, is the collection of sets $\mathcal{I}_1 \cap \mathcal{I}_2$.*

Notice that $\mathcal{I}_1 \cap \mathcal{I}_2$, in definition 132, consists of subsets of A which are simultaneously in \mathcal{I}_1 and in \mathcal{I}_2 ; independent sets themselves are *not* intersected.

Example 133 *Let M_1 and M_2 be matroids on ground set $A = \{a, b, c, d\}$ with M_1 having bases $\{a, b\}$ and $\{b, c\}$ and M_2 having bases $\{b, c\}$ and $\{c, d\}$. Specify $M_1 \cap M_2$.*

Solution Since the collection of independent sets is closed under taking subsets (Definition 21(2) in Chapter 1, section 1.2.2) we can list all the independent sets of M_1 and M_2 :

$$\begin{aligned}\mathcal{I}_1 &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}, \\ \mathcal{I}_2 &= \{\emptyset, \{b\}, \{c\}, \{d\}, \{b, c\}, \{c, d\}\}.\end{aligned}$$

Now it is an easy matter to specify the common independent sets: $M_1 \cap M_2 = \{\emptyset, \{b\}, \{c\}, \{b, c\}\}$. We can represent the intersection graphically as shown in Figure 3.1.

*This idea seems to date back to the 1970s—Nicholas Harvey attributes it to Nobuaki Tomizawa and Masao Iri (see N.J.A Harvey, “Algebraic Algorithms for Matching and Matroid Problems”, *SIAM Journal on Computing*, 39(2), 679-702, 2009).

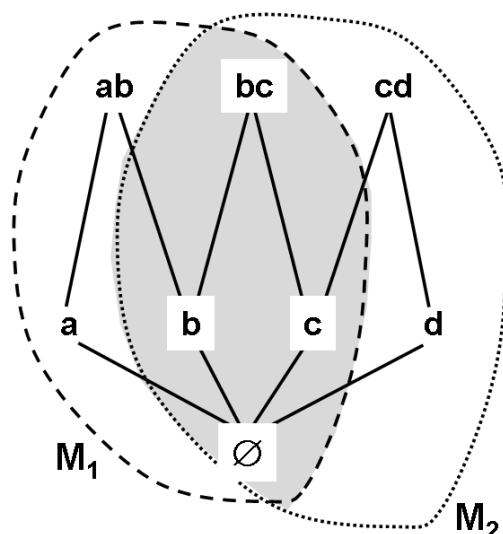


Figure 3.1: intersection of the two matroids M_1 and M_2 in Example 133.

The matroid intersection in Example 133 is easily checked to be a matroid in its own right. This is unfortunately not always the case, as the next exercise reveals:

Exercise 134 Two matroids M_1 and M_2 on the same ground set $A = \{a, b, c, d\}$ have independent sets

$$\begin{aligned}\mathcal{I}_1 &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}\} \text{ and} \\ \mathcal{I}_2 &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{b, c\}, \{c, d\}, \{a, d\}\}.\end{aligned}$$

Show that $M_1 \cap M_2$ is not a matroid.

We shall see presently that being able to solve the following optimisation problem

MAX COMMON INDEPENDENT SET

Input: Matroids M_1 and M_2 on the same ground set A

Output: A set X of maximum size which is independent in M_1 and M_2

will open up a new vista of OR problems which we can solve. But because $M_1 \cap M_2$ is not normally a matroid, we cannot hope to solve MAX COMMON INDEPENDENT SET by using the Greedy Algorithm. Indeed, we no longer even have the crucial **maximal=maximum** property of matroids. For instance in Exercise 134 we saw that $\{c\}$ was a maximal set in $M_1 \cap M_2$ but it was certainly not one of the largest sets in the intersection.

In section 2.3.3 of the last chapter we saw the Binet-Cauchy Theorem pick out the spanning trees of a graph, as nonzero determinants within a representation, $B_i(G)$, of its cycle matrix. This gives us a clue about how to solve MAX COMMON INDEPENDENT SET: we shall try to match independent sets among the columns of a representation of M_1 with those among the columns of a representation of M_2 . Before we elaborate further we shall apply this idea to revisit Exercise 134:

Example 135 Let M_1 and M_2 be the matroids of Exercise 134. Find representations of these matroids and confirm the Binet-Cauchy theorem by applying it to these two matrices.

Solution It is not too hard to find representations for these small matroids, once we spot they are both

partition matroids: $M_1 = M(\{\{a\}, \{b, c, d\}\})$ and $M_2 = M(\{\{a, c\}, \{b, d\}\})$ with representations:

$$R_1 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad R_2 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Now the Binet-Cauchy theorem evaluates the determinant of

$$R_1 \times R_2^T = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} a & b \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}$$

by summing over products of determinants of matching pairs of submatrices in R_1 and R_2 . In fact there are only two such pairs which both give nonzero determinants. For instance, columns 1 and 3 of R_1 , and the corresponding rows of R_2^T give a contribution to the Binet-Cauchy summation of

$$\det \begin{pmatrix} a & c \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \times \det_c \begin{pmatrix} a & b \\ 1 & 0 \\ 1 & 0 \end{pmatrix} = 1 \times 0 = 0.$$

Ignoring all such pairs, the theorem correctly asserts that

$$2 = \det \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} = \det \begin{pmatrix} a & b \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \times \det_b \begin{pmatrix} a & b \\ 1 & 0 \\ 0 & 1 \end{pmatrix} + \det \begin{pmatrix} a & d \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \times \det_d \begin{pmatrix} a & b \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 1 \times 1 + 1 \times 1 = 2.$$

The important point about the summation in this example is that it *identifies* precisely the subsets of $\{a, b, c, d\}$ which are independent in both matroids, that is: $\{a, b\}$ and $\{a, d\}$. However, the calculation in Example 135 finished by just giving us a numerical answer: 2. This told us that there were two independent sets of size 2 in $M_1 \cap M_2$ but not which sets these were. Worse, even that answer depended on the choice of representation.

Exercise 136 The two matroids M_1 and M_2 in Example 135 may also be represented by

$$R_1 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad R_2 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

How does this affect the summation of products in the Binet-Cauchy Theorem?

You will remember that, in the Mirsky-Perfect representation of transversal matroids, the use of indeterminates prevented cancellation of terms in determinant evaluation (see Example 116 in section 2.4.1). Suppose, for the ground set A of our matroids we define $\text{diag}(A)$ to be the matrix whose i -th diagonal element is an indeterminate representing the i -th element of A ; and whose off-diagonal elements are zero. The trick is to place this matrix in the middle of the Binet-Cauchy product $R_1 \times R_2^T$ of two representations. It has the effect of ‘physically’ labelling each square submatrix with the subset it represents. We shall refer to this ‘labelled’ product as the *Binet-Cauchy intersection*. Let us try our example again:

Example 137 Let M_1 and M_2 be the matroids of Exercise 134. Choose representations of these matroids and use the Binet-Cauchy theorem to find the maximum independent sets in $M_1 \cap M_2$.

Solution With the representations in Exercise 136 the independent sets appeared with different signs in the Binet-Cauchy Theorem and cancelled each other out. This is a good choice for showing the power of the indeterminate matrix $\text{diag}(A)$:

$$R_1 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad R_2 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

Interposing the matrix $\text{diag}(A)$ our Binet-Cauchy intersection becomes

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & c & d \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} a & 0 \\ c & b-d \end{pmatrix}.$$

Notice we have discretely dropped the row and column labelling — the $\text{diag}(A)$ matrix is doing that job now; indeed, its effect is to weight any column of the first matrix (or any row of the second, but not both) with the corresponding ground set element.

Let us apply the Binet-Cauchy Theorem yet again:

$$a(b-d) = \det \begin{pmatrix} a & 0 \\ c & b-d \end{pmatrix} = \det \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \times \det \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \det \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix} \times \det \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = ab - ad.$$

The result is a polynomial in the ground set indeterminates whose terms are exactly the maximum independent sets in $M_1 \cap M_2$.

Exercise 138 Use the indeterminate Binet-Cauchy approach to identify the maximum independent set in $M_1 \cap M_2$ for the two matroids in Example 133.

Let us formalise things as an algorithm.

Algorithm 139 (Binet-Cauchy Intersection)

```

INPUT:         instance of MAX COMMON INDEPENDENT SET problem
OUTPUT:       a polynomial enumerating all optimal solutions for this problem instance

1   D:=diag(A)
2   R1:= representation of M1; R2:= representation of M2
3   Φ := R1 × D × R2T
   // the Binet-Cauchy intersection. //
4   K := min(no. of rows of R1, no. of rows of R2)
5   optpoly := 0;
6   for i = K downto 1 while optpoly = 0 do
7       S := all pairs of subsets of {1, ..., K} of size i
8       for (U, V) ∈ S do
9           optpoly := optpoly + det Φ[U|V];
           // recall that the notation [U|V] means the submatrix of rows indexed by U and //
           // columns indexed by V. //
10      od
11  od
12  return optpoly

```

Perhaps Algorithm 139 looks more complicated than what we have just been doing! We have to allow for two facts:

1. M_1 and M_2 may have different ranks (basis sizes). $M_1 \cap M_2$ certainly cannot contain common independent sets of size greater than the minimum rank of M_1 and M_2 , hence the definition of K in step 4 of the algorithm.

2. Even if M_1 and M_2 have the same rank there may be no independent sets of this size in $M_1 \cap M_2$. If this is the case, the loop in steps 5–11 will take smaller and smaller submatrices of the Binet-Cauchy intersection corresponding to subsets of the rows of R_1 and the subsets of columns of R_2 until a nonzero determinant is found or we run out of submatrices. Even without the cost of computing with indeterminates, this is potentially very time-consuming which is why Algorithm 139 is more of a conceptual tool than a computational one. Practical solution of MAX COMMON INDEPENDENT SET is achieved by Edmonds' Algorithm (see section 3.4).

Exercise 140 Let M_1 and M_2 be matroids on the ground set $A = \{a, b, c, d\}$ with M_1 having just one basis, $\{a, b, c\}$ and M_2 similarly having just one basis, $\{a, b, d\}$. Apply Algorithm 139 to find the set of all maximal independent sets in $M_1 \cap M_2$.

Now we have assembled the necessary theory it is time to apply it to some new problems. Don't worry if, at this moment, the theory seems very complicated! In the next section we shall apply it in five different applications and each application will look almost identical—after a bit everything will begin to seem quite familiar!

3.3 Applications of matroid intersection

Our concern in this section is to intersect the various different kinds of matroid we have met, graphic, cographic, partition, transversal etc, with each other to solve various OR problems. With the Binet-Cauchy Intersection algorithm at our disposal we need hardly do more than explain each application and give an example — we have already done most of the hard work in Chapter 2 by constructing the representation matrices which are the main inputs to the algorithm. That said, it is worth noting that we have probably reached the most advanced part of this study guide. We make two observations:

1. Everything that follows in this chapter depends heavily on the Binet-Cauchy Theorem and its use in Algorithm 139. It will do no harm to revisit the previous section and also subsection 2.2.3 in Chapter 2 now or at any point as you work through the examples which you are about to meet.
2. Matrix calculations, particularly those involving determinants, are quite laborious to perform by hand. The examples we shall meet have been deliberately kept to a small scale. Even so, you should expect the exercises to take longer than those you have met previously.

3.3.1 Bipartite matching

One of the most useful applications of matroid intersection is also one of the simplest. You may recall, at the end of section 2.3.2 in Chapter 2, deriving two partition matroids from a bipartite graph whose edges connected two vertex sets S and T . In one matroid, which we called $M(\mathcal{A}(S))$, the independent sets were those subsets consisting of edges incident with different vertices of S ; in $M(\mathcal{A}(T))$, meanwhile, independent sets were subsets of edges incident with different vertices of T . Suppose the edges in some subset are simultaneously incident with different vertices in S and with different vertices in T . Then the subset is a matching! We saw in Chapter 1 that maximum matchings could not be found greedily. They can, it now transpires, be found by matroid intersection.

The actual optimisation problem was:

MIN-WEIGHT MAXIMAL MATCHING

Input:	graph G with edge set E , function $w : E \rightarrow \mathbb{R}$
Output:	a maximal matching of G of minimum weight

and Algorithm 139 does not deal with weighted edges — we will assume for now that all weights are equal, but Exercise 143 will show that including weights is not hard. Equal weights suffice at any rate to complete our solution of the Job Assignment and Committee Selection problems from section 1.3.4.

Example 141 The graph on the left in Figure 3.2 appeared in Figure 1.11 in Chapter 1. It was established that $\{W_1, W_2, W_3, W_5\}$ was a minimum-weight transversal: a cheapest choice of workers to do jobs J_1, \dots, J_4 . Find an actual matching of these workers to jobs.

Solution On the right in Figure 3.2, we have restricted the bipartite graph to include only the selected workers. We have also labelled the edges with indeterminates, with the edge from job i to worker j being labelled t_{ij} . We have a partition matroid $M(J)$ for the jobs whose representation is

$$R_J = \begin{matrix} & t_{11} & t_{12} & t_{15} & t_{22} & t_{23} & t_{31} & t_{35} & t_{45} \\ \begin{matrix} J_1 \\ J_2 \\ J_3 \\ J_4 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix},$$

and a partition matroid $M(W)$ for the workers whose representation is

$$R_W = \begin{matrix} & t_{11} & t_{12} & t_{15} & t_{22} & t_{23} & t_{31} & t_{35} & t_{45} \\ \begin{matrix} W_1 \\ W_2 \\ W_3 \\ W_5 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}.$$

Notice that the columns are ordered the same in both matrices — this is very important in order for the intersection to make sense. This must be same order as the indeterminates appear in the matrix $D = \text{diag}(t_{11}, \dots, t_{45})$. Now we derive the Binet-Cauchy intersection, the Φ matrix as in step 3 of Algorithm 139, carrying out the necessary multiplications to get:

$$\Phi = R_1 \times D \times R_2^T = \begin{pmatrix} t_{11} & t_{12} & 0 & t_{15} \\ 0 & t_{22} & t_{23} & 0 \\ t_{31} & 0 & 0 & t_{35} \\ 0 & 0 & 0 & t_{45} \end{pmatrix}.$$

It is easiest to evaluate the determinant of Φ by taking all permutation matrices, using Theorem 63 from subsection 2.2.2. There is in fact only one permutation matrix all of whose nonzero entries match nonzero entries in Φ , these entries being t_{12} , t_{23} , t_{31} , and t_{45} , so that $\det \Phi = t_{12}t_{23}t_{31}t_{45}$ and this is the matching of jobs to workers that we need: job 1 to W_2 , job 2 to W_3 , job 3 to W_1 and job 4 to W_5 .

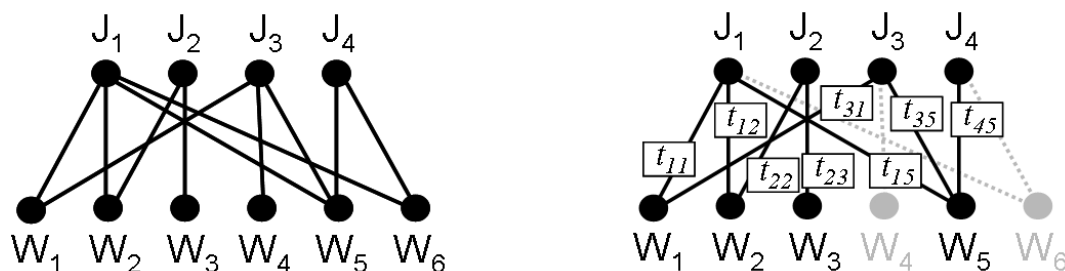


Figure 3.2: bipartite graphs for Example 141.

Exercise 142 Find all maximal matchings in the bipartite graph of Figure 3.3(a).

Exercise 143 Suppose that M is a matroid over ground set A and that the elements of A are weighted, with $a \in A$ having weight $w(a)$. In step 1 of Algorithm 139 adjust the definition of D so that the i -th diagonal entry in $\text{diag}(A)$ is now multiplied by $s^{w(a)}$, where s is a new indeterminate. Now repeat Exercise 142 but using the graph in Figure 3.3(b) which has weighted edges. What do you notice about the powers of s in the final determinant that you compute?



Figure 3.3: bipartite graphs for (a) Exercise 142 and (b) Exercise 143.

3.3.2 Graceful trees

We have looked at intersecting together two partition matroids; now let us look at an intersection involving two matroids of different types. We take an example which does not belong to OR at all but which is rather simple and appealing: we shall intersect the partition matroid which describes graceful graphs (subsection 2.3.2 in the last chapter) with the cycle matroid which describes forests in the complete graph and get ... graceful trees.

To recap on the definition: a *gracefully labelled tree* is one which, its $n + 1$ vertices being labelled $1, \dots, n + 1$ and each edge being labelled with the difference of its end vertices, has an edge with each label in $\{1, \dots, n\}$. As we suggested, all the work has already been done in Chapter 2.

Example 144 Find all gracefully labelled trees on four vertices.

Solution We will take the partition matroid which assigns edge ij of K_4 to partition set $|i - j|$, see Figure 3.4(a). Then we intersect this with the cycle matroid of K_4 . Bases in the first matroid are gracefully labelled graphs; bases in the second are spanning trees. Bases in the intersection are graceful trees. The ground set is the set of edges of K_4 and we can use indeterminate t_{ij} to represent edge ij . Let us multiply the representation matrix, R_1 , say, for the partition matroid by $D_{K_4} = \text{diag}(t_{12}, t_{23}, \dots, t_{14})$. Remembering that post-multiplying by D_{K_4} weights the columns, we get

$$R_1 \times D_{K_4} = \begin{pmatrix} t_{12} & t_{23} & t_{34} & 0 & 0 & 0 \\ 0 & 0 & 0 & t_{13} & t_{24} & 0 \\ 0 & 0 & 0 & 0 & 0 & t_{14} \end{pmatrix}.$$

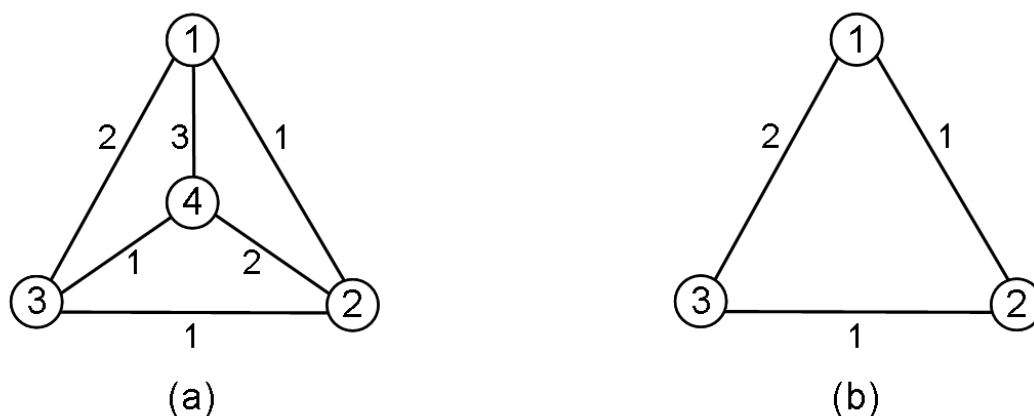


Figure 3.4: labelled complete graphs for (a) Example 144 and (b) Exercise 146.

The Binet-Cauchy intersection Φ is now formed by multiplying by the transpose of the cycle matroid representation:



$$R_2^T = B(K_4)^T = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}^T.$$

This looks a more arduous calculation than our previous examples but we must remember that each row of $R_1 D_{K_4}$ produces a weighted sum of the rows of $B(K_4)^T$: the first row says add the first three rows of $B(K_4)^T$; the second row says add the next two rows and so on. We get:

$$\Phi = \begin{pmatrix} t_{12} & t_{23} - t_{12} & t_{34} - t_{23} & -t_{34} \\ t_{13} & t_{24} & -t_{13} & -t_{24} \\ t_{14} & 0 & 0 & -t_{24} \end{pmatrix}.$$

Since (step 4 in the algorithm) $K := \min(\text{no. of rows of } R_1, \text{no. of rows of } R_2) = 3$ we will take the determinant of Φ with one column deleted — a 3×3 matrix. We get four graceful trees:

$$\text{optpoly} = -t_{14}t_{12}t_{13} + t_{14}t_{23}t_{13} - t_{14}t_{24}t_{23} + t_{14}t_{24}t_{34}.$$

The first and last terms give so-called ‘graceful labellings’ of the tree , the middle two label the tree . (We might mention, it is a notorious unsolved problem in combinatorics to decide if every tree has a graceful labelling.)

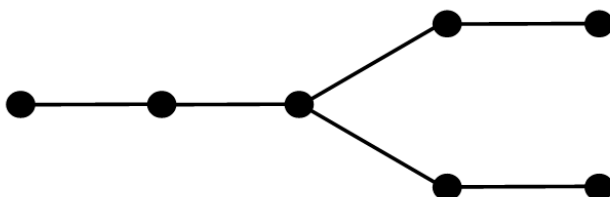


Figure 3.5: a 7-vertex tree for Exercise 145.

Exercise 145 Try to find a graceful labelling for the tree in Figure 3.5. (This has nothing to do with OR, it is meant as light relief. It is surprisingly hard!)

In Example 141 our Binet-Cauchy intersection had more columns than rows and we took determinants of all square submatrices to build our polynomial optpoly. In Example 144 we again had more columns than rows but we took only one submatrix. Why was this? Because we used the incidence matrix $B(K_4)$ as our representation of the cycle matroid and we know that the rows of this matrix are linearly dependent. We could have taken all four possible 3×3 submatrices but because of this linear dependence they would all give us the same polynomial. So we could have deleted one of the rows of $B(K_4)$ to start with, but this would have sacrificed the nice symmetrical shape that you can see in the Binet-Cauchy intersection Φ in Example 144. We exploit this symmetry at the end of the next exercise.

Exercise 146 Repeat Example 144 for K_3 , the complete graph of three vertices (Figure 3.4(b)). Infer, from the Binet-Cauchy intersections for K_3 and K_4 , what the matrix will be for K_5 .

Exercise 147 There is an example of a 5-vertex graceful tree on the far right in Figure 2.4 in Chapter 2, section 2.3.2. This will be included in the polynomial which is the determinant of any 4×4 submatrix of the Φ for K_5 you were asked to write down in Exercise 146. Find the permutation matrix whose nonzero entries match this term in the polynomial.

3.3.3 Branchings

We have just looked at intersecting a partition matroid with the cycle matroid of a graph. Now we give another (and more useful!) example of the same thing. Here are a couple of motivating problems:

Web Searching: part of the web is to be searched for keywords. The search should fan out from a given URL and should follow those links with the highest ranking (as supplied by a search engine, for instance);

Depot Location: A city location is to be chosen from which deliveries may be made. Routes from this location to all parts of the city are to be assigned, choosing those streets which generally carry least traffic.

As usual, the OR professional has much freedom as to how to interpret and tackle these problems. We shall solve them in terms of the following optimisation problem:

MIN-WEIGHT MAXIMAL BRANCHING

Input:	A directed graph with weighted arcs, the weights in \mathbb{R}
Output:	A minimum weight subset of arcs forming a forest in which no two arcs enter the same vertex.

We have not met directed graphs before but they differ from undirected graphs only in so far as each edge now has a direction, and is called an *arc* to make the distinction. For two vertices u and v the arc uv is different from the arc vu : in the first we say u is the *tail* and v is the *head* of the arc. This changes the definition for connectivity since walks in a directed graph must follow the direction of the arcs. We can still talk about the ‘underlying graph’, in which we ignore arc directions, being connected, and we can talk about a collection of arcs forming a spanning tree, also ignoring directions. But the web searching and deliveries (presumably respecting one-way streets!) referred to in our OR problems must clearly follow a sense of direction.

A *branching* in a directed graph is a subset of the arcs which forms a forest in which no two arcs enter the same vertex. If v is a vertex then we refer to the number of arcs having head v as the *indegree* of v , while the number of arcs having tail v is the *outdegree* of v . So branchings are forests in which no vertex has indegree greater than 1. It is not hard to see that each separate tree in such a forest must contain exactly one vertex of indegree zero and this is called its *root vertex*. If we can find a branching which is also a spanning tree then the single root vertex will be the location of our depot in the Depot Location problem and will be the root URL for our Web Searching problem.

The MIN-WEIGHT MAXIMAL BRANCHING problem seeks to find a branching of maximal size and, among these, to find the one with least weight. With the MIN-WEIGHT SPANNING TREE problem in Chapter 1 it was possible in a connected graph to extend any forest to a spanning tree; with branchings this may not be possible. Consider the directed graph in Figure 3.6, for example. A spanning tree in the underlying graph is given by the arcs $\{a, b, c, d, f\}$ but this fails to be a branching because arcs d and f have the same head (vertex 5). There are subsets of arcs which satisfy the head condition but fail to be trees, for instance in the cycle on the edges $\{b, c, d, e\}$ no two arcs have the same head. Such subsets must also be eliminated in our search for maximal branchings. In fact, this directed graph has no spanning branching: for the Depot Location problem (and indeed for anyone using it) this is clearly a very irritating street plan.

Exercise 148 *There are three maximum-cardinality branchings in the directed graph in Figure 3.6. One of these is $\{a, b, d, e\}$; find the other two.*

The maximum branchings you found in Exercise 148 are obviously also *maximal* but they are not the only maximal branchings. The two sets $\{b, c, d\}$ and $\{c, d, e\}$ are both maximal. So the collection of subsets which constitute branchings in a directed graph fail to satisfy maximal=maximum — they cannot be the independent sets of a matroid. Even in cases where we do have maximal=maximum we cannot guarantee that a greedy approach will work.

Exercise 149 *The directed graph in Figure 3.7 does contain spanning branchings. Suppose we try to construct a minimum-weight branching B rooted at vertex a by (1) taking $B = \{a\}$ and (2) repeatedly*

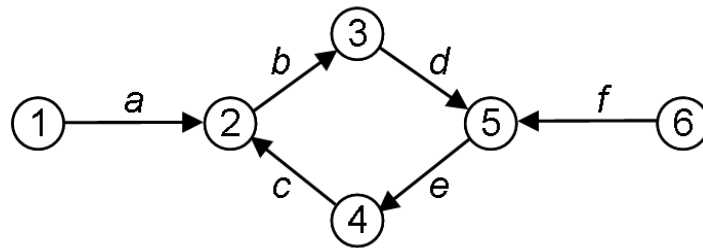


Figure 3.6: a directed graph with no spanning branching.

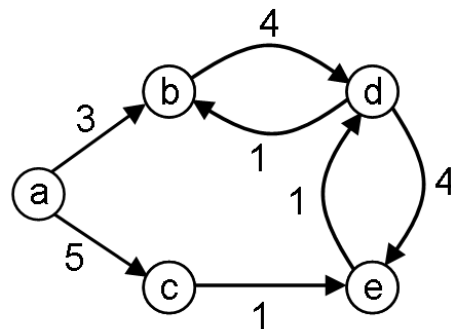


Figure 3.7: min-weight spanning branchings cannot be found greedily.

adding to B the lowest weight edge directed away from B . Show that this greedy approach fails to construct a minimum-weight branching rooted at a .

Luckily, the maximum branchings in a directed graph can be found in the intersection of two matroids. One of these is naturally the cycle matroid of the underlying graph. The other is a partition matroid:

Definition 150 Let G be a directed graph with arc set A . Let \mathcal{A} be the collection of all subsets of A in which no two arcs have the same head. Then (A, \mathcal{A}) is a matroid called the head partition matroid of the directed graph, which we will denote $M^-(G)$.

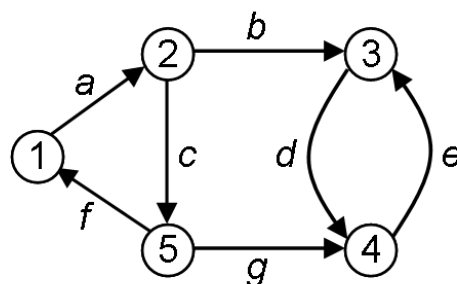


Figure 3.8: directed graph for Example 151.

Example 151 Find all the maximum branchings in the graph of Figure 3.8 which have vertex 1 as a root vertex.

Solution Following our usual procedure, we represent the two matroids which we want to intersect, in this case R_1 will represent the head partition matroid $M^-(G)$ which we have just mentioned; R_2 will represent the cycle matroid:

$$R_1 = \begin{matrix} & a & b & c & d & e & f & g \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}, \quad R_2 = \begin{matrix} & a & b & c & d & e & f & g \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 & -1 \end{pmatrix} \end{matrix}.$$

The incidence matrix, $R_2 = B(G)$, does indeed ignore the directions on the arcs: the columns for arcs d and e are identical. Taking the diagonal matrix of indeterminates to be $D = \text{diag}(a, \dots, g)$ and calculating $\Phi = R_1 \cdot D \cdot R_2^T$ we get the Binet-Cauchy intersection:

$$\begin{aligned} \Phi = R_1 \cdot D \cdot R_2^T &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & f & 0 \\ a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b & 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & d & 0 & 0 & g \\ 0 & 0 & c & 0 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 & -1 \end{pmatrix}^T \\ &= \begin{pmatrix} f & 0 & 0 & 0 & -f \\ a & -a & 0 & 0 & 0 \\ 0 & b & -b+e & -e & 0 \\ 0 & 0 & d & -d+g & -g \\ 0 & c & 0 & 0 & -c \end{pmatrix}. \end{aligned}$$

The determinant of this Φ matrix turns out to be zero. This is no surprise since we know that $B(G)$ has row rank 4 and such a *rank deficiency* will always be inherited in a matrix product: in Φ the columns are linearly dependent (*columns* since $B(G)$ is transposed in the product). Nevertheless, we see that the product of the terms on the diagonal makes a *nonzero* contribution to the determinant, namely $f(-a)(-b+e)(-d+g)(-c)$, even though any subset of edges containing f , a and c must contain a cycle and so cannot be a branching! There is no contradiction here, though: some other permutation matrices will contribute equal terms but of opposite sign to cancel out this diagonal product.

Since Φ will give $\text{optpoly} = 0$ in Algorithm 139 we must take 4×4 submatrices. Now, just as in the graceful tree calculation (see the paragraph following Exercise 145) all choices of column deletion for Φ will give the same results. Different choices of *row* deletion, on the other hand, make a big difference! Deleting row i has the same effect as deleting the row i in R_1 , the head partition matroid representation. This means disqualifying those arcs which had vertex i as their head. So we will get a determinant which lists the branchings in which vertex i has indegree zero — precisely the branchings with root vertex i . If we delete column 1 of Φ and also delete row 1 then the determinant gives branchings rooted at vertex 1: $abcd - abcg + aceg$.

Exercise 152 Delete the other columns of the matrix Φ in Example 151 and take determinants to find all maximum branchings rooted at vertices 2, 3, 4 and 5.

Exercise 153 Repeat the analysis of Example 151 but using the graph in Figure 3.6.

Exercise 154 A street network is represented schematically as shown in Figure 3.9. The weights on the arcs of this directed graph indicate average traffic loads. By using matroid intersection with an extra indeterminate, as in Exercise 143, justify the choice of vertex 3 as a siting for a depot which is optimal in terms of delivering to the other locations via the least busy streets.

If you look back to Chapter 2 to what we called the Laplacian matrix in Example 100 in section 2.3.3 you may notice a similarity between it and the Binet-Cauchy intersection for branchings. We shall return to this point shortly.

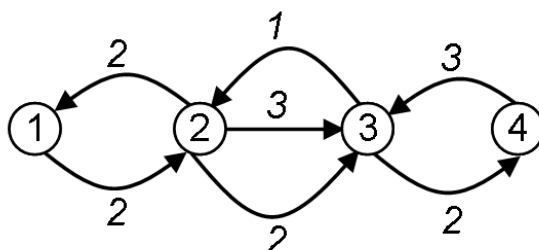


Figure 3.9: directed graph for Exercise 154.

3.3.4 Edge-disjoint spanning trees

We shall solve another new problem in this section:

Fault-tolerant Networking: It is desired to find two sets of connections in a computer network both of which allow each node to reach all others. The first network should choose the cheapest available connections; the second need not be optimised but should have no connections in common with the first.

A solution to this problem will ensure that communication can still be maintained (albeit at possibly greater cost) even if arbitrarily many communication links in the low-cost network fail.

We emphasise, as usual, that the OR professional must take many real-world factors into account in solving such a problem. We proceed immediately to model it as a combinatorial optimisation problem:

EDGE-DISJOINT SPANNING TREES

Input: Graph G with m vertices and at least $2m - 2$ edges

Output: Pair of spanning trees of G having no edges in common

Note that each spanning tree will require $m - 1$ edges so the lower bound on the number of edges is essential. Note also, that we have omitted to mention the cost of the links in this problem. We have seen that cost considerations can be included as an extra indeterminate in our Binet-Cauchy intersection.

We shall solve EDGE-DISJOINT SPANNING TREES as a matroid intersection. First though, we shall show how *not* to solve it, since it will prove to be an interesting red herring.* This red herring is the idea that intersecting one cycle matroid with another will produce two trees, one from each. Well, let us try this for a particular example.

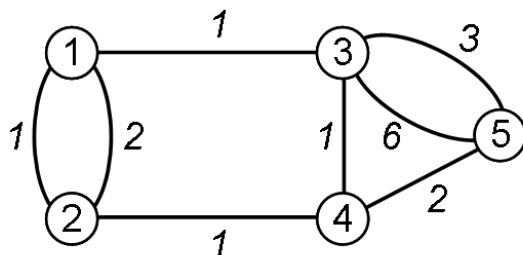


Figure 3.10: graph representing a computer network.

*The invaluable *Brewer's Dictionary of Phrase and Fable* tells us that a 'red' (smoked or salted) herring, trailed on the ground, would lead hounds astray from the scent of the fox.

Let us denote by G the computer network graph in Figure 3.10. It has incidence matrix:

$$B(G) = \begin{matrix} & t_{12} & t'_{12} & t_{13} & t_{24} & t_{34} & t_{35} & t'_{35} & t_{45} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{matrix}.$$

Note that we have chosen to reverse our usual order of the -1 and 1 in each column; we are free to do this and it will make things simpler later on. We have labelled the edges to represent the vertex pairs they connect (it does not matter at this stage which edge joining vertices 1 and 2 is t_{12} and which is t'_{12}). Now we propose to take the Binet-Cauchy intersection of $B(G)$ with itself. We will recall our remarks directly following Exercise 145 — $B(G)$ must have rank 4 so we will be looking for 4×4 submatrices in Algorithm 139 and it makes no difference which row we delete. So we will delete the first row in advance to get

$$B_1(G) = \begin{matrix} & t_{12} & t'_{12} & t_{13} & t_{24} & t_{34} & t_{35} & t'_{35} & t_{45} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{matrix}.$$

The Binet-Cauchy intersection for this reduced matrix, taken with itself, is:

$$\Phi = \begin{pmatrix} t_{12} + t'_{12} + t_{24} & 0 & -t_{24} & 0 \\ 0 & t_{13} + t_{34} + t_{35} + t'_{35} & -t_{34} & -t_{35} - t'_{35} \\ -t_{24} & -t_{34} & t_{24} + t_{34} + t_{45} & -t_{45} \\ 0 & -t_{35} - t'_{35} & -t_{45} & t_{35} + t'_{35} + t_{45} \end{pmatrix}$$

and the determinant of this matrix which is the outcome of Algorithm 139 has 31 terms!*

$$\begin{aligned} \text{optpoly} = & t_{12}t_{13}t_{45}t_{35} + t_{12}t_{13}t_{24}t_{35} + t_{12}t_{13}t_{24}t'_{35} + t_{12}t_{13}t_{24}t_{45} + t_{12}t_{13}t_{34}t_{35} \\ & + t_{12}t_{13}t_{34}t'_{35} + t_{12}t_{13}t_{34}t_{45} + t_{12}t_{13}t_{45}t'_{35} + t_{12}t_{34}t_{24}t_{35} + t_{12}t_{34}t_{24}t'_{35} + t_{12}t_{34}t_{24}t_{45} \\ & + t_{12}t_{35}t_{24}t_{45} + t_{12}t'_{35}t_{24}t_{45} + t'_{12}t_{13}t_{45}t_{35} + t'_{12}t_{13}t_{24}t_{35} + t'_{12}t_{13}t_{24}t'_{35} + t'_{12}t_{13}t_{24}t_{45} \\ & + t'_{12}t_{13}t_{34}t_{35} + t'_{12}t_{13}t_{34}t'_{35} + t'_{12}t_{13}t_{34}t_{45} + t'_{12}t_{13}t_{45}t'_{35} + t'_{12}t_{34}t_{24}t_{35} + t'_{12}t_{34}t_{24}t'_{35} \\ & + t'_{12}t_{34}t_{24}t_{45} + t'_{12}t_{35}t_{24}t_{45} + t'_{12}t'_{35}t_{24}t_{45} + t_{24}t_{13}t_{45}t_{35} + t_{24}t_{13}t_{34}t_{35} + t_{24}t_{13}t_{34}t'_{35} \\ & + t_{24}t_{13}t_{34}t_{45} + t_{24}t_{13}t_{45}t'_{35}. \end{aligned}$$

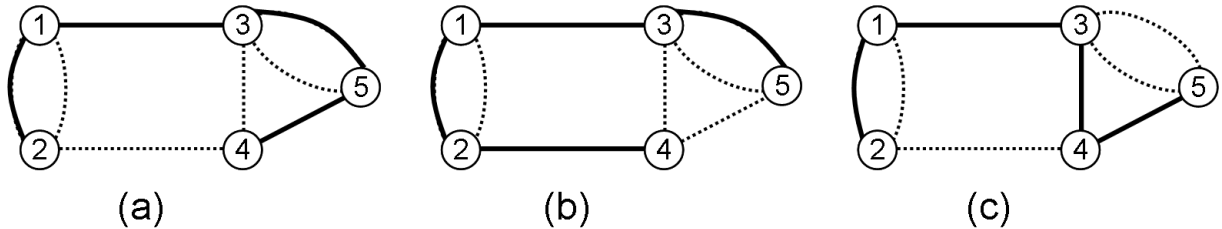


Figure 3.11: spanning trees in the graph of Figure 3.10 (edge weights not included).

We can inspect the terms of this polynomial to see what we have achieved. The first one, $t_{12}t_{13}t_{45}t_{35}$, works well — it corresponds to the spanning tree shown as bold edges in Figure 3.11(a); you will see that the

*This example was worked out on a computer — you will not be expected to carry out such laborious calculations by hand, either in the exercises or in your final examination!

remaining, dotted, edges also form a spanning tree, so that the pair of trees solves EDGE-DISJOINT SPANNING TREES. However, the second tree, $t_{12}t_{13}t_{24}t_{35}$, shown in bold in Figure 3.11(b) does *not* solve our problem! This is because its edges form a *cutset* in the graph: deleting these edges separate the vertices of the graph into two connected components having no edges between them. If the edges 13 and 24, in particular, were network links which failed then communication with network nodes 3, 4 and 5 would no longer be possible for nodes 1 and 2.

Exercise 155 Which term in the polynomial *optpoly* corresponds to Figure 3.11(c) and does this spanning tree give a solution to EDGE-DISJOINT SPANNING TREES?

Exercise 156 Although we do not, in principle, give values to indeterminates, we can experiment with setting to 1 all the t_{ij} in the matrix Φ above (so that the entry in the first row and column will become $1 + 1 + 1 = 3$, for example). Write down the matrix which results and find its determinant. In view of the Matrix Tree Theorem (Theorem 99 in Chapter 2, section 2.3.3) what do you deduce about the polynomial *optpoly*?

What we have done, we realise, is to intersect the cycle matroid of our graph G with itself and get ... just the cycle matroid! The Binet-Cauchy intersection counts maximum independent sets in the matroid intersection, which in this case is still just the spanning trees of G . Intersection has achieved nothing!

Luckily we have a matroid which specifically avoids cut sets — the cocycle matroid. Intersecting with this matroid is what we need to do.

Example 157 Find a representation of the cocycle matroid $M^*(G)$ of the graph G of Figure 3.10. Hence find the intersection of this matroid with $M(G)$ and all solutions to EDGE-DISJOINT SPANNING TREES for this graph.

Solution We shall reorder the columns of the matrix $B_1(G)$ from earlier and perform elementary row operations to get a standard representation:

$$\begin{array}{cccccccc} & t_{12} & t_{13} & t_{34} & t_{45} & t'_{12} & t_{24} & t_{35} & t'_{35} \\ \begin{array}{l} 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} & \xrightarrow{\begin{array}{l} R2+R3+R4 \\ R3+R4 \end{array}} & \begin{array}{l} 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

(Our earlier choice of how to order the +1's and -1's in each column was deliberately aimed at making this reduction as easy as possible.) Let us denote this latter matrix R_M . We learnt in Chapter 2 (Theorem 109) that when matroid M has a standard representation $R_M = [I_r | X]$ its dual M^* has representation $R_M^* = [X^T | I_{n-r}]$, n being the size of the ground set. So the cocycle matroid of G in this case has representation

$$R_M^* = \begin{pmatrix} t_{12} & t_{13} & t_{34} & t_{45} & t'_{12} & t_{24} & t_{35} & t'_{35} \\ -1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We are ready for our intersection! We calculate $R_M \times \text{diag}(t_{12}, t_{13}, t_{34}, t_{45}, t'_{12}, t_{24}, t_{35}, t'_{35}) \times R_M^{*T}$ to get

$$\Phi = \begin{pmatrix} t_{12} + t'_{12} & -t_{12} - t_{24} & 0 & 0 \\ 0 & t_{13} + t_{24} & 0 & 0 \\ 0 & t_{34} + t_{24} & t_{34} + t_{35} & t_{34} + t'_{35} \\ 0 & 0 & t_{45} + t_{35} & t_{45} + t'_{35} \end{pmatrix},$$

with determinant $(t_{12} + t'_{12})(t_{13} + t_{24})(t_{34}t'_{35} + t_{45}t_{35} - t_{34}t_{35} - t_{45}t'_{35})$, which expands to give 16 terms. One of these is $t_{12}t_{13}t_{45}t_{35}$: the spanning tree shown in Figure 3.11(a). In our matroid intersection the bold edges in Figure 3.11(a) now represent simultaneously a spanning tree and a maximal non-cut; since it does not contain a cut set of edges, the remaining, dotted, edges must contain a spanning tree.

Exercise 158 Refer back to Figure 3.10 and make a note of the edge weights. Now inspect the different terms in the expansion of $\det \Phi$ in Example 157 to find a minimum weight spanning tree, thus completely solving the Fault-tolerant Networking problem.

Exercise 159 By incorporating suitable powers of an additional indeterminate s in the matrix

$$\text{diag}(t_{12}, t_{13}, t_{34}, t_{45}, t'_{12}, t_{24}, t_{35}, t'_{35})$$

of Example 157, exhibit the minimum cost spanning trees in Figure 3.10 as terms in the expansion of $\det \Phi$. [Hint: remind yourself of the answers to exercises 143 and 154.]

3.3.5 Simultaneous transversals

Our last application of intersecting two matroids amounts to a generalisation of bipartite matching. In section 3.3.1 we found the intersection of two partition matroids; we will now intersect general transversal matroids. Again, the hard work was done in Chapter 2 in ensuring that we can produce representations of transversal matroids.

We will solve the following problems:

Room Booking: Suppose that n professors P_1, \dots, P_n are to teach their courses on a particular day. There are n teaching rooms R_1, \dots, R_n and teaching takes place in k timetabled hours, T_1, \dots, T_k . It is required to book each a room for each professor for an hour when both professor and room are available.*

Radio Frequency Allocation: It is required to allocate one of n radio frequencies f_1, \dots, f_n to each of n mobile phones P_1, \dots, P_n located within a certain region. There are k radio masts M_1, \dots, M_k . Certain frequencies are unused and available at each mast and each mobile phone is within range of some of the masts but not others. No two masts may use the same frequency within the region and each mast can only broadcast on one frequency at a given time.

These problems are both partially solved as instances of the following optimisation problem:

MAXIMAL SIMULTANEOUS TRANSVERSAL

Input: Ground set A and collections \mathcal{A}_1 and \mathcal{A}_2 of subsets of A

Output: Maximal set X which is simultaneously a transversal for \mathcal{A}_1 and \mathcal{A}_2

They are only *partially* solved because, as with the Job Assignment problem, each transversal requires a corresponding matching in order to specify an allocation (of rooms or radio frequencies or whatever). We shall give an example of solving the Radio Frequency Allocation problem.

Example 160 In Figure 3.12, four mobile phones a, b, c , and d are shown in the vicinity of six radio masts. Four radio frequencies are being used but only some of these are available at each mast. Find a set of masts which may allocate available frequencies to phones which are near them, so that every phone is allocated a frequency and no frequency is used more than once in the region.

Solution We can produce two transversal matroids, one matching phones against masts and the other matching frequencies against masts. Each matroid has, as its ground set, the set of masts. Since no frequencies are available at mast 6 we can remove this from our analysis, leaving $\{M_1, \dots, M_5\}$ as our ground set. The phone and frequency subset collections, denoted \mathcal{A}_P and \mathcal{A}_f , respectively, are:

$$\mathcal{A}_P = \begin{array}{c|cccc} \text{Phone:} & a & b & c & d \\ \hline \text{Near masts:} & 2,3 & 5 & 1,2 & 3,4,5 \end{array} \quad \mathcal{A}_f = \begin{array}{c|cccc} \text{Frequency:} & f_1 & f_2 & f_3 & f_4 \\ \hline \text{Available at masts:} & 1,2,5 & 2,5 & 4 & 2,3 \end{array}$$

These may be represented by their incidence structures which, as we learnt in Chapter 2, will coincide with the structure of the Piff-Welsh representation of the transversal matroids. We call these matroids $M(P)$ and

*This application of matroid intersection may be found in R. J. Wilson's *Introduction to Graph Theory*, see Appendix A.

$M(f)$, respectively, and give their representations $R_{M(P)}$ and $R_{M(f)}$ directly — they are produced according to the procedure given in section 2.4.2 and you are encouraged to reproduce them yourself as an exercise:

$$R_{M(P)} = \begin{matrix} & M_1 & M_2 & M_3 & M_4 & M_5 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{matrix} \quad R_{M(f)} = \begin{matrix} & M_1 & M_2 & M_3 & M_4 & M_5 \\ \begin{matrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{matrix}.$$

(If you do not have time to work out these representations for yourself you should at least note that the -1 in the final column of $R_{M(f)}$ is not a misprint! It cannot be $+1$ because otherwise columns 2, 3 and 5 would become linearly dependent contradicting the fact that masts M_2 , M_3 and M_5 form a partial transversal for the frequency availabilities.)

The Binet-Cauchy intersection is given by

$$\Phi = R_{M(P)} \times \text{diag}(M_1, \dots, M_5) \times R_{M(f)}^T = \begin{pmatrix} M_2 & M_2 & 0 & M_2 + M_3 \\ M_5 & -M_5 & 0 & 0 \\ M_1 + M_2 & M_2 & 0 & M_2 \\ M_5 & -M_5 & M_4 & M_3 \end{pmatrix},$$

which has determinant

$$\det \Phi = -2M_2M_3M_4M_5 - M_1M_2M_4M_5 - M_1M_3M_4M_5.$$

In this small example it is easy to see that, for instance, the first term indicates that there are two ways in which M_2 , M_3 , M_4 and M_5 may allocate available frequencies to phones a , b , c and d . Thus, M_3 allocates f_4 to a and M_4 allocates f_3 to d , while M_2 and M_5 allocate f_1 and f_2 to b and c in either order. In larger examples, this final allocation requires two bipartite matchings, just as in section 3.3.1 we used bipartite matching to complete the Job Assignment problem.

Exercise 161 A university timetables its lectures in one hour slots between 0900 and 1800, with an unused hour from 1200-1300 for lunch:

Slot:	T_1	T_2	T_3	lunch	T_4	T_5	T_6	T_7	T_8
Time:	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18

Five professors are to teach on a particular day and their availability is given as a collection of subsets \mathcal{A}_P shown:

Professor:	P_1	P_2	P_3	P_4	P_5
Availability:	T_1-T_3	T_4-T_6	T_7-T_8	T_2-T_3	T_2-T_5

and there are five rooms in which they can teach, whose availability is given as the collection of subsets \mathcal{A}_R :

Room:	R_1	R_2	R_3	R_4	R_5
Availability:	T_1	T_3-T_4	T_1, T_7-T_8	T_1-T_3	T_5-T_8

The transversal matroids $M(\mathcal{A}_P)$ and $M(\mathcal{A}_R)$ have representations

$$R_{M(P)} = \begin{matrix} & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \\ \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

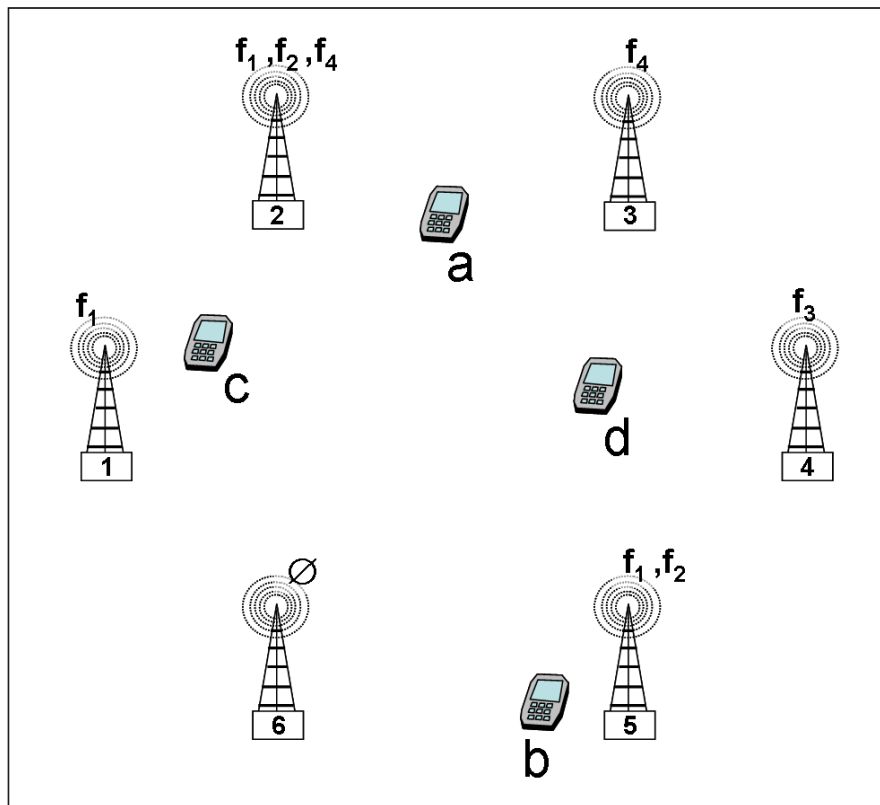


Figure 3.12: mobile phone network for Example 160.

and

$$R_{M(R)} = \begin{matrix} & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \\ \begin{matrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 \end{pmatrix} \end{matrix}$$

find all simultaneous transversals and hence find a choice of professors and rooms that are simultaneously available.

3.4 Beyond matroids

It is natural to ask whether we can solve even more optimisation problems by intersecting together three or more matroids. This turns out to be a question about which problems can and which, apparently, cannot be solved efficiently. Since this is obviously of great importance to OR we devote some time to it.

3.4.1 Good characterisations and hard problems

Chvatál's Theorem (Exercise 39 at the end of section 1.3.3 in Chapter 1) gave a sufficient condition for a graph to contain a cycle which visits every vertex exactly once. The problem of finding such a *Hamilton cycle* is closely related to the Travelling Salesman Problem or TSP, in which one is required to make a round trip of a set of cities by the shortest possible route. We saw however, (in Exercise 39), that Chvatál's Theorem was not a *necessary* condition—the graph in question failed the condition but nevertheless

contained a Hamilton cycle.

In fact, no condition is known which is both necessary *and* sufficient for a graph to be *Hamiltonian*, that is, for the graph to possess a Hamilton cycle. This assertion needs some clarification: we mean that no general test is known, which is not just a restatement of the definition of a Hamilton cycle, whereby it can be decided if any given graph is Hamiltonian. To be more precise we have to discuss computational complexity and the issue of polynomial versus exponential-time algorithms. You met these topics in the unit *Software Engineering, Algorithm Design and Analysis* and we shall open Chapter 4 by covering them in greater depth.

Another way of expressing the difficulty of finding Hamilton cycles is to say that no *good characterisation* of Hamiltonian graphs is known. The idea of a good characterisation is due to Jack Edmonds: for a given property P , of graphs, say, we want to be able to exhibit either a *certificate* to confirm that property P holds or a certificate to confirm that property P does *not* hold. For instance, a certificate to confirm that a graph is not bipartite would be an odd-length cycle, while a certificate that it is bipartite would be a partition of the vertices into sets S and T containing no internal edges. Euler's Theorem (Exercise 38 in section 1.3.3) gives a good characterisation of graphs which have an Eulerian trail: a listing of the vertex degrees provides a certificate either way, since having at most two odd degrees is necessary and sufficient for this property.

It is easy to produce *potential* certificates for *non-Hamiltonicity* of a graph G , that is, absence of a Hamilton cycle. For instance, if the graph in Figure 3.13(a) appears as a proper subgraph* of G , and vertices u and v have degree 2 in G then no cycle can pass both u and v except the 4-cycle that constitutes the subgraph itself. However, this certificate for non-Hamiltonicity creates *false negatives* since there are graphs which contain no such subgraph and yet are non-Hamiltonian, for example the famous *Petersen graph* in Figure 3.13(b). In other words, absence of the subgraph in Figure 3.13(a) is necessary but not sufficient for Hamiltonicity.

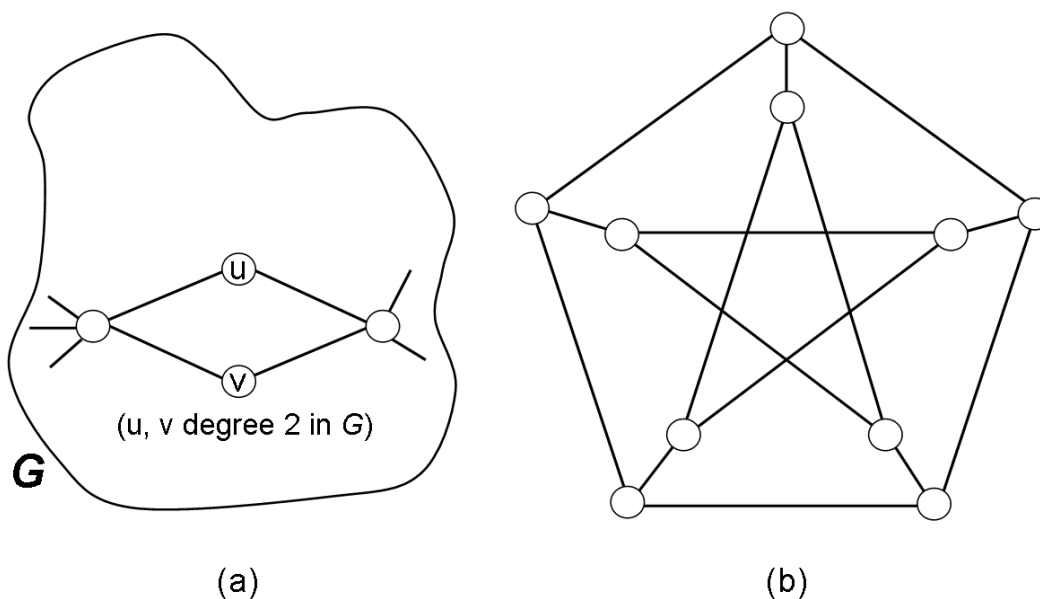


Figure 3.13: (a) a potential certificate for non-Hamiltonicity (b) the Petersen graph which eludes this certificate.

Chvátal's Theorem, on the other hand, although it is one of the strongest conditions known for existence of Hamilton cycles, cannot provide a reliable certificate for non-Hamiltonicity: we saw in Exercise 39 that a graph with a degree sequence failing to meet the condition could still have a Hamilton cycle: a *false positive*.

*The word 'proper' in mathematics usually means 'not consisting of everything'. Thus a proper subset of S is one which omits at least some elements of S ; a proper subgraph of G is one which omits at least some edges or vertices of G .

Exercise 162 A variant of Euler's Theorem (Exercise 38) says that a graph G has an Euler tour, that is a closed walk which traverses every edge of G , if and only if every vertex has even degree.

1. Show that this version of Euler's Theorem follows from the one in Exercise 38;
2. Explain how this provides a good characterisation of Eulerian graphs, i.e. those possessing an Euler tour.

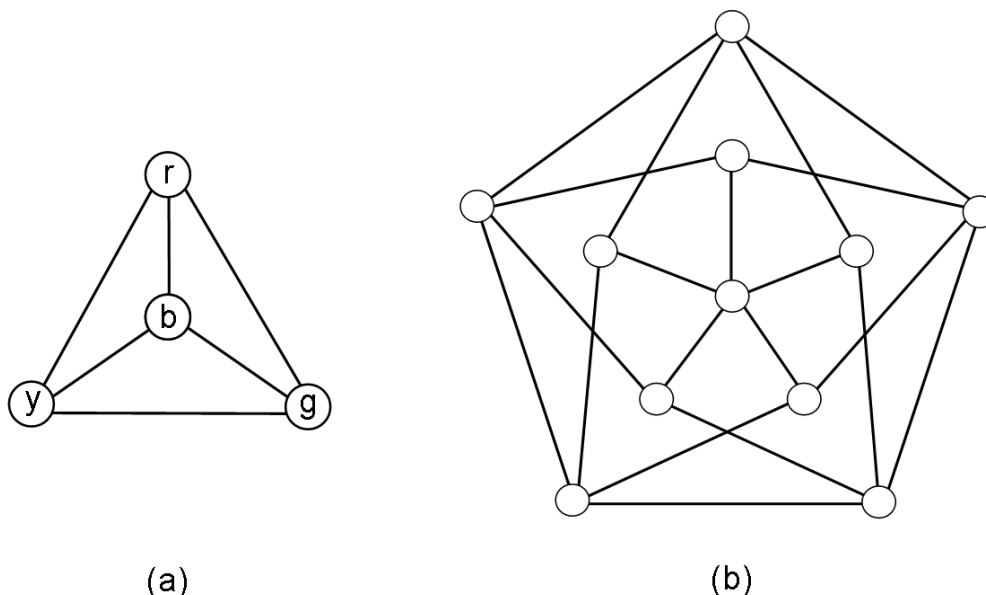


Figure 3.14: (a) a four-coloured K_4 : a potential certificate for non-3-colourability (b) the Grötzsch graph which eludes this certificate.

Exercise 163 A 3-colouring of a graph G is a partition of the vertices of G into three sets R , S and T so that no edges of G connect vertices in the same partition set. In other words, we can colour the vertices with three colours so that no vertices of the same colour are joined by an edge. Given that the Grötzsch graph in Figure 3.14(b) is not 3-colourable discuss the adequacy of the complete graph on four vertices K_4 as a certificate for non-3-colourability.

Having a good characterisation for Hamiltonicity does not automatically give us a fast algorithm for finding Hamilton cycles! However, having a good characterisation is usually an indication that it is worth looking for a fast algorithm. Indeed, the only current example of a natural problem which has a good characterisation but no fast algorithm is the problem of finding a non-trivial factor for a given integer N .*

3.4.2 Three-matroid intersection

Do the Binet-Cauchy intersections which we have been studying in this chapter provide good characterisations for properties? For instance, for the Room Booking problem, we are asking if a simultaneous transversal exists for two subset collections of size, say, n . Any such transversal is immediately a certificate to confirm that the answer is yes. The Binet-Cauchy intersection is an $n \times n$ matrix Φ whose determinant lists such transversals. Now $\det \Phi$ will be very difficult to evaluate computationally — we should emphasise again that our use of indeterminates for describing matroid intersection has bought clarity at the expense of practicality — but if we can find a non-zero vector of

*Any factor of N other than 1 or N itself obviously certifies the existence of a factor. A certificate for *not* having a factor was discovered in 1975 by Vaughan Pratt, who also introduced the term 'certificate'.

weights $u = (a_1, \dots, a_n)$ for which $\Phi \times u^T = 0$ then, together, the pair (Φ, u) is a certificate proving that no simultaneous transversal exists. So we do get a good characterisation.

Now confirming that matroid intersections provide good characterisations is not particularly useful because there are fast algorithms, not relying on matroid representation, to solve these problems; notably algorithms due to Jack Edmonds and Eugene Lawler. And we can argue that, for a property which may be checked quickly, the problem instance itself — the graph or the subset collections or whatever — is as good a certificate as any both for having a solution and for not having one. However, we can ask hopefully whether problems posed as the intersection of three or more matroids may also produce certificates in terms of matrices.

The answer to this question unfortunately appears to be no. Here is a modification of our room booking problem:

Two-party Room Booking: Suppose that n demonstrators P_1, \dots, P_n are to do organic chemistry experiments with n groups of students on a particular day. There are n teaching laboratories L_1, \dots, L_n and teaching takes place in k timetabled hours, T_1, \dots, T_k . It is required to book a laboratory for each demonstrator to teach one of the groups for an hour when the demonstrator and the student group are both free and the laboratory is available. No demonstrator will consent to teach twice and no student group will consent to having two lessons! Also no laboratory may be used twice because the experiments make such a terrible mess.

We will not bother to formulate an optimisation problem which solves this OR problem, since it is so similar to the original Room Booking. It should also be clear that a solution to the problem will be an independent set of size n that lies in the intersection of three transversal matroids over the ground set T of timetable hours. Unfortunately, no fast algorithm is known for solving this problem, so it is unlikely that a good characterisation can be found. Correspondingly, we have no three-way analogue of the Binet-Cauchy Theorem by which to produce an intersection matrix whose determinant might evaluate three-way simultaneous transversals.

Hamiltonicity of graphs provides an even more compelling example of the difficulty of intersecting three matroids. The problem is usually presented in terms of directed graphs, for which the matroids are easy to specify:

DIRECTED $s - t$ HAMILTON PATH

Input:	A directed graph G , vertices s and t of G
Output:	A directed path in G starts at s , ends at t and visits each vertex exactly once

Exercise 164 Show that if we could solve DIRECTED $s - t$ HAMILTON PATH then we could also find Hamilton cycles in undirected graphs (the concern of Chvátal's Theorem). Illustrate your answer by finding a Hamilton cycle in the graph of Figure 3.15.* [Hint: replace every edge by two arcs with opposite directions]

In section 3.3.3 we formulated the MIN-WEIGHT MAXIMAL BRANCHING problem as the intersection of the cycle matroid $M(G)$ of G and the head partition matroid $M^-(G)$ whose partition sets were the arcs entering each vertex. Now we just add a tail partition matroid M^+ whose partition sets are the arcs leaving each vertex. A branching in which every vertex has outdegree at most 1 must be a directed path; if it is a spanning branching it is a Hamilton path.

It might almost appear that the two matrices M^- and M^+ can solve DIRECTED $s - t$ HAMILTON PATH on their own! Figure 3.16(a) shows how independence in both these matroids rules out the two obvious failures to get a Hamilton path, so it seems as though we might be nearly there...

Example 165 Find a Hamilton path from vertex 5 to vertex 2 in the graph G of Figure 3.16(b).

*This graph appears in Appendix III of J.A. Bondy and U.S.R. Murty's *Graph Theory with Applications*, see Appendix A. It is observed that it is the smallest graph in which all vertices have degree 3 and which has no symmetries. You might care to try constructing a so-called 3-regular graph at random on 10 vertices. You will find you can always redraw it so that it displays some symmetry.

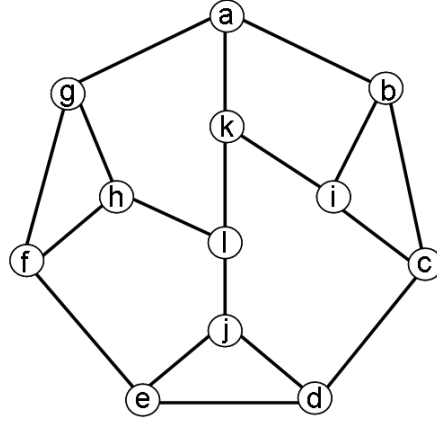


Figure 3.15: graph for Exercise 164.

Solution The head partition matroid $M^-(G)$ and tail partition matroid $M^+(G)$ have representations:

$$R^- = \begin{matrix} & t_{12} & t_{23} & t_{25} & t_{37} & t_{41} & t_{43} & t_{54} & t_{61} & t_{63} & t_{65} & t_{76} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}$$

and

$$R^+ = \begin{matrix} & t_{12} & t_{23} & t_{25} & t_{37} & t_{41} & t_{43} & t_{54} & t_{61} & t_{63} & t_{65} & t_{76} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

The Binet-Cauchy intersection is:

$$\Phi = R^+ \times \text{diag}(t_{12}, t_{23}, \dots, t_{76}) \times R^- = \begin{pmatrix} 0 & t_{12} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{23} & 0 & t_{25} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & t_{37} \\ t_{41} & 0 & t_{43} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_{54} & 0 & 0 & 0 \\ t_{61} & 0 & t_{63} & 0 & t_{65} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_{76} & 0 \end{pmatrix}.$$

We recall from our branching intersection that we must remove the column of this matrix corresponding to the root of the branching, which must have indegree zero: this will be vertex 5. By analogy we must remove the row corresponding to the vertex which is to end our Hamilton path: row 2. Finally we get our

determinant:

$$\det \Phi[\{1, 3, 4, 5, 6, 7\} | \{1, 2, 3, 4, 6, 7\}] = \det \begin{pmatrix} 0 & t_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_{37} \\ t_{41} & 0 & t_{43} & 0 & 0 & 0 \\ 0 & 0 & 0 & t_{54} & 0 & 0 \\ t_{61} & 0 & t_{63} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_{76} & 0 \end{pmatrix} = t_{12}t_{37}t_{54}t_{76}(t_{41}t_{63} - t_{43}t_{61}).$$

Success! The term $-t_{12}t_{37}t_{54}t_{76}t_{43}t_{61}$ is a Hamilton path from 5 to 2. But what is the other term?

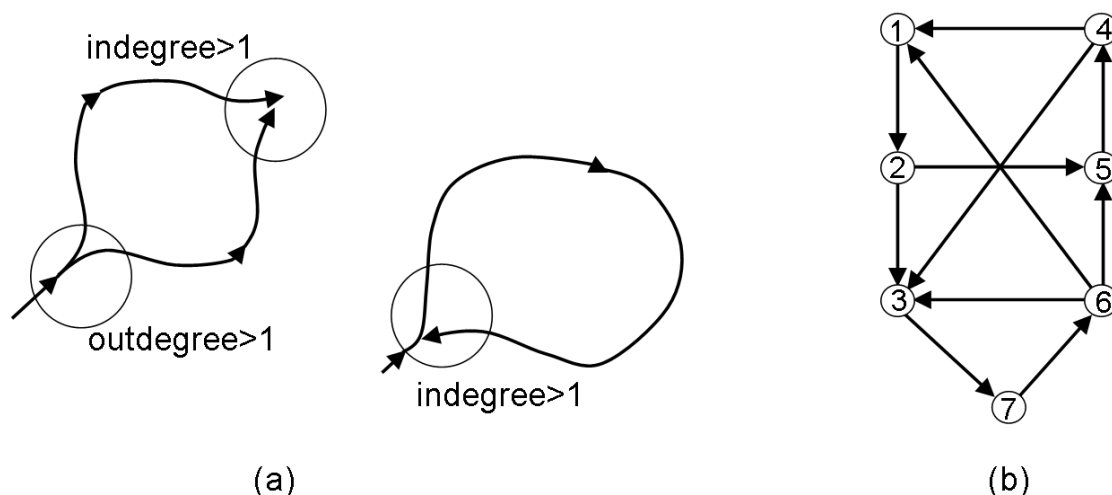


Figure 3.16: (a) violations of the partition matroids M^- and M^+ ; (b) a directed graph for Example 165.

Exercise 166 *What is the other term?*

The answer to Exercise 166 dashes our hopes of producing a certificate for absence of $s - t$ Hamilton paths. We can find terms in our determinants which correspond to subgraphs containing isolated cycles. These cycles are certainly independent in both the partition matroids: all vertices have indegree and outdegree 1; neither obstacle from Figure 3.16(a) is involved. And perhaps all terms are of this form — in this case there is no Hamilton path but we cannot produce a matrix with zero determinant. We are in the same position as we were when we tried to use a K_4 subgraph as a certificate of non-3-colourability, our certificate can lead to false negatives.

Of course, if we could only intersect the cycle matroid with our two partition matroids it would eliminate exactly the terms in $\det \Phi$ which contain cycles. But, as was the case with three transversal matroids, there appears to be no way of incorporating a third matrix into our calculation.

Chapter 4

Linear and integer linear programming

4.1 Introduction

The study of matroids has introduced us to many of the essential concepts and techniques of OR within a unified and mathematically elegant framework. At the end of Chapter 2 and again at the end of the last chapter we found ourselves reflecting on the limitations of this framework: if a matroid was not representable an ad-hoc implementation would be required for the theoretical concept of an oracle, in order to operate the greedy algorithm; if a problem required the intersection of more than two matroids we had no effective means of finding maximum common independent sets. Naturally enough, experts in OR and computer science and mathematics have explored these difficulties in great depth and in the context of great advances in the theory of computation and combinatorial mathematics.

The broader picture for combinatorial optimisation was established in the late 1970s and early 1980s when a new understanding of the limits of computing coincided with breakthroughs in the geometry of linear optimisation. It is this broader picture which we want to portray in this chapter. We shall start by describing the ideas of algorithmic and computational complexity. You have met these ideas in *Software Engineering, Algorithm Design and Analysis*; here we want to pursue them in more depth and link them more closely to problems in OR and combinatorics. The main theme is the concept of **NP**-completeness and it is this concept which allows us to make a formal distinction between those linear optimisation problems which may be effectively solved and those which, at least to the best of our belief, cannot. We thus move on to describe and distinguish between linear programming and its specialisation to what is apparently inherently harder: integer linear programming. Linear programming identifies optimal solutions as the vertices of convex n -dimensional bodies called polyhedra. In some cases these vertices can be guaranteed to have integer values and then integer linear programming can efficiently solve **NP**-complete problems.

4.2 Complexity of algorithms and problems

When we invent a method for solving some optimisation problem we must eventually implement the method algorithmically if it is to be applied to problem instances of a realistic size. There is a well-developed approach to assessing how effective the algorithm turns out to be; this concentrates on the long-term behaviour of the algorithm as problem instances get larger and larger — it takes an *asymptotic* view of algorithm speed, blurring the distinction between speeds which differ by a constant factor in order to focus on speeds which differ on a logarithm scale. At the extreme this leads us to what appears to be a fundamental divide in algorithm design, between those algorithms whose running time is a polynomial function of input size and those whose running time is exponential. The question then arises: is this more than a property of individual algorithms? Are there problems for which no polynomial algorithms exist? The study of these questions is the domain of computational, as opposed to algorithmic, complexity.

4.2.1 Algorithmic complexity

Here is a made-up problem:

Take Five: *In a list c of n integers, $n \geq 5$, find the maximum value of a sum of five elements.*

Here are two different algorithms which solve it:

Algorithm 167 (Take Five (I))

INPUT: list c of $n \geq 5$ integers
OUTPUT: maximum possible sum of five entries of L

```

1  best :=  $-\infty$ 
2  for  $p = 1$  to  $n - 4$  do
3      for  $q = p + 1$  to  $n - 3$  do
4          for  $r = q + 1$  to  $n - 2$  do
5              for  $s = r + 1$  to  $n - 1$  do
6                  for  $t = s + 1$  to  $n$  do
7                       $sum5 := c[p] + c[q] + c[r] + c[s] + c[t]$ ;
8                      if  $sum5 > best$  then  $best := sum5$  fi
9  od od od od od
12 return best

```

Algorithm 168 (Take Five (II))

INPUT: list c of $n \geq 5$ integers
OUTPUT: maximum possible sum of five entries of c

```

1  best :=  $-\infty$ 
2  for all  $n$ -bit binary strings  $x$  do
3      if  $x$  has exactly five 1's then
4           $sum5 := c \times x^T$ ; // selects and adds the elements of  $c$  indexed by 1's in  $x$  //
5          if  $sum5 > best$  then  $best := sum5$  fi
6      fi
7  od
12 return best

```

Take Five (I) does the same as (II) except that it guarantees in advance that it is choosing a subset of exactly five elements.

Which algorithm runs most quickly? (I) has nested **for** loops and these are a typical way in which algorithmic complexity increases:

One loop: n steps — *linear* time complexity
Two loops: n steps repeated n times — n^2 or *quadratic* time complexity
Three loops: n^2 steps repeated n times — n^3 or *cubic* time complexity
...

An important idea which you met in the unit *Software Engineering: Algorithm Design and Analysis* was that an algorithm which has three nested loops in one place and two nested loops elsewhere still has cubic time complexity because the time taken by the two nested loops is unimportant compared with the three nested loops. We write $n^3 + n^2 = O(n^3)$ to express this idea, the O standing for *order of growth*: we take the highest power of n and ignore its coefficient. In the same way, the fact that in Take Five (I) each level of nesting runs a shorter loop does not alter the fact that this is an n^5 algorithm. It has to locate exactly $\binom{n}{5} = n!/(n-5)!5!$ subsets of L in all, giving a time complexity of

$$\begin{aligned} \frac{1}{5!} n(n-1)(n-2)(n-3)(n-4) &= \frac{1}{120} n^5 - \frac{1}{12} n^4 + \frac{7}{24} n^3 - \frac{5}{12} n^2 + \frac{1}{5} n \\ &= O(n^5) \text{ (taking 5 as the highest power and ignoring the coefficient of } 1/120). \end{aligned}$$

Exercise 169 Solve the Take Five problem for the list

$$L := 7, 1, -3, 4, 11, -2, 8, 6, 1, 4$$

as rapidly as you can. Did you use either of the algorithms (I) and (II)? What time complexity did your approach have?

Now suppose our list L has length $n = 10$. There are 252 subsets of L of size 5. Algorithm Take Five (II) has to consider *all* 2^n subsets of an n element set; in this case, there are 1024 subsets. So (II) is definitely much slower than (I). However, if we compare n^5 with 2^n with $n = 10$ we get a different picture:

$10^5 = 100000$ two orders of magnitude greater than $2^{10} = 1024$. So surely to say that (I) is an $O(n^5)$ algorithm is to give a misleading picture?

In fact, if you compare n^5 with 2^n you will find that the latter is smaller until beyond $n = 20$. After that, though, things change dramatically as shown in Figure 4.1; and it is the *long term* performance of an algorithm which is the preoccupation of algorithm designers. In the long run the difference between n^5 and $\binom{n}{5}$ is completely negligible compared to how they both differ from 2^n . What is more, the difference between n^5 and $\binom{n}{5}$ eventually becomes negligible compared to how they both differ from n^4 — big-Oh applies in both directions, as the next exercise reveals.

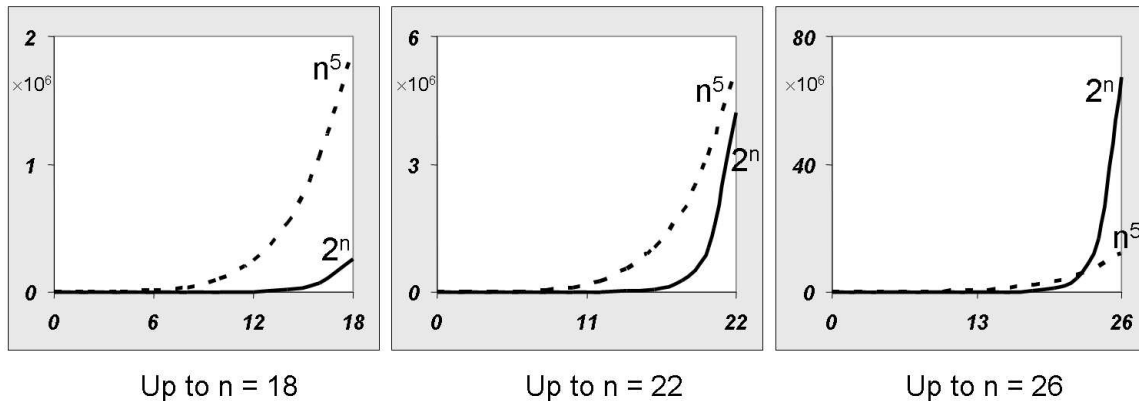


Figure 4.1: graphs of n^5 against 2^n for increasing n .

Exercise 170 If you have access to mathematical software or to a spreadsheet package* (such as the freely downloadable Calc by OpenOffice) calculate the values of n^4 , n^5 , 2^n and $\binom{n}{5}$ for $n = 5, \dots, 150$. A spreadsheet will calculate $\binom{n}{5}$ as something like “=COMBINAT(cell,5)”, where **cell** is the spreadsheet cell containing n ; this will probably give an error if you try to apply it to $n < 5$. Plot the curves of these functions of n for (a) $n = 1 \dots, 30$ and (b) $n = 1 \dots, 150$. What do you notice about how the growths of these functions compare as n becomes large?

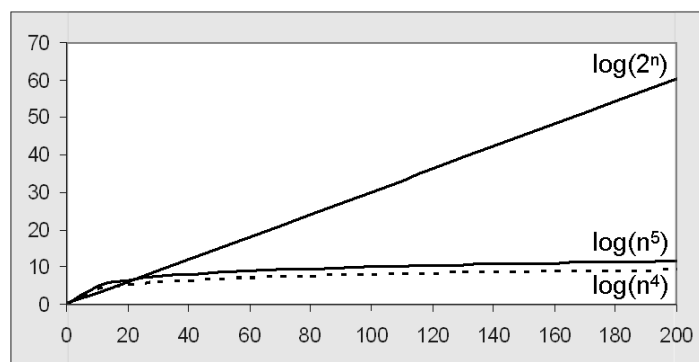


Figure 4.2: graphs of $\log n^4$ and $\log n^5$ against $\log 2^n$ for increasing n .

The difference between algorithms which run in a polynomial number of time steps compared to those which require exponential time is best illustrated by plotting the relevant graphs on a log scale. In Figure 4.2, $\log n^4$ and $\log n^5$ are plotted against $\log 2^n$. The logs are taken to base 10 so that the values being plotted are roughly the number of digits in the numbers. Now $\log 2^n = n \log 2$ so this gives a straight

*If not, don't worry — turn to the solution for this exercise in Appendix C.4, cover up graph (b) and predict, on the basis of graph (a), what will happen as n approaches 150. Now compare your prediction with graph (b).

line with constant slope $\log 2 \approx 0.3$. Meanwhile, when k is a constant, $\log n^k = k \log n$ whose slope decreases with n . The result is, that as n increases from 100 to 200, the number of digits in each of n^4 and n^5 goes up by roughly 1; the number of digits in 2^n roughly *doubles* to 60, an astronomical number!*

The above remarks lead us to make a fundamental distinction between *polynomial-time algorithms*, those which run, in the worst-case in a number of steps bounded by a polynomial function of the input size; and *exponential-time algorithms* for which the worst-case running time is exponential in the input size. Just how fundamental is this distinction we shall now investigate.

4.2.2 Computational complexity

It was observed in the 1960s that many optimisation problems seemed to defy all efforts to design algorithms to solve them that ran in less than exponential time. Checking whether graphs had Hamilton cycles or 3-colourings were typical examples and this apparent intractability seemed to be linked to the question of finding good characterisations, as we saw at the end of Chapter 3 in section 3.4.1. Almost simultaneously, in the early 70s, Stephen Cook, in the USA and Leonid Levin in the USSR proved the seminal result that there existed a ‘hardest’ problem whose solution would automatically yield a solution to every member of a certain well-defined class of ‘decision problems’. This class was called **NP**.

A problem is called a *decision problem* if its output is a Yes or No. Given a graph G we may ask for a longest cycle: this is an optimisation problem; or we may ask if a cycle of length at least K exists in G , for some K ; this is a decision problem. Informally, we may define the class **NP** to be all decision problems for which a Yes-certificate can be specified and checked for accuracy in a polynomial number of steps, the polynomial being defined in terms of the size of the original problem input.

Example 171 Give a decision version of the Take Five problem and show that it is in **NP**.

Solution The simplest decision problem would be: given a list L , does L contain five elements whose sum exceeds K ? A certificate would then be a list of five indices i_1, \dots, i_5 for which it could be checked that $\sum L[i_j] > K$. A simpler way of showing this decision problem is in **NP** is to observe that it may be solved in polynomial time, so the list L itself can act as a Yes-certificate!

In fact, all decision problems solved by some polynomial-time algorithm are automatically, and not very interestingly, members of **NP**. We say that such problems form a class **P** and express the relationship with **NP** as $\mathbf{P} \subseteq \mathbf{NP}$. Our discussion of good characterisations in section 3.4.1 has already provided us with several more problems in **NP**: checking Hamiltonicity (the Yes-certificate is a proposed Hamilton cycle); checking 3-colourability (the Yes-certificate is a proposed assignment of colours to vertices); three simultaneous transversals (the Yes certificate is a proposed transversal for three different subset collections).

Exercise 172 Does the problem of deciding whether three matroids have a common independent set of size K belong to **NP**?

Given that $\mathbf{P} \subseteq \mathbf{NP}$ it is natural to ask whether this inclusion is proper or not. That is, we would like to know if there are problems which are in **NP** without being in **P**. This question is a tremendously important one because of the ‘hardest problem’ theorem of Cook and Levin, which may be informally stated as:

Theorem 173 The class **NP** contains a problem X which has the property that an instance of any other problem in **NP** can, in polynomial time, be reduced to (i.e. reformulated as) an instance of X .

A problem having the property asserted by Theorem 173 is known as **NP-complete**. The existence of such problems is of sufficient theoretical interest but Cook and Levin’s theorem was based on confirming **NP-completeness** for an actual problem: a certain problem in logic known as the Satisfiability problem. This was immediately seized upon by other researchers, foremost among them Richard Karp at the

* 10^{60} is very approximately the number of Planck time units until our sun becomes a red giant and destroys the Earth, *Planck time* being about 0.5×10^{-44} seconds, a fundamental lower limit on the time a computational step could take in some futuristic computer. If this futuristic computer began today to run Algorithm Take Five (II) on a list of 200 entries it would probably not have finished before the death of the Solar System; Take Five (I), however, would have finished within one billionth of a *yoctosecond* (10^{-33} s).

University of California, Berkeley, as a starting point for further polynomial-time reductions. Thus, if instances of Satisfiability can be polynomial-time reduced to instances of, say, Hamilton cycle, then that means that Hamilton cycle is also **NP**-complete: solving it in polynomial time also solves Satisfiability which, by Cook-Levin, solves everything in **NP**. Figure 4.3 shows the approach schematically.

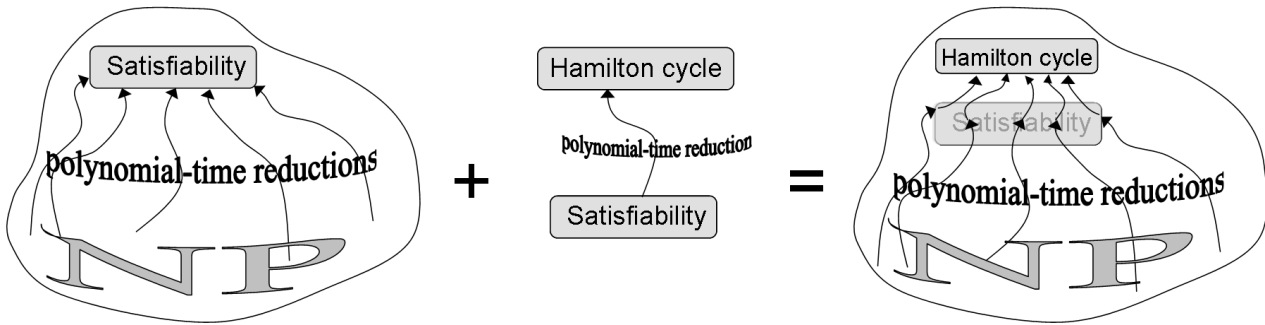


Figure 4.3: proving **NP**-completeness of new problems.

Very quickly a huge tree of polynomial-time reductions had grown, branching out from Satisfiability. Some branches were easy to establish. For example, if we know that Hamiltonian cycle is **NP**-complete then, since we confirmed at the end of Chapter 3 that three-matroid intersection solves Hamilton cycle, it follows that three-matroid intersection is also **NP**-complete (in a suitable decision problem form—see Exercise 172). Most reductions involve a lot more work and involve the construction of ingenious instances of target problems. For instance, we might construct, for a given graph G , a new graph G' in such a way that G has a Hamilton cycle if and only if G' has a 3-colouring. Provided that G' is constructed in polynomial time this shows that 3-colourability is at least as hard to check as Hamiltonicity. Then 3-colouring becomes **NP**-complete.

It would be disappointing not to give at least one typical example of a polynomial-time reduction. Let us define a slightly simplified version of Satisfiability, which is nevertheless **NP**-complete, and show that it can be solved if we can solve the following problem.

MIN VERTEX COVER

Input:	A graph G with vertex set V and edge set SE
Output:	A subset $V' \subseteq V$, of size at most K , such that every edge in E is incident with some vertex in V'

The simplified logic problem is known as 3-SAT: a list of Boolean variables x_1, \dots, x_n is given, together with their negations x'_1, \dots, x'_n . A list of triples is given, each containing three elements from the list $x_1, \dots, x_n, x'_1, \dots, x'_n$. The problem is to decide if there exists an assignment of truth values T and F to the x_i so that every triple contains at least one T . Such an assignment is called a *satisfying truth assignment*. If x_i is assigned T then its negation x'_i is automatically assigned F and vice versa. For example if $n = 4$ and $x_1 = T, x_2 = T, x_3 = F$ and $x_4 = T$ then

$$\begin{array}{cccc} (x_1, x'_2, x_3) & (x_2, x'_3, x_4) & (x'_1, x'_2, x'_3) & (x'_2, x_3, x_4) \\ \text{evaluates to} & (T, F, F) & (T, T, T) & (F, F, T) \end{array}$$

and every triple contains at least one T . However, the same assignment for the triples

$$\begin{array}{cccc} (x'_1, x_2, x_3) & (x'_2, x_3, x'_4) & (x'_1, x'_2, x'_3) & (x'_1, x_3, x_4) \\ \text{evaluates to} & (F, T, F) & (F, F, F) & (F, F, T) \end{array}$$

with the second triple containing only F values. So this is a satisfying truth assignment for the first collection of triples but not the second.

Exercise 174 Find a satisfying truth assignment for the second set of triples, or prove that no such assignment is possible.

It is **NP**-complete to decide if a collection of triples has a satisfying truth assignment. Now suppose we have such a collection S . We are going to construct a graph G and an integer K such that S has a satisfying truth assignment if and only if G has a vertex cover of size K or less. If we can always do this in a number of steps bounded by a polynomial in the size of S then we can always solve 3-SAT in polynomial time if we can solve MIN VERTEX COVER in polynomial time. Then MIN VERTEX COVER is confirmed to be **NP**-complete. Figure 4.4 adapts our earlier schematic to show how this works.

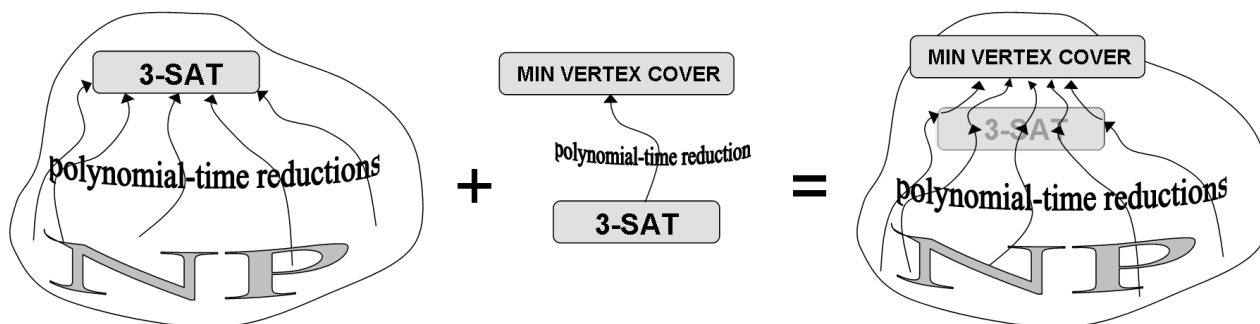


Figure 4.4: proving **NP**-completeness of MIN VERTEX COVER.

Here is how the reduction, that is, the arrow in the middle of Figure 4.4, is implemented. G has an edge $x_i x'_i$ for every logic variable and in addition a cycle of three edges for every triple. Join each vertex for each triple to the corresponding $x_i x'_i$ edge: if the triple begins, say, x_1 , then join the first vertex of its cycle to x_1 ; if it is x'_1 join it to x'_1 . This sounds more complicated than it is: Figure 4.5 will make it clear.

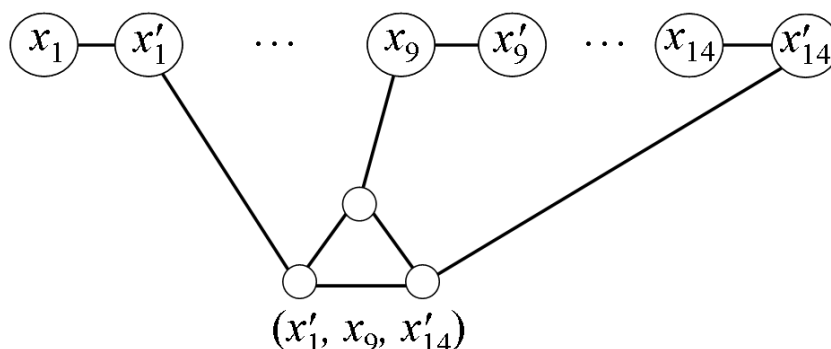


Figure 4.5: constructing the vertex cover graph for 3-SAT: encoding the triple (x'_1, x_9, x'_14) .

Exercise 175 Figure 4.6 shows a ‘building kit’ with all the necessary parts for building the vertex cover graph for the 3-SAT instance (x_1, x'_2, x_3) , (x_2, x'_3, x_4) , (x'_1, x'_2, x'_3) , (x'_2, x_3, x_4) . Follow the pattern in Figure 4.5 to construct the graph.

We set $K = \text{no. of variables} + 2 \times \text{no. of triples}$. For the instance of 3-SAT in Exercise 175, $K = 4 + 2 \times 4 = 12$. The graph you constructed had 20 vertices—we try to find a vertex cover of size at most 12, that is a subset V' of 12 vertices so that every edge is incident with some vertex in V' . The crucial point is, a vertex cover graph has a vertex cover V' of size at most K if and only if the original 3-SAT problem has a satisfying truth assignment. What if, in the graph we have made (we are looking at the solution graph, Figure C.18), we just take all the vertices in the triangles: that is 12 vertices, and all the

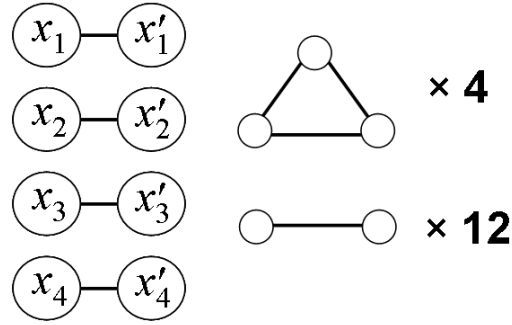


Figure 4.6: component parts for building the vertex cover graph for (x_1, x'_2, x_3) , (x_2, x'_3, x_4) , (x'_1, x'_2, x'_3) , (x'_2, x_3, x_4) .

triangle edges are incident to one of them. Well, this is no good because none of the edges $x_i x'_i$ are joined to any of the triangles. We definitely have to include at least one vertex from each of the $x_i x'_i$ edges for a start.

Here is how we would like to construct a vertex cover:

$$\begin{aligned}
 \text{one vertex from each } x_i x'_i \text{ edge} &= \text{no. of variables} \\
 + \text{two vertices from each triangle} &= 2 \times \text{no. of triples} \\
 \text{total} &= K.
 \end{aligned}$$

This is the best we can do: each truth assignment corresponds to including in V' a subset of vertices from the $x_i x'_i$ edges of G : include x_i if x_i is set to T and x'_i if it is set to F . So far, V' covers every $x_i x'_i$ edge and any edges to triples that have a T entry. Such triples require two additional vertices to be added to V' to cover all their edges; any triple without a T entry requires the addition of three new vertices, and this will take the size of V' beyond K .

Figure 4.7 shows the graph G for the second collection of triples above, with the non-satisfying truth assignment indicated by highlighting the appropriate $x_i x'_i$ edge vertices. For this instance of 3-SAT we have $K = 12$ and you should check that more than eight additional vertices are required to cover all edges of G .

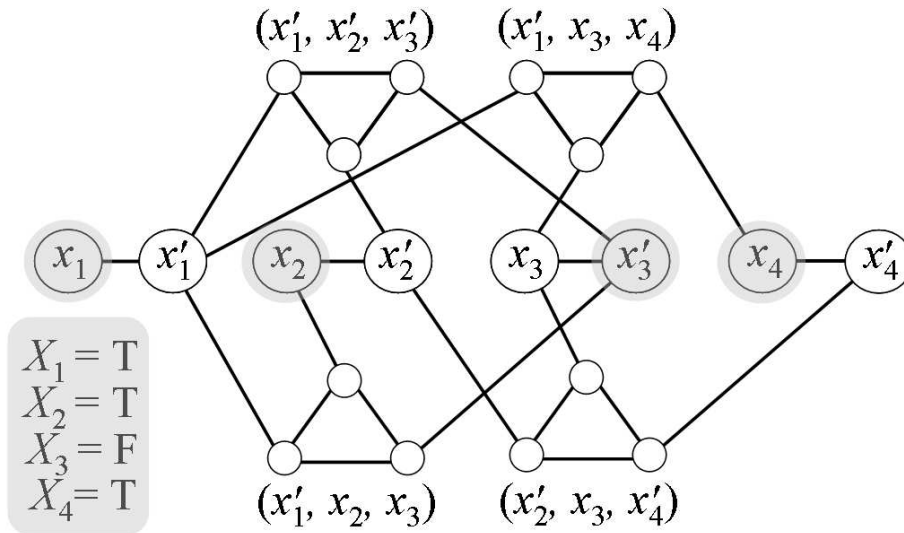


Figure 4.7: constructing the vertex cover graph for 3-SAT instance (x'_1, x_2, x_3) , (x'_2, x_3, x'_4) , (x'_1, x'_2, x'_3) , (x'_1, x_3, x_4) , with the non-satisfying truth assignment $x_1 = T$, $x_2 = T$, $x_3 = F$, $x_4 = T$ highlighted.

Exercise 176 Show that a satisfying truth assignment for the graph in Figure 4.7 gives a vertex cover of size 12.

Later in this chapter we shall see that both 3-SAT and MIN VERTEX COVER may be formulated as linear programmes. The apparent intractability that is indicated by their **NP**-completeness can then be viewed as instances of the intractability of solving linear programmes in integers.

4.2.3 Does $P = NP$?

The question Does $P = NP$? is at once the most briefly stated and most widely known open problem in the whole of mathematics and computer science. Equality will hold if any single **NP**-complete problem can be shown to have a polynomial-time algorithm. If so then, by definition, *every* problem in **NP** has a polynomial-time algorithm. This is often referred to as the “collapse of the polynomial hierarchy” and it is widely assumed not to happen — it is believed that $P \neq NP$. All attempts to resolve this question have so far been unsuccessful.*

We can now view the concept of good characterisations of properties from a more formal standpoint. We can define the class **co-NP** to consist of all those decision problems which possess a No-certificate which can be checked in polynomial time. Together with a Yes-certificate this would provide a necessary and sufficient condition for the property addressed by the problem, so not surprisingly hard problems which are simultaneously in **NP** and **co-NP** are very elusive. In particular, all the evidence so far indicates that $NP \cap co-NP$ contains no **NP**-complete problems. Indeed, if it *does* contain **NP**-complete problems then it would follow that the two classes are identical. It is generally felt that the opposite is the case — that $NP \neq co-NP$; but a proof of this fact can be shown to imply that $P \neq NP$, so this too remains an open question.

Our earlier remarks that polynomial-time problems trivially have both Yes and No-certificates amount to saying that $P \subseteq NP \cap co-NP$ and indeed it is conceivable that all properties with good characterisations can be tested in polynomial-time — that $NP \cap co-NP \subseteq P$, so that equality holds. The problem of testing primality of integers has been known to belong to $NP \cap co-NP$ since Pratt proved in 1975, but it has only quite recently been settled whether it belongs to **P**—it does, thanks to an algorithm discovered in 2002 by Manindra Agrawal, Neeraj Kayal and Nitin Saxena. Remember, testing primality is a decision problem: actually finding factorisations of composite integers appears to be more difficult and is thought not to have a polynomial-time algorithm.†

An important and non-trivial example of a problem which belongs to $NP \cap co-NP$ will lead us forward to our discussion of linear programming. If x is a vector, we will write $x \geq 0$ to mean that every entry in x is non-negative, and similarly for $x < 0$. The following result, which we present in two equivalent forms, is due to the nineteenth century Hungarian scientist Gyula Farkas:

Lemma 177 (Farkas’ Lemma) Let A be an $m \times n$ matrix with entries in \mathbb{R} and let b be a column vector in \mathbb{R}^m . Then for each of the following pairs of statements, exactly one is true:

Standard Form: either (1) there is some column vector $x \geq 0$ in \mathbb{R}^n for which $Ax = b$;
 or (2) there is some column vector y in \mathbb{R}^m such that $A^T y \geq 0$ and $b^T y < 0$.

Canonical Form: either (1′) there is some column vector x in \mathbb{R}^n for which $Ax \leq b$;
 or (2′) there is some column vector $y \geq 0$ in \mathbb{R}^m such that $A^T y = 0$ and $b^T y < 0$.

The words ‘Standard’ and ‘Canonical’ relate to terms in linear programming and do not need to be understood yet, but you should note carefully the difference between the two forms of the Lemma. You may also care to remind yourself of how equations are being represented in matrix form here, by turning back to Figure 2.3 in section 2.2.3 of Chapter 2. At that point we solved the equations $Ax = b$ by reducing the matrix A to row-echelon form and using back substitution; the equations (1) and (1′) above involve a more difficult problem: in the standard form the solution must be non-negative; in the canonical form we

*Indeed it is famously the subject of a \$1 million prize offered by the Clay Institute, see Appendix A.

†And it is to be hoped this is true, given that much of e-commerce security is based on large and supposedly factorisation-proof integers.

have an inequality. These can be important conditions: in Exercise 78, for example, negative production targets had to be avoided. It is not at all clear that the task of meeting such conditions is in **P**; it is, however, clearly in **NP** since a Yes-certificate for either (1) or (1') is any suitably chosen vector x . The two forms of Farkas' Lemma gives us the opposite — a No-certificate which we can produce if there is no solution. The fact that solving $Ax \leq b$ and solving $Ax = b$ non-negatively are problems in **NP** \cap **co-NP** encourages us to think they might, in fact, lie in **P**; and indeed we shall see that they do, although, as with testing primality, this is a difficult mathematical result.

Exercise 178 Use the Standard Form of Farkas' Lemma to produce a No-certificate confirming that $Ax = b$ has no non-negative solutions, where

$$A = \begin{pmatrix} -2 & 3 & 1 \\ 3 & -4 & -1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -7 \\ 9 \end{pmatrix}.$$

4.3 Polyhedral optimisation

We had a double purpose in proposing, at the beginning of this chapter, the rather facile little Take Five problem. Here it is again:

Take Five: In a list of n integers, find the maximum value of a sum of five elements.

We shall now re-specify it in a form reminiscent of Farkas' Lemma (Lemma 177):

$$\begin{array}{ll} \text{maximise} & cx \\ \text{subject to} & Ax = b \\ & x \leq 1 \\ & x \geq 0 \\ \text{where} & A = (1, \dots, 1), \quad b = (5) \end{array}$$

This is a so-called *linear programme*: the maximisation or minimisation of a *linear function*, i.e. a weighted sum of variables, subject to a number of *linear constraints*. The main constraint in our example, $(1, \dots, 1) \times x = 5$ is essentially the test used by Algorithm 168: Take Five (II) to select only subsets of the list c of size five. The aim of the constraint is to exclude all but those vectors x which contain 0's and 1's and sum to 5, therefore indexing five elements of c . There is a big difference though: linear programmes optimise over the real numbers; the constraint does not apply to all 2^n possible 0-1 vectors of length n but to an infinite number of vectors (up to the limits of computer representation, of course!) We cannot automatically exclude the possibility that x is the vector $x = (1, 1/2, 1/3, \dots, 1/83, 0, \dots, 0)$, the sum of whose nonzero elements, to four decimal places, is 5.0021, satisfying the constraint $Ax = b$ to within less than 1/20th of a percent.

Actually, we *can* exclude this possibility for this particular linear programme and for a very good reason: the matrix A is totally unimodular, a property which we met in Chapter 2 in section 2.3.3 in connection with incidence matrices. We shall see in section 4.3.4 that such constraint matrices guarantee integer solutions to linear programmes and that this allows special cases of hard optimisation problems to be solved in polynomial time. Running a standard linear programme solver on the above linear programme for a list c of length 200 produced the required maximum sum more or less instantaneously. The $O(n^5)$ algorithm Take Five (I) took over two hours.

4.3.1 Linear programming

We should anchor our discussion immediately with a formal definition:

Definition 179 Let A be an $m \times n$ real matrix, called the constraint matrix. Let c be a row vector in \mathbb{R}^n and b a column vector in \mathbb{R}^m . Let x be a variable column vector of length n : $x = (x_1, \dots, x_n)^T$. A linear

programme is an optimisation problem in one of the following forms:

$$\begin{array}{ll} \text{maximise} & cx \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

Standard Form

$$\begin{array}{ll} \text{maximise} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array}$$

Canonical Form

The product cx is called the objective function. The conditions following the words “subject to”, including the non-negativity condition $x \geq 0$, are called the constraints. In addition to the above two forms, we may have mixed constraints where some rows of the constraint matrix give ‘=’, some ‘ \leq ’ and some ‘ \geq ’ (this is sometimes called General Form). And we may choose to minimise instead of maximising, in which case ‘ \leq ’ is replaced with ‘ \geq ’ in the canonical form.

You may wish to turn back to Farkas’ Lemma (Lemma 177 in section 4.2.3) to compare its Standard and Canonical forms with those above. The Take Five linear programme with which we began this section was, in fact, a General Form maximisation programme. The different forms of maximisation programme may be easily converted into each other, in the sense that, if we have an algorithm for solving one form then any programme may be adapted to be solved in this form. The details are important if an actual algorithm is to be specified but our discussion is going to be mainly geometrical so we shall avoid the technicalities.

We must immediately ask whether a linear programme, in whatever form, has a solution at all. If there is at least one value of the vector x which satisfies the constraints then this vector is said to be *feasible* and the programme likewise is called *feasible*. A linear programme which has no feasible vectors is called *infeasible*. Even if there are feasible vectors, maximisation may still be impossible since the objective function may be able to take arbitrarily large values. In this case the programme is called *unbounded*.

Exercise 180 For the two linear programmes

$$\begin{array}{ll} \text{maximise} & P + Q + R \\ \text{subject to} & P + 2Q - R = 1 \\ & 5P + 9Q = 3 \\ & P, Q, R \geq 0 \end{array}$$

(1)

$$\begin{array}{ll} \text{maximise} & P + Q + R \\ \text{subject to} & P + Q - R = 1 \\ & -2P + Q = 1 \\ & P, Q, R \geq 0 \end{array}$$

(2)

use the Standard Form of Farkas’ Lemma (Lemma 177) to show that (1) is infeasible; show that (2) is unbounded by deriving a general solution vector, as in Chapter 2 (section 2.2.3).

Linear programming is fundamentally a geometric approach to formulating problems. Of course, this is hard to visualise unless the dimension is almost trivially low — so this is where we shall continue! Consider an even more simple problem than Take Five, we shall call it Take Two:

$$\begin{array}{ll} \text{maximise} & cx \\ \text{subject to} & (1, \dots, 1)x = 2 \\ & x \leq 1 \\ & x \geq 0 \end{array}$$

Suppose we are trying to solve this linear programme for a list c of length 3. We abandon our vector x for the moment to use notation from standard 3-dimensional geometry with our vector now being (x, y, z) : we must maximise $c_1x + c_2y + c_3z$ subject to

$$x + y + z = 2, \tag{4.1}$$

$$0 \leq x \leq 1, \tag{4.2}$$

$$0 \leq y \leq 1, \tag{4.3}$$

$$0 \leq z \leq 1. \tag{4.4}$$

These constraints are depicted in Figure 4.8.

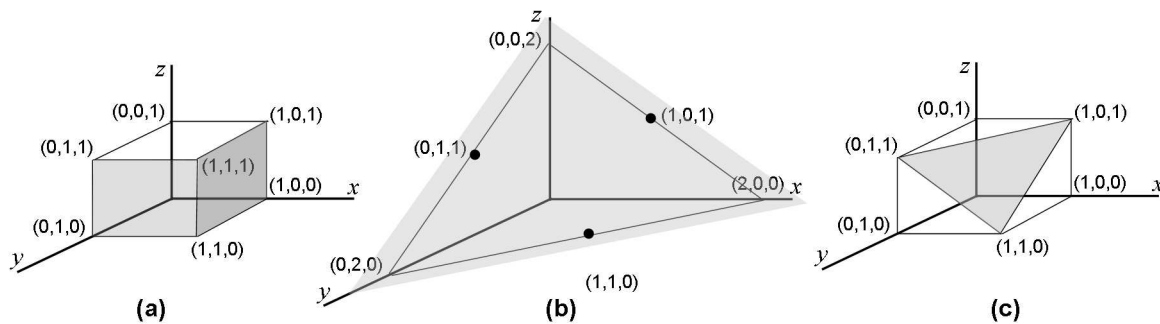


Figure 4.8: Convex polytopes formed by constraints of Take Two: (a) the constraints $0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq z \leq 1$; (b) the constraint $x + y + z = 2$; (c) all constraints.

Any inequality or collection of inequalities defines a region called a *convex polyhedron*, in the present example in 3-dimensional space. For instance, the equality $x = 1$ defines a (2-dimensional) plane parallel with the yz plane and passing through the point $(1, 0, 0)$ —the plane which extends the right-hand end of the box in Figure 4.8(a)). This plane has dimension one lower than the space in which it is defined and is called a *hyperplane* (if it passed the origin it would qualify as a vector space and be called a *hyperspace*). Now any hyperplane defines two convex polyhedra called *half-spaces*, one which replaces its defining equality with \leq and one which replaces it with \geq . The inequality $x \leq 1$ is the half-space to the left of (and including) this plane, as it is drawn in Figure 4.8(a). The box in the figure is completely contained within this half-space.

Now we can specify the box more completely by adding more of the constraints from (4.2)–(4.3). Adding constraints to a convex polyhedron corresponds to *intersecting* it with another convex polyhedron. The result of this intersection is again a convex polyhedron, since it is just defined by a larger set of inequalities. By intersecting progressively more convex polyhedra with each other, the region defined by the intersection may become bounded. A bounded convex polyhedron is called a *convex polytope*. An n -dimensional convex polytope consists of $(n - 1)$ -dimensional *faces* which meet in $(n - 2)$ -dimensional *edges* which in turn meet in $(n - 3)$ -dimensional *vertices*.*

In Figure 4.8(a), the collection of the six inequalities (4.2)–(4.4) have confined the region within a convex polytope — a unit cube. It has six faces, eight edges and eight vertices: the eight linear combinations $a_1(1, 0, 0) + a_2(0, 1, 0) + a_3(0, 0, 1)$, where each a_i is zero or 1.

Recall that the equality $x = 1$ defined a plane, that is a 2-dimensional region. This makes sense because, by requiring x to be fixed at value 1, we have reduced our *degrees of freedom* from three to two: only y and z remain as free choices. Similarly, the first constraint of Take Two, (4.1), is a plane since it is an equality; there are two degrees of freedom because z is not a free choice: it has to be equal to $2 - x - y$. The plane is shown in Figure 4.8(b): it is the whole plane in which the triangle lies.

What happens if we intersect constraint (4.1) with constraints (4.2)–(4.3)? We are intersecting a 3-dimensional convex polytope with a 2-dimensional convex polyhedron, so the result will be 2-dimensional: it will be the diagonal slice through our unit cube shown in Figure 4.8(c).

Exercise 181 What convex polyhedron would result if constraint (4.1) were replaced by an inequality: $x + y + z \leq 1$? Would this be a convex polytope? What about if the inequality was the other way: $x + y + z \geq 1$?

Linear programming, as we remarked, optimises over (some computational representation of) the real numbers. Let us take an actual instance of the Take Two problem: suppose the list c is given by $(3, -2, 1)$. The linear program tries to maximise the function $3x - 2y + z$ subject to the constraints (4.1)–(4.4). Keeping (4.1) as an equality means the search is restricted to the bounded triangle (a 2-dimensional convex polytope) which is the intersection of the hyperplane $x + y + z = 2$ with the unit cube. This would seem to

*In two dimensions, convex polytopes are just *convex polygons* (triangles, rectangles, pentagons etc) which do not really have faces; or rather, the 1-dimensional faces are the edges and the 0-dimensional edges are the vertices.

offer a lot of possible values for x , y and z . If we allow (4.1) to become an inequality, replacing '=' by ' \leq ' as in Exercise 181, then we must search in the whole 3-dimensional convex polytope shown in Figure 4.8(c) — an even greater infinity of points!

The whole theory of linear programming rests on the fact that, in order to optimise a linear objective function, we only have to consider values of the function at vertices of a convex polyhedron. This is not true if the objective function is non-linear: we need calculus to find maxima and minima in this case. It is also not true for a general polyhedron (a union of convex polyhedra) and we have been careful to qualify our polyhedra and polytopes as 'convex' in the preceding paragraphs. Convexity is the crucial property, fortunately bestowed by specifying linear inequalities, which makes linear optimisation a finite problem.

Exercise 182 What are the vertices of the convex polytope in Figure 4.8(c)? If '=' in constraint (4.1) is replaced by ' \geq ' we saw in Exercise 181 that we get a new convex polytope which is a tetrahedron. What are its vertices?

4.3.2 Convexity

Informally, 'convex' means that any straight line joining two points of the polytope will always remain inside the polytope. An example of a non-convex 3-dimensional body is shown in Figure 4.9: the straight line joining a to b remains inside the body but those joining a and b to c pass outside it.

Exercise 183 (A bit of relaxation!) Try to think of symbols in some alphabet or character set which are convex. It is not easy — there are none in the Roman alphabet unless you count the letter 'I' in a sans-serif font. Design a convex alphabet (in two dimensions). Why do you think alphabets, historically, tend not to be convex?

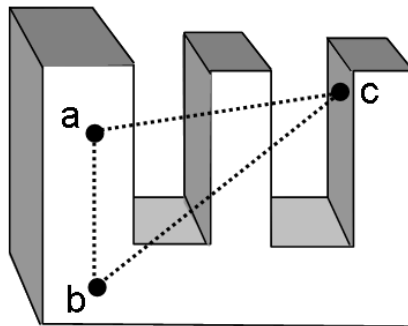


Figure 4.9: a non-convex body in three dimensions.

Convexity may be given a formal definition in terms of a special kind of linear combination:

Definition 184 If x and y are points in \mathbb{R}^n then a linear combination $\alpha x + \beta y$ is called convex if $\alpha, \beta \geq 0$ and $\alpha + \beta = 1$. A region in \mathbb{R}^n is convex if, for any two points x and y in S , all convex combinations of x and y are in S . If S is a finite set of points in \mathbb{R}^n then the convex hull of S , written $\text{conv}(S)$, is the set of all convex combinations of pairs of points in S .

The whole of linear programming is based on the following theorem*:

Theorem 185 Let f be a real-valued linear function defined on the points of a convex polyhedron P . Then:

1. If P bounded (it is a convex polytope) and non-empty then f attains its maximum value at some vertex of P ;

*This theorem is sometimes known as the Extreme Point Theorem. It comes in many forms, the one given here coming from *Elementary Linear Programming with Applications* by B. Kolman, R. E. Beck, see Appendix A.

2. If P is not bounded and is non-empty and f attains a maximum value in P then this value is attained at a vertex of P ;
3. If f does not attain a maximum value in P then P is either unbounded or empty.

'Maximum' may be replaced by 'minimum' in each of the above.

Suddenly we no longer have to worry about checking values at infinitely many points in an n -dimensional body: we need only look at a finite (although possibly large) number of points, its vertices! We observe that the function f in Theorem 185 may attain its maximum value at other points of P than vertices. But in this case this same maximum value will also be attained at a vertex. A very simple example is shown in Figure 4.10. If the objective function is $f(x, y) = x + y$ then its maximum value is clearly 1, and this is attained at two vertices: $(1, 0)$ and $(0, 1)$. But it is also attained everywhere on the line joining these two points—everywhere on one face of the convex polytope. On the other hand if the objective function is $f(x, y) = 2x + y$ then the maximum is attained only at $(1, 0)$.

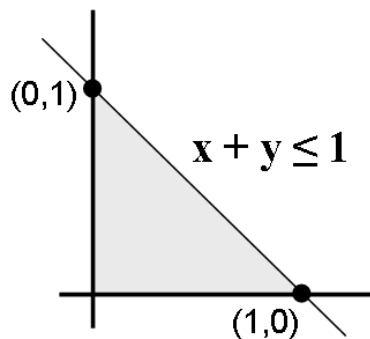


Figure 4.10: the convex polytope defined by $x + y \leq 1$ and $x, y \geq 0$.

Exercise 186 Sketch the convex polytope defined by the following linear programme:

$$\begin{array}{ll}
 \text{maximise} & c(x, y, z)^T \\
 \text{subject to} & x + z \leq 2 \\
 & y + z \leq 1 \\
 & x \leq 2 \\
 & y \leq 1 \\
 & z \leq 1/2 \\
 & x, y, z \geq 0
 \end{array}$$

Hence solve the linear programme when

- (a) $c = (1, 1, 1)$ (b) $c = (1, -1, 1)$ (c) $c = (-1, 1, 1)$ (d) $c = (-1, 1, 2)$.

Find further values of c so that, altogether, you have maximised $c(x, y, z)$ at every vertex of the polytope for some choice of c .

In 1947, George Dantzig, an American mathematician of East European extraction* invented an algorithm which 'intelligently' searched through the vertices of a convex polytope to identify one at which a given linear function was maximised or minimised. This so-called *Simplex algorithm*, although it is provably exponential for some worst-case linear programmes, is very effective in practice and has remained to this day the standard method for solving linear optimisation problems. In 1979 a Russian mathematician, Leonid Genrikhovich Khachiyan† astonished the optimisation world with a completely new approach to solving linear programming. This is known as the *Ellipsoid algorithm* since it works by capturing some

*His father was Russian, his French mother's maiden name, Ourisson, is of Polish origin.

†who died in April 2005, at the tragically early age of 52, coincidentally just a fortnight before Dantzig.

solution to a set of linear equalities within successively smaller ellipsoids. After a polynomial number of iterations the ellipsoids become sufficiently small that the solution is identified, to an arbitrary degree of accuracy, or it is ascertained that no solution exists. The linear equalities can be defined so that the output of the Ellipsoid algorithm yields a solution to the desired maximisation problem. The polynomial-time performance of the algorithm was a tremendous theoretical breakthrough, bringing a key member of $\mathbf{NP} \cap \mathbf{co-NP}$ into the class \mathbf{P} . However, the actual running time of the algorithm is $O(n^2 \log L)$ where L is the size of the linear programme (in terms of bits required to represent it) and n is the dimension of the space in which we are working. This is much slower than the Simplex algorithm in practice and so the Ellipsoid algorithm has proved more important as a vehicle for research into computational complexity than as a tool for OR professionals*.

We are not going to describe the Simplex algorithm or the Ellipsoid algorithm. It would require too many pages to do justice to either. The Simplex algorithm is beautifully described in many places and we particularly recommend the books by Papadimitriou and Steiglitz and by Kolman and Beck, given in Appendix A. The former also describes the difficult Ellipsoid algorithm with great clarity. *Game Theory* by Jones, also listed in the Appendix, is another admirable source.

4.3.3 Duality

As with so many mathematical objects (matroids included!) each linear programme is equipped with a dual. When comparing a linear programme to its dual we may refer to the original programme as the *primal*. If the primal is a canonical-form maximisation programme then the dual is a minimisation programme with the constraint matrix inequalities reversed. We will write down the canonical programme and its dual for comparison. As usual A is $m \times n$, b is a column vector in \mathbb{R}^m and c a row vector in \mathbb{R}^n . Let y be a variable column vector of length m . We have:

$$\begin{array}{ll}
 \text{maximise} & cx \\
 \text{subject to} & Ax \leq b \\
 & x \geq 0
 \end{array}
 \qquad
 \begin{array}{ll}
 \text{minimise} & b^T y \\
 \text{subject to} & A^T y \geq c^T \\
 & y \geq 0
 \end{array}
 \tag{4.5}$$

Primal programme
Dual programme

These two related linear programmes are very important and there are many vectors of potentially different sizes being manipulated. It may help to have a schematic image to refer to and this is given in Figure 4.11.

If a linear programme is feasible and bounded then it has an optimal solution. It will turn out that the value of its objective function at this optimal point is identical to the optimal value we get from the dual programme. It is not obvious that the dual should actually have a solution but the next result confirms that this is the case:

Theorem 187 (1) *If a canonical form linear programme is unbounded then its dual is infeasible.* (2) *If the linear programme is infeasible and its dual is feasible then the dual is unbounded.*

Exercise 188 Suppose P is a primal linear programme which is feasible and bounded. Explain why Theorem 187(1) and (2) together ensure that its dual D must also be feasible and bounded.

Example 189 Show, using the canonical form of Farkas' Lemma 177, that the linear programme

$$\begin{array}{ll}
 \text{maximise} & (-2, -2, 2)x \\
 \text{subject to} & Ax \leq b \\
 & x \geq 0
 \end{array}
 \quad \text{where} \quad A = \begin{pmatrix} 1 & -1 & 2 \\ -2 & 2 & 4 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

*A variant of the Ellipsoid algorithm which rivals Simplex was developed by Narendra Karmarkar in 1984. You can find a description in *Elementary Linear Programming with Applications* by B. Kolman, R. E. Beck, see Appendix A.

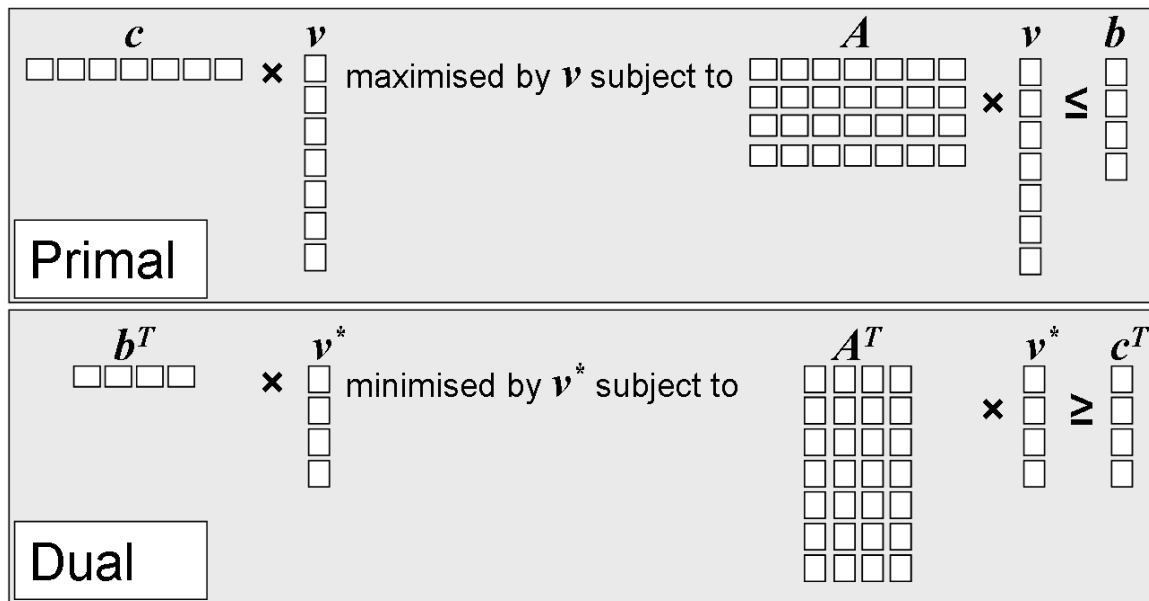


Figure 4.11: schematic image of a primal programme being solved by \mathbf{v} and the corresponding dual programme being solved by \mathbf{v}^* . Not that, in each case, the quantity being maximised or minimised is a product of two vectors and is therefore a *single number*.

is infeasible. Construct the dual programme and sketch its polyhedron. Hence verify that the dual programme is feasible and confirm that it is unbounded, as asserted by Theorem 187.

Solution The canonical form of Farkas Lemma requires us to find a non-negative column vector \mathbf{y} of length 2 such that $A^T \mathbf{y} = 0$ and $b^T \mathbf{y} < 0$. Now the condition $A^T \mathbf{y} = 0$ is exactly what it means for the columns of A^T — that is the rows of A — to be linearly dependent. So one row must be a multiple of the other, but it must be a *negative* multiple since that is what will give us $\mathbf{y} \geq 0$. To actually apply the Lemma as it is written, we can take $\mathbf{y} = (2, 1)^T$ and then

$$A^T \mathbf{y} = \begin{pmatrix} 1 & -2 \\ -1 & 2 \\ 2 & -4 \end{pmatrix} \times \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad b^T \mathbf{y} = (-1, 1) \times \begin{pmatrix} 2 \\ 1 \end{pmatrix} = -2 \times 1 < 0.$$

So the linear programme is certainly infeasible. Using the vector $\mathbf{c} = (-2, -2, 2)$ from the primal programme, the dual programme may be written down as

$$\begin{array}{ll} \text{minimise} & -y_1 + y_2 \\ \text{subject to} & y_1 - 2y_2 \geq -2 \\ & -y_1 + 2y_2 \geq -2 \\ & 2y_1 - 4y_2 \geq 2 \\ & y_1, y_2 \geq 0 \end{array}$$

and it is easy to sketch the polyhedron since this is a 2-dimensional programme. The result is shown in Figure 4.12, with any point in the shaded region being feasible. This shaded polyhedron has two vertices where the two lower straight lines meet the y_1 axis but it is unbounded, so it cannot be a polytope and Theorem 185(1) does not apply. Any point on the line $-y_1 + 2y_2 \geq -2$ satisfies the constraints of the dual programme and on this line the objective function $-y_1 + y_2$ has value $-y_1 + y_1/2 - 1 = -y_1/2 - 1$. Clearly this can be made into an arbitrarily large negative number, so we can confirm that the dual programme is unbounded.

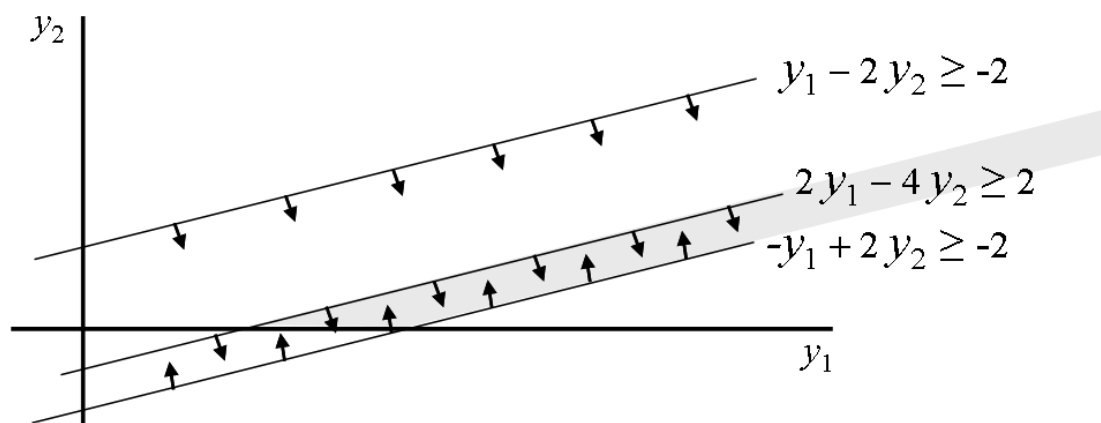


Figure 4.12: polyhedron described in Example 189. Arrows indicate the direction in which the inequalities constrain the plane. The polyhedron is shaded and extends unboundedly to the right.

In Figure 4.12, suppose the constraint defining the upper line, $y_1 - 2y_2 \geq -2$, was changed slightly so that this line had a slightly greater gradient. This would affect neither the unboundedness nor the feasibility of the dual programme but it would mean that the two columns of the constraint matrix for this programme were no longer linearly dependent. Then Farkas Lemma would no longer apply, so the primal programme would have values x with $Ax \leq b$. However, we know by Theorem 187(1) that the primal programme is infeasible. This is not a contradiction: we conclude that none of the solutions to $Ax \leq b$ satisfy non-negativity, so that not all of the constraints of the primal programme can be met.

In Exercise 188, you checked that Theorem 187 was sufficient to ensure that for a linear programme with a solution the dual must also have a solution. One of the fundamental results of linear programming is that the optimum values given by these solutions are equal (you may like to refer back to Figure 4.11 here to remind yourself how these values arise):

Theorem 190 (The Duality Theorem of Linear Programming) *If the canonical primal linear programme given in equation (4.5) is optimised by some feasible column vector v then the dual programme is optimised by some feasible column vector v^* and the optimal value for both objective functions is the same and is given by:*

$$cv = b^T v^* = (v^*)^T Av.$$

Exercise 191 Write down the dual of the linear programme

$$\begin{array}{ll} \text{minimise} & -w - 3x - 6y + 5z \\ \text{subject to} & -2w + x - 3y + z \geq -4 \\ & w - 6x - y + z \geq 2 \\ & w, x, y, z \geq 0 \end{array}$$

Unless you have a strong intuition for 4-dimensional geometry the primal programme in Exercise 191 will defy your efforts to solve it geometrically. This is not a problem of course — linear programmes that occur in practical applications may have hundreds or even thousands of variables; they are solved without any recourse to geometry except as it is embodied in the Simplex algorithm. However, you will have observed in your solution to Exercise 191 that you obtained a 2-dimensional programme which *can* be solved geometrically if desired.

Exercise 192 By sketching the polytope for the dual programme you found in Exercise 191, find the optimal value of its objective function. Hence give the optimal value for the primal programme by appealing to the Duality Theorem of Linear Programming.

If you worked your way through the last exercise and checked its solution you will come to the conclusion

that solving linear programmes by hand, even in four dimensions and with the aid of the Duality Theorem, is very laborious. Let us honour the memory of George Dantzig, whose Simplex algorithm has for over sixty years worked so tirelessly and ingeniously so that we do not have to!

4.3.4 Integer linear programming

We explained at the beginning of section 4.3 that our Take Five problem could be formulated as a linear programme and that this linear programme somehow avoided optimal solutions that were produced by vectors of many non-integer entries instead of the five 1's that we needed. The first fact is not surprising: almost all optimisation problems, and certainly all those we have met in this study guide, can be formulated as linear programmes. That we could rely on getting an integer solution was a much rarer event.

Let us look at a problem which we know is much harder than Take Five: the MIN VERTEX COVER problem. We shall see that we can formulate this, but not solve it, using linear programming. But we shall see that we can nevertheless derive some useful information for optimisation.

We met MIN VERTEX COVER in section 4.2.2 purely as an example of a natural optimisation problem that was at least as hard as 3-SAT in the sense that any instance of 3-SAT could be reformulated as a MIN VERTEX COVER problem. What if we are actually interested in solving MIN VERTEX COVER? Here is an OR application:

Congestion Charging: Camera stations are to be situated on a road network so the registration numbers of cars entering the network can be recorded. It is required that every stretch of road in the network is surveyed by a camera and that the number of stations is kept to a minimum.

The Congestion Charging problem is solved by MIN VERTEX COVER: the road network is modelled as a graph with vertices representing junctions and edges representing stretches of road. Each vertex represents a potential site for a camera station. It is perhaps necessary to add extra vertices to edges that represent very long stretches of road, or that represent stretches of road that bend. This model is illustrated in Figure 4.13. A vertex cover will be a choice of vertices as camera sites so that every edge of the graph is 'covered' by a camera.

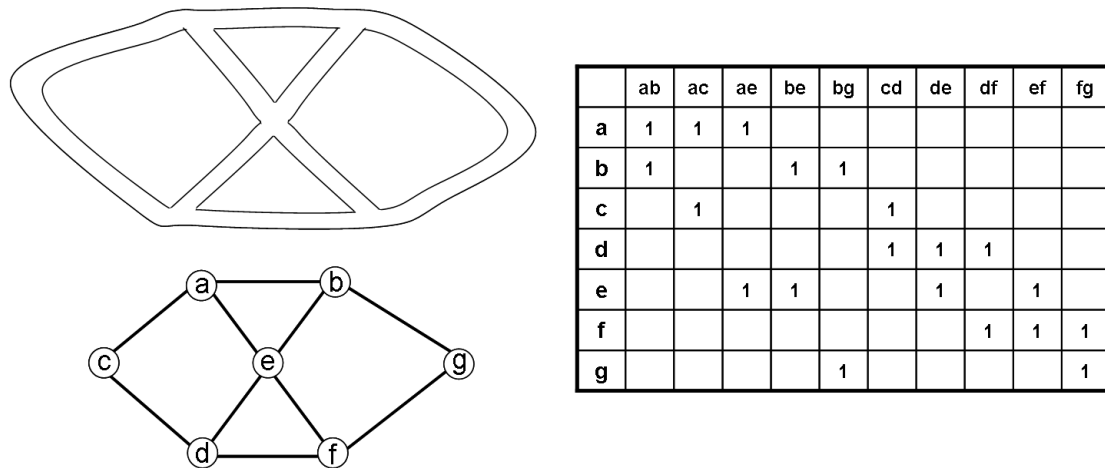


Figure 4.13: a road network modelled as a graph, and its incidence matrix.

We can exploit the incidence matrix to produce inequalities which will turn MIN VERTEX COVER into a linear programme. Notice that each column of the matrix in Figure 4.13 represents an edge of the graph and each 1 in that column records a vertex which is incident with that edge. So, if we choose a subset X of the vertices then it is a vertex cover if the corresponding rows, between them, provide a 1 in every column. The rows of the matrix corresponding to vertices *not* in X , are removed from consideration: we can think of this as multiplying the rows in question by zero. Suppose the incidence matrix is A . Let y be a vector

which has an entry for each vertex: 1 for the entries in X and zero for those not in X . Let c be a row vector consisting of n 1s where n is the number of edges of the graph. Then y specifies a vertex cover if and only if $A^T y \geq c^T$.

Exercise 193 Figure 4.14 shows a 3-vertex graph (K_3) and its incidence matrix. Write down the products $A^T y$ as explained above, where A is the incidence matrix and y is (a) $(0, 1, 0)$, and (b) $(1, 1, 0)$. Hence show that $\{b\}$ is not a vertex cover but that $\{a, b\}$ is a vertex cover. What does the value of 2 in the second product represent?

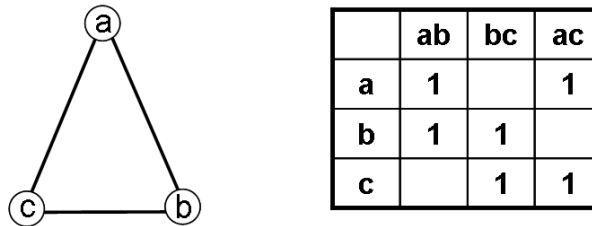


Figure 4.14: graph and incidence matrix for Exercise 193.

We can thus formulate MIN VERTEX COVER as a linear programme in dual form, as in (4.5) in section 4.3.3:

$$\begin{array}{ll} \text{minimise} & b^T y \\ \text{subject to} & A^T y \geq c^T \\ & y \geq 0 \end{array}$$

with b being the all-1s vector which sums the entries in y .

Exercise 194 Suppose the y vector in Exercise 193 is written $y = (y_1, y_2, y_3)$. Write down explicitly the constraints of the dual linear programme for this instance of MIN VERTEX COVER.

We can certainly solve this linear programme efficiently. The problem is that for a solution to be meaningful it is absolutely necessary for this solution to be a 0-1 vector. If you solve the linear programme in Exercise 194 you will get the solution vector $(1/2, 1/2, 1/2)$. For our Congestion Charging application this is suggesting that the cheapest solution is to put half a camera site at every location; but presumably the number of camera sites is minimised partly, or even predominantly, because of the costs associated with installing and maintaining each site: having fewer, smaller sites might well *increase* the overall cost.

Why do the solutions of this linear programme not include the minimum vertex cover set $\{a, b\}$ which you found in Exercise 193? It is worth looking at the convex polyhedron given by the inequalities. It is constructed in Figure 4.15. Since it is rather hard to visualise, the polyhedra for two subsets of the inequalities are shown, as well as the actual polyhedron which is on the right.

This polyhedron is not a polytope since it is not a convex hull—it extends to infinity along each axis. However, Theorem 185(2) applies because this is a *minimisation* programme. It has four vertices: $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$ and $(1/2, 1/2, 1/2)$ and the latter is the unique minimum-value vertex for the programme. So although all the possible minimum vertex covers are vertices of the polyhedron they will never be selected by a linear programme solver. Notice that the point $(1, 1, 1)$ is also a (non-minimum) vertex cover but not a vertex of the polyhedron.

The same problem besets our larger minimisation problem in Figure 4.13: if we use the incidence matrix in Figure 4.13 to specify a linear programme then it will also yield a minimum at a non-integer point, the vector $(1/2, 1/2, 1/2, 1/2, 1/2, 1/2)$ to be precise. The elements in this vector sum to 3.5 which is less than the elements of, say, $(1, 1, 0, 1, 0, 1, 0)$ which represents the minimum vertex cover $\{a, b, d, f\}$ but has element sum 4. However, although we cannot use the linear programme solution to position our congestion charge cameras it does tell us something. We require an integer number of camera sites and a vertex cover of size 3 would give a solution to our linear programme which is smaller than its optimal value which is impossible. So we need at least 4 camera sites. This is a general principle: *if the objective*

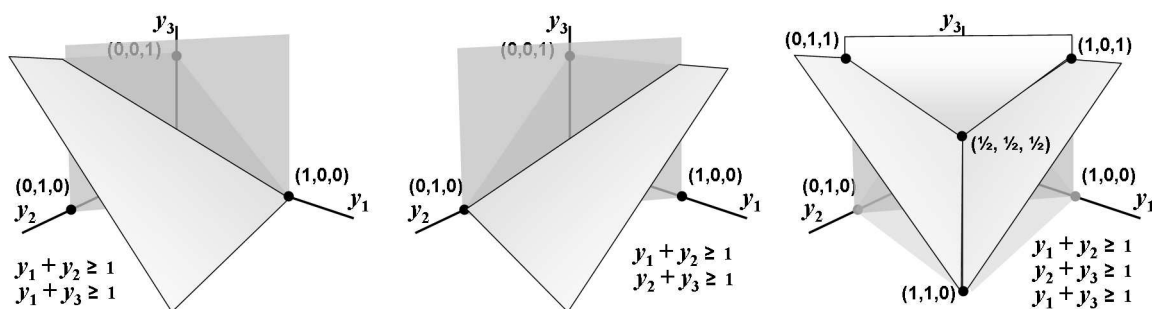


Figure 4.15: the polyhedron associated with the linear programme of Exercise 194. The polyhedra left and centre show subsets of the inequalities to help illustrate the construction of the final polyhedron on the right.

function $\sum x_i$ has minimum value M subject to the constraints of a linear programme, then any 0-1 vector satisfying these constraints must have at least $\lceil M \rceil$ 1s, where $\lceil M \rceil$ is the smallest integer greater than M .

We should not be surprised to find that MIN VERTEX COVER could not be solved by linear programming since we know that it is *NP*-complete. However, even problems in *P* may not be solvable by linear programming if they require integer solutions. Let us look at another example and introduce some terminology. We will return to Chapter 1 and a problem we left unsolved — MIN-WEIGHT MAXIMAL MATCHING. We saw at the end of Chapter 2 that the matching matroid would, in principle, solve the problem but only by relying on an oracle for telling us when sets of vertices belonged to matchings. Now let us try linear programming. We shall try to find a maximum weight matching in a graph G with weighted edges — this will best fit our canonical form and we know that we can replace weights by their negatives to recover the original minimisation problem.

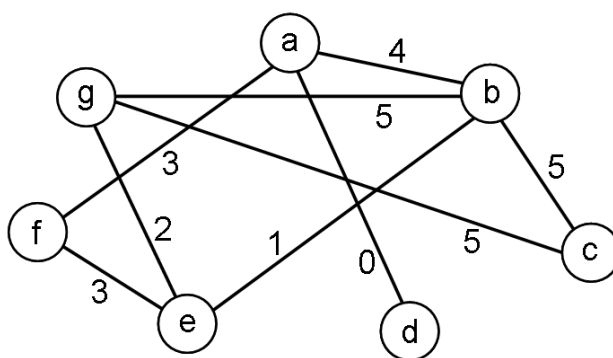


Figure 4.16: the graph originally given in Figure 2.9 in section 2.3.4 of Chapter 2.

Recall that a matching in a graph is a subset of edges with no vertex incident with more than one edge in the subset. To illustrate how linear programming applies to maximum matchings we will borrow a graph to which we applied Network Pruning in section 2.3.4 of Chapter 2 — for convenience it is reproduced in Figure 4.16. The objective function is a natural choice: there are nine edges; we shall have a variable for each and try to maximise a weighted sum of these variables, with the weights being the edge weights from the graph:

$$\begin{aligned} &\text{maximise} && cx \\ &\text{where} && c = (4, 0, 3, 5, 1, 5, 5, 3, 2) \\ &\text{and} && x = (x_{ab}, x_{ad}, x_{af}, x_{bc}, x_{be}, x_{bg}, x_{cg}, x_{ef}, x_{eg})^T \end{aligned}$$

What we want is for the linear programme to pick some edges and not others, just as the Take Five programme picked subsets of five list entries. So $x = (0, 1, 0, 0, 1, 0, 1, 0, 0)$, picking edges ad , be and cg

would be a good start (it is a matching) and $x = (1, 0, 0, 0, 0, 0, 1, 1, 0)$ would be better (ab, cg, ef is a maximum-weight matching). But we must avoid outputs such as $x = (0, 0, 0, 1, 0, 1, 1, 0, 0)$ which (greedily) selects all three edges of weight 5 and results in each of vertices b, c and g being incident with more than one edge. This last example gives us a clue on how to choose our constraints: we shall do as we did with Take Five and set up sums of x elements:

$$\begin{array}{ll}
 \text{maximise} & cx \\
 \text{subject to} & x_{ab} + x_{ad} + x_{af} \leq 1 \\
 & x_{ab} + x_{bc} + x_{be} + x_{bg} \leq 1 \\
 & x_{bc} + x_{cg} \leq 1 \\
 & x_{ad} \leq 1 \\
 & x_{be} + x_{ef} + x_{eg} \leq 1 \\
 & x_{af} + x_{ef} \leq 1 \\
 & x_{bg} + x_{cg} + x_{eg} \leq 1 \\
 & x_{ab}, \dots, x_{eg} \geq 0 \\
 \text{where} & c = (4, 0, 3, 5, 1, 5, 5, 3, 2)
 \end{array}$$

We are almost ready to run! But we know from our experience with MIN VERTEX COVER that there is a danger of getting solution vectors such as $(\frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ which obey our constraints but are meaningless in terms of our graph. There are two ways to proceed:

Integer Linear Programming (ILP): we add a constraint saying “all x_{ij} are integer”;

Linear Programming Relaxation: we add a constraint saying “all x_{ij} satisfy $x_{ij} \leq 1$ ”.

In our MIN VERTEX COVER example, the non-integer solution nevertheless obeyed the Relaxation constraint so we must accept that this will not always resolve the difficulty. ILP *does* guarantee a complete solution to maximum matching but there is a different and apparently fatal difficulty:

if our constraints produce a convex polytope whose vertices are not integers then we have no way to locate the optimal integer points inside the polytope.

We shall see shortly that ILP is **NP**-complete: we probably have no choice but to accept Relaxation. As a matter of fact, if you input the above problem, exactly as it is written, to a linear programme solver it will actually give you the correct answer. This is pure luck! If you change the weight of the edge ad from zero to 2 then an optimal solution to the Relaxation is

$$(0, 1, 0, \frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}, 1, 0)$$

which has no interpretation in terms of the graph.

Exercise 195 Given the modified weight vector $c = (4, 2, 3, 5, 1, 5, 5, 3, 2)$ what are the values of the cx for $x = (1, 0, 0, 0, 0, 0, 1, 1, 0)$ and $x = (0, 1, 0, 1/2, 0, 1/2, 1/2, 1, 0)$?

We have now seen two examples of ILP in action, neither of which were very successful. We will explain our lack of success by returning to the theory we built up in section 4.2. Recall from section 4.2.2 that the 3-SAT problem comprises a set of Boolean variables x_1, \dots, x_n together with their negations x'_1, \dots, x'_n , and a list of triples over this set. An optimisation version of 3-SAT (as distinct from the decision problem) is to find a satisfying truth assignment: an assignment of truth values T and F to the x_i so that every triple contains at least one T . In an earlier example we had the triples

$$(x_1, x'_2, x_3), (x_2, x'_3, x_4), (x'_1, x'_2, x'_3), (x'_2, x_3, x_4)$$

and a satisfying truth assignment was $x_1 = T, x_2 = T, x_3 = F$ and $x_4 = T$.

The optimisation version of 3-SAT is no harder than maximum matching to capture by ILP. We shall use the same x_i variables (but not their negations) and for each triple we shall have a constraint of the form

$$\sum_i x_i + \sum_j (1 - x_j) \geq 1$$

where the first summation is over the x_i which appear without negation and the second summation is over those which appear negated: so x'_j is represented by $(1 - x_j)$. For our example of four triples this gives

$$\begin{array}{ll} \text{minimise} & cx \\ \text{subject to} & \begin{array}{rcl} x_1 + x_3 + (1 - x_2) & \geq & 1 \\ x_2 + x_4 + (1 - x_3) & \geq & 1 \\ (1 - x_1) + (1 - x_2) + (1 - x_3) & \geq & 1 \\ x_3 + x_4 + (1 - x_2) & \geq & 1 \\ x_1, x_2, x_3, x_4 & \geq & 0 \end{array} \end{array}$$

What is the vector c ? The question is, what exactly are we minimising? It does not matter! Although we said this was an optimisation version of 3-SAT, that is only to say that we ask for an actual truth assignment rather than just an answer: “Yes, an assignment exists,” or “No, no assignment exists.”

Any feasible vector will provide a satisfying truth assignment *as long as all its entries are zeros and 1's*. For example, the assignment $x_1 = T$, $x_2 = T$, $x_3 = F$ and $x_4 = T$ would correspond to the x vector $(1, 1, 0, 1)^T$ and you can check that each constraint in our example linear programme is satisfied by this vector.

Exercise 196 Write down the constraints which model the 3-SAT instance

$$(x'_1, x_2, x_3), (x'_2, x_3, x'_4), (x'_1, x'_2, x'_3), (x'_1, x_3, x_4).$$

Find an assignment of zeros and 1's to the x_i which satisfy the constraints and confirm that it corresponds to a satisfying truth assignment for these triples.

We will assume that our cx for minimisation is just the sum of all the x_i (on the basis that setting a logic gate input to T probably costs more than setting it to F , so the fewer 1s the better!) Now we are in the same position as for maximum matching in that we must try to avoid non-integer solutions to our linear programme. The above example will actually be solved as a Relaxation problem by the zero vector. However making a few changes to give

$$(x_1, x_2, x_4), (x'_1, x_2, x_4), (x'_1, x'_2, x'_3), (x'_2, x_3, x_4)$$

gives a linear programme which is solved by $x = (1/2, 1/4, 0, 1/4)^T$.

At this point we can observe that we have done what we did for MIN VERTEX COVER in section 4.2.2: we have shown that a polynomial algorithm for solving ILP would allow us to solve 3-SAT. So we can add ILP to the network of problems in Figure 4.3:

Theorem 197 ILP is NP-complete.

4.3.5 Totally unimodular matrices

While studying the representations of graphic matroids in section 2.3.3 of Chapter 2 we identified the property of incidence matrices of graphs which made the Matrix Tree Theorem work: total unimodularity (definition 97). This obscure-sounding property of having all nonzero determinants equal to ± 1 for all submatrices, turns out to run very deep in combinatorics: it appears in all sorts of applications and constructions; and wherever it appears it brings good news to the optimiser:

Theorem 198 For any linear programme having constraints $Ax \leq b$ or $Ax = b$, where the constraint matrix A is totally unimodular and b is integer-valued, all vertices of the corresponding convex polyhedron are integer-valued. Moreover, this property is unchanged if the constraints $0 \leq x \leq u$ are added, where $x = (x_1, \dots, x_n)^T$ and $u = (u_1, \dots, u_n)^T$ is any non-negative integer column vector.

You will recall from Theorem 185 that if a linear programme objective function is maximised or minimised, then the maximum or minimum is attained at vertices of its polyhedron. If all vertices are integer-valued then an ILP having constraints $Ax \leq b$ or $Ax = b$, where A is totally unimodular and b is integer-valued, will be solved in its Relaxation form.

Suppose we return to our difficulty with the linear programming relaxation for the maximum matching problem. It was more convenient not to write out the constraints as a large matrix at the time, but we will do so now:

$$A = \begin{matrix} & \begin{matrix} ab & ad & af & bc & be & bg & cg & ef & eg \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix} & \begin{pmatrix} 1 & & 1 & & & & & & \\ 1 & 1 & & & & & & & \\ & & & 1 & 1 & 1 & & & \\ & 1 & & & & & 1 & & \\ & & & & 1 & & & 1 & 1 \\ & & 1 & & & & & 1 & \\ & & & & & 1 & 1 & & 1 \end{pmatrix} \end{matrix}$$

(we omit the zeros for clarity). Notice that this is the *unsigned* incidence matrix of the graph. If this matrix were only unimodular then we would be able to find maximum matchings via linear programming relaxation. But we know this cannot be the case — we have seen an instance where fractional values occurred in the solution. Sure enough, if we take the following square submatrix:

$$\begin{matrix} & \begin{matrix} bc & bg & cg \end{matrix} \\ \begin{matrix} b \\ c \\ g \end{matrix} & \begin{pmatrix} 1 & 1 & \\ 1 & & 1 \\ & 1 & 1 \end{pmatrix} \end{matrix}$$

then we find it has determinant -2, violating the definition of total unimodularity.

There is something to be rescued, however. In the case of bipartite graphs, linear programming relaxation is guaranteed to work:

Theorem 199 *The unsigned incidence matrix of a bipartite graph is totally unimodular.*

Well, we can already solve maximum matchings in bipartite graphs by matroid intersection (Chapter 3, section 3.3.1) but it is nice to know that the linear programming polyhedron for this problem has integer-valued vertices. In fact, the two approaches are intimately connected as we shall see shortly.

Where else can we apply total unimodularity? One example is our Take Five problem. Its constraint matrix was $A = (1, \dots, 1)$ which is trivially totally unimodular. We can also put to use the fact that signed incidence matrices of graphs are totally unimodular. Here is an application:

Capacitated Transportation: It is required to transport a number K of units as cheaply as possible from a point s to a point t through a network whose links each carry a certain tariff per unit; additionally, each link has a capacity and the number of units transported through each link must not exceed this capacity.

An example is given in Figure 4.17. Each arc (directed edge) has two numbers: a cost or tariff c for transporting a unit through that arc and a capacity u —the highest number of units which the arc can carry.

Capacitated Transportation is modelled as a classical problem in directed graphs:

Definition 200 *Suppose that G is a directed graph, and that two vertices, s and t , of G are nominated as a ‘source’ and ‘target’. A flow in G is an assignment of a non-negative integer value to each arc such that:*

1. *each arc into s has value zero (nothing is carried into the source);*
2. *each arc out of t has value zero (nothing is carried away from the target);*
3. *at each other vertex, the sum of the incoming values is equal to the sum of the outgoing values (nothing is lost or gained at intermediate points during transport); and*
4. *the sum of the values leaving s equals the sum of the values arriving at t (nothing is lost or gained overall during transportation).*

The flow value is defined to be the sum of the values leaving s (and arriving at t). A K -flow is a flow whose value is K .

An example of a 3-flow is shown in Figure 4.18.

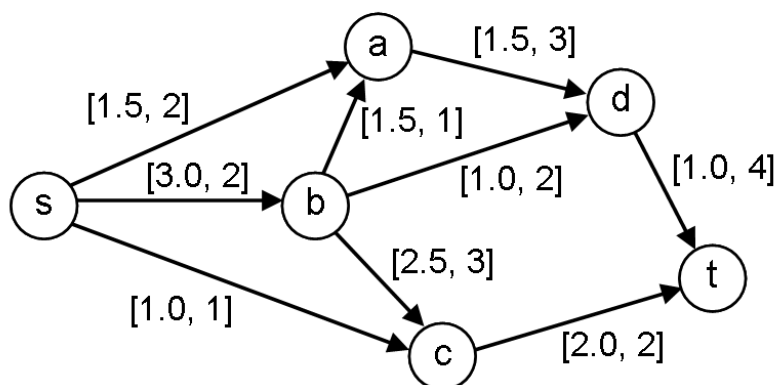


Figure 4.17: a network in which an $s - t$ flow must be found. The label of each edge, $[c, u]$, is the cost of using the edge and its capacity.

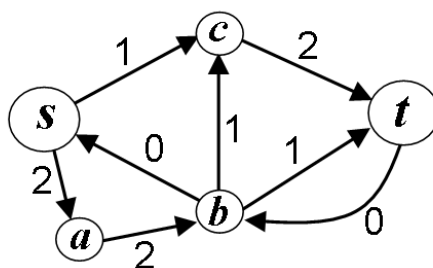


Figure 4.18: an example of a 3-flow.

Exercise 201 Suppose the flow through arc bc in Figure 4.18 is increased to 2 units. If no arcs are allowed to decrease their current flow how will this affect the flow through the whole network?

MIN-FLOW

Input:	Directed graph G , with each arc e having a cost $c(e)$ and a maximum capacity $u(e)$; two vertices s and t of G ; a positive integer K
Output:	an $s - t$ K -flow of minimum total edge cost in which no arc e has value exceeding $u(e)$

An example is shown in Figure 4.17. A 0-flow is trivial to find — each edge carries no goods. A 1-flow is not much more complicated since it can only be a path from s to t , each arc having value 1. In Figure 4.17 there are five possible paths, the cheapest being the one taking edge sc followed by edge ct .

Exercise 202 One possible $s - t$ 2-flow in the network of Figure 4.17 is given by sending value 1 on arcs sb , sc , and bc and value 2 on arc ct . What is the cost of this flow, using the cost values given in the figure? Is this a minimum-cost 2-flow for this network?

Finding flows manually, even in such a small network, is quite arduous. For instance, Figure 4.19 shows a minimum cost 5-flow in the same network. It is quite hard to convince yourself with pen and paper that this is indeed the cheapest way of transporting 5 units from s to t .

Now we can put to good use the fact that the incidence matrix of a graph is totally unimodular and also interpret, at last, the + and - sign which appears in each column. We will represent our directed graph for

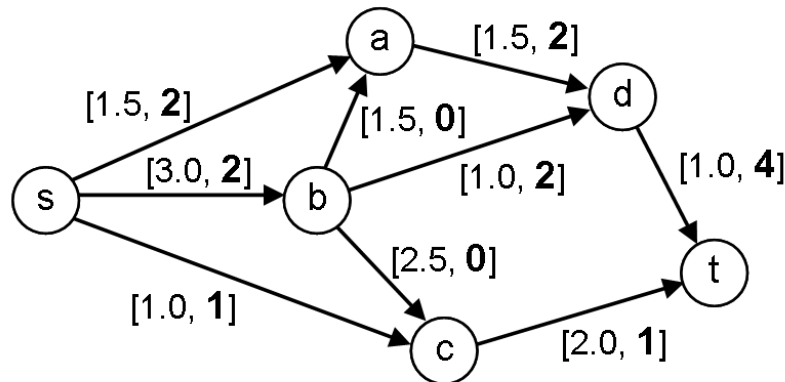


Figure 4.19: a minimum-cost 5-flow in the network of Figure 4.17 (values used in each arc are in bold).

the flow problem by the incidence matrix of its underlying undirected graph but in each column we shall use the -1 for the vertex which is the tail of the corresponding arc and the +1 for the head of this arc. The matrix for the network in Figure 4.17 is:

$$A = \begin{matrix} & \begin{matrix} sa & sb & sc & ba & bc & ad & bd & ct & dt \end{matrix} \\ \begin{matrix} s \\ a \\ b \\ c \\ d \\ t \end{matrix} & \begin{pmatrix} -1 & -1 & -1 & & & & & & \\ 1 & & & 1 & & -1 & & & \\ & 1 & & -1 & -1 & & -1 & & \\ & & 1 & & 1 & & & -1 & \\ & & & & & 1 & 1 & & -1 \\ & & & & & & & 1 & 1 \end{pmatrix} \end{matrix}$$

Now we can specify a linear programme for the flow problem. Let us define a column vector b which has an entry for each vertex in our network:

1. the entry for the source vertex s is $-K$;
2. the entry for the target vertex t is $+K$;
3. all other entries are zero,

(so the b vector is encoding rules 3 and 4 of definition 200).

$$\begin{array}{ll} \text{minimise} & cx \\ \text{subject to} & Ax = b \\ & x \leq u \\ & x \geq 0 \end{array}$$

The objective function cx is taking account of the cost vector c ; the constraints $x \leq u$ take account of the capacities in the arcs; the b vector is encoding the definition of a flow. The whole programme is guaranteed to produce only integer entries in the x vector of arc values because of the total unimodularity of A .

Exercise 203 Write out in full the linear programme for finding a 5-flow in the network of Figure 4.17, i.e. write each individual equation rather than specifying the programme with matrices and vectors.

4.4 Matroids again

It should not be thought that the examples we have given of totally unimodular matrices are the only ones; as we said they occur throughout combinatorics. We should mention that they have been given a good characterisation by Paul Seymour; it is beyond our scope but, as we might hope, leads to a polynomial time algorithm for testing for unimodularity.

Nor should it be thought that, just because a linear programme fails to have a totally unimodular constraint matrix, it cannot have integer-valued vertices in its polyhedron. We shall end this chapter and the study guide with perhaps the two most famous examples: the Birkhoff polytope and the polytopes that Jack Edmonds derived for matroids and matroid intersections. There are many others: for example the 0-1 matrix which records the incidence of vertices with maximal complete subgraphs of a graph, defines a convex polytope which has integer vertices precisely if the graph in question is a *perfect graph** a lovely and unexpected link between geometry and graph structure discovered by the Czech mathematician Vášek Chvátal.

4.4.1 The Birkhoff polytope

We introduced permutation matrices in section 2.2.2 of Chapter 2. They provided a useful definition of the determinant function and then reappeared to help in the search for transversals in section 2.4.1; now they briefly take a starring role.

An $n \times n$ matrix is called *doubly stochastic* if each of its rows and columns consists of non-negative real numbers summing to 1. As you might imagine, such matrices are very important in probability theory but our interest comes from the following remarkable theorem:

Theorem 204 (The Birkhoff-von Neumann Theorem) *The $n \times n$ doubly stochastic matrices form a convex polytope whose vertices are the $n \times n$ permutation matrices.*

You need to think of the $n \times n$ matrices in this theorem as being written out, row by row, so that their elements become a vector of length n^2 , as illustrated in Figure 4.20, so that the polytope, which is called the *Birkhoff polytope*, is a subset of \mathbb{R}^{n^2} .

The Birkhoff-von Neumann Theorem, together with the Extreme Point Theorem (Theorem 185) tells us that if we specify the property of being doubly stochastic by means of a set of linear constraints then a linear programme which optimises subject to these constraints will locate an optimum value at a 0-1 vector corresponding to a permutation matrix. For an $n \times n$ doubly stochastic matrix X there are $2n$ constraints, one for each row and column, as shown below:

$$\begin{array}{ccc}
 \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ & & \cdots & & \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nn} \end{pmatrix} & \longrightarrow & \begin{array}{l} x_{11} + x_{12} + x_{13} + \cdots + x_{1n} = 1 \\ x_{21} + x_{22} + x_{23} + \cdots + x_{2n} = 1 \\ \cdots \\ x_{n1} + x_{n2} + x_{n3} + \cdots + x_{nn} = 1 \end{array} \\
 \downarrow & & \text{row sums equal 1} \\
 \begin{array}{l} x_{11} + x_{21} + x_{31} + \cdots + x_{n1} = 1 \\ x_{12} + x_{22} + x_{32} + \cdots + x_{n2} = 1 \\ \cdots \\ x_{1n} + x_{2n} + x_{3n} + \cdots + x_{nn} = 1 \end{array} & & \text{column sums equal 1}
 \end{array} \tag{4.6}$$

and there are n^2 constraints $x_{ij} \geq 0$ for $1 \leq i, j \leq n$. We can now create a standard form maximisation linear programme (Definition 179 in section 4.3.1) and maximise some objective function cx where

*By the Strong Perfect Graph Theorem, a graph G is perfect if and only if for any odd-length cycle, C_t , $t \geq 5$, if either C_t or K_t minus C_t is a subgraph then G has additional edges joining its vertices but not belonging to the subgraph. That there is more to perfect graphs than would appear from this definition can be inferred from the fact that the proof of the Strong Perfect Graph Theorem is over 150 pages long.

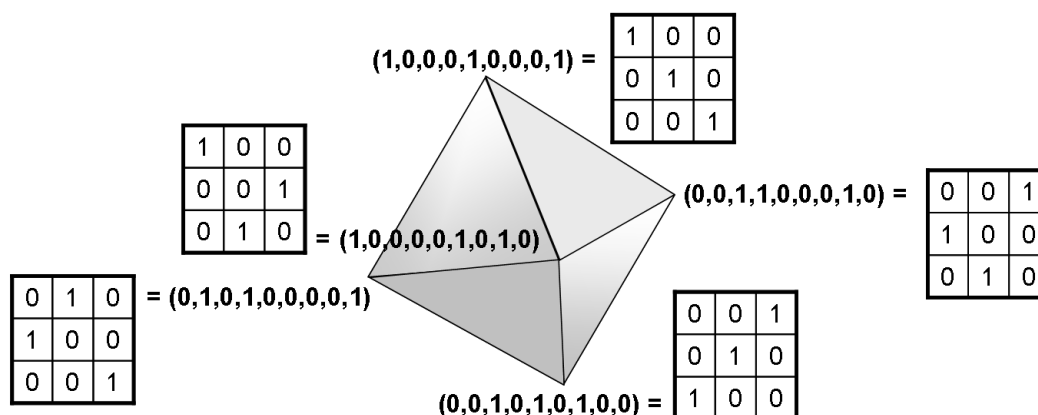


Figure 4.20: a sketch of the Birkhoff polytope for 3×3 doubly stochastic matrices. Since these live in 9-dimensional real space this 2-dimensional picture of a 3-dimensional polytope is not meant to do more than convey the general idea!

$$x = (x_{11}, x_{12}, \dots, x_{nn})^T.$$

Example 205 Specify a standard form linear programme which will sort the list $(2, 7, 3, 5)$ into ascending order.

Solution This is, as the idiom goes, ‘using a sledge hammer to crack a nut’. But it is a good example of how a permutation matrix can be selected as a vertex of the Birkhoff polytope and put to use. We observe that the solution to our sorting problem is given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \\ 5 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 5 \\ 7 \end{pmatrix},$$

so we need a way to identify this particular permutation matrix as being the ‘optimum’ for this unsorted list. So let us take an arbitrary point in the Birkhoff polytope in \mathbb{R}^{16} : $(x_{11}, x_{12}, \dots, x_{44})$. We know from Theorem 204 that, whatever linear optimisation we perform, it will be achieved by a permutation matrix. So the product

$$(1, 2, 3, 4) \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \\ 5 \end{pmatrix} = 2x_{11} + 7x_{12} + \dots + 12x_{43} + 20x_{44},$$

will be a permutation of $(2, 7, 3, 5)$ with the weightings $(1, 2, 3, 4)$ given to its elements. This will be maximised exactly when the input vector is in ascending order: with 2 in the 1st position, 3 in the 2nd position etc. So if we solve the standard form linear program with

$$c = (2, 7, 3, 5, 4, 14, 6, 10, 6, 21, 9, 15, 8, 28, 12, 20),$$

then the result will be

$$x^T = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{21} & x_{22} & x_{23} & x_{24} & x_{31} & x_{32} & x_{33} & x_{34} & x_{41} & x_{42} & x_{43} & x_{44} \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix},$$

which gives the maximum value for the objective function $cx = 2 + 6 + 15 + 28 = 51$.

Example 205 is a simple way of showing the Birkhoff polytope in action, and it illustrated the point that almost any problem can be formulated as a linear programme. However, sorting vectors of length n by assigning values to n^2 unknowns x_{ij} is not a sensible application of linear programming since sorting can be done in worst-case time $n \log n$ using Heapsort, as you may recall from the level 2 unit *Software Engineering, Algorithm Design and Analysis*. A more realistic application will now be given by paying one last visit to the transversal problem.

Suppose, for simplicity we have n sets defined over a ground set of n elements, $\{a_1, \dots, a_n\}$. The sets may be represented by an incidence matrix and we saw in Chapter 2, section 2.4.1 that transversals correspond to permutation matrices whose non-zero elements match non-zero elements in the incidence matrix. So if we take the Mirsky-Perfect representation of a transversal matroid and maximise the sum of its entries subject to these entries satisfying the constraints of the Birkhoff polytope then a transversal will be selected.

Example 206 In Exercise 56, at the end of Chapter 1, you derived a bipartite graph for the Committee Representation problem whose incidence structure was:

$$X_{\mathcal{A}} = \begin{matrix} & \begin{matrix} A & B & C & D & E & F \end{matrix} \\ \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix} & \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{matrix}.$$

We wish to show that a transversal is provided by employees A, B, C and D . Give three matrices from the relevant Birkhoff polytope with just one of them providing an optimal solution to the relevant standard linear programme.

Solution Since we are restricting to a 4×4 submatrix of the above incidence matrix we can follow the approach of Example 205: we apply the constraints of equation (4.6) and optimise the function which assigns zero to those x_{ij} which are zero in our incidence structure and 1 to the others:

$$x_{11} + x_{13} + x_{14} + x_{24} + x_{31} + x_{32} + x_{43} + x_{44}.$$

The following three matrices all satisfy the doubly stochastic constraints of equation (4.6):

$$M_1 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \quad M_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The values of the objective function evaluate as follows:

$$\begin{array}{rcccccccccccl} & x_{11} & + & x_{13} & + & x_{14} & + & x_{24} & + & x_{31} & + & x_{32} & + & x_{43} & + & x_{44} \\ M_1 : & \frac{1}{2} & & \frac{1}{2} & & 0 & & \frac{1}{2} & & \frac{1}{2} & & 0 & & 0 & & \frac{1}{2} & = \frac{5}{2} \\ M_2 : & 0 & & 0 & & 1 & & 0 & & 0 & & 1 & & 0 & & 0 & = 2 \\ M_3 : & 1 & & 0 & & 0 & & 1 & & 0 & & 1 & & 1 & & 0 & = 4 \end{array}$$

with M_3 attaining the maximum value for the objective function and therefore identifying a matching of employees A, B, C and D to the products P_1, \dots, P_4 .

You may recall that in section 2.2.2 of Chapter 2 we had a theorem (Theorem 63) which asserted that the value of the determinant of a square matrix X was $\det X = \sum_P \text{sgn}(P) \Pi(X, P)$, the sum being taken over all permutation matrices P (so that is all vertices of the Birkhoff polytope) and with $\Pi(X, P)$ being the product of all elements of X which matched non-zero entries in P . The linear programme in Example 206 can be rephrased as calculating $\max_P \Pi(X, P)$, where X is the incidence matrix of a set system.

4.4.2 Matroid polytopes

There are many other polytopes with integer vertices. To end with, and as a kind of recapitulation on our main theme, we mention the polytope that arises from a matroid. Suppose we have a matroid $M = (A, \mathcal{I})$ over ground set A , specified in terms of its independent sets \mathcal{I} . We may represent each independent set as a 0-1 incidence vector whose entries are indexed by A . Now the convex hull of these vectors will be a polytope, $P(M)$, called the matroid polytope whose vertices are, by definition, these same vectors. It is thus exactly the kind of polytope with integer-valued vertices that we have just been investigating. Not surprisingly, it satisfies linear programme-type constraints: suppose $|A| = n$ and let x be a vector in \mathbb{R}^n , with entries indexed by A . If $Y = y_1, \dots, y_t$ is a subset of A then write $x(Y)$ for $x_{y_1} + \dots + x_{y_t}$. Then it is not hard to see that any vector x in the polytope $P(M)$ must satisfy:

$$x(Y) \leq \text{rank}(Y), \text{ for each } Y \subseteq A \quad (4.7)$$

$$x_a \geq 0, \text{ for all } a \in A \quad (4.8)$$

$\text{rank}(Y)$ being the matroid rank function (the size of a maximum independent set contained in Y). In the early 70s Jack Edmonds proved these conditions are not only necessary but sufficient to describe the matroid polytope:

Theorem 207 *For any matroid $M = (A, \mathcal{I})$, with $|A| = n$, the set of all vectors in \mathbb{R}^n satisfying conditions (4.7) and (4.8) is precisely the polytope $P(M)$.*

He also proved that, given two matroids M_1 and M_2 on the same ground set A , the intersection of their polytopes is exactly described by these same conditions but with (4.7) replaced by

$$x(Y) \leq \min(\text{rank}_{M_1}(Y), \text{rank}_{M_2}(Y)), \text{ for each } Y \subseteq A. \quad (4.9)$$

It follows that this intersection polytope again has integer-valued vertices. What happens when we intersect three matroids? As we might expect from our investigation at the end of Chapter 3, the polytope which arises is no longer guaranteed to have integer-valued vertices; correspondingly, the conditions of Edmonds' theorem cannot extend to this case.

We have already seen two matroid polytopes, one important, one trivial: the polytope which we constructed for bipartite matching arises in this way. And the polytope in Figure 4.8(c), which came from our linear programme to solve Take Two, is identical to the matroid polytope $P(M)$ for the matroid defined on $A = \{a, b, c\}$ by the independent sets $\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}\}$.

We have come almost full circle, and will end, as we began with an exercise:

Exercise 208 *Sketch the matroid polytope for the matroid on $\{a, b, c\}$ whose bases are $\{a, b\}$ and $\{a, c\}$.*

Appendix A

Further reading

A version of this material will be found on the CD-ROM which accompanies this study guide: web links on the CD-ROM will be clickable which will save you typing them into your browser; they may also be slightly more up-to-date.

A.1 Books

Some of the books below appear because they are cited in footnotes: they are not intended to be a secondary source of study for this unit; the annotation after each entry will indicate if it is recommended for this purpose.

Applegate D.L., Bixby, R.E., Chvatal, V. and Cook, W.J. *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2007, ISBN 0691129932.

A major case study in modern optimisation. The background to the Concorde software found on the CD-ROM accompanying the study guide for this unit.

Bondy, J.A. and Murty, U.R.S *Graph Theory with Applications*, North Holland, 1976, ISBN 0333226941.

An excellent text if you can get a copy (I believe it is now out of print). The same authors have coauthored *Graph Theory*, published by Springer, 2007, ISBN 1846289696, which is excellent but different from, and more advanced than, their 1976 book.

Cameron, P.J. *Introduction to Algebra*, Oxford University Press, 2007, second edition, ISBN 0198527934.

Highly recommended as an introduction to linear algebra and to mathematical concepts generally.

Garey, M.R. and Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H.Freeman & Co Ltd, 1979, ISBN 0716710455.

The classic text on computational complexity, still in print after 30 years of developments in the area. Extremely readable and with what is still one of the most useful compendia of NP-complete problems.

Gowers T., ed. *The Princeton Companion to Mathematics*, Princeton University Press, 2008, ISBN 0691118809.

One of the big mathematics publishing events of the decade. All the main topics of this study guide get coverage but this is a book that anybody involved with mathematics will treasure.

Hillier, F.S. and Lieberman, G.J. *Introduction to Operations Research*, McGraw-Hill Higher Education, 2004, eighth edition, ISBN 007123828X.

Covers the practice of linear programming in much more depth than needed for this unit as well as many standard other OR topics such as queuing theory. Includes CD-ROM of optimisation software.

Jones, A.J. *Game Theory: Mathematical Models of Conflict*, Horwood Publishing Ltd, new edition, 2000, ISBN 1898563144.

Presents game theory, an important topic for OR which is not covered in this guide, with, moreover, a strong (and very clear and self-contained) emphasis on linear programming.

Kolman, B. and Beck, R.E. *Elementary Linear Programming with Applications*, second edition Academic Press, 1995, ISBN 012417910X.

Sound and practical book on linear programming, with CD-ROM containing 'student-oriented' linear programming software SMPX. Gives extensive coverage of integer programming.

Lindley, D.V. *Making Decisions*, John Wiley & Sons, 1985, second edition, ISBN 0471908088.

A book that everybody involved in quantitative management should read.

Mertens, S and Moore, C. *The Nature of Computation*, Oxford University Press, 2010. ISBN 0199233217.

Accessible introduction to the latest developments in computation theory, covering much of the material of Chapter 4 of this study guide.

Nishimura, H. and Kuroda, S., eds. *A Lost Mathematician, Takeo Nakasawa: The Forgotten Father of Matroid Theory*, Birkhäuser, 2009, ISBN 3764385723. A fascinating account of how matroid theory was invented in Japan, simultaneously with and completely independently of, developments in America.

Oxley, J.G. *Matroid Theory*, Oxford University Press, new edition, 2006, ISBN 0199202508.

Now the standard text in the subject. Oxley is currently preparing a new edition, so it might be worth checking if you are thinking of buying.

Papadimitriou, C.H. *Computational Complexity*, Addison Wesley, 1993, ISBN 0201530821.

A very readable and authoritative introduction to computational complexity and NP-completeness.

Very expensive for a paperback, but with luck it will eventually make into a cheap Dover edition like its stable mate *Combinatorial Optimization*.

Papadimitriou, C.H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications Inc, Mineola, New York, 2000, ISBN 0486402584.

A low-cost Dover reprint of a famous text from the 1980s. It covers everything in this study guide except matroid representations with exceptional clarity and with many examples and exercises.

Welsh, D.J.A *Matroid Theory*, Academic Press Inc., U.S., 1976, ISBN 012744050X.

A classic text, now sadly out of print.

White, N. *Matroid Applications*, Cambridge University Press, 1992, ISBN 0521381657.

More advanced than is needed for this course but the first chapter, by Walter Whiteley on “Matroids and Rigid Structures”, is valuable further reading for those intrigued by the application of cographic matroids to rigidity described in section 2.3.4 in Chapter 2.

Wilson, R.J. *Introduction to Graph Theory*, Prentice Hall, fifth edition, 2010, ISBN 027372889X.

An exceptionally clear and authoritative introduction to the subject which moreover contains a chapter on matroid theory.

Winston, W.L. *Operations Research: Applications and Algorithms*, Brooks Cole 1998, fourth edition, ISBN 0534423620.

As with Hillier and Lieberman covers the practice of linear programming in great depth. Includes CD-ROM of optimisation software.

A.2 Journals

You will generally need access to a reasonable university library or to a web facility such as JSTOR to make use of these. I give ISSN's for print versions of journals only but many journals can be found on the web by searching for their names and they will often make some pages, and at least contents pages, available. The website www.optimization-online.org (see under A.3) is a good source of information about electronic publications.

Annals of Operations Research published by Springer Netherlands, ISSN 0254-5330.

Computational Optimization and Applications published by Springer US, ISSN: 0926-6003.

European Journal of Operational Research published by Elsevier, ISSN: 0377-2217.

IEEE Transactions on Evolutionary Computation published by IEEE, ISSN: 1089-778X.

IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans published by IEEE, ISSN: 1083-4427.

International Journal of Mathematics in Operational Research published by InderScience Publishers, ISSN: 1757-5850.

Journal of Combinatorial Optimization published by Springer US, ISSN: 1382-6905.

Journal of the Operational Research Society published by Palgrave Macmillan, ISSN: 0160-5682.

Journal of Optimization Theory and Applications published by Springer, ISSN: 0022-3239.

Mathematical Methods of Operations Research published by Springer, ISSN: 1432-2994.

Mathematical Programming published by Springer Berlin/Heidelberg, ISSN 0025-5610.

Mathematics of Operations Research published by the Institute for Operations Research and the Management Sciences, ISSN: 0364-765X.

Operations Research published by the Institute for Operations Research and the Management Sciences, ISSN: 0030-364X.

Optimization published by Taylor & Francis, ISSN: 0233-1934.

SIAM Journal on Optimization published by the Society of Industrial and Applied Mathematics, ISSN: 1052-6234.

A.3 Websites

Websites are notoriously ephemeral things. I have tried to confine myself to the home pages of sites which appear reasonably permanent. I have excluded very many beautiful contributions posted by individuals; such links tend to change as people move on from, or reorganise, their virtual filing cabinets.

It is good practice, when citing web pages, to specify a date of access, as an indication of how up-to-date the reference is likely to be. All the websites listed below were accessed in July 2010. As we mentioned above, the copy of the list given on the CD-ROM accompanying this study guide may be more recent.

All URLs below are understood to be prefaced by 'http://'. This may generally be omitted*, even for non-standard URLs such as <http://jgaa.info>.

www.ams.org/featurecolumn The Feature Column of the American Mathematical Society, currently edited by David Austin, Bill Casselman, Joe Malkevitch and Tony Phillips, has a wonderful archive going back ten years, with well over 100 authoritative articles to download.

www.claymath.org/millennium The problem $P = NP$ is the fourth of seven \$1 million prizes offered by the Clay Institute. The official description of the problem is by Steve Cook and is a very clear and succinct overview at a more technical level.

www.combinatorics.org The Electronic Journal of Combinatorics which publishes peer-reviewed, free-to-view articles. The home page also has useful links to related resources.

code.google.com/p/sympy home page for the SymPy symbolic computation project led by Ondřej Čertík, a version of which is packaged on the CD-ROM accompanying this guide.

www2.informs.org/Resources The resources collection of the Institute for Operations Research and the Management Sciences.

www.ifors.org Website of the International Federation of Operational Research Societies.

jgaa.info The Journal of Graph Algorithms and Applications, which offers peer-reviewed, free-to-view articles. Volumes 1-8 have been published in print version by World Scientific.

ocw.mit.edu/courses/mathematics/ Authoritative lecture notes from, at the last count, about 50 undergraduate mathematics courses and perhaps the same number of graduate courses.

www.mathprog.org/ The Mathematical Programming Society: supports the mathematical end of the OR profession. Awards the Fulkerson Prizes, one of the most prestigious in the field, every three years at its International Conference.

www.numericana.com A well-known site created by Gérard P. Michon, offering help and information on hundreds of mathematical topics.

*I once attended a presentation by Tim Burners-Lee at the Royal Society in London. When asked what he would have done differently, with the benefit of hindsight, when inventing the world-wide-web, he replied "I would have left out the http!".

www.optimization-online.org a repository of e-prints about optimization and related topics.

www.scienceofbetter.org website provided by INFORMS (see above) to promote their tag-line ‘The Science of Better’.

www.theoremoftheday.org A collection of one-page descriptions of theorems, usually with a concrete illustration. Many of these are to do with combinatorics and computer science.

www.theorsociety.com The OR Society. This link appears, rather disconcertingly, to take you directly to their shop. Don’t be put off — there are dozens of links to useful information.

www.tsp.gatech.edu/index.html The essential website for all things regarding the Travelling Salesman Problem. A version of the Concorde software by William Cook and others is available on the CD-ROM accompanying this study guide.

www2.isye.gatech.edu/~wcook/qsopt/ The home page for the QSopt linear programming solver at Georgia Institute of Technology.

Appendix B

Specimen examination

B.1 Examination paper

This paper is intended to be representative of the examination you will take when you have studied this unit.

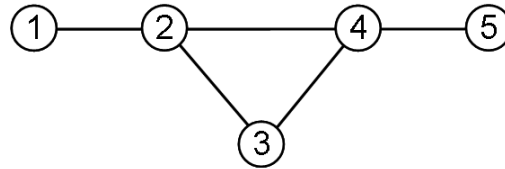
Model answers are provided immediately after the specimen examination paper. You will derive most benefit from the paper if you make a serious attempt at each question before looking at the model answer.

Note that details of the examination process may change from year to year and you are strongly advised to check the latest information provided by the University of London before sitting the examination for this unit.

Specimen rubric: Answer ALL FIVE questions. Each question carries 20 marks and there are a total of 100 marks available on this paper.

1. Let G be a graph with vertex set V . Let $\mathcal{A}(G)$ be the collection of sets consisting of all those subsets X of V such that G has a matching whose vertex set contains X .

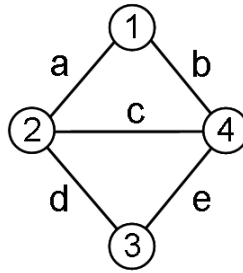
Let G be the graph depicted below:



- (a) Show that both of $\{1, 2, 3\}$ and $\{1, 4, 5\}$ are in $\mathcal{A}(G)$; [3]
 - (b) Show that the set $\{1, 3, 5\}$ is *not* in $\mathcal{A}(G)$; [3]
 - (c) If $X = \{1, 3\}$ and $Y = \{1, 4, 5\}$ explain how X and Y satisfy the exchange axiom for matroids. [6]
 - (d) Given that $\mathcal{A}(G)$ specifies the independent sets of a matroid (the *matching matroid*) with ground set V and supposing that the label of a vertex is also taken to be its weight (vertex 1 has weight 1, etc), describe the steps by which the greedy algorithm will construct a maximal independent set of minimum weight. [8]

2. Suppose we have a collection of subsets $\mathcal{A} = \{A, B, C\}$ of the set $X = \{a, b, c, d\}$, where A, B and C are given by: $A = \{a, b, c\}$, $B = \{b, c\}$, and $C = \{d\}$.
 - (a) State the definition of a *partial transversal* of \mathcal{A} ? [3]
 - (b) Which of the sets $\{a, b, c\}$ and $\{b, c, d\}$ is a basis for the transversal matroid $M(\mathcal{A})$. Justify your answer. [6]
 - (c) Write down the Mirsky-Perfect representation for $M(\mathcal{A})$. By calculating the determinants of suitable submatrices of this representation, list all complete transversals for \mathcal{A} , stating which subset of \mathcal{A} is represented by which element of X . [11]

3. The following graph shows a small railway network:



Three train companies R , S and T operate on this network, using the links shown:

$$\begin{aligned} R &: a, b \\ S &: c, d \\ T &: a, d, e \end{aligned}$$

It is desired to select a minimal subset of the links which will allow goods to be transported throughout the network. For business reasons it is also desired to use each of the train companies on at least one link in the subset.

- (a) Describe how this problem may be solved as the intersection of two matroids M_1 and M_2 . Explain briefly why these matroids may be represented, respectively, by the following two matrices:

$$X = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \end{matrix} \qquad Y = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} R \\ S \\ T \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

[6]

- (b) Prove that the collection of subsets of links which are simultaneously bases in both M_1 and M_2 is specified by the determinant of the matrix

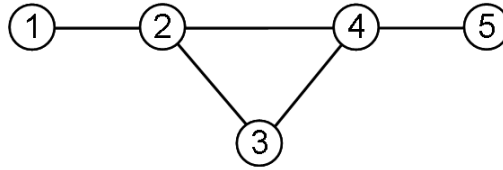
$$\begin{pmatrix} a+b & 0 & a \\ -a & c+d & d-a \\ 0 & -d & e-d \end{pmatrix}.$$

[8]

- (c) By calculating the value of the determinant in part (b) or otherwise, give all solutions to the problem of selecting a suitable subset of railway links.

[6]

4. (a) For the following graph



explain why the vertex subset $\{1, 3, 4\}$ is a **minimal** but not **minimum** vertex cover. [5]

- (b) Suppose that x_1, x_2, x_3 and x_4 are Boolean variables with negations x'_1, x'_2, x'_3, x'_4 . Explain why setting $x_i = T$, for $i = 1, \dots, 4$, where T stands for True, is not a satisfying truth assignment for the 3-SAT instance

$$(x_1, x_2, x_3), (x'_1, x'_2, x'_4), (x'_2, x'_3, x'_4). \quad [2]$$

- (c) Explain briefly the significance of 3-SAT in the theory of computational complexity. [3]

- (d) Construct a graph G on 26 vertices in which any minimum vertex cover corresponds to a satisfying truth assignment for the 3-SAT instance in part (b). Give a satisfying truth assignment for this instance and indicate on your graph the corresponding minimum vertex cover. [10]

5. (a) Write down the dual of the linear programme

$$\begin{array}{ll} \text{minimise} & 5w + x + 9y - z \\ \text{subject to} & w - x + y - 3z \geq -2 \\ & w + x + 3y + z \geq 2 \\ & w, x, y, z \geq 0 \end{array} \quad [8]$$

- (b) By sketching the polytope for the dual programme you found in part (a), find a vector which optimises its objective function and state the optimal value. Hence give the optimal value for the primal programme by appealing to the Duality Theorem of Linear Programming. [12]

B.2 Model answers

1. (a) Edges 12 and 43 are disjoint and so are a matching which contains $\{1, 2, 3\}$. Edges 12 and 45 are a matching which contains $\{1, 4, 5\}$. [3]
- (b) To contain $\{1, 3, 5\}$ in a matching we need to have edges 12 and 45 present since these are the only edges containing 1 and 5. But this excludes edges 23 and 34 which are not disjoint from 12 and from 45, respectively. But these are the only edges containing 3. So $\{1, 3, 5\}$ cannot be contained in a matching. [3]
- (c) The exchange axiom says that there is $y \in \{1, 4, 5\} \setminus \{1, 3\}$ such that $\{1, 3\} \cup \{y\}$ is independent. [3]
We saw that we cannot add $\{5\}$ to $\{1, 3\}$ and keep independence, so we must check that $\{1, 3, 4\}$ is independent. But edges 12 and 34 are a matching containing $\{1, 3, 4\}$ so we are done. [3]
- (d) The Greedy Algorithm will start with the empty independent set $X = \emptyset$. Now we add minimum-weight elements under the constraint of preserving independence: [2]
 add 1 since $\{1\}$ is independent and 1 has minimum weight;
 add 2 since $\{1, 2\}$ is independent (e.g. edge 12) and 2 is min weight;
 add 3 since $\{1, 2, 3\}$ is independent (edges 12, 34) and 3 is min weight;
 add 4 since $\{1, 2, 3, 4\}$ is independent (edges 12, 24) and 4 is min weight. [4]

Now the algorithm terminates since we cannot add 5 to an independent set containing 1 and 3 (as seen in part (b)). [2]

2. (a) A partial transversal of \mathcal{A} is a subset U of X such that there is a 1–1 mapping $\phi : U \rightarrow \mathcal{A}$ satisfying $x \in \phi(x)$ for all $x \in U$. [3]
- (b) The set $\{b, c, d\}$ is a basis for $M(\mathcal{A})$ since we can specify a mapping ϕ by $\phi(b) = A$, $\phi(c) = B$ and $\phi(d) = C$ and ϕ satisfies the requirement for a partial transversal. [3]
The set $\{a, b, c\}$ cannot be a transversal since the required mapping ϕ cannot map any element to C . So all three elements will be mapped to $\{A, B\}$ which prevents the mapping from being 1–1. [3]
- (c) The Mirsky-Perfect representation is

$$\begin{array}{c} a \quad b \quad c \quad d \\ A \quad \begin{pmatrix} t_{11} & t_{12} & t_{13} & 0 \\ 0 & t_{22} & t_{23} & 0 \\ 0 & 0 & 0 & t_{34} \end{pmatrix} \\ B \\ C \end{array}$$

If \mathcal{A} has complete transversals then this matrix must have non-singular 3×3 submatrices. All such submatrices must include column d to prevent their last row being zero. [2]

So we have:

$$\begin{array}{c} a \quad b \quad d \\ \det \begin{pmatrix} A & t_{11} & t_{12} & 0 \\ B & 0 & t_{22} & 0 \\ C & 0 & 0 & t_{34} \end{pmatrix} = t_{11}t_{22}t_{34}, \text{ giving } a \in A, b \in B, d \in C \\ a \quad c \quad d \\ \det \begin{pmatrix} A & t_{11} & t_{13} & 0 \\ B & 0 & t_{23} & 0 \\ C & 0 & 0 & t_{34} \end{pmatrix} = t_{11}t_{23}t_{34}, \text{ giving } a \in A, c \in B, d \in C \\ b \quad c \quad d \\ \text{and } \det \begin{pmatrix} A & t_{12} & t_{13} & 0 \\ B & t_{22} & t_{23} & 0 \\ C & 0 & 0 & t_{34} \end{pmatrix} = (t_{12}t_{23} - t_{13}t_{22})t_{34} = t_{12}t_{23}t_{34} - t_{13}t_{22}t_{34}, \\ \text{giving } b \in A, c \in B, d \in C \text{ and } c \in A, b \in B, d \in C, \text{ respectively.} [7] \end{array}$$

3. (a) A minimum weight subset of edges connecting the network will be a spanning tree. This can be found using the cycle matroid of the graph: M_1 . In order to represent each train company we must choose a complete transversal of the subsets of links used by the different companies. This can be found using the transversal matroid of this collection of subsets: M_2 . [3]

Now the cycle matroid is represented by the incidence matrix. We recognise that X is the incidence matrix, with its last row deleted since the incidence matrix has rank 3. The transversal matroid is represented by the incidence matrix of the subsets but with some non-zero entries possibly taking values other than 1. The matrix Y is this incidence matrix and we can confirm that the non-singular 3×3 submatrices do indeed correspond to complete transversals. [3]

$$X = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \end{matrix} \quad Y = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} R \\ S \\ T \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

- (b) To find the intersection of M_1 and M_2 we form the Binet-Cauchy Intersection:

$$\begin{aligned} X \times \text{diag}(a, b, c, d, e) \times Y^T &= \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \times \text{diag}(a, b, c, d, e) \times \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} a & b & 0 & 0 & 0 \\ -a & 0 & c & d & 0 \\ 0 & 0 & 0 & -d & e \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} a+b & 0 & a \\ -a & c+d & d-a \\ 0 & -d & e-d \end{pmatrix}. \end{aligned} \quad [4]$$

- (c) Now the maximal sets independent in both M_1 and M_2 are enumerated by the determinant of this 3×3 matrix: [1]

$$\begin{aligned} \det \begin{pmatrix} a+b & 0 & a \\ -a & c+d & d-a \\ 0 & -d & e-d \end{pmatrix} &= (a+b)((c+d)(e-d) + d(d-a)) + a^2d \\ &= (a+b)(ce - cd + de - ad) + a^2d \\ &= ace - acd + ade - a^2d + bce - bcd + bde - abd + a^2d \\ &= ace - acd + ade + bce - bcd + bde - abd \end{aligned} \quad [4]$$

These five products each correspond to a solution to the railway problem. [1]

4. (a) $\{1, 3, 4\}$ is minimal since if we remove any element we no longer have cover:

$\{1, 3\}$ does not cover edge 45

$\{1, 4\}$ does not cover edge 23

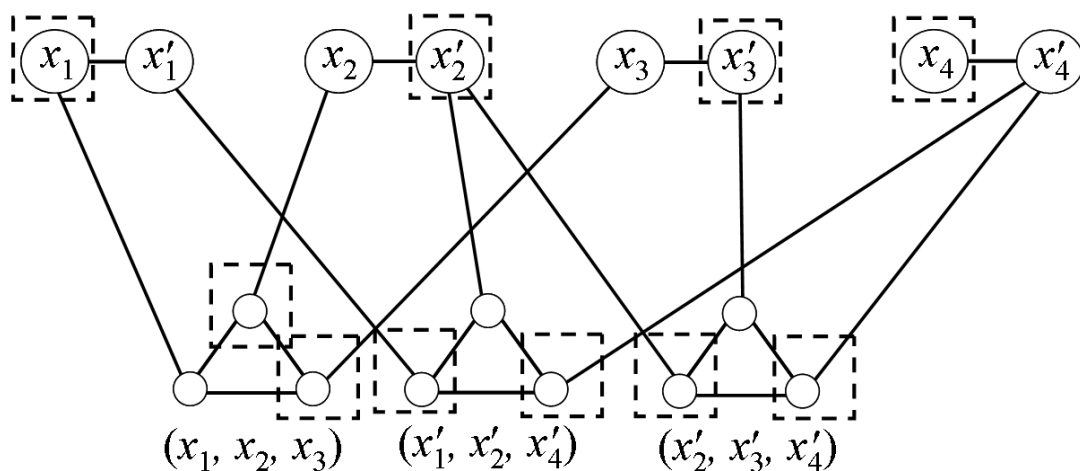
$\{3, 4\}$ does not cover edge 12 [3]

It is not minimum since $\{2, 4\}$ is a vertex cover of smaller cardinality. [2]

- (b) If $x_1 = x_2 = x_3 = x_4 = T$ then clause 2, (x'_1, x'_2, x'_4) evaluates to (F, F, F) and is not satisfied. So this is not a satisfying truth assignment. [2]

- (c) 3-SAT is **NP**-complete—it is as hard as any problem in **NP**, in the sense that a polynomial algorithm for 3-SAT would imply one for all problems in **NP**. [3]

- (d) The graph required is:



[6]

A minimum vertex cover (of size 10) is indicated by boxed vertices. In particular there is a vertex for each Boolean variable and its negation and among these the boxed vertices indicate the truth assignment:

$$x_1 = x_4 = T, \quad x_2 = x_3 = F.$$

[4]

5. (a) Write the primal programme as

$$\text{minimise } cx \text{ subject to } Ax \geq b,$$

where $c = (5, 1, 9, -1)$ and $A = \begin{pmatrix} 1 & -1 & 1 & -3 \\ 1 & 1 & 3 & 1 \end{pmatrix}$ and $b = (-2, 2)^T$. Then the dual programme is:

$$\text{maximise } b^T y \text{ subject to } A^T y \leq c,$$

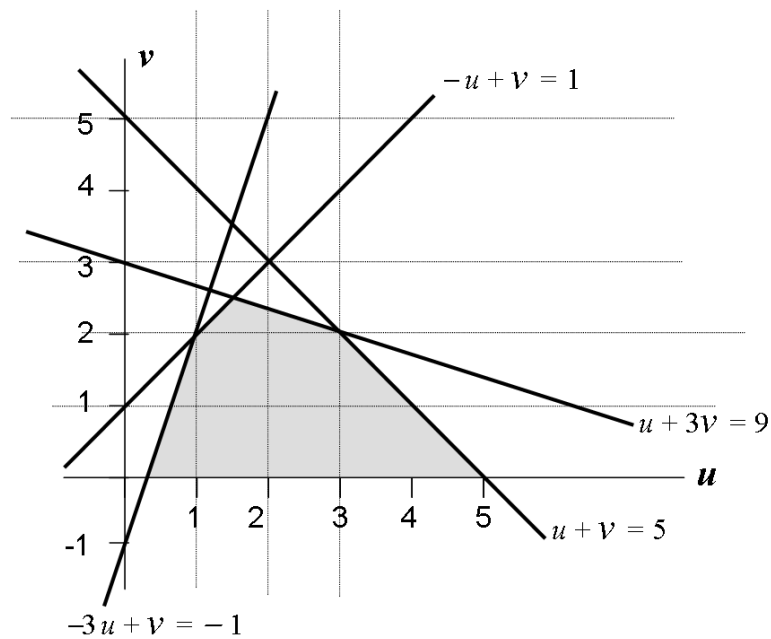
[4]

thus:

$$\begin{array}{ll} \text{maximise} & -2u + 2v \\ \text{subject to} & u + v \leq 5 \\ & -u + v \leq 1 \\ & u + 3v \leq 9 \\ & -3u + v \leq -1 \\ & u, v \geq 0 \end{array}$$

[4]

- (b) The polytope is as shown here:



[6]

We know that the objective function is maximised at a vertex. The vertices are

$$(0, \frac{1}{3}), (1, 2), (\frac{3}{2}, \frac{5}{2}), (3, 2), (5, 0),$$

(clockwise from bottom left), with values of the objective function $-2u + 2v$ respectively:

$$2/3, 2, 2, -2, -10.$$

So either $(1, 2)$ or $(\frac{3}{2}, \frac{5}{2})$ will maximise the objective function.

[4]

Now the Duality Theorem of Linear Programming says that the objective function for the primal programme minimises to the same value as the optimum for the dual. So this will also be 2.

[2]

Appendix C

Solutions to exercises

Solutions appear below grouped by chapter and numbered as they appear in the text. The fact that Exercise 3 is followed by Exercise 5 indicates that a definition or theorem has intervened, not that solutions have been omitted.

C.1 Solutions for Chapter 1

Exercise 2 No — because \mathbb{R}^3 is a collection of triples, whereas \mathbb{R}^5 contains only 5-tuples. For \mathbb{R} the answer is still no — the individual entries in a vector might be real numbers but this does not make \mathbb{R} a subset of \mathbb{R}^n for any $n > 1$.

Exercise 3

- (a) $(1, 1)$ in \mathbb{R}^2 ;
- (b) $-2 \times (2, 0, 1, 0, 2, 1)$ in \mathbb{R}^6 ;
- (c) 0.424 in \mathbb{R} ;
- (d) 8.540 in \mathbb{R} , to three decimal places (written more explicitly, the product is $3.142 \times (2.718)$ but the distinction between \mathbb{R} and \mathbb{R}^1 is often blurred).

Exercise 5

- (a) Yes, since $22/7$ and $-\pi$ are scalars in \mathbb{R} ;
- (b) Yes, since 0 is a scalar in \mathbb{R} . (Note that the sum here is $(0, 0, 0)$, but this does not show that $(1, 1, 1)$ and $(-1, 0, 0)$ are linearly dependent since neither scalar in the weighted sum is nonzero.)
- (c) Yes, since 4 is a scalar in \mathbb{R} (the other vector is assumed to be present but multiplied by zero);
- (d) No, since the vectors are not the same length — the addition is not defined;
- (e) Yes, since x^2 and $-y^2$ are scalars in \mathbb{R} ;
- (f) No, unless we define ‘squaring a triple’ by $(a, b, c)^2 = (a^2, b^2, c^2)$ (this would be non-standard);
- (g) No, unless x and y are both non-negative, since $\sqrt{-1}$, for example, is not a scalar in \mathbb{R} ;

Exercise 7 If $a_1 \neq 0$ then $(-1, -2) + (a_2/a_1) \times (0, 3) = (0, 0)$ which is impossible since the first components give $-1 + 0 = 0$.

Exercise 9 The three vectors are as shown in fig. C.1(a). When multiplied by the scalars derived in Example 8, we have the vectors shown in fig. C.1(b). The sum, obtained by ‘sliding’ $(0, 2/3)$ along $(2, -4)$ and then $(-2, 2)$ along both vectors, gives an arrow pointing *towards* the origin — the zero vector.

Exercise 12 You should get something like:

$$\begin{pmatrix} 2 & 3 & 5 \\ 7 & 11 & 13 \\ 17 & 19 & 23 \end{pmatrix} \xrightarrow[R3 - \frac{17}{2} \times R1]{R2 - \frac{7}{2} \times R1} \begin{pmatrix} 2 & 3 & 5 \\ 0 & \frac{1}{2} & -\frac{9}{2} \\ 0 & -\frac{13}{2} & -\frac{39}{2} \end{pmatrix} \xrightarrow{R3 + 13 \times R2} \begin{pmatrix} 2 & 3 & 5 \\ 0 & \frac{1}{2} & -\frac{9}{2} \\ 0 & 0 & -78 \end{pmatrix}.$$

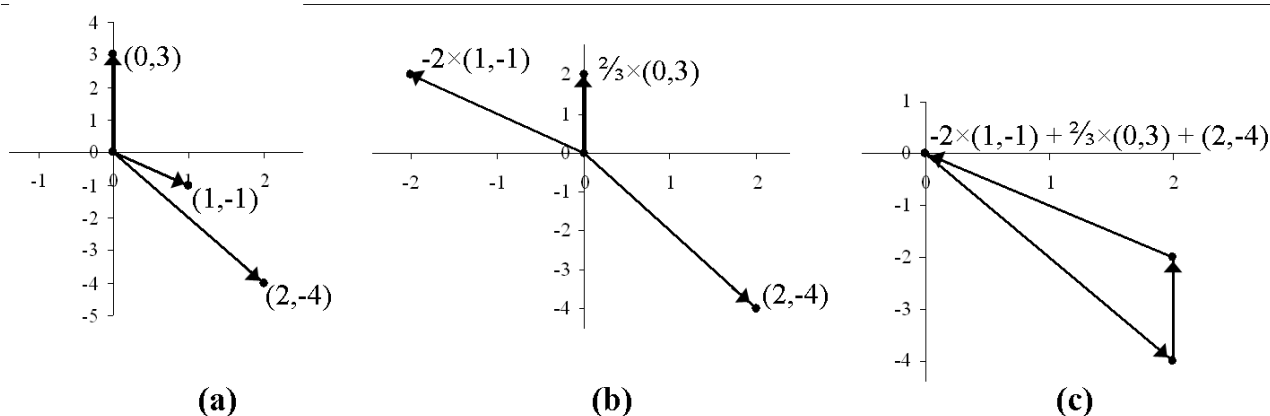


Figure C.1: linearly dependent vectors.

Exercise 16 It is a good idea to start by getting zeros in the first column:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \xrightarrow{\substack{R2-5\times R1 \\ R3-9\times R1 \\ R4-13\times R1}} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & -8 & -24 & -36 \\ 0 & -12 & -24 & -36 \end{pmatrix}.$$

Indeed, we see at once that no independent set of size three or more can contain the first vector, which can produce collinearity among any pair out of the other three. This easily leads to zero rows:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & -8 & -16 & -24 \\ 0 & -12 & -24 & -36 \end{pmatrix} \xrightarrow{\substack{R3-2\times R2 \\ R4-3\times R2}} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Rows 1 and 2, however, are not collinear, so they are linearly independent. This suggests that there is only ‘room’ for two vectors in any linearly independent subset of A . Another way of looking at it: we could have started by subtracting off multiples of *any* row. So we can argue that *no* vector can belong to an independent set of size three or more. Our matroid consists of all subsets of A of size at most two.

Exercise 19 The question asks us to find three scalars A , B and C , not all zero, such that

$$A(a_{11}v_1 + a_{12}v_2) + B(a_{21}v_1 + a_{22}v_2) + C(a_{31}v_1 + a_{32}v_2) = 0. \quad (\text{C.1})$$

Rearranging the left-hand side by linearity gives:

$$(Aa_{11} + Ba_{21} + Ca_{31})v_1 + (Aa_{12} + Ba_{22} + Ca_{32})v_2.$$

But we are told that both the first and second components of the vector sum

$$b_1(a_{11}, a_{12}) + b_2(a_{21}, a_{22}) + b_3(a_{31}, a_{32})$$

are zero, i.e. $b_1a_{11} + b_2a_{21} + b_3a_{31} = 0$ and $b_1a_{12} + b_2a_{22} + b_3a_{32} = 0$. So setting $A = b_1$, $B = b_2$ and $C = b_3$ will satisfy equation C.1.

Exercise 20 Suppose that we could find two bases X and Y of the vector space with $|X| < |Y|$. Suppose Y' is some subset of Y of size $|X| + 1$. By the Steinitz Exchange Lemma, we can find $y \in Y'$ with $y \notin X$ and $X \cup \{y\}$ linearly independent. But this contradicts the maximality of X , so X and Y must have the same size.

Exercise 23 Abbreviate the colours to 'r', 'b', 'g' and 'p'. Two possible matroids (A, \mathcal{I}) with $|\mathcal{I}| = 8$ have

$$\begin{aligned}\mathcal{I}_1 &= \{\emptyset, \{r\}, \{b\}, \{g\}, \{p\}, \{r,b\}, \{r,g\}, \{r,p\}\} \text{ and} \\ \mathcal{I}_2 &= \{\emptyset, \{r\}, \{b\}, \{g\}, \{p\}, \{r,b\}, \{g,b\}, \{p,b\}\}.\end{aligned}$$

These are isomorphic under the relabelling which swaps the colours red and blue.

Exercise 24 We have $\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{4\}, \{6\}, \{1, 2\}, \{1, 4\}, \{1, 6\}\}$. Notice that this is isomorphic to the matroids in Exercise 23, one possible relabelling being $r \rightarrow 1, b \rightarrow 2, g \rightarrow 4, p \rightarrow 6$.

Exercise 26 The row rank of the matrix is 2 since it is in row echelon form and has no zero rows. This is also the column rank by Corollary 14, so $\text{rank}(M) = 2$. Columns 1, 2 and 3 again form a matrix in row echelon form, so this is a subset of columns having rank 2; it is not a basis since it cannot be an independent set, having more columns than its rank. Columns 2 and 3, are collinear, so this is a subset of rank 1 and cannot be a basis. Notice that this means that columns 1 and 2 must form a basis (in fact they are the two unit vectors of \mathbb{R}^n and form the so-called *standard basis*). Finally, columns 4 and 5 are not collinear, since one is not a multiple of the other, so they form a subset of rank 2; this is also a maximal independent set and is therefore a basis for M .

Exercise 29 Closed walks: **cac** length 2, weight 4;

cbc length 2, weight 4;

cdc length 2, weight 8;

Cycles: **cbac** 3 cycles of length 3, weights $2 + 1 + 2 = 5$, $2 + 1 + 2 = 5$ and $2 + 3 + 2 = 7$;

cadc 2 cycles of length 3, weights $2 + 2 + 4 = 8$ and $2 + 3 + 4 = 9$;

cbdc 1 cycle of length 3, weight $2 + 6 + 4 = 12$;

the reverses of these cycles: **cabc**, **cdac**, **cdbc**.

Exercise 30 There are six spanning forests of weight 4, as shown in Figure C.2 (each one is a disconnected subgraph containing all four vertices, regardless of the number of edges, so each is spanning but none is a tree).

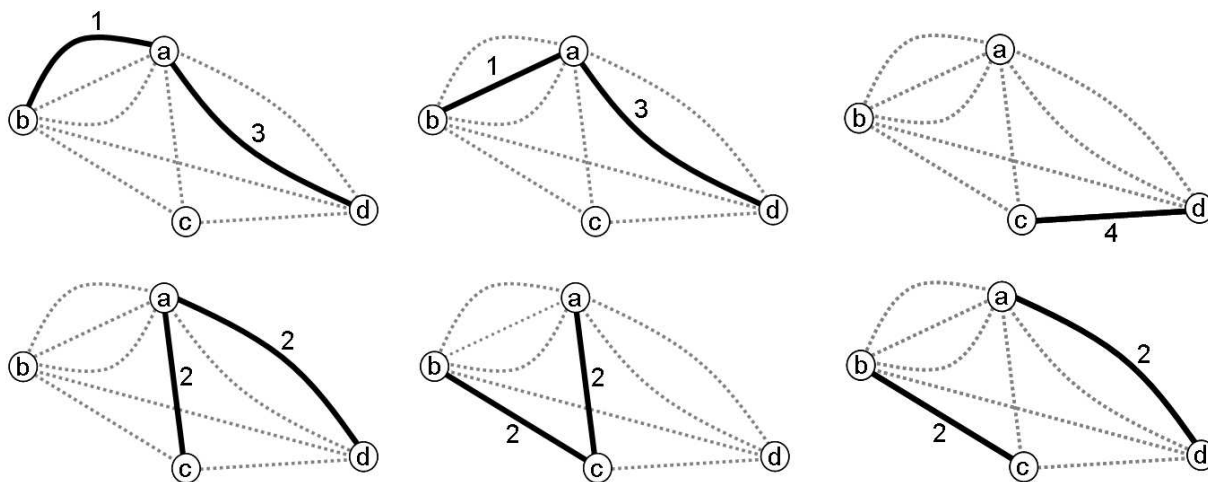


Figure C.2: spanning forests of weight 4 in the graph of Exercise 32.

Exercise 32 The other three minimum-weight spanning trees are as shown in Figure C.3.

Exercise 33 There are many possible solutions, one being shown in Figure C.4. All solutions have edge weights summing to 7.25.

Exercise 34 One possible set of rankings is shown in Figure C.5 (only for two of the solutions to Exercise 32, since the other is identical for the purposes of this exercise). The rankings (a) identify a single

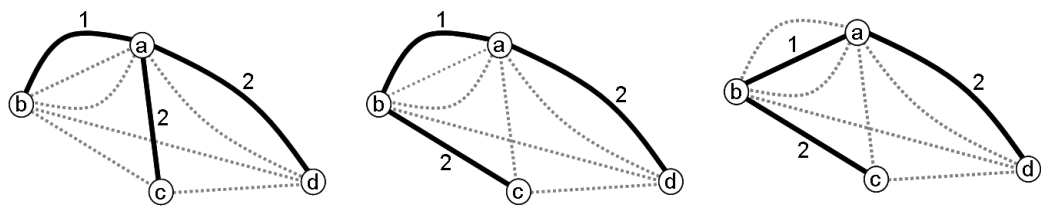


Figure C.3: minimum weight spanning trees for Exercise 32.

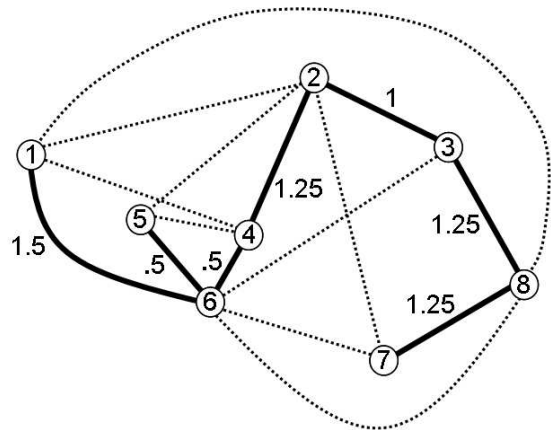


Figure C.4: minimum weight spanning tree for Exercise 33.

‘worst’ product to eliminate. For the rankings (b) a coin toss is needed to decide between b and d for elimination. A different approach might be simply to pair up the products, make a comparison between each pair and then choose at random between the losers of each comparison. This will involve only roughly half as many comparisons as the minimum spanning tree approach but will usually mean elimination is done on the basis of much less information. These are the kinds of trade-offs that OR professionals must negotiate with their clients.

Exercise 35 Applying the greedy approach to the graph of Figure 1.2 will first select edge ab of cost 1 and then edge cd of cost 4. This gives a perfect matching of cost 5. However, a non-greedy approach could first choose edge bc with cost 2 and then edge ad also with cost 2, giving a perfect matching of lower total cost 4.

Exercise 37 We examine each vertex to establish its degree, as shown in Figure C.6.

vertex	1	2	3	4	5	6	7	8
degree	4	5	3	4	3	6	3	4
degree with edge 37	4	5	4	4	3	6	4	4

In each case the number of vertices of odd degree is even. Adding the extra edge decreased the number by 2; had it been added between vertices of even degree the number would have *increased* by 2; had it been added between vertices with different parity degrees the number would remain unchanged.

Exercise 38 If you have tried to construct an Euler trail in the graph of Figure 1.7 you may have come to realise that having four vertices of odd degree must always cause you to fail. For two of these vertices must occur somewhere in the middle of the trail; the trail may pass these vertices several times but each time it must enter and leave by a different edge, increasing the degree by an even number; so these vertices cannot have odd degree.

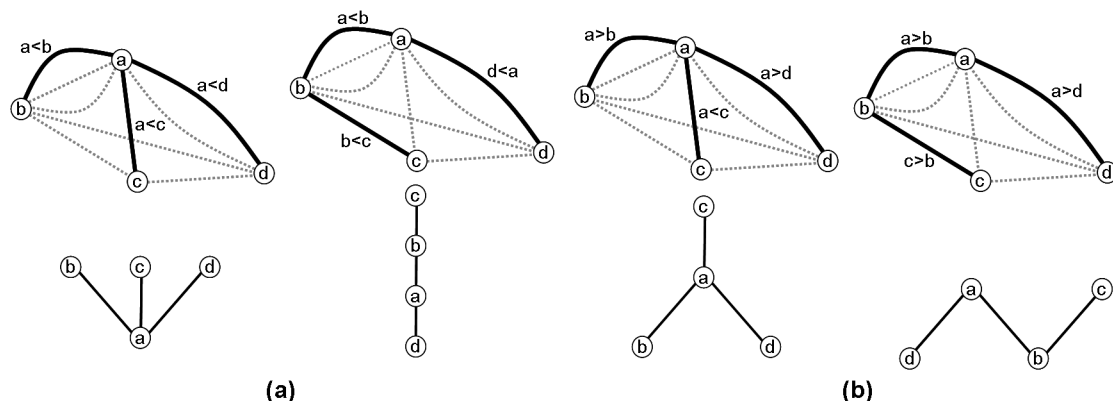


Figure C.5: solution to Exercise 34.

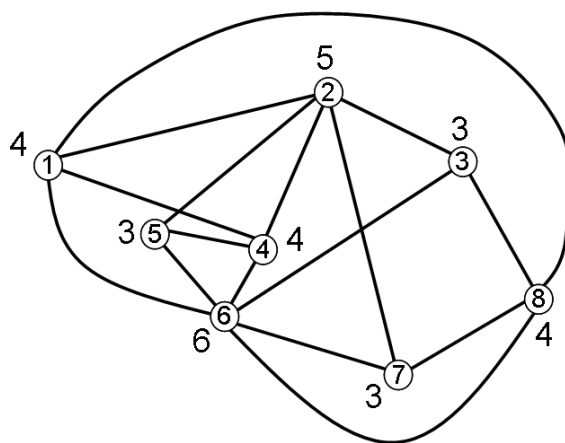


Figure C.6: graph of Figure 1.7 with vertex degrees.

With the edge 37 added there are two vertices of odd degree, 2 and 5. An Euler trail beginning at vertex 2 is:

245641678123863725.

Exercise 39 We arrange the vertex degrees in ascending order:

i	1	2	3	4	5	6	7	8
d_i	3	3	3	4	4	4	5	6
d_i with edge 37	3	4	4	4	4	4	5	6

Without the edge 37 we find that $d_3 = 3 \not\geq 3$ and $d_{8-3} = 4 \not\geq 5$. So the condition of Chvátal's Theorem fails to hold. However, with the edge 37 in place we have $d_1 > 1$, $d_2 > 2$ and $d_3 > 3$, so the condition holds. A Hamilton cycle in the original graph is

123876541.

Now this cycle consists of eight edges and we can take alternate edges to form one perfect matching, with the remaining edges forming another.

Exercise 40 Suppose there are three edge-disjoint matchings. We can think of them as being colourings of certain edges either red, blue or green. Now consider the hexagon formed by the cycle 2387652. The vertices 3, 5 and 7 have degree 3, so every edge incident with these vertices must be either red, blue or

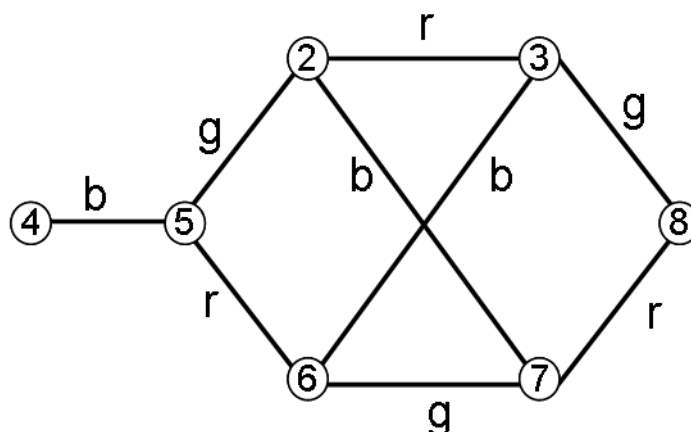


Figure C.7: partial matchings for Exercise 40.

green. Since the order of the colours does not matter we can fix the colours at one of these vertices; let us say that the edges incident with vertex 3 are coloured thus: 23=red, 36=blue and 38=green. Now notice that the edge 78 must be either red or blue. Suppose it is red. Then the colours of the remaining edges of the hexagon (and its chords) are forced to be as shown in Figure C.7. But we see that red or green each form perfect matchings for the hexagon so that a perfect matching for the whole graph must add a red edge from 1 to 4 and a green edge from 1 to 4 which is impossible. Colouring edge 78 blue instead similarly leads to a situation where it is impossible to complete the three perfect matchings, so no such matchings can be found.

Exercise 43 Although the maximal sets of \mathcal{A} are both maximum (size 2), this is not enough on its own to guarantee that the greedy algorithm will work. If $w(a) = w(c) = 1$, $w(b) = 3$ and $w(d) = 2$ then the greedy algorithm may first pick a . It must then add b since $\{a, b\}$ is the only maximal set of \mathcal{A} containing a . This is not an optimal solution since $\{c, d\}$ is a maximal set of lower weight. Since w is a choice of weight function for which the greedy algorithm fails, \mathcal{A} cannot be a matroid by Theorem 42. Indeed, the choice of $X = \{a\}$ and $Y = \{c, d\}$ fails to satisfy axiom 3 of definition 21.

Exercise 45 $\mathcal{I} = \{\emptyset, \{x\}, \{y\}, \{z\}, \{xy, yz\}, \{xz, yz\}, \{xy, xz\}\}$.

Exercise 47 The sequence of edges added is shown in Figure C.8

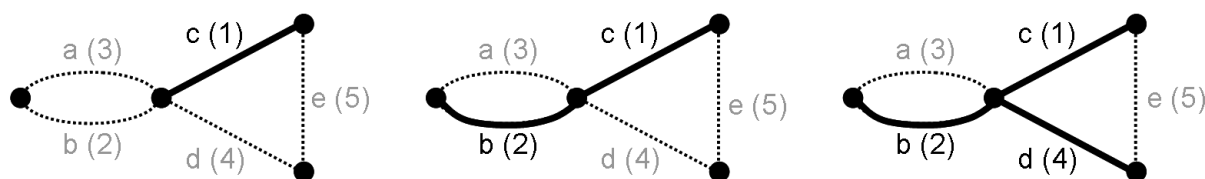


Figure C.8: sequence of partial spanning trees for Exercise 47.

Exercise 48 If G is the graph in Figure C.9 then $M(G)$ is the relevant matroid.

Exercise 49 In the cycle matroids of the two graphs in Figure C.10 both $\{b\}$ and $\{c\}$ are dependent since the corresponding edges are self-loops, forming non-trivial cycles. So $\{a\}$ is the only nonempty set which is independent.

Exercise 51 The subset collection is $\mathcal{A} = \{\{W_1, W_2, W_5, W_6\}, \{W_2, W_3\}, \{W_1, W_4, W_5\}, \{W_5, W_6\}\}$. A cheapest transversal is $\{W_1, W_2, W_3, W_5\}$. A most expensive one is $\{W_3, W_4, W_5, W_6\}$.

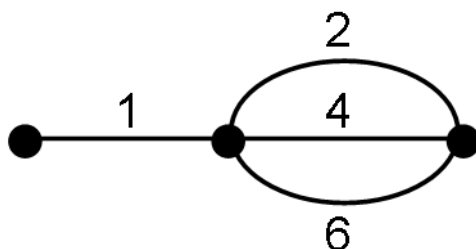


Figure C.9: graph whose cycle matroid is isomorphic to that given in Exercise 48.

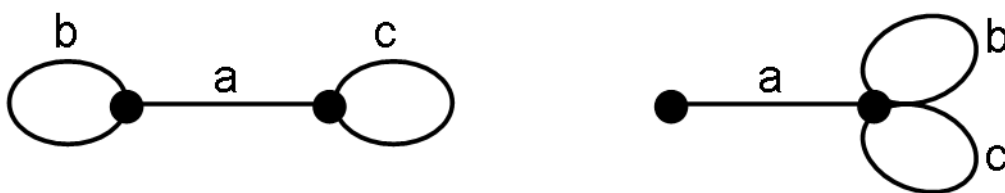


Figure C.10: two graphs with isomorphic cycle matroids (exercise 49).

Exercise 53 No complete transversal T for \mathcal{A} is possible since A_1 , A_3 and A_4 between them share only 2 distinct elements of A : we cannot represent all three sets in T using these two elements. The partial transversal $\{a, b, c\}$ is therefore maximal, with a representing A_1 , b representing A_2 and c representing A_3 . However, a cheaper maximal transversal is $\{a, c, d\}$, with a representing A_2 , c representing A_3 and d representing A_1 .

Exercise 56 The bipartite graph will have four vertices for the products and six for the sales representatives, connected as shown in Figure C.11.

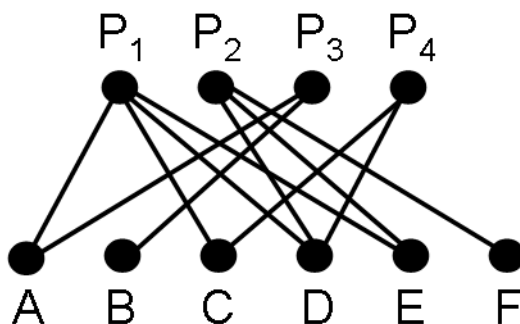


Figure C.11: bipartite graph for exercise 56 showing experience with different products.

A transversal for the sets of people adjacent to each product will give us a representative on our committee for each product. If, in addition, we weight each person according to how many different products they have experience of, then a transversal of *maximum* weight will form the ideal committee. Since Dominique knows three products (vertex D has degree 3) we greedily chose her first, followed by Abdul, Cheung and Enrique (vertices of degree 2). This gives a maximum weight transversal: $\{A, C, D, E\}$. Notice that we *cannot* assign its members greedily to products; this is the matching problem which our matroid does not solve. So, for instance, if A is asked to represent P_1 on our committee, C to represent P_4 , and D to represent P_2 , then this leaves E unable to represent any product they have experience with. A perfect matching pairs A with P_3 , C with P_4 , D with P_1 and E with P_2 . Notice, also, that we *maximised* weights in

this problem rather than minimising them; there is really no difference between being greedy for the largest or being greedy for the smallest.

C.2 Solutions for Chapter 2

Exercise 61 The summations for the negative (top-right to bottom-left) and the positive (top-left to bottom-right) diagonals in the extended matrices are:

$$\begin{array}{llllllllll} \text{(a)} & -8 & + & 0 & + & -(-9) & + & -3 & + & 0 & + & 12 & = & 10 \\ \text{(b)} & -12 & + & -(-3) & + & 0 & + & 0 & + & -9 & + & 8 & = & -10 \\ \text{(c)} & -10 & + & 0 & + & -12 & + & 4 & + & 0 & + & 18 & = & 0 \end{array}$$

Exercise 65 The six permutation matrices P together with the values of $\text{sgn}(P)\Pi(X, P)$ are as shown in Figure C.12

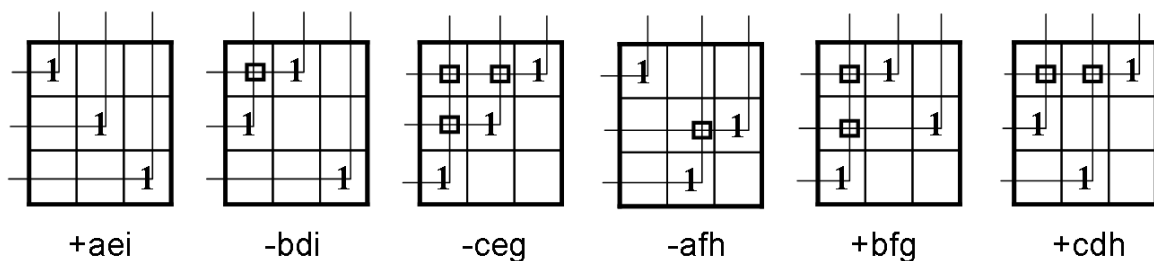


Figure C.12: the six permutation matrices of order 3 (Exercise 65).

Exercise 67 In the first determinant on the right-hand side there are two equal rows. In the second determinant, subtracting the third row from the second gives the first row. In both cases reducing to row echelon form will produce a row of zeros and therefore determinants of zero.

Exercise 70 This would be laborious to calculate by Gaussian elimination. Using Corollary 68 it is easy:

$$\begin{aligned} \det \begin{pmatrix} 3 & 0 & 0 & 0 \\ 117 & 2 & 5 & 0 \\ -192 & -69 & 813 & 7 \\ -357 & 3 & 2 & 0 \end{pmatrix} &= (-1)^{1+1} \times 3 \times \det \begin{pmatrix} 2 & 5 & 0 \\ -69 & 813 & 7 \\ 3 & 2 & 0 \end{pmatrix} \quad \text{'expanding' about } i=1, j=1, \\ &= 3 \times (-1)^{2+3} \times 7 \times \det \begin{pmatrix} 2 & 5 \\ 3 & 2 \end{pmatrix} \quad \text{this time with } i=2, j=3, \\ &= 3 \times (-7) \times (4 - 15) \quad \text{by the Rule of Sarrus,} \\ &= 231. \end{aligned}$$

Exercise 72 You should have the following:

$$\begin{array}{lll} \text{(a)} \begin{pmatrix} -\frac{8}{3} \\ \frac{26}{3} \\ -\frac{2}{3} \end{pmatrix} & \text{(b)} \begin{pmatrix} 1 & 1 & 2 \\ 3 & 1 & 0 \\ 4 & 2 & -3 \end{pmatrix} & \text{(c)} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Note that part (b) reverses the row echelon reduction in example 71(c). Indeed, the solution to part (c) is the result of multiplying this reduction matrix by its *inverse* to get the 3×3 identity matrix I_3 : ones on the main diagonal and zeros everywhere else.

Exercise 76 Firstly, the product is

$$XX^T = \begin{pmatrix} 0 & 1 & -1 & 0 \\ 2 & 0 & 0 & 2 \\ 1 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 2 & 1 \\ 1 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -1 \\ 0 & 8 & 2 \\ -1 & 2 & 2 \end{pmatrix},$$

whose determinant, using the Rule of Sarrus, is $32 - 16 = 16$.

Now extract the four 3×3 submatrices of X . The corresponding submatrices of X^T are just their transposes, so we can apply Lemma 60(b): whereby the Binet-Cauchy Theorem becomes a sum of squares:

$$\begin{aligned} & \left(\det \begin{pmatrix} 0 & 1 & -1 \\ 2 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \right)^2 + \left(\det \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & 2 \\ 1 & 0 & 0 \end{pmatrix} \right)^2 + \left(\det \begin{pmatrix} 0 & -1 & 0 \\ 2 & 0 & 2 \\ 1 & 1 & 0 \end{pmatrix} \right)^2 + \left(\det \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix} \right)^2 \\ &= (0 - 2)^2 + (2 - 0)^2 + (-2 - 0)^2 + (0 - 2)^2 = 16. \end{aligned}$$

Exercise 78 The number of bicycles produced and sold is 1000 so $P + Q + R = 1000$. If the profit target of \$9000 is met then we must have $10P + 12Q - 2R = 9000$. This gives the matrix equation:

$$\begin{pmatrix} 1 & 1 & 1 \\ 10 & 12 & -2 \end{pmatrix} \begin{pmatrix} P \\ Q \\ R \end{pmatrix} = \begin{pmatrix} 1000 \\ 9000 \end{pmatrix},$$

which we solve by reducing to row-echelon form:

$$\begin{pmatrix} 1 & 1 & 1 \\ 10 & 12 & -2 \end{pmatrix} \xrightarrow{R2-10R1} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & -12 \end{pmatrix} \quad \begin{pmatrix} 1000 \\ 9000 \end{pmatrix} \xrightarrow{R2-10R1} \begin{pmatrix} 1000 \\ -1000 \end{pmatrix}.$$

Now we find that $2Q = -1000 + 12R$ so $Q = -500 + 6R$ and, from the first equation, $P = 1500 - 7R$. All solutions have the form $(1500 - 7R, -500 + 6R, R)$ but we must presumably avoid negative production at each factory. A realistic option might be $R=150$, giving $P=450$ and $Q=400$.

Exercise 81 Experimentation will convince you that no more than four spanning trees can be obtained in a graph on four edges (either a 4-cycle or a pair of multiple edges will give this). These are the bases of the cycle matroid of the graph. Now the bases of $U_{2,4}$ are all subsets of $\{a, b, c, d\}$ of size 2. There are $\binom{4}{2}$ of them:

$$\begin{matrix} \{a, b\} \\ \{a, c\} & \{b, c\} \\ \{a, d\} & \{b, d\} & \{c, d\} \end{matrix}.$$

Now if $U_{2,4}$ were graphic then there would have to be some graph with four edges a, b, c and d for which the above sets were the spanning trees. Since there can be no more than four spanning trees this is impossible, so $U_{2,4}$ is not graphic.

Exercise 83 The submatrices corresponding to the three graceful graphs are identical:

$$\begin{matrix} 12 & 13 & 25 & 15 & 23 & 13 & 25 & 15 & 12 & 13 & 14 & 15 \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}.$$

Since these matrices are in row echelon form, their linear independence follows from Corollary 14.

Exercise 86 The partition sets of $\mathcal{A}(T)$ are $\{a, b\}$, $\{c, d\}$, $\{e, f\}$. The partition matroid $M(\mathcal{A}(T))$ is

represented by the matrix:

$$\begin{array}{c} \{a,b\} \\ \{c,d\} \\ \{e,f\} \end{array} \begin{array}{ccccc} a & b & c & d & e & f \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array},$$

and there are $3 \times 3 \times 3 = 27$ independent sets.

Exercise 88 Using the labelling provided in Figure 2.4, we get

$$B(K_5) = \begin{array}{c} \begin{matrix} & 12 & 23 & 34 & 45 & 13 & 24 & 35 & 14 & 25 & 15 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix} \end{array}$$

Exercise 92 Transposing the submatrix formed by columns 23, 13, 25 and 15 of the incidence matrix of K_5 and applying elementary row operations:

$$\begin{array}{c} 23 \\ 13 \\ 25 \\ 15 \end{array} \begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix} \xrightarrow{R4-R3+R1-R2} \begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

shows that the column rank of the matrix is 3, so this is not a linearly independent set of columns. Indeed, the reduction of R_4 to zero using R_1 , R_2 and R_3 shows that the corresponding four edges of the graph form a cycle.

Exercise 93 The transposed submatrix formed by columns 12, 13, 14 and 15 of the incidence matrix of K_5 is:

$$\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 12 \\ 13 \\ 14 \\ 15 \end{matrix} \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array}; \text{ or, after reordering vertices: } \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 & 1 \end{matrix} \\ \begin{matrix} 12 \\ 13 \\ 14 \\ 15 \end{matrix} \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \end{array}.$$

The reordered version is in row echelon form with four nonzero rows so that all four original column vectors were linearly independent. So the final subgraph in Figure 2.4 is a basis for the cycle matroid (a spanning tree).

Exercise 94 For $B'(G)$ to represent the cycle matroid we would require subsets of columns to be linearly independent if and only if they correspond to a subgraph containing no cycle. This fails when all three columns are included: we have a square matrix which, since the three columns correspond to a cycle, should have zero determinant by Lemma 60(a). But we have $\det B'(G) = -2$ (you may like to check that if the second 1 in each column of $B'(G)$ is replaced by a -1 , giving $B(G)$, then the determinant becomes zero).

Exercise 95 The first four columns give the 4×4 submatrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

which is not in row echelon form but which has a zero row which must obviously remain when elementary row operations are performed. So this submatrix will have zero determinant. If we take columns 1, 2, 3 and 5, on the other hand, we have a submatrix whose row echelon form reduction gives a nonzero diagonal

product:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix} \xrightarrow{R2+R1} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

The incidence matrix of a 5-vertex graph G must, by definition, have a 1 and a -1 in each column and have five rows. Since every column of our matrix already has a 1 and -1 except columns 2 and 4, we must add a final row having a -1 in precisely these positions, giving:

$$B(G) = \begin{matrix} & a & b & c & d & e & f \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & -1 & 0 & 0 \end{pmatrix} \end{matrix}.$$

This matrix, to which vertex and edge names have been added for convenience, may be decoded according to the definition of the incidence matrix: column a indicates that edge a joins vertex 1 to vertex 2, etc, and we get the graph shown in fig. C.13. We note that columns 1, 2, 3 and 5 correspond to edges a, b, c and e which indeed form a spanning tree.

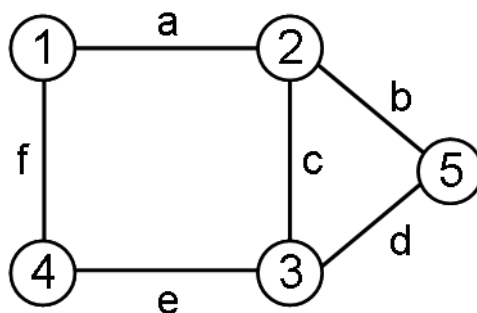


Figure C.13: graph whose incidence matrix is constructed for Exercise 95.

Exercise 98 (a) unimodular; (b) unimodular; (c) not unimodular, since $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ appears as a 2×2 submatrix, with determinant -2; (d) not unimodular since, although all 2×2 submatrices have determinant 1 or -1, the element x constitutes a 1×1 submatrix with determinant -2.

Exercise 101 The submatrix is

$$\begin{pmatrix} 4 & -2 & -1 \\ -2 & 3 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

which, applying the Rule of Sarrus, has value $20 - 15 = 5$. The five spanning trees are: $e_1e_2e_4$, $e_1e_3e_4$, $e_1e_2e_5$, $e_1e_3e_5$, and $e_1e_4e_5$.

Exercise 104 The weighted $K_{3,5}$ is shown in Figure C.14(a). The edges are weighted so that struts in row one even though they are to be avoided, have *lowest* weight because this means the greedy algorithm, operating on the bipartite graph, will choose them first for removal. Now all but one of the edges from u_1 can be removed without destroying connectedness; next, all but one edge from u_2 may be removed. This means that every weight 3 edge must be left intact. The remaining edges correspond to the strutted cells shown in Figure C.14(b). There are seven struts in all, one more than the six in Figure 2.7(a) which failed to give rigidity.

Exercise 105 The lowest weight edge that can be removed is be (edge ad , with weight zero, is the only edge connecting vertex d to the rest of the graph and so must be retained). Next, edge ge may be removed.

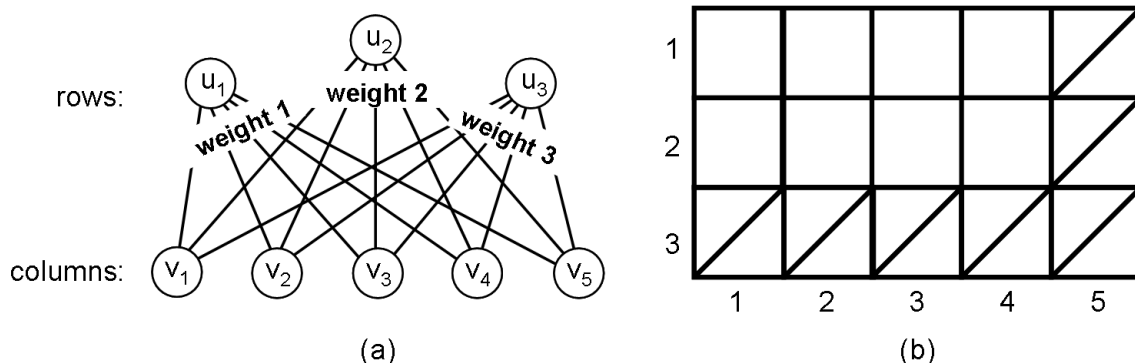


Figure C.14: (a) weighted $K_{3,5}$ and (b) rigid min-weight configuration for Exercise 104.

Now all the edges of weight 3 and 4 are required for connectivity; the edges weighted 5 are in a cycle so any one of these may be removed and the Greedy Algorithm will now terminate since all remaining edges are essential to connectivity.

Exercise 107 The independent sets of $M(\mathcal{A})$ can take one element from each of $\{a, b\}$ and $\{c, d, e\}$. So the bases are: $\{a, c\}$, $\{a, d\}$, $\{a, e\}$, $\{b, c\}$, $\{b, d\}$ and $\{b, e\}$. Now the bases of $M^*(\mathcal{A})$ are found by subtracting these sets from $\{a, b, c, d, e\}$ giving: $\{b, d, e\}$, $\{b, c, e\}$, $\{b, c, d\}$, $\{a, d, e\}$, $\{a, c, e\}$ and $\{a, c, d\}$ and these bases define the dual matroid. In the graph G_1 in Figure 2.10 the spanning trees are precisely the bases of $M(\mathcal{A})$ while in G_2 the spanning trees are the bases of $M^*(\mathcal{A})$. So both matroids are graphic. Conversely, the maximal non-cut edge sets in G_1 and G_2 are precisely the bases of $M^*(\mathcal{A})$ and $M(\mathcal{A})$, respectively. So both matroids are also cographic.

Exercise 108 By bringing columns 3 and 4 to the left we create:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 \end{pmatrix}.$$

This has the form $[I_r | X]$ where $r = 3$ and X is 3×2 .

Exercise 111 The dual representation is

$$\begin{pmatrix} 1 & 0 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Exercise 113 By reordering the columns we get a standard representation:

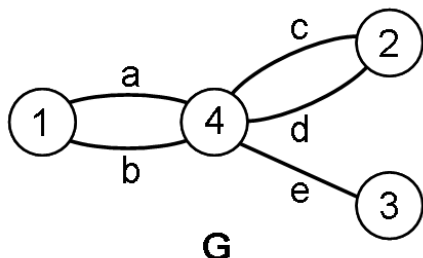
$$\begin{array}{ccccc} a & c & e & b & d \\ \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

with the dual matroid having representation:

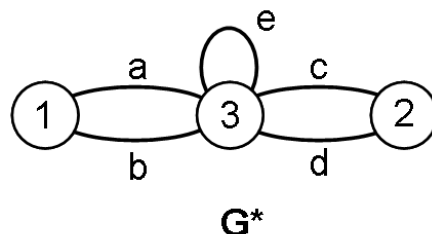
$$\begin{array}{ccccc} a & c & e & b & d \\ \begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Adding an extra row to make each column sum to zero we get the corresponding graphs:

$$\begin{array}{c} a \quad b \quad c \quad d \quad e \\ \begin{array}{l} 1 \left(\begin{array}{ccccc} 1 & -1 & 0 & 0 & 0 \end{array} \right) \\ 2 \left(\begin{array}{ccccc} 0 & 0 & 1 & -1 & 0 \end{array} \right) \\ 3 \left(\begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \end{array} \right) \\ 4 \left(\begin{array}{ccccc} -1 & 1 & -1 & 1 & -1 \end{array} \right) \end{array}$$



$$\begin{array}{c} a \quad c \quad e \quad b \quad d \\ \begin{array}{l} 1 \left(\begin{array}{ccccc} -1 & 0 & 0 & 1 & 0 \end{array} \right) \\ 2 \left(\begin{array}{ccccc} 0 & -1 & 0 & 0 & 1 \end{array} \right) \\ 3 \left(\begin{array}{ccccc} 1 & 1 & 0 & -1 & -1 \end{array} \right) \end{array}$$



Exercise 114 Elementary row operations reduce the matrix to

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

so it has rank 3 and its determinant is zero.

Exercise 117 The Mirsky-Perfect representation is

$$X_{\mathcal{A}} = \begin{array}{c} \{a,b,c\} \\ \{a,d\} \\ \{c\} \end{array} \begin{array}{c} a \quad b \quad c \quad d \\ \begin{pmatrix} t_{11} & t_{12} & t_{13} & 0 \\ t_{21} & 0 & 0 & t_{24} \\ 0 & 0 & t_{33} & 0 \end{pmatrix} \end{array}$$

There are four 3×3 submatrices which, including row and column labels, are:

$$\begin{array}{c} \{a,b,c\} \\ \{a,d\} \\ \{c\} \end{array} \begin{array}{c} a \quad b \quad c \\ \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & 0 & 0 \\ 0 & 0 & t_{33} \end{pmatrix} \end{array}, \quad \begin{array}{c} a \quad b \quad d \\ \begin{pmatrix} t_{11} & t_{12} & 0 \\ t_{21} & 0 & t_{24} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad \begin{array}{c} a \quad c \quad d \\ \begin{pmatrix} t_{11} & t_{13} & 0 \\ t_{21} & 0 & t_{24} \\ 0 & t_{33} & 0 \end{pmatrix} \end{array}, \quad \begin{array}{c} b \quad c \quad d \\ \begin{pmatrix} t_{12} & t_{13} & 0 \\ 0 & 0 & t_{24} \\ 0 & t_{33} & 0 \end{pmatrix} \end{array}.$$

The second matrix has a zero row and so has zero determinant; for each of the other three there is precisely one 3×3 permutation matrix which matches nonzero entries. The final list is:

$$-t_{12}t_{21}t_{33} \qquad 0 \qquad -t_{11}t_{24}t_{33} \qquad -t_{12}t_{24}t_{33}.$$

Exercise 118 We abbreviate the names by letters: A, B, C, D, E and F . Recall that the product experience subsets for the four products are: $P_1 = \{A, C, D, E\}$, $P_2 = \{D, E, F\}$, $P_3 = \{A, B\}$ and $P_4 = \{C, D\}$. We get the matrix

$$X_{\mathcal{A}} = \begin{array}{c} P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} A \quad B \quad C \quad D \quad E \quad F \\ \begin{pmatrix} t_{11} & 0 & t_{13} & t_{14} & t_{15} & 0 \\ 0 & 0 & 0 & t_{24} & t_{25} & t_{26} \\ t_{31} & t_{32} & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{43} & t_{44} & 0 & 0 \end{pmatrix} \end{array} \text{ and columns } A, B, C, F \text{ give submatrix: } \begin{pmatrix} t_{11} & 0 & t_{13} & 0 \\ 0 & 0 & 0 & t_{26} \\ t_{31} & t_{32} & 0 & 0 \\ 0 & 0 & t_{43} & 0 \end{pmatrix}.$$

Now t_{26} and t_{43} are the only nonzero entries for rows 2 and 4, respectively, of this submatrix and t_{32} is the only possible nonzero for column 2. This forces the selection of t_{11} for row and column 1 and therefore the

only nonzero term in the determinant of this submatrix is $t_{11}t_{26}t_{32}t_{43}$. This corresponds to the transversal: Abdul represents product P_1 on the committee, Franz represents P_2 , Baldish represents P_3 and Cheung represents P_4 . Notice that this is still not a complete solution to the Committee Selection problem since weights have not been taken into account — so diversity of product experience has not been maximised.

Exercise 120

- The determinants have values -1 and +1, respectively. In the R vector they appear multiplied by $(-1)^1$ because, in the corresponding rows, there is an X symbol between the L and R columns.
- No X symbol appears in column t_{43} because this column is identical to t_{33} so these columns can never combine to give a non-singular matrix: we require columns that give non-singular matrices when combined with L and with R columns.
- The indeterminate currently being assigned a value is t_{33} (the highlighted R column).
- If we make the assignment $t_{33} = -1$ then $L + (-1) \times R$ has no zero entries and this is therefore a valid assignment.
- The next tableau will now look like:

a	b	c	d	t_{42}	t_{43}
1	1	1	1	0	0
1	-1	2	0	1	0
1	2	-1	0	0	1
×	×		3	-1	
×		×	-3	2	
×			-1	1	×
	×	×	-3	3	
	×		1	1	×
		×	-2	1	×
			L	R	

- You should confirm that each of the values 1, -1 or 2 to the indeterminate t_{42} will give zeros in the vector $L + t_{42} \times R$. The value $t_{42} = -2$, however, gives no zeros and is a valid assignment.

Exercise 123 The first missing tableau from example 122 is:

Iteration 3: indeterminate t_{31} :

A	B	C	D	E	F	t_{31}	t_{43}	t_{44}
1	0	1	1	1	0	0	0	0
0	0	0	1	-1	1	0	0	0
0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	1
						L	R	

Any permutation matrix not containing the R column must contain column B which will then make X_R singular. So no non-singular matrix pairs are possible and the assignment is $t_{31} = 1$. The next tableau is:

Iteration 4: indeterminate t_{43} :

A	B	C	D	E	F	t_{43}	t_{44}
1	0	1	1	1	0	0	0
0	0	0	1	-1	1	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1
						L	R

This time, any permutation matrix not containing the R column must contain column t_{44} which will then make X_R singular. Again, no non-singular matrix pairs are possible and the assignment is $t_{43} = 1$.

The two 1's found here are to be seen in the tableau of iteration 5 in example 122.

Exercise 124 The completed tableau is

a	b	c	d	t_{44}
1	1	0	0	0
1	0	0	1	0
0	1	1	0	0
0	0	1	0	1
\times	\times	\times	1	-1
			L	R

with the assignment $t_{44} = -1$ being valid.

Exercise 126 The first tableau offers no pairs of non-singular matrices so we set $t_{21} = 1$. We present the continuation without comment:

Iteration 2: indeterminate t_{22} :

a	b	c	d	e	t_{22}	t_{23}	t_{24}	t_{25}
1	1	1	1	1	0	0	0	0
1	0	0	0	0	1	1	1	1
\times	-1				1			
			L		R			

$t_{22} = -1$

Iteration 3: indeterminate t_{23} :

a	b	c	d	e	t_{23}	t_{24}	t_{25}
1	1	1	1	1	0	0	0
1	-1	0	0	0	1	1	1
\times		-1			1		
	\times	1			1		
			L		R		

$t_{23} = 2$

Iteration 4: indeterminate t_{24} :

a	b	c	d	e	t_{24}	t_{25}
1	1	1	1	1	0	0
1	-1	2	0	0	1	1
\times			-1		1	
	\times		1		1	
		\times	-2		1	
			L		R	

$t_{24} = -2$

Iteration 5: indeterminate t_{25} :

a	b	c	d	e	t_{25}
1	1	1	1	1	0
1	-1	2	-2	0	1
\times				-1	1
	\times			1	1
		\times		-2	1
			\times	2	1
			L	R	$t_{25} = 3$

We will apply elementary row operations to the final matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 2 & -2 & 3 \end{pmatrix} \xrightarrow{R_2 - R_1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & -2 & 1 & -3 & 2 \end{pmatrix} \xrightarrow{R_1 - R_2} \begin{pmatrix} 1 & 3 & 0 & 4 & -1 \\ 0 & -2 & 1 & -3 & 2 \end{pmatrix}.$$

This does not affect linear dependence between the columns so the validity of the representation is preserved. We can also rearrange columns (this is just reordering the ground set) to give a standard representation of $U_{2,5}$:

$$\begin{pmatrix} 1 & 0 & 3 & 4 & -1 \\ 0 & 1 & -2 & -3 & 2 \end{pmatrix} = [I_2 | X].$$

Now we can derive the dual of $U_{2,5}$ by Theorem 109:

$$[X^T | I_{5-2}] = \begin{pmatrix} 3 & -2 & 1 & 0 & 0 \\ 4 & -3 & 0 & 1 & 0 \\ -1 & 2 & 0 & 0 & 1 \end{pmatrix}.$$

If you look back to the representation of $U_{3,5}$ presented at the end of section 2.3.1 you will see that it was precisely this matrix (although here our column headings would be different since we reordered the columns).

Exercise 127 Inserting the value $t_{24} = -2$ into the incidence matrix we have the final tableau:

a	b	c	d	t_{43}
1	1	1	1	0
1	-1	2	-2	0
1	2	-1	0	1
\times	\times		5	-2
\times		\times	-7	1
	\times	\times	-9	3
			L	R

You might have thought the assignment $t_{43} = 3$ was likely, given that 2 and -2 have already been used. In fact, we see that this is one of the few assignments which is *not* valid, because $L + 3 \times R$ will have zero in the last entry. An assignment of $t_{43} = 1$ is valid here and gives the final representation of $U_{3,4}$:

$$\begin{pmatrix} a & b & c & d \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 2 & -2 \\ 1 & 2 & -1 & 1 \end{pmatrix}.$$

Exercise 130 A subset of $X = \{a, b, c, d, e, f\}$ is independent in $M_X(G)$ if it is covered by some matching. The edge dC may be included in any matching, so d is in any basis of $M_X(G)$. The remaining edges contribute partial matchings as separated into three columns below:

A	a	A	b	A	b
B	c	B	a	B	c
D	e	D	c	D	e
	or f		or e		or f
			or f		

The result is seven subsets of X covered by maximal matchings and thus bases of $M_X(G)$:

1	2	3	4	5	6	7
d	d	d	d	d	d	d
a	a	b	b	b	b	b
c	c	a	a	a	c	c
e	f	c	e	f	e	f

Referring back to example 129 we notice that the vertices A, B, C and D in G are adjacent to exactly the subsets of X contained in the family $\mathcal{A} = \{\{a, b\}, \{a, c\}, \{d\}, \{c, e, f\}\}$. So our bases are exactly the same as the bases of the transversal matroid $M(\mathcal{A})$. We deduce that any transversal matroid is a matching matroid.

Exercise 131 In the Mirsky-Perfect representation:

$$\begin{pmatrix} a & b & c & d & e & f \\ \{a, b\} & t_{11} & t_{12} & 0 & 0 & 0 & 0 \\ \{a, c\} & t_{21} & 0 & t_{23} & 0 & 0 & 0 \\ \{d\} & 0 & 0 & 0 & t_{34} & 0 & 0 \\ \{c, e, f\} & 0 & 0 & t_{43} & 0 & t_{45} & t_{46} \end{pmatrix}$$

any 4×4 submatrix which does not contain column d will have zero third row and therefore zero determinant.

C.3 Solutions for Chapter 3

Exercise 134 The intersection of these two matroids is as shown as the shaded region in Figure C.15. This collection of subsets is not a matroid since it contains $\{c\}$ and $\{a, b\}$ but neither of the subsets $\{a, c\}$ nor $\{b, c\}$, contradicting the exchange axiom for matroids (definition 21(3)).

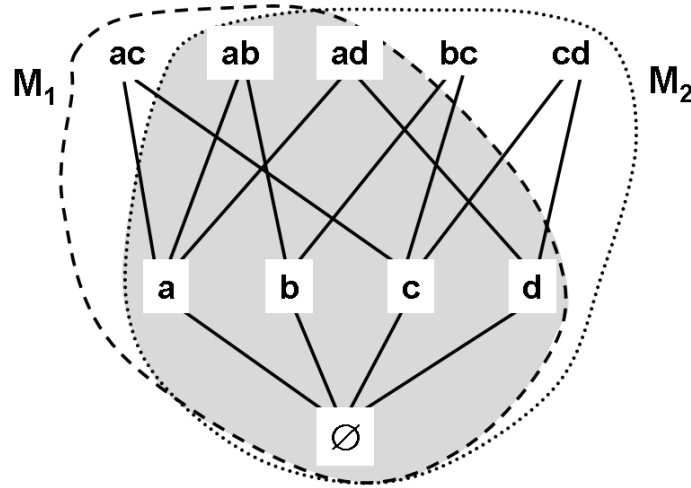


Figure C.15: intersection of the two matroids in Exercise 134.

Exercise 136 For this choice of representations, the Binet-Cauchy calculation asserts that:

$$0 = \det \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = \det \begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix} \times \det_b^a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \det \begin{pmatrix} a & d \\ 1 & 0 \end{pmatrix} \times \det_d^a \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = 1 - 1 = 0,$$

which is correct but tells us nothing about independent sets in $M_1 \cap M_2$.

Exercise 138 We may use the representations

$$R_1 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad R_2 = \begin{pmatrix} a & b & c & d \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The Binet-Cauchy product is:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} a & 0 & c & 0 \\ 0 & b & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & c \\ b & 0 \end{pmatrix}.$$

Now $\det \begin{pmatrix} 0 & c \\ b & 0 \end{pmatrix} = -bc$ so $\{b, c\}$ is identified as the only independent set of size 2 in $M_1 \cap M_2$.

Exercise 140 Because of subset closure, the independent sets of M_1 and M_2 must be all subsets of $\{a, b, c\}$ and $\{a, b, d\}$, respectively. The initial steps of Algorithm 139 are as follows:

$$1. D := \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix};$$

$$2. \text{ Representations of the matroids are } R_1 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, R_2 = \begin{pmatrix} a & b & c & d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$3. \Phi := R_1 \times D \times R_2^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot D \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

4. $K := 3$ since both matroids have rank 3;

5. $\text{optpoly} := 0$;

The first pass of the loop (steps 5–11) will try to construct a nonzero polynomial using all three rows and columns of Φ . But $\det \Phi$ is clearly zero. The loop will then examine all 2×2 submatrices and will find

$U = V = \{1, 2\}$ with $\det \Phi[U|V] = \det \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} = ab$. Since this is the only square submatrix of order 2 with nonzero determinant, the algorithm will return the value $\text{optpoly} := ab$. Notice that if we continued the algorithm down to 1×1 matrices then we would find precisely the elements in the ground set which are common to both matroids, namely $\{a\}$ and $\{b\}$.

Exercise 142 Let $S = \{a, b, c\}$ and $T = \{w, x, y, z\}$. Define $D := \text{diag}(aw, ax, bw, by, cx, cz)$. We have the partition matroids $M(S)$ and $M(T)$ represented by:

$$R_S = \begin{matrix} & aw & ax & bw & by & cx & cz \\ \begin{matrix} a \\ b \\ c \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}, \text{ and } R_T = \begin{matrix} & aw & ax & bw & by & cx & cz \\ \begin{matrix} w \\ x \\ y \\ z \end{matrix} & \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}.$$

The Binet-Cauchy intersection is

$$\Phi = R_S \cdot D \cdot R_T^T = \begin{pmatrix} aw & ax & 0 & 0 \\ bw & 0 & by & 0 \\ 0 & cx & 0 & cz \end{pmatrix}.$$

Since $K := \min(\text{no. of rows of } R_S, \text{no. of rows of } R_T) = 3$ (step 4 in the algorithm) we must take determinants of 3×3 submatrices, deleting one column of Φ at a time:

$$\det \begin{pmatrix} ax & 0 & 0 \\ 0 & by & 0 \\ cx & 0 & cz \end{pmatrix} = ax \cdot by \cdot cz, \quad \det \begin{pmatrix} aw & 0 & 0 \\ bw & by & 0 \\ 0 & 0 & cz \end{pmatrix} = aw \cdot by \cdot cz$$

$$\det \begin{pmatrix} aw & ax & 0 \\ bw & 0 & 0 \\ 0 & cx & cz \end{pmatrix} = -bw \cdot ax \cdot cz, \quad \det \begin{pmatrix} aw & ax & 0 \\ bw & 0 & by \\ 0 & cx & 0 \end{pmatrix} = -aw \cdot by \cdot cx.$$

So Algorithm 139 will return a value for optpoly (step 12) of $ax \cdot by \cdot cz + aw \cdot by \cdot cz - bw \cdot ax \cdot cz - aw \cdot by \cdot cx$ whose terms give the maximal matchings: $\{ax, by, cz\}$, $\{aw, by, cz\}$, $\{bw, ax, cz\}$ and $\{aw, by, cx\}$.

Exercise 143 Let $S = \{a, b, c\}$ and $T = \{w, x, y\}$. Now edge aw has weight 2, so we will attach a product of indeterminates to this edge: $aw \times s^2$. Using powers of s similarly for the other edges, we define

$D := \text{diag}(aws^2, axs^3, ays, bxs^2, bys, cws)$. We have the partition matroids $M(S)$ and $M(T)$ represented by:

$$R_S = \begin{matrix} & \begin{matrix} aws^2 & axs^3 & ays & bxs^2 & bys & cws \end{matrix} \\ \begin{matrix} a \\ b \\ c \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}, \text{ and } R_T = \begin{matrix} & \begin{matrix} aws^2 & axs^3 & ays & bxs^2 & bys & cws \end{matrix} \\ \begin{matrix} w \\ x \\ y \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{matrix}.$$

The Binet-Cauchy intersection is

$$\Phi = R_S \cdot D \cdot R_T^T = \begin{pmatrix} aws^2 & axs^3 & ayt \\ 0 & bxs^2 & by \\ cwt & 0 & 0 \end{pmatrix}.$$

This is a square matrix, so we can evaluate the determinant directly:

$$\text{optpoly} = cw.ax.by.s^5 - cw.ay.bx.s^4$$

The maximal matchings are $\{ax, by, cw\}$ and $\{ay, bx, cw\}$ and we observe that the powers of t attached to these are precisely the sums of their edge weights. So a min-weight maximal matching is $\{ay, bx, cw\}$ of weight 4.

Exercise 145 One of several possible labellings is shown in Figure C.16.

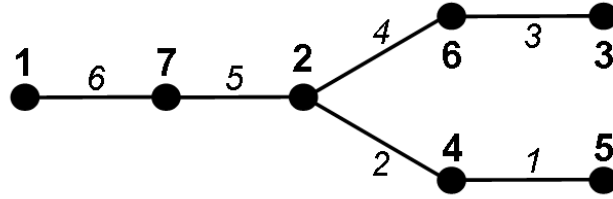


Figure C.16: labelling of the 7-vertex tree for Exercise 145.

Exercise 146 We intersect the cycle matroid of K_3 with the partition matroid which assigns edge ij of K_3 to partition set $|i - j|$. The ground set is the set of edges of K_3 and we use indeterminate t_{ij} to represent edge ij . Multiply the incidence matrix R_1 of the partition matroid by $D_{K_3} = \text{diag}(t_{12}, t_{23}, t_{13})$ to get

$$R_1 \times D_{K_3} = \begin{pmatrix} t_{12} & t_{23} & 0 \\ 0 & 0 & t_{13} \end{pmatrix}.$$

The Binet-Cauchy intersection Φ is now formed by multiplying by the transpose of the cycle matroid representation:

$$R_2 = B(K_3)^T = \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & -1 \end{pmatrix}^T.$$

We get:

$$\Phi = \begin{pmatrix} t_{12} & t_{23} - t_{12} & -t_{23} \\ t_{13} & 0 & -t_{13} \end{pmatrix}.$$

Since $K := \min(\text{no. of rows of } R_1, \text{no. of rows of } R_2) = 2$ we will take the determinant of Φ with one column deleted — a 2×2 matrix. We get two graceful trees:

$$\text{optpoly} = -t_{23}t_{13} + t_{12}t_{13}.$$

There is a pattern in these Binet-Cauchy intersections which may be continued:

$$\begin{pmatrix} t_{12} & t_{23} - t_{12} & -t_{23} \\ t_{13} & 0 & -t_{13} \end{pmatrix}, \begin{pmatrix} t_{12} & t_{23} - t_{12} & t_{34} - t_{23} & -t_{34} \\ t_{13} & t_{24} & -t_{13} & -t_{24} \\ t_{14} & 0 & 0 & -t_{24} \end{pmatrix}, \begin{pmatrix} t_{12} & t_{23} - t_{12} & t_{34} - t_{23} & t_{45} - t_{34} & -t_{45} \\ t_{13} & t_{24} & t_{35} - t_{13} & -t_{24} & -t_{35} \\ t_{14} & t_{25} & 0 & -t_{14} & -t_{25} \\ t_{15} & 0 & 0 & 0 & -t_{15} \end{pmatrix}, \dots$$

Exercise 147 If we delete the first column of the final matrix in solution 146 then we can evaluate the determinant of the 4×4 matrix remaining by summing over permutation matrices. The permutation matrix which is just the identity matrix I_4 will match the terms in the matrix identified below:

$$\begin{pmatrix} \boxed{t_{23} - t_{12}} & t_{34} - t_{23} & t_{45} - t_{34} & -t_{45} \\ t_{24} & \boxed{t_{35} - t_{13}} & -t_{24} & -t_{35} \\ t_{25} & 0 & \boxed{-t_{14}} & -t_{25} \\ 0 & 0 & 0 & \boxed{-t_{15}} \end{pmatrix},$$

which will contribute the product $(t_{23} - t_{12})(t_{35} - t_{13})(-t_{14})(-t_{15})$ to the determinant. This includes the term $+t_{12}t_{13}t_{14}t_{15}$ which specifies our tree. You may like to check that this term will appear regardless of what column of Φ is deleted, in accordance with the remarks following Exercise 145. If you have the energy to continue the sequence of Φ matrices from Exercise 146 up to 7 vertices you can repeat this exercise for your solution to Exercise 145.

Exercise 148 The other two maximum branchings are: $\{b, c, e, f\}$ and $\{a, b, e, f\}$.

Exercise 149 The greedy approach will construct B by adding arcs in the following order: ab, bd, de and ac . Each arc has the lowest weight of those which leave B at each stage. The total weight is $3 + 4 + 4 + 5 = 16$. But the branching consisting of arcs ac, ce, ed and db has only half this weight: $5 + 1 + 1 + 1 = 8$.

Exercise 152 The determinants are:

Row 2 deleted: $bcd f - cbfg + cefg$;

Row 3 deleted: 0 (since there is no directed walk from vertex 3 to vertices 1, 2 and 3);

Row 4 deleted: 0 (ditto);

Row 5 deleted: $abcd - abcg + aceg$

Exercise 153 The head partition matroid and cycle matroids have representations

$$R_1 = \begin{matrix} & a & b & c & d & e & f \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}, \quad \text{and } R_2 = \begin{matrix} & a & b & c & d & e & f \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{matrix}.$$

We take the diagonal matrix of indeterminates to be $D = \text{diag}(a, \dots, f)$ and calculate $\Phi = R_1 \cdot D \cdot R_2^T$, giving the Binet-Cauchy intersection:

$$\Phi = R_1 \cdot D \cdot R_2^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ a & -a + c & 0 & -c & 0 & 0 \\ 0 & b & -b & 0 & 0 & 0 \\ 0 & 0 & 0 & e & -e & 0 \\ 0 & 0 & d & 0 & -d + f & -f \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

We know that we can delete any column to reflect the rank deficiency of R_2 , so assume we delete column 6. Clearly we must delete rows 1 and 6 since they are zero. Since deleting row i corresponds to setting indegree of vertex i to zero this means our maximum branchings will be rooted at both vertices 1 and 6 (as we saw was the case in Exercise 148). Now the value of optpoly in Algorithm 139 is given by summing

over the five possible addition column deletions which give us a 4×4 submatrix:

Column 1 deleted: $-bcef$;
 Column 2 deleted: $bcef$;
 Column 3 deleted: $abef - bcef$;
 Column 4 deleted: $-abef + bcef$;
 Column 5 deleted: $-abde + abef - bcef$;

with a grand total of $-abde + abef - bcef$.

Exercise 154 We need to show that there is a maximum branching rooted at vertex 3 and that this branching has the minimum weight among maximum branchings in the graph. We represent the appropriate head partition and cycle matroids thus:

$$R_1 = \begin{matrix} & t_{12} & t_{21} & t_{23} & t'_{23} & t_{32} & t_{34} & t_{43} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}, \quad \text{and } R_2 = \begin{matrix} & t_{12} & t_{21} & t_{23} & t'_{23} & t_{32} & t_{34} & t_{43} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix} \end{matrix},$$

with t_{23} and t'_{23} being the indeterminates which represent the two arcs from vertex 2 to vertex 3. Let $D = \text{diag}(t_{12}s^2, t_{21}s^2, t_{23}s^2, t'_{23}s^3, t_{32}s, t_{34}s^2, t_{43}s^3)$ with the powers of t representing the traffic loads on the individual streets. Now the Binet-Cauchy intersection is

$$R_1 \cdot D \cdot R_2^T = \begin{pmatrix} t_{21}s^2 & -t_{21}s^2 & 0 & 0 \\ t_{12}s^2 & -t_{12}s^2 + t_{32}s & -t_{32}s & 0 \\ 0 & t_{23}s^2 + t'_{23}s^3 & -t_{23}s^2 - t'_{23}s^3 + t_{43}s^3 & -t_{43}s^3 \\ 0 & 0 & t_{34}s^2 & -t_{34}s^2 \end{pmatrix}.$$

We know from previous examples that any column can and must be deleted in order to account for the rank deficiency of R_2 . We also know that deleting row i will give maximum branchings rooted at vertex i . When we delete column 1 and row 3 we get a 3×3 matrix with determinant $-t_{21}t_{32}t_{34}s^5$ which identifies a branching of weight 5 using arcs 21, 32 and 34. Deleting any other row is found to give branchings of greater weight, for example, deleting column 1 and row 1 gives a 3×3 matrix of determinant $-t_{12}t_{23}t_{34}s^6 - t_{12}t'_{23}t_{34}s^7$.

Exercise 155 The term is $t_{1,2}t_{1,3}t_{3,4}t_{4,5}$ (the second term on the second line of `optpoly` as it is printed). It does not solve EDGE-DISJOINT SPANNING TREES since edges 13, 34 and 45 are a cut set giving two connected components: on vertices $\{1, 2, 4\}$ and $\{3, 5\}$.

Exercise 156 The matrix is

$$\begin{pmatrix} 3 & 0 & -1 & 0 \\ 0 & 4 & -1 & -2 \\ -1 & -1 & 3 & -1 \\ 0 & -2 & -1 & 3 \end{pmatrix}.$$

Let us simplify the calculation of its determinant by using elementary row operations (this is legitimate by Lemma 60 in Chapter 2, section 2.2.2):

$$\begin{pmatrix} 3 & 0 & -1 & 0 \\ 0 & 4 & -1 & -2 \\ -1 & -1 & 3 & -1 \\ 0 & -2 & -1 & 3 \end{pmatrix} \xrightarrow[R2+2R4]{R1+3R3} \begin{pmatrix} 0 & -3 & 8 & -3 \\ 0 & 0 & -3 & 4 \\ -1 & -1 & 3 & -1 \\ 0 & -2 & -1 & 3 \end{pmatrix} \xrightarrow{R4-\frac{2}{3}R1} \begin{pmatrix} 0 & -3 & 8 & -3 \\ 0 & 0 & -3 & 4 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -\frac{19}{3} & 5 \end{pmatrix}.$$

This has not produced row-echelon form but we can apply Theorem 63 from section 2.2.2: there are now only two permutation matrices whose nonzero entries match with nonzero entries in our simplified matrix, giving two products: $(-1)(-3)(-3)(5)$ and $(-1)(-3)(4)(-19/3)$, the permutations having signs $+1$ and -1 , respectively. So the determinant has value $(+1) \times 45 + (-1) \times -76 = 31$. We can infer that the 31 terms in the polynomial `optpoly` are precisely the 31 spanning trees in our graph, as counted by the Matrix Tree

Theorem.

Exercise 158 There are two minimum-cost spanning trees in the graph of Figure 3.10, one with edge set $\{12, 13, 34, 35\}$ and the other with edge set $\{12, 24, 34, 35\}$. The corresponding terms in the determinant are $t_{12}t_{13}t_{34}t_{35}$ and $t_{12}t_{24}t_{34}t_{35}$. Both have weight 6. Other choices from the determinant give spanning trees of higher weight, e.g. $t_{12}t_{13}t_{35}t_{45}$ and $t_{12}t_{24}t_{35}t_{45}$ both have weight 7.

Exercise 159 The diagonal matrix will become

$$\text{diag}(t_{12}s, t_{13}s, t_{34}s, t_{45}s^2, t'_{12}s^2, t_{24}s, t_{35}s^3, t'_{35}s^6).$$

The Binet-Cauchy intersection becomes

$$\begin{pmatrix} t_{1,2}s + t'_{1,2}s^2 & -t_{1,2}s - t_{2,4}s & 0 & 0 \\ 0 & t_{1,3}s + t_{2,4}s & 0 & 0 \\ 0 & t_{3,4}s + t_{2,4}s & t_{3,4}s + t_{3,5}s^3 & t_{3,4}s + t'_{3,5}s^6 \\ 0 & 0 & t_{4,5}s^2 + t_{3,5}s^3 & t_{4,5}s^2 + t'_{3,5}s^6 \end{pmatrix}$$

with determinant

$$(t_{1,2}s + t'_{1,2}s^2)(t_{1,3}s + t_{2,4}s)(t_{3,4}t'_{3,5}s^7 + t_{3,5}t_{4,5}s^5 - t_{3,4}t_{3,5}s^4 - t'_{3,5}t_{4,5}s^8)$$

and the terms with minimum weight are those in the lowest power of s . These must take $t_{12}s$ from the first bracket, either term from the second bracket, and $-t_{3,4}t_{3,5}s^4$ from the third bracket, giving:

$$-t_{1,2}t_{1,3}t_{3,4}t_{3,5}s^6 - t_{1,2}t_{2,4}t_{3,4}t_{3,5}s^6.$$

Thus we find the same two spanning trees as we did in Exercise 158.

Exercise 161 We form the Binet-Cauchy intersection:

$$\Phi = R_{M(P)} \times \text{diag}(T_1, \dots, T_8) \times R_{M(R)}^T = \begin{pmatrix} T_1 & T_3 & 0 & T_2 + T_3 & 0 \\ 0 & T_4 & 0 & 0 & T_5 + T_6 \\ 0 & 0 & T_7 + T_8 & 0 & T_7 - T_8 \\ 0 & -T_3 & 0 & T_2 - T_3 & 0 \\ 0 & -T_3 + T_4 & 0 & T_2 - T_3 & -T_5 \end{pmatrix}.$$

We can begin to evaluate $\det \Phi$ by observing that any permutation matrix can match nonzero entries in the first and third columns only in the first and third rows, respectively. These entries will contribute $T_1 \times (T_7 + T_8)$ to the determinant. The remaining rows and columns then form a 3×3 submatrix to which we can apply elementary row operations:

$$\begin{pmatrix} T_4 & 0 & T_5 + T_6 \\ -T_3 & T_2 - T_3 & 0 \\ -T_3 + T_4 & T_2 - T_3 & -T_5 \end{pmatrix} \xrightarrow{R3-R1-R2} \begin{pmatrix} T_4 & 0 & T_5 + T_6 \\ -T_3 & T_2 - T_3 & 0 \\ 0 & 0 & -2T_5 - T_6 \end{pmatrix}.$$

Finally we revert to the permutation matrix viewpoint: the only permutation matrix whose nonzero entries match nonzero entries in the right-hand matrix above is the identity matrix I_3 . So we have:

$$\det \Phi = T_1(T_7 + T_8)T_4(T_2 - T_3)(-2T_5 - T_6).$$

This gives us 16 simultaneous transversals: the minus signs we can ignore; the factor of 2 means there are two ways to simultaneously allocate rooms to professors during the indicated timetable slots: Choosing the first term in each bracket we have $-2T_1T_2T_4T_5T_7$, the two allocations being

$$\begin{array}{ccccc} P_1 & P_4 & P_2 \text{ or } P_5 & P_2 \text{ or } P_5 & P_3 \\ & & \text{in} & & \\ R_1 & R_4 & R_2 & R_5 & R_3 \\ & & \text{during timetable period} & & \\ T_1 & T_2 & T_4 & T_5 & T_7 \end{array}$$

There is an assumption that no room should be used twice (a little unrealistic) and (eminently reasonable!) that no professor should teach twice.

Exercise 162

1. An Euler trail which is not closed must begin and end at a vertex of odd degree (see solution to Exercise 38). Suppose every vertex of G has even degree and delete an edge uv . This creates exactly two vertices of odd degree, so by Euler's Theorem there is an Euler trail beginning at u and ending at v ; replacing the edge uv will turn this trail into a closed walk, which is an Euler tour. Thus having all degrees even is a *sufficient* condition for being Eulerian. It is also *necessary* since if there is a vertex of odd degree then there must be at least two, by the Hand-shaking Lemma (Exercise 37). Now by Euler's Theorem we can at most have an Euler trail beginning and ending at vertices of odd degree. Such a trail is not closed and is therefore not an Euler tour.
2. A witness for a graph to be Eulerian would be any Euler tour. A witness for the graph to be non-Eulerian would be any vertex of odd degree.

Exercise 163 We definitely cannot colour the vertices of K_4 with three colours without some edge joining two vertices of the same colour. So the presence of K_4 as a subgraph certainly prevents a three-colouring. However, K_4 is not a subgraph of the Grötzsch graph (which contains no 3-cycles, for instance). So given that the latter is not three-colourable, we cannot rely on K_4 subgraphs as a witness for non-three-colourability: it can lead to false negatives.

Exercise 164 Replace every undirected edge by two arcs, one in each direction to get a new graph G' . Now search for a directed Hamilton path from a to b . If this fails, try a to g and so on for each vertex adjacent to a . For a and g , for example, we will find $a k i b c d e j l h f g$. Now since ag is an edge this will complete a directed Hamilton cycle in G' which will correspond to an undirected cycle in G .

Exercise 166 The other term in the evaluation of $\det \Phi[\{1, 3, 4, 5, 6, 7\} | \{1, 2, 3, 4, 6, 7\}]$ is $t_{12}t_{37}t_{54}t_{76}t_{41}t_{63}$ and this corresponds to a directed path of length 3: edges 54, 41 and 12; and a cycle of length 3: 37, 76 and 63. The path certainly begins at $s = 5$ and ends at $t = 2$ as required, but it is not a Hamilton path.

C.4 Solutions for Chapter 4

Exercise 169 Your approach may have been to sort the list into ascending order:

$$-3, -2, 1, 1, 4, 4, 6, 7, 8, 11$$

and then add the last five numbers to get 36. The quickest sorting algorithms take $O(n \log n)$ time but for a small list like this you would probably just search the list n times to identify the successively larger entries. This takes time

$$n + (n - 1) + \dots + 1 = \frac{1}{2}n(n + 1) = O(n^2).$$

A much quicker option is to search the list five times to identify the largest entry, the second largest, and so on. If we remove each largest entry that is remaining we will need to inspect

$$n + (n - 1) + (n - 2) + (n - 3) + (n - 4) = 5n - 10 = O(n)$$

entries in all.

Even quicker would be to keep a working array of length 5, sorted into ascending order which initially contains the first five list entries. Now we scan the remaining $n - 5$ entries, discarding the smallest array entry and inserting a new list entry as appropriate. The insertion would be done by insertion sort which you may recall meeting on the unit *Software Engineering, Algorithm Design and Analysis*. In the worst case we may still have to scan all five entries in our working array for each list entry, but on average we can expect to scan only $5/2$ entries. So we would say that this method has $5n$ worst-case complexity but $5n/2$ average complexity. Of course, 'big Oh' does not distinguish between these: both are just $O(n)$.

Exercise 170 You should have graphs which look like Figure C.17. There is never any doubt that n^5

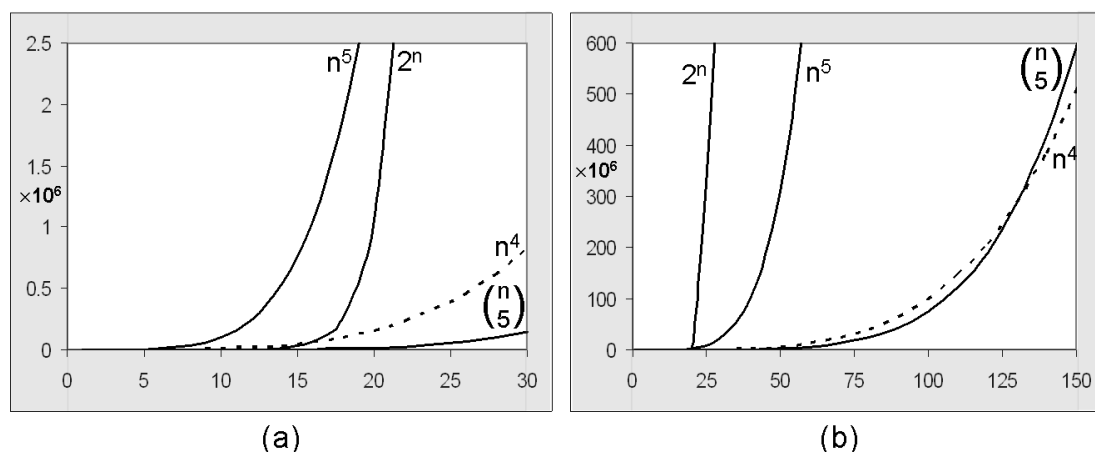


Figure C.17: graphs of n^4 , n^5 , 2^n and $\binom{n}{5}$ for (a) $n \leq 30$ and (b) $n \leq 150$.

climbs more steeply than n^4 . In Figure C.17(a) there still appears to be a doubt about whether 2^n will ever catch up with n^5 , and you would confidently predict, on the basis of Figure C.17(a), that $\binom{n}{5}$ will never exceed n^4 . In Figure C.17(b) things have reversed somewhat. As we saw in Figure 4.1 of Chapter 4, 2^n suddenly shoots past n^5 around $n = 23$. The value of $\binom{n}{5}$, meanwhile, has passed n^4 by $n = 132$.

Exercise 172 If each of the three matroids has a polynomial-time algorithm for checking independence then deciding if they have a common independent set of size K is in **NP**. In particular this decision problem is in **NP** for representable matroids. However, if we do not know such polynomial-time algorithms exist then we cannot assert that the problem is in **NP** since this depends on checking certificates in polynomial time. We may perhaps be relying on an oracle to determine independence (see the beginning of Chapter 2) in which case we would talk about having an *oracle-polynomial-time algorithm* for checking Yes-certificates.

Exercise 174 You will find that changing the value of any one of the x_i still gives an (F, F, F) triple. However, on changing both x_1 and x_2 to F the triples evaluate as

$$\begin{array}{cccc} (x'_1, x_2, x_3) & (x'_2, x_3, x'_4) & (x'_1, x'_2, x'_3) & (x'_1, x_3, x_4) \\ (T, F, F) & (T, F, F) & (T, T, T) & (T, F, T) \end{array}$$

with every triple containing at least one T .

Exercise 175 The vertex cover graph for the 3-SAT instance $(x_1, x'_2, x_3), (x_2, x'_3, x_4), (x'_1, x'_2, x'_3), (x'_2, x_3, x_4)$ is given in Figure C.18.

Exercise 176 A copy of the vertex cover graph is shown in Figure C.19 with a vertex cover of size 12 highlighted, corresponding to the satisfying truth assignment given in the solution to Exercise 174.

Exercise 178 One choice of No-certificate is the vector $y = (4, 3)^T$ since $A^T \times (4, 3)^T = (1, 0, 1)^T$ and $b^T y = (-7, 9) \times (4, 3)^T = -1 < 0$.

Exercise 180 The constraint matrix for programme (1) is

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 5 & 9 & 0 \end{pmatrix} \quad \text{and we have} \quad b = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

Now we can find a column vector y such that $A^T y \geq 0$ and $b^T y < 0$. An example is $y = (-4, 1)^T$, for which $A^T y = (1, 1, 4)^T$ and $b^T y = -1$. So by the Standard Form of Farkas' Lemma, the equations $Ax = b$ have no non-negative solutions and therefore programme (1) has no feasible vectors.

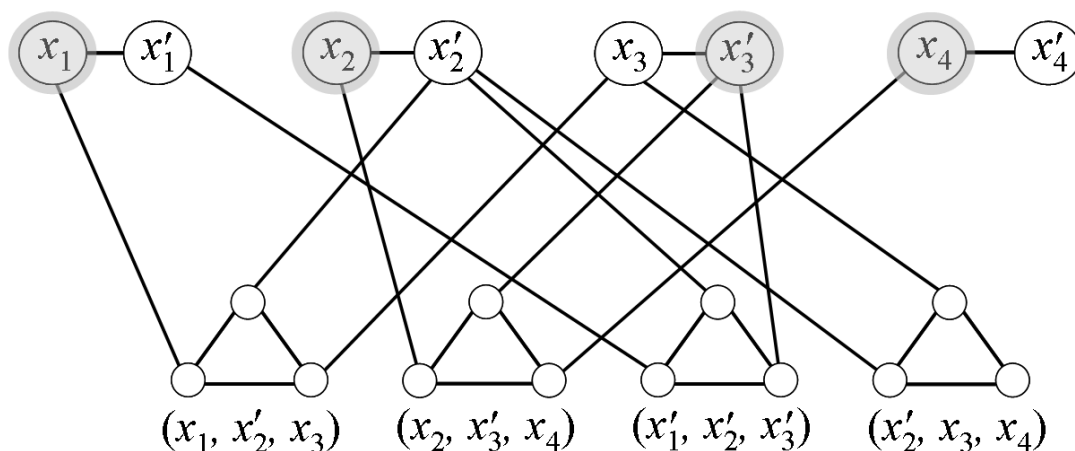


Figure C.18: vertex cover graph for 3-SAT instance (x_1, x'_2, x_3) , (x_2, x'_3, x_4) , (x'_1, x'_2, x'_3) , (x'_2, x_3, x_4) .

The constraint matrix for programme (2) is

$$A = \begin{pmatrix} 1 & 1 & -1 \\ -2 & 1 & 0 \end{pmatrix} \quad \text{and we have} \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Applying elementary row operations to A and b :

$$\begin{pmatrix} 1 & 1 & -1 \\ -2 & 1 & 0 \end{pmatrix} \xrightarrow{R2+2R1} \begin{pmatrix} 1 & 1 & -1 \\ 0 & 3 & -2 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \xrightarrow{R2+2R1} \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

So $3Q - 2R = 3$ and $Q = 2R/3 + 1$. Back substitution into the first equation gives $P + 2R/3 + 1 - R = 1$ so $P = R/3$. We have discovered that all vectors of the form $(R/3, 2R/3 + 1, R)$ solve the constraints. This makes the objective function $R/3 + 2R/3 + 1 + R = 2R + 1$ which can take arbitrarily large values, so programme (2) is unbounded.

Exercise 181 The inequality $x + y + z \leq 2$ gives us the half-space below the sloping plane and therefore includes everything in the unit cube below this plane. This is therefore a bounded region, so it is a polytope. It is a box with one sloping face, as shown in Figure 4.8(c). With $x + y + z \geq 2$ we will instead include everything *above* the sloping plane—this will again be a polytope, this time a tetrahedron, whose base is the sloping plane and whose apex is the point $(1, 1, 1)$.

Exercise 182 The polytope shown has seven vertices: $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(1, 1, 0)$, $(1, 0, 1)$ and $(0, 1, 1)$. We still get a polytope if ' \geq ' is used instead of ' $=$ ' since the resulting halfspace must still intersect with the unit cube. Now there are only four vertices: $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$ and $(1, 1, 1)$.

Exercise 183 An attempt at a convex alphabet is shown in Figure C.20. Requiring convexity places great restrictions on the amount of variety you can create in symbols. This increases the danger of confusion when symbols are hand-written. If humans were blessed with perfect optical symbol recognition and perfect calligraphic skills then an arbitrarily large convex alphabet could be created by using regular polygons.

Exercise 186 Your sketch should identify something like the 'wedge' shape in Figure C.21(b), perhaps by way of first building the box defined by the individual inequalities that bound x , y and z , and then adding the hyperplanes that are defined by the first two inequalities: Figure C.21(a). Now by Theorem 185 the value of $c(x, y, z)$ is maximised, for any choice of c , at one of the eight vertices of the polytope. We can find by trial and error which vertex gives the highest value:

(a) when $c = (1, 1, 1)$ the value of $x + y + z$ in the polytope is maximised at $x = 2, y = 1, z = 0$;

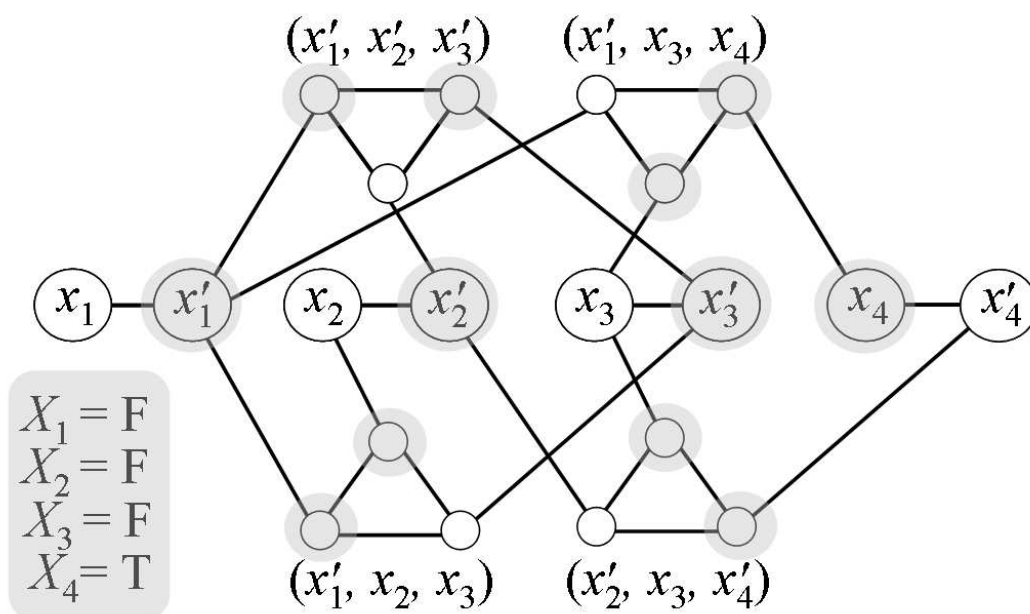


Figure C.19: vertex cover graph for 3-SAT instance (x'_1, x_2, x_3) , (x'_2, x_3, x'_4) , (x'_1, x'_2, x'_3) , (x'_1, x_3, x_4) and truth assignment $x_1 = F$, $x_2 = F$, $x_3 = F$ and $x_4 = T$.

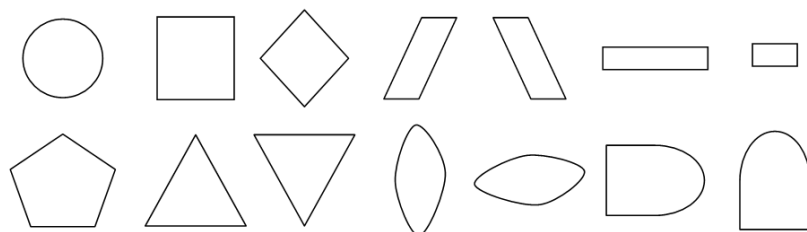


Figure C.20: a convex alphabet.

- (b) when $c = (1, -1, 1)$ the value of $x - y + z$ is maximised at $(2, 0, 0)$ and also at $(3/2, 0, 1/2)$;
- (c) when $c = (-1, 1, 1)$ the value of $-x + y + z$ is maximised at $(0, 1, 0)$ and at $(0, 1/2, 1/2)$;
- (d) when $c = (-1, 1, 2)$ the value of $-x + y + 2z$ is maximised at $(0, 1/2, 1/2)$.

There are three more vertices of the polytope at which $c(x, y, z)$ has yet to be maximised: $(0, 0, 0)$, $(0, 0, 1/2)$ and $(3/2, 1/2, 1/2)$. We can choose c to be $(-1, -1, -1)$, maximised at $(0, 0, 0)$; $(-1, -1, 1)$, maximised at $(0, 0, 1/2)$ and $(1, 2, 4)$ maximised at $(3/2, 1/2, 1/2)$.

Exercise 188 We have that P is feasible and bounded. Now D cannot be unbounded otherwise, by Theorem 187(1), its dual, P , would be infeasible which is false. Suppose D was infeasible. P , the dual of D , is feasible, so by Theorem 187(2), this would mean P must be unbounded but again this is false, so D is feasible.

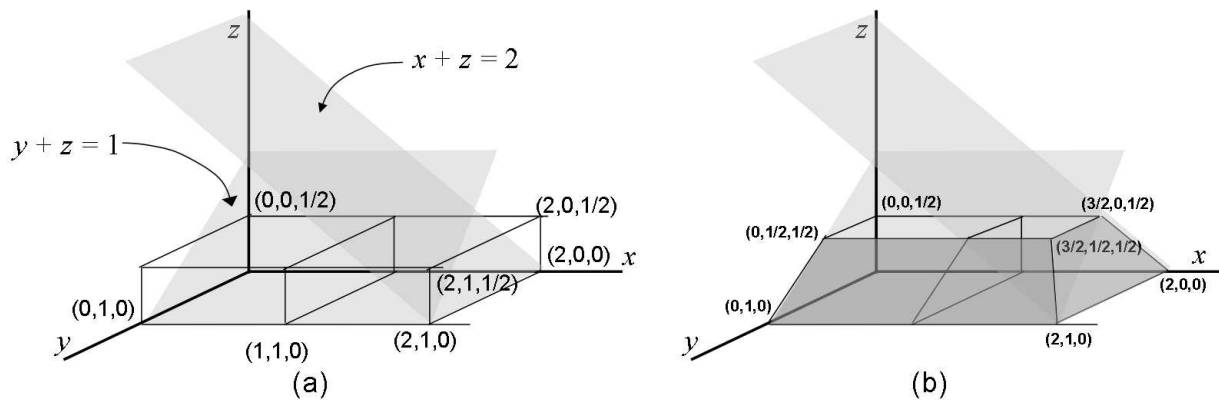


Figure C.21: sketch of convex polytope for Exercise 186.

Exercise 191 The dual programme is:

$$\begin{array}{ll}
 \text{maximise} & -4x + 2y \\
 \text{subject to} & -2x + y \leq -1 \\
 & x - 6y \leq -3 \\
 & -3x - y \leq -6 \\
 & x + y \leq 5 \\
 & x, y \geq 0
 \end{array}$$

Exercise 192 There are four straight lines to draw. Probably it is easiest to sketch, for each inequality, a straight line of the form $y = mx + c$, with the inequality telling you whether the polyhedron lies above or below this line. Thus from $x - 6y \leq -3$ we derive the straight line equation $y = x/6 + 1/2$ which bounds the polyhedron from *below*. The inequality $x + y \leq 5$ gives the equation $y = -x + 5$ bounding the polyhedron from above. The complete sketch is given in Figure C.22. The polyhedron in this case is a polytope with four vertices. The point of intersection of each pair of straight lines is a vertex and its coordinates must be substituted into the objective function $-4x + 2y$ to find the value of the linear programme at that point. Once the maximum value is found then Theorem 185 guarantees that this is a point in polytope at which the objective function is maximised; Theorem 190 tells us that this minimises the objective function of the primal programme. Here are the possibilities:

equations	solution	value of $-4x + 2y$
$x - 6y = -3$ $x + y = 5$	$(27/7, 8/7)$	$-92/7$
$x + y = 5$ $-2x + y = -1$	$(2, 3)$	-2
$-2x + y = -1$ $-3x - y = -6$	$(7/5, 9/5)$	-2
$-3x - y = -6$ $x - 6y = -3$	$(33/19, 15/19)$	$-102/19$

We see that there are actually two vertices of the polytope which optimise the objective function for the dual programme, one at $v^* = (2, 3)^T$ and one at $v^* = (7/5, 9/5)^T$. The optimum value of the primal programme is also -2 by Theorem 190.

Exercise 195 The 0-1 value for x gives $cx = 12$; the version with fractional values gives 12.5.

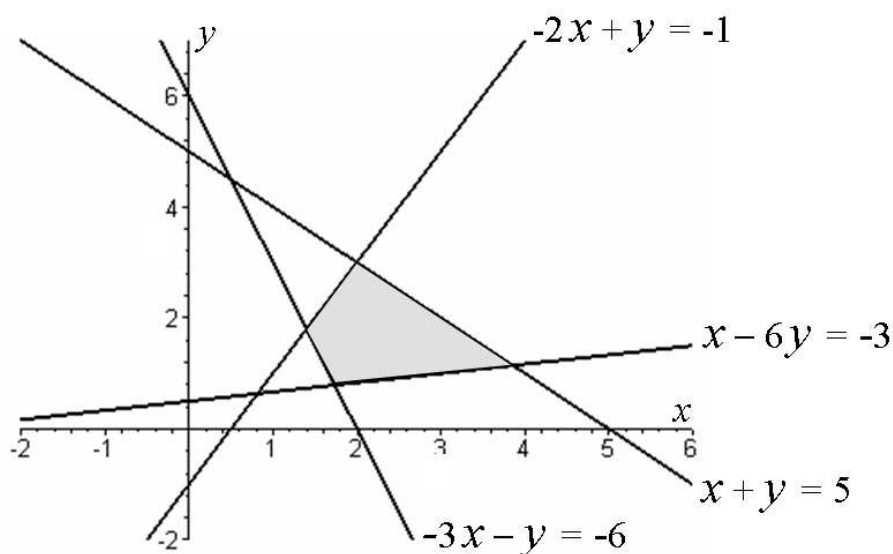


Figure C.22: sketch of convex polytope for Exercise 192.

Exercise 196 The constraints are as given in the following linear programme:

$$\begin{array}{ll}
 \text{minimise} & cx \\
 \text{subject to} & x_2 + x_3 + (1 - x_1) \geq 1 \\
 & x_3 + (1 - x_2) + (1 - x_4) \geq 1 \\
 & (1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1 \\
 & x_3 + x_4 + (1 - x_1) \geq 1 \\
 & x_1, x_2, x_3, x_4 \geq 0
 \end{array}$$

A vector satisfying these constraints is $x = (0, 1, 1, 0)$ and the values $x_1 = x_4 = F$ and $x_2 = x_3 = T$ do indeed satisfy the original triples:

$$\begin{array}{cccc}
 (x'_1, x_2, x_3) & (x'_2, x_3, x'_4) & (x'_1, x'_2, x'_3) & (x'_1, x_3, x_4) \\
 \text{evaluates to} & (T, T, T) & (F, T, T) & (T, F, F) & (T, T, F)
 \end{array}$$

Exercise 193 The vector $y = (0, 1, 0)$ represents the subset $\{b\}$ of the vertices while $y = (1, 1, 0)$ represents the subset $\{a, b\}$. Now the matrix products $A^T y$ for these cases are:

$$\text{(a) } \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad \text{(b) } \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix},$$

respectively. The first product has zero third entry which means that the third edge, ac , is not incident with any vertex in the subset $\{b\}$. The second product has no zeros and therefore every edge of the graph is incident with some vertex of $\{a, b\}$. The entry in the second product which has value 2 corresponds to an edge both of whose ends are located in the vertex subset. Thus the first edge, ab , has both end vertices in the set $\{a, b\}$.

Exercise 194 The whole programme, written out explicitly, is:

$$\begin{array}{ll}
 \text{minimise} & y_1 + y_2 + y_3 \\
 \text{subject to} & y_1 + y_2 \geq 1 \\
 & y_2 + y_3 \geq 1 \\
 & y_1 + y_3 \geq 1 \\
 & y_i \geq 0
 \end{array}$$

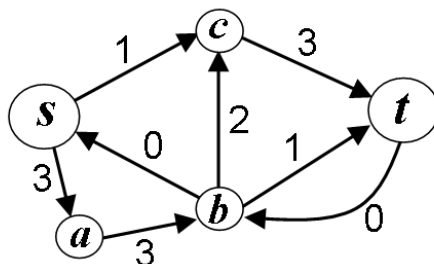


Figure C.23: The 3-flow in Figure 4.18 increased to a 4-flow.

Exercise 201 We must increase the flow in arc ab to 2 in order to satisfy rule 3 of the definition of a flow (definition 200). This then means that arc sa increases to 3. Finally, increasing the flow in arc ct to 3 will balance the flow at c and also balance the overall flow from s to t at a new value of 4. The result is shown in Figure C.23.

Exercise 202 The cost of the flow on arcs sb , sc , bc and ct is 3.0, 1.0, 2.5 and 2.0, respectively, so the total cost for this flow is 8.0. A cheaper flow uses arcs sa , ad , dt , sc , ct with costs 1.5, 1.5, 1.0, 1.0, 2.0 and total cost 7.0. This is in fact the minimum cost 2-flow.

Exercise 203 Using the cost and capacity values on the arcs in Figure 4.17 we have

$$\begin{array}{ll}
 \text{minimise} & (1.5, 3.0, 1.0, 1.5, 2.5, 1.5, 1.0, 2.0, 1.0) \times (x_{sa}, x_{sb}, x_{sc}, x_{ba}, x_{bc}, x_{ad}, x_{bd}, x_{ct}, x_{dt})^T \\
 \text{subject to} & -x_{sa} - x_{sb} - x_{sc} = -5 \\
 & x_{sa} + x_{ba} - x_{ad} = 0 \\
 & x_{sb} - x_{ba} - x_{bc} - x_{bd} = 0 \\
 & x_{sc} + x_{bc} - x_{ct} = 0 \\
 & x_{ad} + x_{bd} - x_{dt} = 0 \\
 & x_{ct} + x_{dt} = 5 \\
 & x_{sa} \leq 2, x_{sb} \leq 2, x_{sc} \leq 1, x_{ba} \leq 1, x_{bc} \leq 3, x_{ad} \leq 3, x_{bd} \leq 2, x_{ct} \leq 2, x_{dt} \leq 4 \\
 & x \geq 0
 \end{array}$$

Exercise 208 We shall borrow the triple notation from Figure 4.8 but label the axes with the ground set $\{a, b, c\}$. We have a somewhat simpler polytope since we are excluding one of the triples with two non-zero entries. The result is shown in Figure C.24.

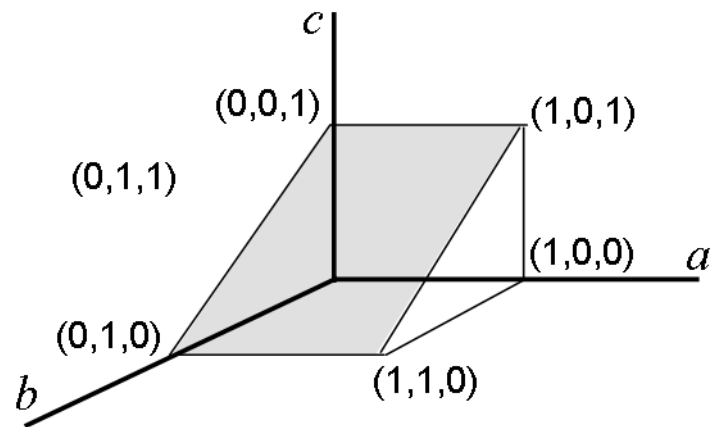


Figure C.24: sketch of convex polytope for Exercise 208.

Index

- 3-SAT, 79
- 3-colouring, 71
- 3-regular graph, 72
- algebraic independence, 42
- algorithm
 - Binet-Cauchy Intersection, 56
 - cubic time, 76
 - Ellipsoid, 87
 - exponential-time, 78
 - Greedy, 15
 - Karmarkar's, 88 footnote
 - linear time, 76
 - oracle-polynomial-time, 138
 - Piff-Welsh merge, 46
 - polynomial-time, 78
 - quadratic time, 76
 - Simplex, 87
 - Take Five (I), 76
 - Take Five (II), 76
- arc, 61
- asymptotic, 75
- augmentation axiom, 6
- axiom, 6
- back substitution, 28
- basis, 5
- binary matroid, 29
- Binet-Cauchy
 - intersection, 55
 - Theorem, 25
- bipartite graph, 31
- bipartite matching, 57
- Birkhoff polytope, 99
- Birkhoff-von Neumann Theorem, 99
- branching, 61
- certificate, 70
- closed walk, 8
- co-NP**, 82
- cocycle matroid, 35, 37, 66
- cographic matroid, 37
- collinear, 3
- complete graph, 30
- complete transversal, 17
- concurrency graph, 35
- connected, 8
- connected component, 66
- constraint matrix, 83
- constraints, 84
- convex
 - combination, 86
 - hull, 86
- convex polygon, 85 footnote
- convex polyhedron, 85
 - bounded, 85
 - intersection, 85
- convex polytope, 85
- cubic
 - time, 76
- cutset, 37, 66
- cycle, 8
- cycle matroid, 15
- decision problem, 78
- deficiency, 63
- determinant, 20
- directed graph, 61
- doubly stochastic matrix, 99
- dual
 - linear programme, 88
 - matroid, 38
- edge
 - of convex polytope, 85
 - of graph, 8
- edge covering, 49
- elementary row operation, 3
- Ellipsoid algorithm, 87
- entry, 20
- error-correcting code, 29
- Euler tour, 71
- Euler trail, 14
- Euler's Theorem, 14
- Eulerian graph, 71
- exchange axiom, 6
- exponential-time algorithm, 78
- Extreme Point Theorem, 86 footnote
- feasible
 - vector, 84
- feasible linear programme, 84
- flow, 96
- flow value, 96
- forest, 9
- framework rigidity, 36
- good characterisation, 70
- graceful
 - graph, 30, 59
 - labelling, 60
 - tree, 59
- graph, 8
 - 3-regular, 72
 - bipartite, 12, 18, 31
 - complete, 30

- concurrency, 35
 - connected, 8
 - directed, 61
 - Eulerian, 71
 - graceful, 30
 - Hamiltonian, 70
 - perfect, 99
 - three-colourable, 71
 - tripartite, 71
 - underlying, 61
- graphic matroid, 16
- Greedy Algorithm, 15
- half-space, 85
- Hamilton cycle, 14, 69
- Hamiltonian graph, 70
- Hand-shaking Lemma, 13
- head, 61
- head partition matroid, 62
- Huffman coding, 7
- hyperplane, 85
- hyperspace, 85
- identity matrix, 22
- ILP, *see* integer linear programming
- incidence matrix, 31, 32
- indegree, 61
- independent edge, 13
- independent set, 5
- indeterminate, 41
- infeasible linear programme, 84
- injection, 29
- injective, 29
- integer linear programming, 94
- Karmarkar's Algorithm, 88 footnote
- Laplace expansion, 21
- Laplacian, 35
- linear
 - combination, 3
 - constraint, 83
 - dependence, 3
 - function, 83
 - independence, 3
 - time, 76
- linear programme, 83
 - canonical form, 84
 - dual, 88
 - feasible, 84
 - infeasible, 84
 - integer, 94
 - primal, 88
 - relaxation, 94
 - standard form, 84
 - unbounded, 84
- linearity, 6
- log scale, 77
- matching, 12
- bipartite, 57
- matroid, 49
- maximal, 12
- perfect, 12
- spanning, 13
- matrix
 - constraint, 83
 - determinant, 20
 - doubly stochastic, 99
 - entry, 20
 - identity, 22
 - incidence, 31, 32
 - Laplacian, 35
 - multiplication, 24
 - non-singular, 21
 - permutation, 22
 - scalar product, 21
 - singular, 21
 - size, 20
 - square, 20
 - submatrix, 20
 - transpose, 4
 - Tree Theorem, 34
 - unimodular, 34
- Matrix Tree Theorem, 34
- matroid, 5
 - axioms for independent sets, 6
 - binary, 29
 - cocycle, 35, 37, 66
 - cographic, 37
 - cycle, 15
 - definition of, 6
 - dual, 38
 - graphic, 16
 - head partition, 62
 - intersection, 53
 - isomorphism, 7, 48
 - matching, 49
 - partition, 31
 - polytope, 102
 - representable, 29
 - tail partition, 72
 - transversal, 18
 - uniform, 29
- maximal vs. maximum, 5, 15
- maximise, *see* optimisation problem
- minimise, *see* optimisation problem
- Mirsky-Perfect Theorem, 41
- mixed constraints, 84
- multiple edge, 8
- necessary and sufficient condition, 14, 70
- neighbourhood set, 13
- network flow, *see* flow
- non-cutset, 37
- non-singular matrix, 21
- NP, 78
- NP, 78
- objective function, 84

- optimisation problem
 - DIRECTED $s - t$ HAMILTON PATH, 72
 - EDGE-DISJOINT SPANNING TREES, 64
 - MAX COMMON INDEPENDENT SET, 54
 - MAXIMAL SIMULTANEOUS TRANSVERSAL, 67
 - MAXIMUM-WEIGHT SUBSET, 15
 - MIN VERTEX COVER, 79
 - MIN-WEIGHT MAXIMAL BRANCHING, 61
 - MIN-WEIGHT MAXIMAL MATCHING, 12, 57
 - MIN-WEIGHT MAXIMAL NON-CUT, 36
 - MIN-WEIGHT MAXIMAL SUBSET, 15
 - MIN-WEIGHT MAXIMAL TRANSVERSAL, 18
 - MIN-WEIGHT SPANNING TREE, 10
- OR problem
 - Capacitated Transportation, 96
 - Committee Selection, 17
 - Congestion Charging, 91
 - Depot Location, 61
 - Fault-tolerant Networking, 64
 - Framework Rigidity, 35
 - Job Assignment, 17, 57
 - Justified Product Elimination, 10
 - Network Pruning, 35
 - Peer-to-peer Updating, 9
 - Radio Frequency Allocation, 67
 - Room Booking, 67
 - Two-party Room Booking, 72
 - Web Searching, 61
- oracle-polynomial-time algorithm, 138
- order
 - of growth, 76
 - of magnitude, 77
- origin, 2
- outdegree, 61
- P**, 78
- partial transversal, 17
- partition
 - of sets, 31
- partition matroid, 31
- path, 8
- perfect graph, 99
- perfect matching, 12
- permutation matrix, 22
- Petersen graph, 70
- Piff-Welsh merge, 46
- Piff-Welsh Theorem, 42
- pivot element, 4
- Plank time, 78 footnote
- polyhedron, *see* convex polyhedron
- polynomial-time algorithm, 78
- polynomial-time reduction, 79
- polytope, *see* convex polytope
 - Birkhoff, 99
 - of a matroid, 102
- precisely (math. idiom), 7 footnote
- proper (math. idiom), 70
- pseudocode, 15
- quadratic time, 76
- rank
 - column rank of matrix, 4
 - deficiency, 63
 - deficient, 21
 - of a matroid, 7
 - of matrix, 4
 - row rank of matrix, 4
- relaxation, 94
- representable matroid, 29
- representation
 - of cocycle matroid, 38
 - of cycle matroid, 32
 - of dual matroid, 38
 - of partition matroid, 30
 - of transversal matroid, 41
 - standard, 38
- rhombus, 36
- root vertex, 61
- row-echelon form, 4
- Rule of Sarrus, 21
- Satisfiability, 78
- satisfying truth assignment, 79
- scalar, 2
- scalar product, 21
- self-loop, 8
- set partition, 31
- sign of a permutation matrix, 22
- Simplex algorithm, 87
- simultaneous
 - equations, 27
 - transversal, 67
- singular matrix, 21
- size, 20
- spanning
 - matching, 13
 - subgraph, 9
 - tree, 9
- square matrix, 20
- standard basis, 30, 117
- standard representation, 38
- Steinitz Exchange Lemma, 6
- Strong Perfect Graph Theorem, 99 footnote
- subgraph, 9
- submatrix, 20
- subset-closed, 6
- tail, 61
- tail partition matroid, 72
- Take Five problem, 75
- three satisfiability, 79
- three-colouring, 71
- totally unimodular, 83

- transpose, 4
- transversal, 17
- transversal matroid, 18
- Travelling Salesman Problem, 69
- tree, 9
- trivial
 - cycle, 9 footnote
- trivial (math. idiom), 8 footnote
- trivial walk, 8
- truth assignment, 79
- TSP, 69

- unbounded linear programme, 84
- underlying graph, 61
- uniform matroid, 29
- unimodular, 34
- unit vector, 2

- vector
 - addition, 2
 - collinearity, 3
 - feasible, 84
 - linear combination, 3
 - linear dependence, 3
 - linear independence, 3
 - space, 2
 - unit, 2
 - zero, 2

- vertex
 - of convex polytope, 85
 - of graph, 8

- walk, 8
- weight, 8

- Yes-certificate, 78
- yoctosecond, 78 footnote

- zero vector, 2

Notes

Notes

Comment form

We welcome any comments you may have on the materials which are sent to you as part of your study pack. Such feedback from students helps us in our effort to improve the materials produced for the University of London.

If you have any comments about this guide, either general or specific (including corrections, non-availability of Essential readings, etc.), please take the time to complete and return this form.

Title of this subject guide:

Name

Address

Email

Student number

For which qualification are you studying?

Comments

This image shows a full page of a document template designed for handwritten notes or essays. It features approximately 28 evenly spaced, thin grey horizontal lines across the entire width of the page. The margins are consistent on all sides, providing ample space for writing. There are no pre-printed questions, headings, or other markings on the page.

Please continue on additional sheets if necessary.

Date:

Please send your completed form (or a photocopy of it) to:

Publishing Manager, Publications Office, University of London, Stewart House, 32 Russell Square, London WC1B 5DN, UK.