Preface

This book has grown out of an attempt to understand the role that the topology of an electrical network plays in its **efficient** analysis. The approach taken is to transform the problem of solving a network with a given topology, to that of solving another with a different topology (and same devices), but with additional inputs and constraints. An instance of this approach is network analysis by multiport decomposition - breaking up a network into multiports, solving these in terms of port variables and finally imposing the port connection conditions and getting the complete solution. The motivation for our approach is that of building more efficient circuit simulators, whether they are to run singly or in parallel. Some of the ideas contained in the book have already been implemented - BITSIM, the general purpose circuit simulator built at the VLSI Design Centre, I.I.T. Bombay, is based on the 'topological hybrid analysis' contained in this book and can further be adapted to use topological decomposition ideas.

Many combinatorial optimization problems arise naturally when one adopts the above approach, particularly the hybrid rank problem and its generalizations. The theory required for the solution of these problems was developed by electrical engineers parallel to, and independent of, developments taking place in the theory of matroids and submodular functions. Consider, for instance, the work of Kishi and Kajitani, Iri, Ohtsuki et al in the late 60's on principal partition and its applications, independent of Edmonds' work on matroid partitions (1965). There is a strong case for electrical network topologists and submodular function theorists being aware of each others' fields. It is hoped that the present book would fill this need.

The topological network analysis that we have considered is to be

distinguished from the kind of work exemplified by 'Kirchhoff's Third Law' which has been discussed in many books published in the 60's (eg. the book by Seshu and Reed [Seshu+Reed61]). In the 70's much interesting work in this area was done by Iri, Tomizawa, Recski and others using the 'generality assumption' for linear devices. Details may be found, for instance, in Recski's book [Recski89]. In the present book devices play a very secondary role. Mostly we manipulate only Kirchhoff's Laws.

Submodular functions are presented in this book adopting the 'elementary combinatorial' as opposed to the 'polyhedral' approach. Three things made us decide in favour of the former approach.

- It is hoped that the book would be read by designers of VLSI algorithms. In order to be convincing, the algorithms presented would have to be fast. So very general algorithms based on the polyhedral approach are ruled out.
- The polyhedral approach is not very natural to the material on Dilworth truncation.
- There is an excellent and comprehensive monograph, due to S.Fujishige, on the polyhedral approach to submodular functions; a book on polyhedral combinatorics including submodular functions from A.Schrijver is long awaited.

In order to make the book useful to a wider audience, the material on electrical networks and that on submodular functions are presented independently of each other. A final chapter on the hybrid rank problem displays the link. An area which can benefit by algorithms based on submodular functions is that of CAD for VLSI - particularly for building partitioners. Some space has therefore been devoted to partitioning in the chapter on Dilworth truncation.

The book is intended primarily for self study - hence the large number of problems with solutions. However, most of the material has been tested in the class room. The network theory part has been used for many years for an elective course on 'Advanced Network Analysis' - a third course on networks taken by senior undergraduates at the EE Dept, I.I.T. Bombay. The submodular function part has been used

for special topics courses on combinatorics taken by doctoral students in Maths and Computer Science. This material can be covered in a semester if the students have a prior background in elementary graphs and matroids, leaving all the starred sections and relegating details and problems to self study.

It is a pleasure to acknowledge the author's indebtedness to his many colleagues, teachers and friends and to express his heartfelt gratitude.

He was introduced to electrical network theory by Professors R.E.Bedford and K.Shankar of the EE Dept., I.I.T. Bombay, and to graph theory by Professor M.N.Vartak of the Dept. of Maths, I.I.T. Bombay. Professor Masao Iri, formerly of the University of Tokyo, now of the University of Chuo, has kept him abreast of the developments in applied matroid theory during the last two decades and has also generously spared time to comment on the viability of lines of research.

He has benefited through interaction with the following: Professors S.D.Agashe,
P.R.Bryant,A.N.Chandorkar,M.Chandramouli,C.A.Desoer,A.Diwan,S.Fujishige, P.L.Hammer,M.V.Hariharan,Y.Kajitani,M.V. Kamath,M.S.Kamath,E.L.Lawler, K.V.V. Murthy,T.Ozawa,S.Patkar,S.K.Pillai,P.G.Poonacha,G.N.Revankar,S.Roy, S.C.Sahasrabudhe,P.C.Sharma,M.Sohoni,V.Subbarao,N.J.Sudarshan,V.K.Tandon, N.Tomizawa, P.P.Varaiya, J.M.Vasi.

The friends mentioned below have critically read parts of the manuscript: S.Batterywala, A.Diwan, N.Jayanthi, S.Patkar, P.G.Poonacha and the '96 batch students of the course 'Advanced Network Analysis'. But for Shabbir Batterywala's assistance (technical, editorial, software consultancy), publication of this book would have been delayed by many months.

Mr Z.A.Shirgaonkar has done the typing in Latex and Mr R.S.Patwardhan has drawn the figures.

The writing of this book was supported by a grant (HN/EE/TXT/95) from the C.D.P., I.I.T. Bombay.

The author is grateful to his mother Lalitha Iyer, wife Jayanthi and son Hari for their continued encouragement and support.

Note to the Reader

This book appears too long because of two reasons:

- it is meant for self study - so contains a large number of exercises and problems with solutions.
- it is aimed at **three** different types of readers:
  - Electrical engineers interested in topological methods of network analysis.
  - Engineers interested in submodular function theory
  - Researchers interested in the link between electrical networks and submodular functions.

To shorten the book for oneself it is not necessary to take recourse to drastic physical measures. During first reading all starred sections, starred exercises and problems may be omitted. If the reader belongs to the first two categories mentioned above, she would already find that only about two hundred pages have to be read.

Sections, exercises and problems have been starred to indicate that they are not necessary for a first reading. Length of the solution is a fair indicator of the level of difficulty of a problem - star does not indicate level of difficulty. There are only a handful of routine (drill type) exercises. Most of the others require some effort. Usually the problems are harder than the exercises.

Many of the results, exercises, problems etc. in this book are well known but cannot easily be credited to any one author. Such results are marked with a '(k)'.

Electrical Engineers interested in topological methods

Such readers should first brush up on linear algebra (say first two chapters of the book by Hoffman and Kunze [Hoffman+Kunze72]), read a bit of graph theory (say the chapter on Kirchhoff's laws in the book by Chua et al [Chua+Desoer+Kuh87] and the first four chapters of the book by Narsingh Deo [Narsingh Deo74]) and then read chapters 2 to 8. The chapter on graphs contains material on contraction and restriction which is not easily available in textbooks on circuit theory, but which is essential for an understanding of subsequent chapters. So this chapter should be read carefully, particularly since it is written tersely. The chapter on matroids is optional. The chapter on electrical networks should be easy reading but scanning it is essential since it fixes some notation used subsequently and also because it contains material motivating subsequent chapters, e.g. multiport decomposition. The next three chapters contain whatever the book has to say on topological network analysis.

Engineers interested in submodular functions

Such readers should read Chapters 2 to 4 and Chapters 9 to 13 and the first four sections of Chapter 14. If the reader is not interested in matroids he may skip material (chapters, sections, exercises, examples) dealing with them without serious loss of continuity. This would mean he would have to be satisfied with bipartite graph based instances of the general theory. The key chapter for such a reader is Chapter 9. This is tersely written-so should be gone through carefully.

Researchers interested in the link between submodular functions and electrical networks

The key chapter for such a reader is Chapter 14. To read the first four sections of this chapter the reader has to be familiar with Chapters 5, 6, 7 from the electrical networks part and the unstarred sections of the chapters on submodular functions. If he has some prior familiarity with submodular functions and electrical networks it is possible to directly begin reading the chapter picking up the required results on

submodular functions as and when they are referred to in the text. To read the last section of the chapter, familiarity with Chapter 8 is required.

Comments on Notation

Sometimes, instead of numbering equations, key statements etc., we have marked them with symbols such as $(*), (**), (\surd)$. These marks are used over and over again and have validity only within a local area such as a paragraph, a proof or the solution to a problem.

In some cases, where there is no room for confusion, the same symbol denotes different objects. For instance, usually B denotes a bipartite graph. But in Chapter 4, B denotes a base of a matroid- elsewhere a base is always denoted by b. The symbol E is used for the edge set of a graph, in particular a bipartite graph. But $E(X), X \subseteq V(\mathcal{G})$ denotes the set of edges with both endpoints within X, while $E_L(X), X \subseteq V_L$, in the case of a bipartite graph, denotes the set of all vertices adjacent only to vertices in X.

We have often used brackets to write two statements in one.
Example: We say that set X is **contained** in Y (**properly contained in** Y), if every element of X is also a member of Y (every element of X is a member of Y and $X \neq Y$) and denote it by $X \subseteq Y (X \subset Y)$.
This is to be read as the following two statements.

i. We say that set X is **contained** in Y, if every element of X is also a member of Y and denote it by $X \subseteq Y$.

ii. We say that set X is **properly contained in** Y, if every element of X is a member of Y and $X \neq Y$ and denote it by $X \subset Y$.

List of Commonly Used Symbols

Sets, Partitions, Partial Orders

$\{e_1, e_2, \ldots, e_n\}$	*set whose elements are* $e_1, e_2, \ldots, e_n$
$\{x_i : i \in I\}$	*set whose members are* $x_i,\ i \in I$
$(x_i : i \in I)$	*a family (used only in Chapters* 2 *and* 11)
$x \in X$	*element* x *belongs to set* X
$x \notin X$	*element* x *does not belong to set* X
$\forall x$ *or* $\forall\ x$	*for all elements* x
$\exists x$	*there exists an element* x
$X \subseteq Y$	*set* X *is contained in set* Y
$X \subset Y$	*set* X *is properly contained in set* Y
$X \cup Y$	*union of sets* X *and* Y
$X \cap Y$	*intersection of sets* X *and* Y
$X \uplus Y$	*disjoint union of sets* X *and* Y
$\bigcup_{i=1}^{n} X_i$	*union of the sets* X_i
$\biguplus_{i=1}^{n} X_i$	*disjoint union of the sets* X_i
$X - Y$	*set of elements in* X *but not in* Y
$\bar{X}$	*complement of* X
$X \times Y$	*cartesian product of sets* X *and* Y
$X \oplus Y$	*direct sum of sets* X *and* Y
2^S	*collection of subsets of* S
$\mid X \mid$	*size of the subset* X
$(P, \preceq)$	*preorder on* P
$(P, \leq)$	*partition on* P
Π	*partition* Π
Π_N	*partition that has* N *as a block and all blocks except* N *as singletons*
$\mathcal{P}_X$	*collection of all partitions of* X
$\Pi \leq \Pi'$	*partition* Π *is finer than* Π'

$\Pi \vee \Pi'$	*finest partition coarser than both* Π *and* Π'
$\Pi \wedge \Pi'$	*coarsest partition finer than both* Π *and* Π'

Functions,Set Functions and Operations on Functions

$f(\cdot)$	*function* $f(\cdot)$
$f/Z(\cdot), f(\cdot)$ *on* S	*restriction of* $f(\cdot)$ *to* $Z \subseteq S$
$gf(X), g \circ f(X)$	$g(f(X))$
$(f_1 \oplus f_2)(\cdot)$	*direct sum of functions* $f_1(\cdot)$ *and* $f_2(\cdot)$
$f_{fus \cdot \Pi}(\cdot), f(\cdot)$ *on* 2^S,	*fusion of* $f(\cdot)$ *relative to* Π
	i.e., $f_{fus.\Pi}(X_f)$
	$\equiv f(\bigcup_{T \in X_f} T), X_f \subseteq \Pi$
$f/\mathbf{X}(\cdot), f(\cdot)$ *on* 2^S	*restriction of* $f(\cdot)$ *to* $2^X, X \subseteq S$
	(*usually called*) *restriction of* $f(\cdot)$ *to* X
$f \diamond \mathbf{X}(\cdot), f(\cdot)$ *on* 2^S	*contraction of* $f(\cdot)$ *to* X
	$f \diamond \mathbf{X}(Y) \equiv f((S - X) \cup Y) - f(S - X)$
$f^d(\cdot), f(\cdot)$ *on* 2^S	*contramodular dual of* $f(\cdot)$
	$f^d(X) \equiv f(S) - f(S - X)$
$f^*(\cdot), f(\cdot)$ *on* 2^S	*comodular dual of* $f(\cdot)$
	(*with respect to weight function* $\alpha(\cdot)$)
	$f^*(X) \equiv \alpha(X) - (f(S) - f(S - X))$
$P_f, f(\cdot)$ *on* 2^S	*polyhedron associated with* $f(\cdot)$
	$\mathbf{x} \in P_f$ *iff* $x(X) \leq f(X) \quad \forall X \subseteq S$
$P_f^d, f(\cdot)$ *on* 2^S	*dual polyhedron associated with* $f(\cdot)$
	$\mathbf{x} \in P_f^d$ *iff* $x(X) \geq f(X) \quad \forall X \subseteq S$

Vectors and Matrices

$\mathcal{F}, \Re, \mathcal{C}, \Re_+$	*field* $\mathcal{F}$, *real field, complex field,*
	set of nonnegative reals
$\sum x_i$	*summation of elements* x_i
$\mathbf{f}$	*vector* $\mathbf{f}$
$\mathcal{V}$	*vector space* $\mathcal{V}$
$\mathcal{V}^\perp$	*vector space complementary orthogonal to* $\mathcal{V}$
$\mathbf{x}_1 \oplus \mathbf{x}_2$	*direct sum of* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ (*vector obtained by*

	adjoining components of vectors $\mathbf{x}_1$ *and* $\mathbf{x}_2$)
$\mathcal{V}_S \oplus \mathcal{V}_T, S \cap T = \emptyset$	*direct sum of* $\mathcal{V}_S$ *and* $\mathcal{V}_T$ (*obtained by collecting all possible direct sums of vectors in* $\mathcal{V}_S$ *and* $\mathcal{V}_T$)

$dim(\mathcal{V}), r(\mathcal{V})$	*dimension of vector space* $\mathcal{V}$
$d(\mathcal{V}, \mathcal{V}')$	$r(\mathcal{V} + \mathcal{V}') - r(\mathcal{V} \cap \mathcal{V}')$
$\mathbf{A}(i, j)$	i, j^{th} *entry of matrix* $\mathbf{A}$
$\mathbf{A}^T$	*transpose of matrix* $\mathbf{A}$
$\mathbf{A}^{-1}$	*inverse of matrix* $\mathbf{A}$
$< \mathbf{f}, \mathbf{g} >$	*dot product of vectors* $\mathbf{f}, \mathbf{g}$
$\mathcal{R}(\mathbf{A})$	*row space of* $\mathbf{A}$
$\mathcal{C}(\mathbf{A})$	*column space of* $\mathbf{A}$
$det(\mathbf{A})$	*determinant of* $\mathbf{A}$

Graphs and Vector Spaces

$\mathcal{G}$	*graph* $\mathcal{G}$
$V(\mathcal{G})$	*vertex set of* $\mathcal{G}$
$E(\mathcal{G})$	*edge set of* $\mathcal{G}$
t	*a tree*
f	*a forest*
$\bar{t}$	*cotree* $(E(\mathcal{G}) - t)$ *of* $\mathcal{G}$
$\bar{f}$	*coforest* $(E(\mathcal{G}) - f)$ *of* $\mathcal{G}$
$L(e, f)$	$f -$ *circuit of* e *with respect to* f
$B(e, f)$	$f -$ *cutset of* e *with respect to* f
$r(\mathcal{G})$	*rank of* $\mathcal{G}$ $(=$ *number of edges in a forest of* $\mathcal{G})$
$\nu(\mathcal{G})$	*nullity of* $\mathcal{G}$ $(=$ *number of edges in a coforest of* $\mathcal{G})$
$\mathcal{G}openT$	*graph obtained from* $\mathcal{G}$ *by opening and removing edges* T
$\mathcal{G}shortT$	*graph obtained from* $\mathcal{G}$ *by shorting and removing edges* T
$\mathcal{G} \cdot T$	*graph obtained from* $\mathcal{G}open(E(\mathcal{G}) - T)$ *by removing isolated vertices*,

	restriction of $\mathcal{G}$ *to* T
$\mathcal{G} \times T$	*graph obtained from* $\mathcal{G}short(E(\mathcal{G}) - T)$ *by removing isolated vertices*, *contraction of* $\mathcal{G}$ *to* T

$\mathcal{G}_1 \cong \mathcal{G}_2$	$\mathcal{G}_1$ *is* $2-$*isomorphic to* $\mathcal{G}_2$
$r(T)$	$r(\mathcal{G} \cdot T)$
$\nu(T)$	$\nu(\mathcal{G} \times T)$
$\mathcal{H}$	*hypergraph* $\mathcal{H}$
$B(V_L, V_R, E)$	*bipartite graph with left vertex set* V_L, *right vertex set* V_R *and edge set* E
$\mathbf{A}$	*(usually) incidence matrix*
$\mathbf{A_r}$	*reduced incidence matrix*
$\mathbf{Q}_f$	*fundamental cutset matrix of forest* f
$\mathbf{B}_f$	*fundamental circuit matrix of forest* f
KCE	*Kirchhoff's current equations*
KCL	*Kirchhoff's current Law*
KVE	*Kirchhoff's voltage equations*
KVL	*Kirchhoff's voltage Law*
$\mathcal{V}_i(\mathcal{G})$	*solution space of* KCE *of* $\mathcal{G}$
$\mathcal{V}_v(\mathcal{G})$	*solution space of* KVE *of* $\mathcal{G}$
$\mathcal{V} \cdot T$	*restriction of vector space* $\mathcal{V}$ *to* T
$\mathcal{V} \times T$	*contraction of vector space* $\mathcal{V}$ *to* T
$\xi(T)$ *for* $\mathcal{V}$	$r(\mathcal{V} \cdot T) - r(\mathcal{V} \times T)$

Flow Graphs

$F(\mathcal{G}) \equiv (\mathcal{G}, \mathbf{c}, s, t)$	*flow graph on graph* $\mathcal{G}$ *with capacity function* $\mathbf{c}$, *source* s, *sink* t
(A, B)	$cut(A, B)$
$c(A, B)$	*capacity of* $cut(A, B)$
$f(A, B)$	*flow across* $cut(A, B)$, *from* A *to* B
$\lvert\mathbf{f}\rvert$	*value of flow* $\mathbf{f}$
$F(B, \mathbf{c}_L, \mathbf{c}_R)$	*flowgraph associated with bipartite graph* B *with source to left vertex capacity* $\mathbf{c}_L$, *right*

vertex to sink capacity $\mathbf{c}_R$
and $(left\ to\ right)$ *bipartite graph edge capacity* ∞

Matroids

$\mathcal{M} \equiv (S, \mathcal{I})$	*matroid* $\mathcal{M}$
$\mathcal{I}$	*collection of independent sets*
$\mathcal{M}^*$	*dual of the matroid* $\mathcal{M}$
B	(*only in Chapter* 4) *base of a matroid*
$L(e, B)$	$f-$*circuit of* e *with respect to base* B
$B(e, B)$	$f-$*bond of* e *with respect to base* B
$r(T)$	*rank of the subset* T *in the given matroid*
$r(\mathcal{M})$	*rank of the underlying set of the matroid*
$\nu(T)$	*rank of the subset* T *in the dual of the given matroid*
$\nu(\mathcal{M})$	*rank of the underlying set in the dual matroid*
$\mathcal{M}(\mathcal{G})$	*polygon matroid of the graph* $\mathcal{G}$ (*bases are forests*)
$\mathcal{M}^*(\mathcal{G})$	*bond matroid of the graph* $\mathcal{G}$ (*bases are coforests*)
$\mathcal{M}(\mathcal{V})$	*matroid whose bases are maximal independent columns of a representative matrix of* $\mathcal{V}$
$\mathcal{M}^*(\mathcal{V})$	*dual of* $\mathcal{M}(\mathcal{V})$
$\int(X)$	*span* (*closure*) *of the subset* X *in the matroid*
$\mathcal{M} \cdot T$	*restriction of* $\mathcal{M}$ *to* T
$\mathcal{M} \times T$	*contraction of* $\mathcal{M}$ *to* T
$\mathcal{M}_1 \vee \mathcal{M}_2$	*union of matroids* $\mathcal{M}_1$ *and* $\mathcal{M}_2$

Electrical Networks

$\mathbf{v}$	*voltage vector*
$\mathbf{i}$	*current vector*
$\mathcal{N}$	*electrical network*
$\mathcal{N}_{AP}$	*electrical multiport with port set* P *and remaining edge set* A
$\mathcal{E}$	*set of voltage sources in the network*

$\mathcal{J}$	*set of current sources in the network*
R	*resistance, also collection of resistors or* ‘*current controlled voltage*′ *elements in the network*

G	*conductance, also collection of 'voltage controlled current' elements in the network*
L	*inductance, also collection of inductors in the network*
$\mathcal{L}$	*mutual inductance matrix*
C	*capacitance, also collection of capacitors in the network*
$vcvs$	*voltage controlled voltage source*
$vccs$	*voltage controlled current source*
$ccvs$	*current controlled voltage source*
$cccs$	*current controlled current source*
$\mathcal{D}$	*device characteristic*
$\mathcal{D}_{AB}$	$(\mathbf{v}/A, \mathbf{i}/B)$, *where* $\mathbf{v}, \mathbf{i} \in \mathcal{D}$
$\mathcal{D}_A$	$\mathcal{D}_{AA}$
$\mathcal{D}_{AB} \times \mathcal{D}_{PQ}$	$\{(\mathbf{v}, \mathbf{i}),\ \mathbf{v} = \mathbf{v}_A \oplus \mathbf{v}_P, \mathbf{i} = \mathbf{i}_B \oplus \mathbf{i}_Q$ *where* $(\mathbf{v}_A, \mathbf{i}_B) \in \mathcal{D}_{AB}, (\mathbf{v}_P, \mathbf{i}_Q) \in \mathcal{D}_{PQ}\}$
δ_{AB}	$\{(\mathbf{v}_A, \mathbf{i}_B),\ \mathbf{v}_A$ *is any vector on* $A,\ \mathbf{i}_B$ *is any vector on* $B\}$

Implicit Duality

$\mathcal{K}_{SP} \leftrightarrow \mathcal{K}_P$	$\{f_S : f_S = f_{SP}/S,\ f_{SP} \in \mathcal{K}_{SP}\ \ s.t.\ f_{SP}/P \in \mathcal{K}_P\}$
$\mathcal{K}_{S_1} \leftrightarrow \mathcal{K}_{S_2}$	$\{\mathbf{f} : \mathbf{f} = \mathbf{f}_1/(S_1 - S_2) \oplus \mathbf{f}_2/(S_2 - S_1),\ \mathbf{f}_1 \in \mathcal{K}_{S_1},$ $\mathbf{f}_2 \in \mathcal{K}_{S_2}$ *and* $\mathbf{f}_1/S_1 \cap S_2 = \mathbf{f}_2/S_1 \cap S_2\}$
$\mathcal{K}_{S_1} \rightleftharpoons \mathcal{K}_{S_2}$	$\{\mathbf{f} : \mathbf{f} = \mathbf{f}_1/(S_1 - S_2) \oplus \mathbf{f}_2/(S_2 - S_1),\ \mathbf{f}_1 \in \mathcal{K}_{S_1},$ $\mathbf{f}_2 \in \mathcal{K}_{S_2}$ *and* $\mathbf{f}_1/S_1 \cap S_2 = -\mathbf{f}_2/S_1 \cap S_2\}$
$< \cdot, \cdot >$	*a* $q-$*bilinear operation, usually dot product*
$\mathcal{K}^*$	*collection of vectors* $q-$*orthogonal to those in* $\mathcal{K}$
$d(\mathcal{V}, \mathcal{V}')$	$r(\mathcal{V} + \mathcal{V}') - r(\mathcal{V} \cap \mathcal{V}')$
$\mathcal{K}^p$	*the collection of vectors polar to those in* $\mathcal{K}$

$\mathcal{K}^d$ (*only in Chapter* 7) *the collection of vectors integrally dual to those in* $\mathcal{K}$

Multiport Decomposition

$(\mathcal{V}_{E_1P_1}, \cdots, \mathcal{V}_{E_kP_k}; \mathcal{V}_P)$	$k-$ *multiport decomposition of* $\mathcal{V}_E$ *(i.e.,* $(\bigoplus_i \mathcal{V}_{E_iP_i}) \leftrightarrow \mathcal{V}_P = \mathcal{V}_E$*)*
$((\mathcal{V}_{E_jP_j})_k; \mathcal{V}_P)$	$(\mathcal{V}_{E_1P_1}, \cdots, \mathcal{V}_{E_kP_k}; \mathcal{V}_P)$
$((\mathcal{V}_{E_jP_j})_{j\in I}; \mathcal{V}_{P_I})$	$(\cdots \mathcal{V}_{E_jP_j} \cdots; \mathcal{V}_{P_I})$ *where* $j \in I \subseteq \{1, \cdots, k\}$ *and* $P_I = \cup_{j\in I} P_j$
$(\mathcal{V}_{EP}, P)$	*vector space on* $E \uplus P$ *with ports* P
$(\mathcal{V}_{E_1Q_1}, \cdots, \mathcal{V}_{E_kQ_k}; \mathcal{V}_{QP})$	*matched or skewed decomposition of* $(\mathcal{V}_{EP}, P)$

Functions Associated with Graphs and Bipartite Graphs

$V(X), X \subseteq E(\mathcal{G})$	*set of endpoints of edges* X *in graph* $\mathcal{G}$
$\Gamma(X), X \subseteq V(\mathcal{G})$	*set of vertices adjacent to vertices in vertex subset* X *in graph* $\mathcal{G}$
$\Gamma_L(X), X \subseteq V_L$	*in* $B \equiv (V_L, V_R, E)$*, set of vertices adjacent to vertices in* X
$\Gamma_R(X), X \subseteq V_R$	*in* $B \equiv (V_L, V_R, E)$*, set of vertices adjacent to vertices in* X
$E(X), X \subseteq V(\mathcal{G})$	*set of edges with both endpoints in vertex subset* X *in graph* $\mathcal{G}$
$E_L(X), X \subseteq V_L$	*in* $B \equiv (V_L, V_R, E)$ *set of vertices adjacent* **only** *to vertices in* X
$E_R(X), X \subseteq V_R$	*in* $B \equiv (V_L, V_R, E)$ *set of vertices adjacent* **only** *to vertices in* X
$I(X), X \subseteq V(\mathcal{G})$	*set of edges with atleast one endpoint in vertex subset* X *in graph* $\mathcal{G}$
$cut(X), X \subseteq V(\mathcal{G})$	*set of edges with exactly one endpoint in vertex subset* X *in graph* $\mathcal{G}$
$w(\cdot)$	*usually a weight function*
$w_L(\cdot), w_R(\cdot)$	*weight functions on the left vertex set*

and on the right vertex set respectively
of a bipartite graph

Convolution and PP

$f * g(X)$	*convolution of* $f(\cdot)$ *and* $g(\cdot)$ $(min_{Y \subseteq X}[f(Y) + g(X - Y)])$
$\mathcal{B}_{\lambda_{f,g}}, f(\cdot), g(\cdot)$ *on* 2^S	*collection of sets which minimize* $\lambda f(X) + g(S - X)$ *over subsets of* S
$\mathcal{B}_\lambda$	$\mathcal{B}_{\lambda_{f,g}}$
X^λ, X_λ	*maximal and minimal members of* $\mathcal{B}_\lambda$
$\Pi(\lambda)$	*the partition of* $X^\lambda - X_\lambda$ *induced by* $\mathcal{B}_\lambda$
Π_{pp}	*the partition of* S *obtained by taking the union of all the* $\Pi(\lambda)$
$(\Pi_{pp}, \geq_\pi)$	*partition partial order pair associated with* $(f(\cdot), g(\cdot))$
$\emptyset, E_1, \cdots, E_t$	*(usually) the principal sequence of* $(f(\cdot), g(\cdot))$
$\lambda_1, \cdots, \lambda_t$	*(usually) decreasing sequence of critical values*
$(\geq_R)$	*refined partial order associated with* $(f(\cdot), g(\cdot))$

Truncation and PLP

$\overline{f}(\Pi)$	$\sum_{N_i \in \Pi} f(N_i)$
$f_t(\cdot)$	$f_t(\emptyset) \equiv 0$, $f_t(X) \equiv min_{\Pi \in \mathcal{P}_X}(\sum_{X_i \in \Pi} f(X_i))$
$\mathcal{L}_{\lambda_f}, f(\cdot)$ *on* 2^S	*collection of all partitions of* S *that minimize* $\overline{f - \lambda}(\cdot)$
$\mathcal{L}_\lambda$	$\mathcal{L}_{\lambda_f}$
Π^λ, Π_λ	*maximal and minimal member partitions in* $\mathcal{L}_\lambda$
$\lambda_1, \cdots, \lambda_t$	*(usually) decreasing sequence of critical PLP values of* $f(\cdot)$
$\Pi_{\lambda_1}, \Pi_{\lambda_2}, \cdots, \Pi_{\lambda_t}, \Pi^{\lambda_t}$	*principal sequence of partitions of* $f(\cdot)$
$\Pi'_{fus \cdot \Pi}, \Pi' \geq \Pi$	*partition of* Π *with* N_{fus} *as one of its blocks iff the members of* N_{fus} *are the set of blocks of* Π

	contained in a single block of Π'
$(\Pi_{fus})_{exp.\Pi}$	(Π_{fus}, *a partition of* Π) *a partition with* N *as a block,* iff N *is the union of all blocks of* Π *which are members of a single block of* Π_{fus}

xxx

Contents

1 Introduction **1**

2 Mathematical Preliminaries **19**

2.1 Sets 19

2.2 Vectors and Matrices 21

2.3 Linear Inequality Systems 31

2.3.1 The Kuhn-Fourier Theorem 31

2.3.2 Linear Programming 35

2.4 Solutions of Exercises 38

2.5 Solutions of Problems 38

3 Graphs **41**

3.1 Introduction 41

3.2 Graphs: Basic Notions 41

3.2.1 Graphs and Subgraphs 41

3.2.2 Connectedness 43

3.2.3 Circuits and Cutsets 45

3.2.4 Trees and Forests 48

3.2.5 Strongly Directedness 49

3.2.6 Fundamental Circuits and Cutsets 50

3.2.7 Orientation 51

3.2.8 Isomorphism 53
3.2.9 Cyclically connectedness 54
3.3 Graphs and Vector Spaces 55
3.3.1 The Circuit and Crossing Edge Vectors 58
3.3.2 Voltage and Current Vectors 59
3.3.3 Voltage and Current Vector Spaces and Tellegen's Theorem 61
3.3.4 Fundamental cutset matrix of a forest f 62
3.3.5 Fundamental circuit matrix of a forest f 64
3.4 Basic Operations on Graphs and Vector Spaces 67
3.4.1 Restriction and Contraction of Graphs 68
3.4.2 Restriction and Contraction of Vector Spaces . . 71
3.4.3 Vector Space Duality 73
3.4.4 Relation between Graph Minors and Vector Space Minors 74
3.4.5 Representative Matrices of Minors 76
3.4.6 Minty's Theorem 79
3.5 Problems 81
3.6 Graph Algorithms 87
3.6.1 Breadth First Search 90
3.6.2 Depth First Search 91
3.6.3 Minimum Spanning Tree 93
3.6.4 Shortest Paths from a Single Vertex 94
3.6.5 Restrictions and Contractions of Graphs 96
3.6.6 Hypergraphs represented by Bipartite Graphs . 97
3.6.7 Preorders and Partial Orders 97
3.6.8 Partitions 99
3.6.9 The Max-Flow Problem 101
3.6.10 Flow Graphs Associated with Bipartite Graphs 107

3.7 Duality 111
3.8 Notes 116
3.9 Solutions of Exercises 116
3.10 Solutions of Problems 132

4 Matroids **139**
4.1 Introduction 139
4.2 Axiom Systems for Matroids 139
4.2.1 Independence and Base Axioms 140
4.2.2 Rank Axioms 142
4.2.3 Circuit Axioms 144
4.2.4 Closure Axioms 146
4.3 Dual of a Matroid 148
4.4 Minors of Matroids 153
4.5 Connectedness in Matroids 160
4.5.1 Duality for Matroids 162
4.6 Matroids and the Greedy Algorithm 164
4.7 Notes 167
4.8 Solutions of Exercises 168

5 Electrical Networks **177**
5.1 Introduction 177
5.2 In Terms of Multiterminal Devices 178
5.3 In Terms of 2-Terminal Devices 180
5.4 Standard Devices 181
5.5 Common Methods of Analysis 190
5.5.1 Nodal Analysis 190
5.5.2 Loop Analysis 193
5.5.3 Modified Nodal Analysis 196

5.5.4 Sparse Tableau Approach 201
5.6 Procedures used in Circuit Simulators 201
5.6.1 Example to Illustrate Working of Circuit Simulators 201
5.6.2 Working of General Purpose Circuit Simulators 203
5.7 State Equations for Dynamic Networks 209
5.8 Multiports in Electrical Networks 212
5.8.1 An informal Description of Multiport Decomposition 213
5.8.2 Thevenin-Norton Theorem 215
5.9 Some Elementary Results of Network Theory 218
5.10 Notes 220
5.11 Solutions of Exercises 220

6 Topological Hybrid Analysis 227
6.1 Introduction 227
6.2 Electrical Network: A Formal Description 228
6.2.1 Static and Dynamic Electrical Networks 229
6.2.2 Device Decoupling 231
6.3 Some Basic Topological Results 233
6.3.1 Effect of Voltage Unconstrained and Current Unconstrained Devices on the Topological Constraints 233
6.3.2 Voltage and Current shift 236
6.4 A Theorem on Topological Hybrid Analysis 244
6.4.1 The Networks $\mathcal{N}_{AL}$ and $\mathcal{N}_{BK}$ 244
6.5 Structure of Constraints and Optimization 250
6.5.1 Essential Structure of the Constraints 250
6.5.2 Selection of Minimal L and K 251
6.5.3 Solution of Linear Networks by Topological Hybrid Analysis 257

6.5.4 Decomposition procedure for $\mathcal{N}_{AL}, \mathcal{N}_{BK}$ 260
6.5.5 Hybrid Analysis Equations for Linear Networks 262
6.5.6 The Hybrid Rank 266
6.6 Notes 269
6.7 Solutions of Exercises 269

7 The Implicit Duality Theorem and Its Applications 277
7.1 The Vector Space Version 277
7.1.1 The Implicit Duality Theorem: Orthogonality Case 279
7.1.2 Matched and Skewed Sums 282
7.2 *Quasi Orthogonality 285
7.3 Applications of the Implicit Duality Theorem 288
7.3.1 Ideal Transformer Connections 289
7.3.2 Multiport Decomposition 290
7.3.3 Topological Transformation Of Electrical Networks 292
7.3.4 The Adjoint of a Linear System 297
7.3.5 Rank, Nullity and the Hybrid rank 302
7.4 *Linear Inequality Systems 303
7.4.1 Applications of the Polar Form 308
7.5 *Integrality Systems 309
7.6 Problems 316
7.7 Notes 321
7.8 Solutions of Exercises 321
7.9 Solutions of Problems 334

8 Multiport Decomposition 351
8.1 Introduction 351
8.2 Multiport Decomposition of Vector Spaces 352

8.3 Analysis through Multiport Decomposition 360
8.3.1 Rewriting Network Constraints in the Multiport Form 361
8.3.2 An Intuitive Procedure for Solution through Multiports 362
8.4 Port Minimization 367
8.4.1 An Algorithm for Port Minimization 367
8.4.2 Characterization of Minimal Decomposition . . 373
8.4.3 Complexity of Algorithm (Port minimization 1) for Graphic Spaces and Sparsity of the Output Matrices 380
8.4.4 *Minimal Decomposition of Graphic Vector Spaces to make Component Spaces Graphic 382
8.5 *Multiport Decomposition for Network Reduction . . . 386
8.6 Problems 393
8.7 Solutions of Exercises 397
8.8 Solutions of Problems 409

9 Submodular Functions 419
9.1 Introduction 419
9.2 Submodularity 420
9.3 Basic Operations on Semimodular Functions 425
9.4 *Other Operations on Semimodular Functions 432
9.5 Polymatroid and Matroid Rank Functions 438
9.6 Connectedness for Semimodular Functions 444
9.7 *Semimodular Polyhedra 447
9.8 Symmetric Submodular Functions 457
9.9 Problems 463
9.10 Notes 467
9.11 Solutions of Exercises 467

9.12 Solutions of Problems 477

10 Convolution of Submodular Functions 489

10.1 Introduction 489
10.2 Convolution 490
10.2.1 Formal Properties 490
10.2.2 Examples 492
10.2.3 Polyhedral interpretation for convolution 495
10.3 Matroids, Polymatroids and Convolution 497
10.4 The Principal Partition 500
10.4.1 Introduction 500
10.4.2 Basic Properties of PP 501
10.4.3 Symmetry Properties of the Principal Partition of a Submodular Function 508
10.4.4 Principal Partition from the Point of View of Density of Sets 510
10.4.5 Principal Partition of $f^*(\cdot)$ and $f * g(\cdot)$ 513
10.4.6 The Principal Partition associated with Special Minors 519
10.5 *The Refined Partial Order of the Principal Partition . 525
10.6 Algorithms for PP 533
10.6.1 Basic Algorithms 533
10.6.2 *Building the refined partial order given $(\Pi_{pp}, \geq_\pi)$.539
10.6.3 Algorithm $Convolve_S(w_R(\Gamma_L), w_L)$ 540
10.6.4 Example: PP of $(|\Gamma_L|(\cdot), w_L(\cdot))$ 542
10.7 *Aligned Polymatroid Rank Functions 545
10.8 Notes 558
10.9 Solutions of Exercises 559
10.10Solutions of Problems 576

11 Matroid Union **585**

11.1 Introduction 585

11.2 Submodular Functions induced through a Bipartite Graph 585

11.3 Matroid Union: Algorithm and Structure 594

11.3.1 Introduction 594

11.3.2 The Algorithm 595

11.3.3 Justification and complexity of Algorithm Matroid Union 597

11.3.4 Structure of the Matroid Union 600

11.4 PP of the Rank Function of a Matroid 607

11.4.1 Constructing $\mathcal{B}_{\lambda_{r,|\cdot|}}$ 607

11.4.2 Complexity of constructing $\mathcal{B}_{\lambda_{r,|\cdot|}}$ 609

11.4.3 Example 612

11.5 Notes 616

11.6 Solutions of Exercises 616

12 Dilworth Truncation of Submodular Functions **623**

12.1 Introduction 623

12.2 Dilworth Truncation 624

12.2.1 Formal Properties 624

12.2.2 Examples 631

12.3 The Principal Lattice of Partitions 634

12.3.1 Basic Properties of the PLP 634

12.3.2 PLP from the Point of View of Cost of Partitioning 644

12.4 *Approximation Algorithms through PLP for the Min Cost Partition Problem 652

12.5 The PLP of Duals and Truncations 657

12.5.1 The PLP of Duals 658

12.5.2 The PLP of the Truncation 660

12.5.3 The Cotruncation Operation and the Principal Lattice of Copartitions 663
12.6 *The Principal Lattice of Partitions associated with Special Fusions 665
12.7 Building Submodular Functions with desired PLP . . . 669
12.8 Notes 673
12.9 Solutions of Exercises 673
12.10Solutions of Problems 688

13 Algorithms for the PLP of a Submodular Function 691
13.1 Introduction 691
13.2 Minimizing the Partition Associate of a Submodular function 692
13.2.1 Find (Strong) Fusion Set 694
13.2.2 Min$(\bar{f}, S)$ 696
13.3 Construction of the P-sequence of Partitions 699
13.4 Construction of the DTL 704
13.5 Complexity of construction of the PLP 707
13.6 Construction of the PLP of the dual 708
13.7 PLP Algorithms for $(w_R\Gamma)(\cdot)$ and $-(w_R E_L)(\cdot)$ 708
13.7.1 PLP of $(w_R\Gamma)(\cdot)$ 709
13.7.2 PLP of $(-w_R E_L)(\cdot)$ 715
13.8 Structural Changes in Minimizing Partitions 721
13.9 Relation between PP and PLP 726
13.10Fast Algorithms for Principal Partition of the rank function of a graph 733
13.11Solutions of Exercises 735

14 The Hybrid Rank Problem 743
14.1 Introduction 743

14.2 The Hybrid Rank Problem - First Formulation 744
14.3 The Hybrid Rank Problem - Second Formulation . . . 749
14.3.1 Introduction 749
14.3.2 Membership Problem with Matroid Expansion . 750
14.3.3 Membership Problem with Graphic Matroid Expansion 758
14.3.4 PLP of the rank function of a matroid 765
14.4 The Hybrid Rank Problem - Third Formulation 766
14.4.1 Introduction 766
14.4.2 Fusions and Fissions 769
14.4.3 Relation between the Hybrid Rank of a Graph and its Hybrid Rank relative to a Partition . . . 776
14.5 The Hybrid Rank Problem - Fourth Formulation 779
14.5.1 Introduction 779
14.5.2 Generalized Fusions and Fissions 781
14.5.3 Port Decomposition and Generalized Hybrid Rank784
14.5.4 Relation between the Hybrid Rank of a Representative Matrix of a Vector Space and its Generalized Hybrid Rank relative to a Partition . . 789
14.5.5 Nesting Property of Optimal Subspaces 795
14.6 Solutions of Exercises 799

Chapter 1

Introduction

Topological Methods

The methods described in this book could be used to study the properties of electrical networks that are independent of the device characteristic. We use only topological constraints, namely, KCL and KVL. Our methods could, therefore, be called 'network topological'. However, in the literature, 'topological' is used more loosely for all those results which use topological ideas, e.g. Kirchhoff's Third Law, where the admittance of a resistive multiport is obtained in terms of products of admittances present in all the trees and certain special kinds of subtrees of the network. These latter results, though important, are not touched upon in this book. Here our aim has been to

- give a detailed description of 'topological methods in the strict sense' for electrical networks,
- present applications:
  - to circuit simulation and circuit partitioning
  - to establish relations between the optimization problems that arise naturally, while using these methods, to the central problems in the theory of submodular functions.

Applications

There are two kinds of applications possible for the approach taken in this book:

i. To build better (faster, numerically more rugged, parallelizable) circuit simulators. Typically, our methods will permit us to speak as follows.

> *'Solution of a network $\mathcal{N}$ containing arbitrary devices is equivalent to solution of topologically derived networks $\mathcal{N}_1, \cdots, \mathcal{N}_k$ under additional topological conditions.'*

An obvious application would be for the (coarse grained) parallelization of circuit simulation. We could have a number of machines $M_1, \cdots, M_k$ which could run general/special purpose circuit simulation of the derived networks $\mathcal{N}_1, \cdots, \mathcal{N}_k$. The central processor could combine their solutions using the additional topological conditions. Optimization problems would arise naturally, e.g. 'how to minimize the additional topological conditions?'
There are more immediate applications possible. The most popular general purpose simulator now running, SPICE, uses the modified nodal analysis approach. In this approach the devices are divided into two classes, generalized admittance type whose currents can be written in terms of voltages appearing somewhere in the circuit, and the remaining devices whose current variables will figure in the list of unknowns. The final variables in terms of which the solution is carried out would be the set of all nodal voltages and the above mentioned current variables. The resulting coefficient matrix is very sparse but suffers from the following defects:

- the matrix often has diagonal zeros;
- even for pure RLC circuits the coefficient matrix is not positive definite;
- if the subnetwork containing the admittance devices is disconnected, then the corresponding principal submatrix is singular.

These problems are not very severe if we resort to sparse LU methods [Hajj81]. However, it is generally accepted that for large enough networks ($\approx$ 5000 nodes) preconditioned conjugate gradient methods would prove superior to sparse LU techniques. The main advantage of the former is that if the matrix is close to a positive definite matrix, then we can bound the number of iterations. The above defects make MNA ill suited to conjugate gradient methods.

There is a simple way out - viz. to use hybrid analysis (partly loop and partly nodal), where we partition elements into admittance type and impedance type. The structure of the coefficient matrix that is obtained in this latter case is well suited to solution obtained by the conjugate gradient technique but could easily, for wrong choice of variables, be dense. A good way of making the matrices sparse is to use the result that we call the '$\mathcal{N}_{AL} - \mathcal{N}_{BK}$ theorem' (see Section 6.4). Here the network is decomposed into two derived networks whose solution under additional topological (boundary) conditions is always equivalent to the solution of the original network. We select $\mathcal{N}_{AL}$ so that it contains the admittance type elements and $\mathcal{N}_{BK}$ so that it contains the impedance type elements. We then write nodal equations for $\mathcal{N}_{AL}$ and generalized mesh type equations for $\mathcal{N}_{BK}$. The result is a sparse matrix with good structure for using conjugate gradient methods - for instance for RLC networks, after discretization, we would get a positive definite matrix and for most practical networks, a large submatrix would be positive definite. A general purpose simulator BITSIM has been built using these ideas [Roy+Gaitonde+Narayanan90].

The application to circuit partitioning arises as a biproduct when we try to solve a version of the hybrid rank problem using the operation of Dilworth truncation on submodular functions. Many problems in the area of CAD for VLSI need the underlying graph/hypergraph to be partitioned such that the 'interaction' between blocks is minimized. For instance we may have to partition the vertex set of a graph so that the number of lines going between blocks is a minimum. This kind of problem is invariably NP-Hard. But, using the idea of principal lattice of partitions (PLP), we can solve a relaxation of such problems exactly. This solution can then be converted to an approximate solution of the original problem [Narayanan91], [Roy+Narayanan91], [Patkar92], [Roy93],[Roy+Narayanan93a] [Narayanan+Roy+Patkar96].

ii. A second kind of application is to establish strong relationships between electrical networks and combinatorial optimization, in particular, submodular function theory. There are a number of optimization problems which arise when we view electrical networks from a topological point of view. These motivate, and are solved by, important concepts such as convolution and Dilworth truncation of submodular functions. The hybrid rank problem and its generalizations are important instances. Other algorithmic problems (though not entirely topological) include the solvability of electrical networks under 'generality' conditions (see for instance [Recski+Iri80]). It is no longer possible for electrical engineers to directly apply well established mathematical concepts. They themselves often have to work out the required ideas. The principal partition is a good instance of such an idea conceived by electrical engineers. A nice way of developing submodular function theory, it appears to the author, is to look for solutions to problems that electrical networks throw up.

We now present three examples which illustrate the concepts that we will be concerned with in network analysis.

The following informal rule should be kept in mind while reading the examples (see Theorem 6.3.1 and also the remark on page 235).

> *Let $\mathcal{N}$ be an electrical network (not necessarily linear) with the set of independent current sources $E_{\mathcal{J}}$ and the set of independent voltage sources $E_{\mathcal{E}}$. We assume that the independent source values do not affect the device characteristic of the remaining devices. Then, the structure of the constraints of the network, in any method of analysis, (as far as variables other than voltage source currents and current source voltages are concerned) is that corresponding to setting the independent sources to zero, i.e., short circuiting voltage sources and open circuiting current sources. In particular, for linear networks, the structure of the coefficient matrix multiplying the unknown vector is that corresponding to the network obtained by short circuiting the voltage sources and open circuiting the current sources.*

Example 1.0.1 The $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ method:
Consider the electrical network whose graph is given in Figure 1.1. We

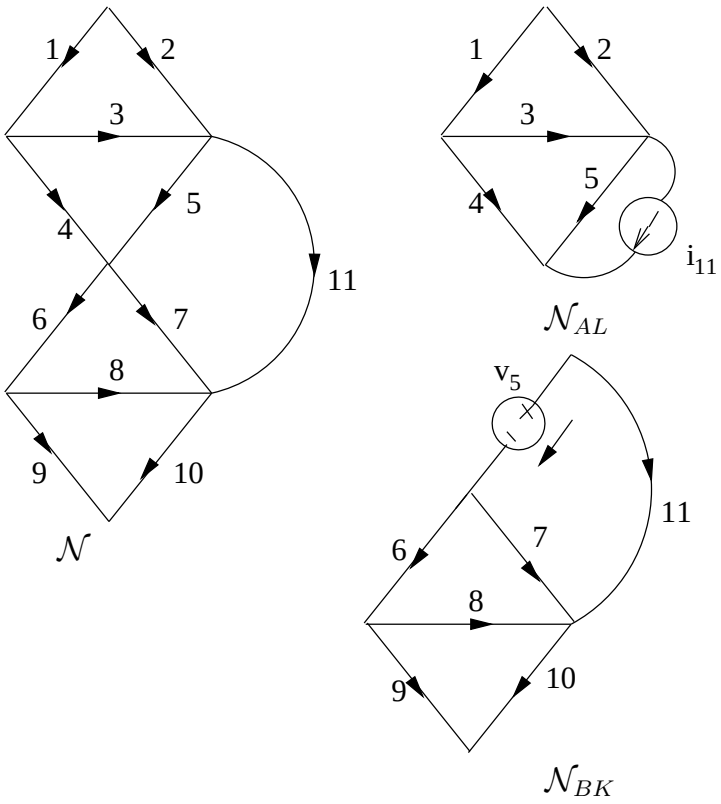


Figure 1.1: To illustrate the $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ Method

assume that the devices associated with branches $\{1,2,3,4,5\}$ $(= A)$ *are independent of those associated with branches* $\{6, 7, 8, 9, 10, 11\}$ $(= B)$. *Then we can show that computing the solution of the network in* $\mathcal{N}$ *in Figure 1.1 is always equivalent to the simultaneous computation of the solutions of the networks* $\mathcal{N}_{AL}, \mathcal{N}_{BK}$, *in the same figure, under the boundary conditions*

$$i_{11} \text{ in } \mathcal{N}_{AL} = i_{11} \text{ in } \mathcal{N}_{BK}.$$

$$v_5 \text{ in } \mathcal{N}_{AL} = v_5 \text{ in } \mathcal{N}_{BK}.$$

Here, in $\mathcal{N}_{AL}$, *the devices in* A *are identical to the corresponding devices in* $\mathcal{N}$. *Similarly in* $\mathcal{N}_{BK}$, *devices in* B *are identical to the corresponding devices in* $\mathcal{N}$. *The subset* $L \subseteq B$ *is a set of branches which, when deleted, breaks all circuits intersecting both* A *and* B. *The subset* $K \subseteq A$ *is a set of branches which, when contracted, destroys all circuits intersecting both* A *and* B. *The graph of* $\mathcal{N}_{AL}$ *is obtained from that of* $\mathcal{N}$ *by short circuiting the branches of* $B - L$. *We denote it by* $\mathcal{G} \times (A \cup L)$. *In this case* $L = \{11\}$. *The graph of* $\mathcal{N}_{BK}$ *is obtained from that of* $\mathcal{N}$ *by open circuiting branches of* $A - K$. *We denote it by* $\mathcal{G}\ .\ (B \cup K)$. *In this case* $K = \{5\}$.

If the network is linear and A *and* B *are of conductance and impedance type respectively, then we can, if we choose, solve* $\mathcal{N}_{AL}$ *by nodal analysis and* $\mathcal{N}_{BK}$ *by loop analysis. So this method can be regarded as a topological generalization of 'Hybrid Analysis.'*

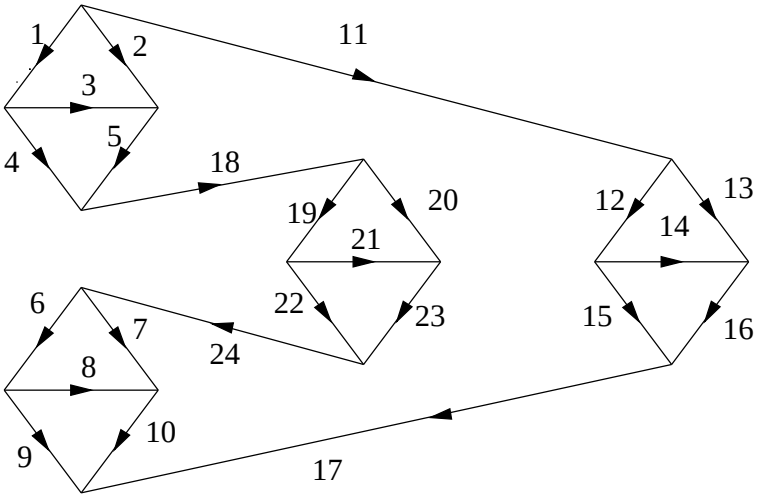
If we so desire, we could try to choose $\mathcal{N}_{AL}$ *or* $\mathcal{N}_{BK}$ *such that they appear (when* i_L, v_k *are set to zero) in several electrically disconnected pieces. So the method can be regarded as a technique of 'Network Analysis by Decomposition'.*

Now we mention some related combinatorial optimization problems.

i. Given a partition of the edges into A *and* B *how to choose* L, K *minimally - this is easy.*

ii. Suppose the network permits arbitrary partitions into A *and* B *and we choose nodal variables for* $\mathcal{N}_{AL}$ *and loop variables for* $\mathcal{N}_{BK}$. *Which partition would give the coefficient matrix of least size?*

It can be shown that the size of the coefficient matrix is $r(\mathcal{G}\ .\ A)+\nu(\mathcal{G}\times B)$*, where* $r(\mathcal{G}\ .\ A), \nu(\mathcal{G}\times B)$ *respectively denote the rank of* $\mathcal{G}\ .\ A$ *and nullity of graph* $\mathcal{G}\times B$*. Minimization of this expression, over all partitions* $\{A, B\}$ *of the edge set* $E(\mathcal{G})$ *of* $\mathcal{G}$*, is the hybrid rank problem which gave rise to the theory of principal partition.*



$\mathcal{N}$

Figure 1.2: A Network to be decomposed into Multiports

Example 1.0.2 Multiport Decomposition:
Let $\mathcal{N}$ *be an electrical network with the graph shown in Figure 1.2. We are given that* $A \equiv \{1, 2\cdots, 10\}$ *and* $B \equiv \{11, \cdots, 24\}$ *(with devices in A and B decoupled). The problem is to split* $\mathcal{N}$ *into two multiports* $\mathcal{N}_{AP_1}, \mathcal{N}_{BP_2}$ *and a 'port connection diagram'* $\mathcal{N}_{P_1P_2}$ *and solve* $\mathcal{N}$ *by solving* $\mathcal{N}_{AP_1}, \mathcal{N}_{BP_2}, \mathcal{N}_{P_1P_2}$ *simultaneously. (In general this would be a problem involving n multiports). It is desirable to choose* P_1, P_2 *minimally. It turns out that*

$$\mid P_1 \mid = \mid P_2 \mid = r(\mathcal{G}\ .\ A) - r(\mathcal{G}\times A) = r(\mathcal{G}\ .\ B) - r(\mathcal{G}\times B).$$

(Here $\mathcal{G}\ .\ A$ *is obtained by open circuiting edges in B, while* $\mathcal{G}\times A$ *is obtained by short circuiting edges in B). In this case this number is 1. The multiports are shown in Figure 1.3.*

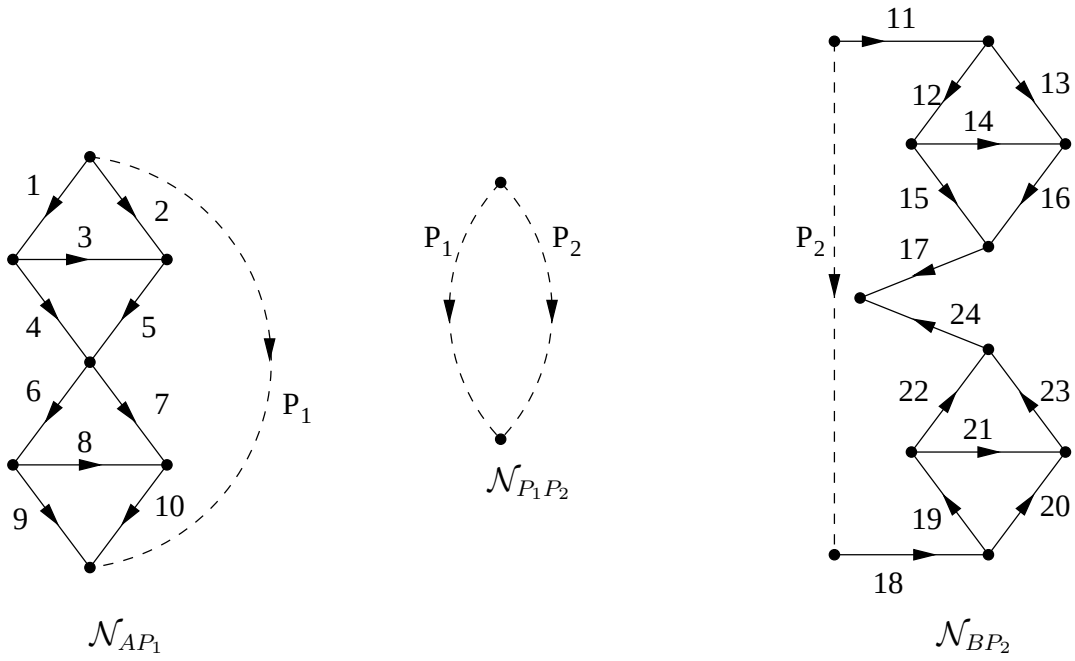


Figure 1.3: Decomposition into Multiports

The general solution procedure using multiport decomposition is as follows: Find the voltage-current relationship imposed on P_1 by the rest of the network in $\mathcal{N}_{AP_1}$, and on P_2, by the rest of the network in $\mathcal{N}_{BP_2}$. This involves solution of $\mathcal{N}_{AP_1}, \mathcal{N}_{BP_2}$ in terms of some of the current/voltage port variables of N_{AP_1} and some of the current/voltage port variables of N_{BP_2}. The voltage-current relationships imposed on P_1, P_2 (as described above) are treated as their device characteristics in the network $\mathcal{N}_{P_1P_2}$. When this is solved, we get the currents and voltages of P_1, P_2. Networks $\mathcal{N}_{AP_1}, \mathcal{N}_{BP_2}$ have already been solved in terms of these variables. So this completes the solution of $\mathcal{N}$. Like the $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ method (to which it is related), this is also a general method independent of the type of network. As before, the technique is more useful when the network is linear.

This method again may be used as a network decomposition technique (for parallelizing) at a different level. Suppose $\mathcal{N}_{AP_1}$ (or $\mathcal{N}_{BP_2}$) splits into several subnetworks when some of the branches P_{O1} (P_{O2}) of $P_1(P_2)$ are opened and others $P_{S1}(P_{S2})$ shorted. Then, by using $i_{P_{O1}}(i_{P_{O2}}), v_{P_{S1}}(v_{P_{S2}})$, as variables in terms of which $\mathcal{N}_{AP_1}(\mathcal{N}_{BP_2})$ are solved, we can make the analysis look like the simultaneous solution of several subnetworks under boundary conditions. There is no restriction on the type of network - we only need the subnetworks to be decoupled in the device characteristic. The optimization problem that arises naturally in this case is the following:

> *Given a partition of the edges of a network $\mathcal{N}$ into $E_1, \cdots, E_k$, find a collection of multiports $\mathcal{N}_{E_1P_1}, \cdots, \mathcal{N}_{E_kP_k}$, and a port connection diagram $\mathcal{N}_{P_1,\cdots,P_k}$, whose combined KCE and KVE are equivalent to those of $\mathcal{N}$, with the size of $\biguplus P_i$ a minimum under these conditions.*

This problem is solved in Chapter 8.

Remark: At an informal level multiport decomposition is an important technique in classical network theory e.g. Thevenin-Norton Theorem, extracting reactances in synthesis, extracting nonlinear elements in nonlinear circuit theory, etc. However, for the kind of topological theory to be discussed in the succeeding pages we need a formal definition of ports that will carry over to vector spaces from graphs. Otherwise the minimization problems cannot be stated clearly, let alone

be solved. In the example described above, it is clear that if we match $\mathcal{N}_{AP_1}$ and $\mathcal{N}_{BP_2}$ along P_1, P_2 we **do not** get back $\mathcal{N}$. A purely graph theoretic definition of multiport decomposition would therefore not permit the decomposition given in this particular example. Such a definition would lead to optimization problems with additional constraints which have no relevance for network analysis. Further, even after optimization according to such a definition, we would end up with more ports than required.

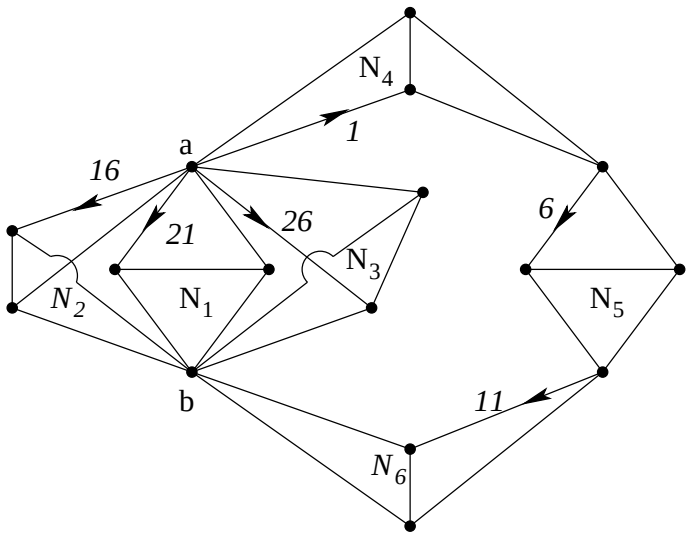


Figure 1.4: Network $\mathcal{N}$ to Illustrate the Fusion-Fission Method

Example 1.0.3 Fusion-Fission method:
Consider the network in Figure 1.4. Six subnetworks have been connected together to make up the network. Assume that the devices in the subnetworks are decoupled. Clearly the networks in Figure 1.4 and Figure 1.5 are equivalent, provided the current through the additional unknown voltage source and the voltage across the additional unknown current source are set equal to zero. But the network in Figure 1.5 is equivalent to that in Figure 1.6 under the additional conditions

$$i_{v1} + i_{v2} + i_{v3} + i = 0$$

$$v_{i1} + v_{i2} + v_{i3} - v = 0$$

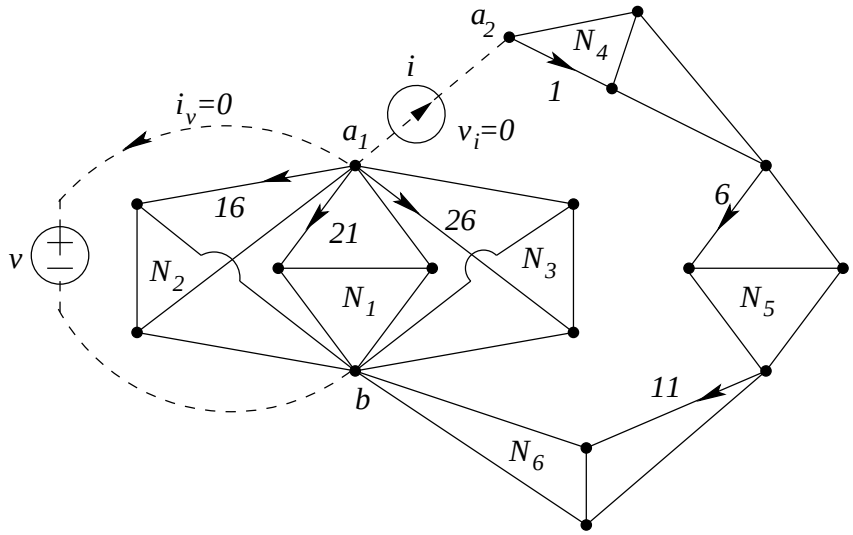


Figure 1.5: A Network equivalent to $\mathcal{N}$ with Virtual Sources

As can be seen, the subnetworks of Figure 1.6 are decoupled except for the common variables v and i and the additional conditions. A natural optimization problem here is the following:

> *Given a partition of the edges of a graph into $E_1, \cdots E_k$, what is the minimum size set of node pair fusions and node fissions by which all circuits passing through more than one E_i are destroyed?*

In the present example the optimal set of operations is to fuse nodes a and b and cut node a into a_1, a_2 as in Figure 1.5. Artificial voltage sources are introduced across the node pairs to be fused and artificial current sources are introduced between two halves of a split node.

It can be shown that this problem generalizes the hybrid rank problem (see Section 14.4). Its solution involves the use of the Dilworth truncation operation on an appropriate submodular function.

We now speak briefly of the mathematical methods needed to derive the kind of results hinted at in the above examples.

The $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ method needs systematic use of the operations of contraction and restriction both for graphs and vector spaces and

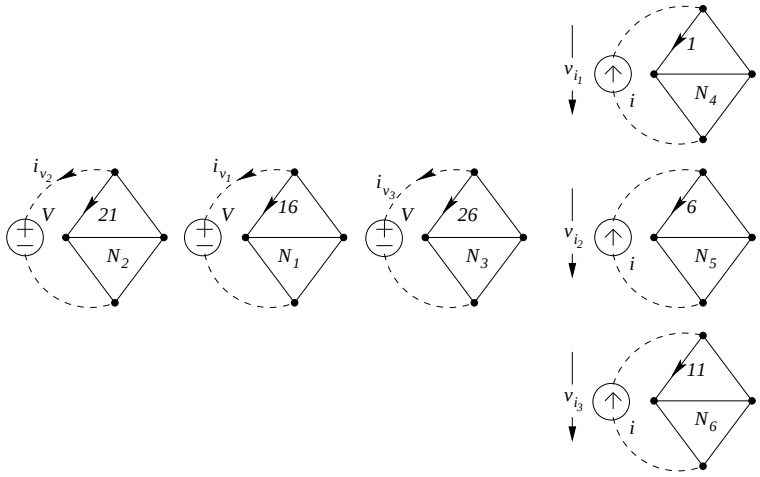


Figure 1.6: Network $\mathcal{N}$ decomposed by the Fusion-Fission Method

the notion of duality of operations on vector spaces. These have been discussed in detail in Chapter 3. The $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ method itself is discussed in Chapter 6.

The multiport decomposition method requires the use of the 'Implicit Duality Theorem'. This result, which should be regarded as a part of network theory folklore, has received too little attention in the literature. We have tried to make amends by devoting a full chapter to it. The optimization problem relevant to multiport decomposition ('port minimization') is discussed in Chapter 8.

The fusion-fission method is a special case of the method of topological transformations discussed in Chapter 7. The solution of the optimization problem that it gives rise to (minimization of the number of fusion and fission operations needed to electrically decouple the blocks of a partition of the edge set) is given in Section 14.4. The solution uses the Dilworth truncation operation on submodular functions.

We next give a chapterwise outline of the book.

Chapter 2 is concerned with mathematical preliminaries such as sets, families, vectors and matrices. Also given is a very brief description of inequality systems.

Chapter 3 contains a very terse description of graphs and their vector space representation. Only material that we need later on in this book is included. Emphasis is placed on complementary orthogonality (Tellegen's Theorem) and the important minor operations linked through duality. The duality described corresponds to complementary orthogonality of vector spaces (and not to the vector space - functional space relation).

Also included is a sketch of the basic algorithms relevant to this book - such as bfs, dfs trees, construction of f-circuits, the shortest path algorithm, algorithms for performing graph minor operations and the basic join and meet operations on partitions. Some space is devoted to the flow maximization problem, particularly certain special ones that are associated with a bipartite graph. (Many of the optimization problems considered in this book reduce ultimately to (perhaps repeated) flow maximization).

Chapter 4 gives a brief account of matroids. Important axiom systems such as the ones in terms of independence, circuit, rank, closure etc. are presented and shown to be equivalent to each other. The minor operations and dual matroids are described. Finally the relation between matroids and the greedy algorithm is presented. This chapter is included for two reasons:

- Some of the notions presented in the previous chapter lead very naturally to their extension to matroids
- matroids are perhaps the most important instance of submodular functions which latter is our main preoccupation in the second half of this book.

Chapter 5 contains a brief introduction to conventional electrical network theory, with the aim of making the book self contained. The intention here is also to indicate the author's point of view to a reader who is an electrical engineer. This chapter contains a rapid sketch of the basic methods of network analysis including a very short description of the procedure followed in general purpose circuit simulators. Also included is an informal account of multiport decomposition and of some elementary results including Thevenin-Norton Theorem.

Chapter 6 contains a description of topological hybrid analysis indicated in Example 1.0.1. This chapter is a formalization of the topological ideas behind Kron's Diakopotics. The methods used involve vector space minors. The main result is Theorem 6.4.1 which has already been illustrated in the above mentioned example.

Chapter 7 contains a detailed description of the Implicit Duality Theorem, its applications and its extensions to linear inequality and linear integrality systems. The operation of generalized minor is introduced and made use of in this chapter. The implicit duality theorem was originally a theorem on ideal transformers and states that if we connect 2-port transformers arbitrarily and expose k-ports, the result would be a k-port ideal transformer. (An ideal transformer, by definition, has its possible port voltage vectors and possible port current vectors as complementary orthogonal spaces.) We show that its power extends beyond these original boundaries. One of the applications described is for the construction of adjoints, another to topological transformations of electrical networks. The latter are used to solve a given network as though it has the topology of a different network, paying a certain cost in terms of additional variables.

Multiport decomposition, from a topological point of view, is the subject of **Chapter 8**. We make free use of the Implicit Duality Theorem of the previous chapter. We indicate that multiport decomposition is perhaps the most natural tool for network analysis by decomposition. It can be shown that multiport decomposition generalizes topological hybrid analysis (see Problem 8.5). We present a few algorithms for minimizing the number of ports for a multiport decomposition corresponding to a given partition of edges of a graph. Finally, we show that this kind of decomposition can be used to construct reduced networks which mimic some of the properties of the original network. In particular we show that any RLMC network can be reduced to a network without zero eigen values (i.e., without trapped voltages or currents) but with, otherwise, the same 'dynamics' as the original network.

The second half of the book is about submodular functions and the link between them and electrical networks.

Chapter 9 contains a compact description of submodular function theory omitting the important operations of convolution and Dilworth truncation. (The latter are developed in subsequent chapters). We begin with the basic definition and some characteristic properties followed by a number of examples of submodular functions which arise in graphs, hypergraphs (represented by bipartite graphs), matrices etc. Basic operations such as contraction, restriction, fusion, dualization etc. are described next. These are immediately illustrated by examples from graphs and bipartite graphs. Some other operations, slightly peripheral, are described next. A section is devoted to the important cases of polymatroid and matroid rank functions. It is shown that any submodular function is a 'translate' through a modular function of a polymatroid rank function. The idea of connectedness is described next. This corresponds to 2-connectedness of graphs. After this there is a very brief but general description of polyhedra associated with set functions in general and with submodular and supermodular functions in particular. The important result due to Frank, usually called the 'Sandwich Theorem' is described in this section. The recent solution, due to Stoer, Wagner and Frank, of the symmetric submodular function minimization problem is described in the next section.

Chapter 10 is devoted to the operation of (lower) convolution of two submodular functions. We begin with purely formal properties and follow it with a number of examples of results from the literature which the operation of convolution unifies. Next we give the polyhedral interpretation for convolution viz. it corresponds to the intersection of the polyhedra of the interacting submodular functions. This is followed by a section in which the operation of convolution is used to show that every polymatroid rank function can be obtained by the fusion of an appropriate matroid rank function.

In the next section, the principal partition (PP) of a submodular function with respect to a strictly increasing polymatroid rank function is dealt with. We begin with the basic properties of PP which give structural insight into many practical problems. An alternative development of PP from the point of view of density of sets is next presented. Finally the PP of naturally derived submodular functions

is related to the PP of the original function.

In the next section, the refined partial order associated with the PP is described. After this we present general algorithms for the construction of the PP of a submodular function with respect to a nonnegative weight function. These use submodular function minimization as a basic subroutine. We consider two important special cases of this algorithm. The first, the weighted left adjacency function of a bipartite graph, is described in this chapter. In this case the submodular function minimization reduces to a flow problem. The second is the PP of a matroid rank function which is taken up in the next chapter. The last (starred) section in this chapter describes a peculiar situation where, performing certain operations on the original submodular function, we get functions whose PP is related in a very simple way to the original PP. This section is developed through problems.

Chapter 11 is on the matroid union operation. In the first section, we give a sketch of submodular functions induced through a bipartite graph and end the section with a proof that 'union' of matroids is a matroid. Next we give Edmond's algorithm for constructing the matroid union. We use this algorithm to study the structure of the union matroid - in particular the natural partition of its underlying set into coloops and the complement, and the manner in which the base of the union is built in terms of the bases of the individual matroids. Finally we use the matroid union algorithm to construct the PP of the rank function of a matroid with respect to the '$|\cdot|$' function.

In **Chapter 12** we study the Dilworth truncation operation on a submodular function. This chapter is written in a manner that emphasizes the structural analogies that exist between convolution and Dilworth truncation. As in the case of convolution, we begin with formal properties and follow it with examples of results from the literature unified by the truncation operation.

In the next section, we describe the principal lattice of partitions (PLP) of a submodular function. This notion is analogous to the PP of a submodular function - whereas in the case of the PP there is a nesting of special sets, in the case of the PLP the special partitions get increasingly finer. We begin with basic properties of the PLP, each

of which can be regarded as a 'translation' of a corresponding property of PP. We then present an alternative development of the PLP in terms of cost of partitioning. In the next section we use this idea for building approximation algorithms for optimum cost partitioning (this problem is of great practical relevance, particularly in CAD for large scale integrated circuits). After this we describe the relation between the PLP of a submodular function and that of derived functions. Here again there is a strong analogy between the behaviours of PP and PLP.

In **Chapter 13**, we present algorithms for building the PLP of a general submodular function. These algorithms are also analogous to those of the PP. The core subroutine is one that builds a '(strong) fusion' set which uses minimization of an appropriately derived submodular function. We specialize these algorithms to the important special cases of the weighted adjacency and exclusivity functions associated with a bipartite graph. (The matroid rank function case is handled in Section 14.3). Next we present some useful techniques for improving the complexity of PLP algorithms for functions arising in practice. Lastly, using the fact that the PP of the rank function of a graph can be regarded, equivalently, as the PLP of the $\mid V(\cdot) \mid$ function on the edge set, we have presented fast algorithms for the former.

The **last chapter** is on the hybrid rank problem for electrical networks. In this chapter, four different (nonequivalent) formulations of this problem are given. The second, third and fourth formulations can be regarded as generalizations of the first. Except in the case of the fourth formulation, we have given fast algorithms for the solution of the problems. This chapter is intended as the link between electrical networks and submodular functions. Each of the formulations has been shown to arise naturally in electrical network theory. The first two formulations require convolution and the third requires Dilworth truncation for its solution. The fourth formulation gives rise to an optimization problem over vector spaces which is left as an open problem.

Chapter 2

Mathematical Preliminaries

2.1 Sets

A **set** (or **collection**) is specified by the **elements** (or **members**) that **belong** to it. If element x belongs to the set (does not belong to the set) X, we write $x \in X$ ($x \notin X$). Two sets are equal iff they have the same members. The set with no elements is called the empty set and is denoted by $\emptyset$. A set is **finite** if it has a finite number of elements. Otherwise it is **infinite**. A set is often specified by actually listing its members, e.g. $\{e_1, e_2, e_3\}$ is the set with members e_1, e_2, e_3. More usually it is specified by a property, e.g. the set of even numbers is specified as $\{x : x$ is an integer and x is even $\}$ or as $\{x, x$ is an integer and x is even $\}$. The symbols $\forall$ and $\exists$ are used to denote 'forall' and 'there exists'. Thus, '$\forall x$' or '$\forall\ x$' should be read as 'forall x' and '$\exists x$' should be read as 'there exists x'. A singleton set is one that has only one element. The singleton set with the element x as its only member, is denoted by $\{x\}$. In this book, very often, we abuse this notation and write x in place of $\{x\}$, if we feel that the context makes the intended object unambiguous.

We say that set X is **contained** in Y (**properly contained in** Y), if every element of X is also a member of Y (every element of X is a member of Y and $X \neq Y$) and denote it by $X \subseteq Y (X \subset Y)$.
The **union** of two sets X and Y denoted by $X \cup Y$, is the set whose members are either in X or in Y (or in both). The **intersection** of

X and Y denoted by $X \cap Y$, is the set whose members belong both to X and to Y. When X and Y do not have common elements, they are said to be **disjoint**. Union of disjoint sets X and Y is often denoted by $X \uplus Y$. Union of sets $X_1, \cdots, X_n$ is denoted by $\bigcup_{i=1}^{n} X_i$ or simply by $\bigcup X_i$. When the X_i are pairwise disjoint, their union is denoted by $\biguplus_{i=1}^{n} X_i$ or $\biguplus X_i$.
The **difference** of X relative to Y, denoted by $X - Y$, is the set of all elements in X but not in Y. Let $X \subseteq S$. Then the **complement** of X **relative to** S is the set $S - X$ and is denoted by $\bar{X}$ when the set S is clear from the context.

A **mapping** $f : X \to Y$, denoted by $f(\cdot)$, associates with each element $x \in X$, the element $f(x)$ in Y. The element $f(x)$ is called the image of x under $f(\cdot)$. We say $f(\cdot)$ maps X into Y. The sets X, Y are called, respectively, the **domain** and **codomain** of $f(\cdot)$. We denote by $f(Z), Z \subseteq X$, the subset of Y which has as members, the images of elements in Z. The set $f(X)$ is called the **range** of $f(\cdot)$. The **restriction** of $f(\cdot)$ to $Z \subseteq X$, denoted by $f/Z(\cdot)$ is the mapping from Z to Y defined by $f/Z(x) \equiv f(x), x \in Z$. A mapping that has distinct images for distinct elements in the domain is said to be **one to one** or **injective**. If the range of $f(\cdot)$ equals its codomain, we say that $f(\cdot)$ is **onto** or **surjective**. If the mapping is one to one onto we say it is **bijective**. Let $f : X \to Y, g : Y \to Z$. Then the **composition** of g and f is the map, denoted by $gf(\cdot)$ or $g \circ f(\cdot)$, defined by $gf(x) \equiv g(f(x)) \ \ \forall x \in X$. The **Cartesian product** $X \times Y$ of sets X, Y is the collection of all ordered pairs (x, y), where $x \in X$ and $y \in Y$. The **direct sum** $X \oplus Y$ denotes the union of disjoint sets X, Y. We use '**direct sum**' loosely to indicate that structures on two disjoint sets are 'put together'. We give some examples where we anticipate definitions which would be given later. The **direct sum** of vector spaces $\mathcal{V}_1, \mathcal{V}_2$ on disjoint sets S_1, S_2 is the vector space $\mathcal{V}_1 \oplus \mathcal{V}_2$ on $S_1 \oplus S_2$ whose typical vectors are obtained by taking a vector $\mathbf{x}_1 \equiv (a_1, \cdots, a_k)$ in $\mathcal{V}_1$ and a vector $\mathbf{x}_2 \equiv (b_1, \cdots, b_m)$ in $\mathcal{V}_2$ and putting them together as $\mathbf{x}_1 \oplus \mathbf{x}_2 \equiv (a_1, \cdots, a_k, b_1, \cdots, b_m)$. When we have two graphs $\mathcal{G}_1, \mathcal{G}_2$ on disjoint edge sets E_1, E_2, $\mathcal{G}_1 \oplus \mathcal{G}_2$ would have edge set $E_1 \oplus E_2$ and is obtained by 'putting together' $\mathcal{G}_1$ and $\mathcal{G}_2$. Usually the vertex sets would also be disjoint. However, where the context permits, we may relax the latter assumption and allow 'hinging' of vertices.

We speak of a **family** of subsets as distinct from a **collection** of subsets. The collection $\{\{e_1, e_2\}, \{e_1, e_2\}, \{e_1\}\}$ is identical to $\{\{e_1, e_2\}, \{e_1\}\}$. But often (e.g. in the definition of a hypergraph in Subsection 3.6.6) we have to use copies of the same subset many times and distinguish between copies. This we do by 'indexing' them. A family of subsets of S may be defined to be a mapping from an index set I to the collection of all subsets of S. For the purpose of this book, the index set I can be taken to be $\{1, \cdots, n\}$. So the family $(\{e_1, e_2\}, \{e_1, e_2\}, \{e_1\})$ can be thought of as the mapping $\phi(\cdot)$ with

$$\begin{aligned} \phi(1) &\equiv \{e_1, e_2\} \\ \phi(2) &\equiv \{e_1, e_2\} \\ \phi(3) &\equiv \{e_1\}. \end{aligned}$$

(Note that a family is denoted using ordinary brackets while a set is denoted using curly brackets).

2.2 Vectors and Matrices

In this section we define vectors, matrices and related notions. Most present day books on linear algebra treat vectors as primitive elements in a vector space and leave them undefined. We adopt a more old fashioned approach which is convenient for the applications we have in mind. The reader who wants a more leisurely treatment of the topics in this section is referred to [Hoffman+Kunze72].

Let S be a finite set $\{e_1, e_2, \ldots, e_n\}$ and let $\mathcal{F}$ be a field. We will confine ourselves to the field $\Re$ of real numbers, the field $\mathcal{C}$ of complex numbers and the $GF2$ field on elements $0, 1$ ($0{+}0 = 0, 0{+}1 = 1, 1{+}0 = 1, 1+1 = 0, 1.1 = 1, 1.0 = 0, 0.1 = 0, 0.0 = 0$). For a general definition of a field see for instance [Jacobson74]. By a vector on S over $\mathcal{F}$ we mean a mapping $\mathbf{f}$ of S into $\mathcal{F}$.The field $\mathcal{F}$ is called the **scalar field** and its elements are called **scalars**. The **support** of $\mathbf{f}$ is the subset of S over which it takes nonzero values. The **sum** of two vectors $\mathbf{f}, \mathbf{g}$ on S over $\mathcal{F}$ is defined by $(\mathbf{f} + \mathbf{g})(e_i) \equiv \mathbf{f}(e_i) + \mathbf{g}(e_i)\ \forall e_i \in S$. (For convenience the **sum** of two vectors $\mathbf{f}$ on S,$\mathbf{g}$ on T over $\mathcal{F}$ is defined by $(\mathbf{f}+\mathbf{g})(e_i) \equiv \mathbf{f}(e_i)+\mathbf{g}(e_i)\ \forall e_i \in S \cap T$, as agreeing with $\mathbf{f}$ on $S-T$, and as agreeing with $\mathbf{g}$ on $T-S$). The **scalar product** of $\mathbf{f}$ by a scalar λ

is a vector $\lambda\mathbf{f}$ defined by $(\lambda\mathbf{f})(e_i) \equiv \lambda(\mathbf{f}(e_i))\ \forall e_i \in S$. A collection $\mathcal{V}$ of vectors on S over $\mathcal{F}$ is a vector space iff it is closed under addition and scalar product. We henceforth omit mention of underlying set and field unless required.

A set of vectors $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ is **linearly dependent** iff there exist scalars $\lambda_1, \ldots, \lambda_n$ not all zero such that $\lambda_1\mathbf{f}_1 + \ldots + \lambda_n\mathbf{f}_n = \mathbf{0}$. (Here the $\mathbf{0}$ vector is one which takes value 0 on all elements of S). Vector $\mathbf{f}_n$ is a linear combination of $\mathbf{f}_1, \ldots, \mathbf{f}_{n-1}$ iff $\mathbf{f}_n = \lambda_1\mathbf{f}_1 + \ldots + \lambda_{n-1}\mathbf{f}_{n-1}$ for some $\lambda_1, \ldots, \lambda_{n-1}$.

The set of all vectors linearly dependent on a collection $\mathcal{C}$ of vectors can be shown to form a vector space which is said to be **generated** by or **spanned** by $\mathcal{C}$. Clearly if $\mathcal{V}$ is a vector space and $\mathcal{C} \subseteq \mathcal{V}$, the subset of vectors generated by $\mathcal{C}$ is contained in $\mathcal{V}$. A maximal linearly independent set of vectors of $\mathcal{V}$ is called a **basis** of $\mathcal{V}$.
In general maximal and minimal members of a collection of sets may not be largest and smallest in terms of size.
Example: Consider the collection of sets $\{\{1,2,3\},\{4\},\{5,6\},\{1,2,3,5,6\}\}$. The minimal members of this collection are $\{1,2,3\},\{4\},\{5,6\}$, i.e., these do not contain proper subsets which are members of this collection. The maximal members of this collection are $\{4\},\{1,2,3,5,6\}$, i.e., these are not proper subsets of other sets which are members of this collection.

The following theorem is therefore remarkable.

Theorem 2.2.1 *All bases of a vector space on a finite set have the same cardinality.*

The number of elements in a basis of $\mathcal{V}$ is called the **dimension** of $\mathcal{V}$, denoted by $dim(\mathcal{V})$, or the **rank** of $\mathcal{V}$, denoted by $r(\mathcal{V})$. Using Theorem 2.2.1 one can show that the size of a maximal independent subset contained in a given set $\mathcal{C}$ of vectors is unique. This number is called the **rank** of $\mathcal{C}$. Equivalently, the rank of $\mathcal{C}$ is the dimension of the vector space spanned by $\mathcal{C}$. If $\mathcal{V}_1, \mathcal{V}_2$ are vector spaces and $\mathcal{V}_1 \subseteq \mathcal{V}_2$, we say $\mathcal{V}_1$ is a **subspace** of $\mathcal{V}_2$.

A mapping $\mathbf{A} : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \longrightarrow \mathcal{F}$ is called a $m \times n$ matrix. It may be thought of as an $m \times n$ array with entries from $\mathcal{F}$. We denote $\mathbf{A}(i, j)$ often by the lower case a_{ij} with i as the row index and j as the column index. We speak of the array $(a_{i1}, \ldots, a_{in})$ as the

ith row of $\mathbf{A}$ and of the array $(a_{1j}, \ldots, a_{nj})$ as the **jth column** of $\mathbf{A}$. Thus we may think of $\mathbf{A}$ as made up of m row vectors or of n column vectors. Linear dependence, independence and linear combination for row and column vectors are defined the same way as for vectors. We say two matrices are **row equivalent** if the rows of each can be obtained by linearly combining the rows of the other. **Column equivalence** is defined similarly. The vector space spanned by the rows (columns) of $\mathbf{A}$ is called its **row space (column space)** and denoted by $\mathcal{R}(\mathbf{A})(\mathcal{C}(\mathbf{A}))$. The dimension of $\mathcal{R}(\mathbf{A})(\mathcal{C}(\mathbf{A}))$ is called the **row rank (column rank)** of $\mathbf{A}$.

If $\mathbf{A}$ is an $m \times n$ matrix then the **transpose** of $\mathbf{A}$ denoted by $\mathbf{A}^T$ is an $n \times m$ matrix defined by $\mathbf{A}^T(i,j) \equiv \mathbf{A}(j,i)$. Clearly the i^{th} row of $\mathbf{A}$ becomes the i^{th} column of $\mathbf{A}^T$ and vice versa. If $\mathbf{B}$ is also an $m \times n$ matrix the **sum** $\mathbf{A} + \mathbf{B}$ is an $m \times n$ matrix defined by $(\mathbf{A} + \mathbf{B})(i,j) \equiv \mathbf{A}(i,j) + \mathbf{B}(i,j)$. If $\mathbf{D}$ is an $n \times p$ matrix, the product $\mathbf{AD}$ is an $m \times p$ matrix defined by $\mathbf{AD}(i,j) \equiv \sum_{k=1}^{n} a_{ik} d_{kj}$. Clearly if $\mathbf{AD}$ is defined it does not follow that $\mathbf{DA}$ is defined. Even when it is defined, in general $\mathbf{AD} \neq \mathbf{DA}$. The most basic property of this notion of product is that it is **associative** i.e. $\mathbf{A}(\mathbf{DF}) = (\mathbf{AD})\mathbf{F}$.

Matrix operations are often specified by **partitioning**. Here we write a matrix in terms of **submatrices** (i.e., matrices obtained by deleting some rows and columns of the original matrix). A matrix may be partitioned along rows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} \\ \ldots \\ \mathbf{A}_{21} \end{bmatrix}$$

or along columns:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} | \mathbf{A}_{12} \end{bmatrix}$$

or both:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \ldots & \mathbf{A}_{1k} \\ \vdots & \vdots & \vdots \\ \mathbf{A}_{p1} & \ldots & \mathbf{A}_{pk} \end{bmatrix}.$$

When two partitioned matrices are multiplied we assume that the partitioning is **compatible**, i.e., for each triple (i,j,k) the number of columns of $\mathbf{A}_{ik}$ equals the number of rows of $\mathbf{B}_{kj}$. Clearly this is achieved if the original matrices $\mathbf{A}, \mathbf{B}$ are compatible for product and

each block of the column partition of $\mathbf{A}$ has the same size as the corresponding row partition of $\mathbf{B}$. The following **partitioning rules** can then be verified.

i. $$\begin{bmatrix} \mathbf{A}_{11} \\ \dots \\ \mathbf{A}_{21} \end{bmatrix} \mathbf{C} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{C} \\ \dots \\ \mathbf{A}_{21}\mathbf{C} \end{bmatrix}$$

ii. $$\mathbf{C} \left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \end{array} \right] = \left[\begin{array}{c|c} \mathbf{C}\mathbf{A}_{11} & \mathbf{C}\mathbf{A}_{12} \end{array} \right]$$

iii. $$\left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \end{array} \right] \begin{bmatrix} \mathbf{C}_{11} \\ \dots \\ \mathbf{C}_{12} \end{bmatrix} = \left[\begin{array}{c} \mathbf{A}_{11}\mathbf{C}_{11} + \mathbf{A}_{12}\mathbf{C}_{12} \end{array} \right].$$

In general if $\mathbf{A}$ is partitioned into submatrices $\mathbf{A}_{ik}, \mathbf{B}$ into submatrices $\mathbf{B}_{kj}$ then the product $\mathbf{C} = \mathbf{AB}$ would be naturally partitioned into $\mathbf{C}_{ij} \equiv \sum_k \mathbf{A}_{ik}\mathbf{B}_{kj}$.

Matrices arise most naturally in linear equations such as $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ and $\mathbf{b}$ are known and $\mathbf{x}$ is an unknown vector. When $\mathbf{b} = \mathbf{0}$ it is easily verified that the set of all solutions of $\mathbf{Ax} = \mathbf{b}$, i.e.,of $\mathbf{Ax} = \mathbf{0}$, forms a vector space. This space will be called the **solution space** of $\mathbf{Ax} = \mathbf{0}$, or the **null space** of $\mathbf{A}$. The **nullity** of $\mathbf{A}$ is the dimension of the null space of $\mathbf{A}$. We have the following theorem.

Theorem 2.2.2 *If two matrices are row equivalent then their null spaces are identical.*

Corollary 2.2.1 *If* $\mathbf{A}, \mathbf{B}$ *are row equivalent matrices then a set of columns of* $\mathbf{A}$ *is independent iff the corresponding set of columns of* $\mathbf{B}$ *is independent.*

The following are **elementary row operations** that can be performed on the rows of a matrix:

i. interchanging rows,

ii. adding a multiple of one row to another,

iii. multiplying a row by a nonzero number.

Each of these operations corresponds to premultiplication by a matrix. Such matrices are called **elementary matrices**. It can be seen that these are the matrices we obtain by performing the corresponding elementary row operations on the unit matrix of the same number of rows as the given matrix. We can define elementary column operations similarly. These would correspond to post multiplication by elementary column matrices.

A matrix is said to be in **Row Reduced Echelon** form (RRE) iff it satisfies the following:
Let r be the largest row index for which $a_{ij} \neq 0$ for some j. Then the columns of the $r \times r$ unit matrix (the matrix with $1s$ along the diagonal and zero elsewhere) $\mathbf{e_1}, \ldots, \mathbf{e_r}$ appear as columns, say $\mathbf{C}_{i_1}, \ldots, \mathbf{C}_{i_r}$ of $\mathbf{A}$ with $i_1 < \ldots < i_r$. Further if $p < i_k$ then $a_{kp} = 0$. We have the following theorem.

Theorem 2.2.3 *Every matrix can be reduced to a matrix in the RRE form by a sequence of elementary row transformations and is therefore row equivalent to such a matrix.*

It is easily verified that for an RRE matrix row rank equals column rank. Hence using Theorem 2.2.3 and Corollary 2.2.1 we have

Theorem 2.2.4 *For any matrix, row rank equals column rank.*

The **rank** of a matrix $\mathbf{A}$, denoted by $r(\mathbf{A})$, is its row rank (= column rank).
Let the elements of S be ordered as $(e_1, \ldots, e_n)$. Then for any vector $\mathbf{f}$ on S we define $\mathbf{R}_f$, the **representative vector** of $\mathbf{f}$, as the one rowed matrix $(\mathbf{f}(e_1), \ldots, \mathbf{f}(e_n))$. We will not usually distinguish between a vector and its representative vector. When the rows of a matrix $\mathbf{R}$ are representative vectors of some basis of a vector space $\mathcal{V}$ we say that $\mathbf{R}$ is a **representative matrix** of $\mathcal{V}$. When $\mathbf{R}, \mathbf{R}_1$ both represent $\mathcal{V}$ they can be obtained from each other by row operations. Hence by Corollary 2.2.1 their column independence structure is identical. An $r \times n$ representative matrix $\mathbf{R}$, $r \leq n$, is a **standard representative matrix** iff $\mathbf{R}$ has an $r \times r$ submatrix which can be obtained by permutations of the columns of the $r \times r$ unit matrix. For convenience we will write a standard representative matrix in the form $[\mathbf{I}|\mathbf{R}_{12}]$ or $[\mathbf{R}_{11}|\mathbf{I}]$. (Here $\mathbf{I}$ denotes the unit matrix of appropriate size).

The **dot product** of two vectors $\mathbf{f}, \mathbf{g}$ on S denoted by $< \mathbf{f}, \mathbf{g} >$ over

$\mathcal{F}$ is defined by $< \mathbf{f}, \mathbf{g} > \equiv \sum_{e \in S} \mathbf{f}(e).\mathbf{g}(e)$. We say $\mathbf{f}, \mathbf{g}$ are **orthogonal** if their dot product is zero. If $\mathcal{C}$ is a collection of vectors on S then $\mathcal{C}^{\perp} \equiv$ set of all vectors orthogonal to every vector in $\mathcal{C}$. It can be verified that $\mathcal{C}^{\perp}$ is a vector space. Let $\mathcal{V}$ be a vector space on S with basis $\mathcal{B}$. Since vectors orthogonal to each vector in $\mathcal{B}$ are also orthogonal to linear combinations of these vectors we have $\mathcal{B}^{\perp} = \mathcal{V}^{\perp}$. If $\mathbf{R}$ is a representative matrix of $\mathcal{V}$, it is clear that $\mathcal{V}^{\perp}$ is its null space. Equivalently $\mathcal{V}^{\perp}$ is the solution space of $\mathbf{Rx} = \mathbf{0}$. If $\mathbf{R}$ is a standard representative matrix with $\mathbf{R} = [\mathbf{I}_{r \times r} | \mathbf{R}_{12}]$, then the solution space of $\mathbf{Rx} = \mathbf{0}$ can be shown to be the vector space generated by the columns of $\begin{bmatrix} -\mathbf{R}_{12} \\ \dots \\ \mathbf{I}_{n-r \times n-r} \end{bmatrix}$, where $n = |S|$.(Here $\mathbf{I}_{k \times k}$ denotes the unit matrix with k rows). Equivalently $\mathcal{V}^{\perp}$ has the representative matrix $[-\mathbf{R}_{12}^T | \mathbf{I}_{n-r \times n-r}]$. The representative matrix of $(\mathcal{V}^{\perp})^{\perp}$ will then be $\mathbf{R}$. We then have the following

Theorem 2.2.5 *i. if $[\mathbf{I}_{r \times r} | \mathbf{R}_{12}]$ is a representative matrix of vector space $\mathcal{V}$ on S then $[-\mathbf{R}_{12}^T | \mathbf{I}_{n-r \times n-r}]$ is a representative matrix of $\mathcal{V}^{\perp}$.*

ii. $r(\mathcal{V}^{\perp}) = |S| - r(\mathcal{V})$

iii. $(\mathcal{V}^{\perp})^{\perp} = \mathcal{V}$. *Hence two matrices are row equivalent iff their null spaces are identical.*

Consider the collection of all $n \times n$ matrices over $\mathcal{F}$. We say that $\mathbf{I}$ is an identity for this collection iff for every $n \times n$ matrix $\mathbf{B}$ we have $\mathbf{IB} = \mathbf{BI} = \mathbf{B}$. If $\mathbf{I}_1, \mathbf{I}_2$ are identity matrices we must have $\mathbf{I}_1 = \mathbf{I}_2 = \mathbf{I}$. The unit matrix (with $1s$ along the diagonal and $0s$ elsewhere) is clearly an identity matrix. It is therefore the only identity matrix. Two $n \times n$ matrices $\mathbf{A}, \mathbf{B}$ are **inverses** of each other iff $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$. We say $\mathbf{A}, \mathbf{B}$ are **invertible** or **nonsingular**. If $\mathbf{A}$ has inverses $\mathbf{B}, \mathbf{C}$ we must have $\mathbf{C} = \mathbf{C}(\mathbf{AB}) = (\mathbf{CA})\mathbf{B} = \mathbf{IB} = \mathbf{B}$. Thus the inverse of a matrix $\mathbf{A}$, if it exists, is unique and is denoted by $\mathbf{A}^{-1}$. We then have the following

Theorem 2.2.6 *i.* $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$

ii. If $\mathbf{A}, \mathbf{D}$ are $n \times n$ invertible matrices, then $(\mathbf{AD})^{-1} = (\mathbf{D}^{-1}\mathbf{A}^{-1})$.

With a square matrix we associate an important number called its determinant. Its definition requires some preparation.

A bijection of a finite set to itself is also called a **permutation**. A permutation that interchanges two elements (i.e. maps each of them to the other) but leaves all others unchanged is a **transposition**. Every permutation can be obtained by repeated application of transpositions. We then have the following

Theorem 2.2.7 *If a permutation σ can be obtained by composition of an even number of transpositions then every decomposition of σ into transpositions will contain an even number of them.*

By Theorem 2.2.7 we can define a permutation to be **even (odd)** iff it can be decomposed into an even (odd) number of transpositions. The **sign** of a permutation σ denoted by $sgn(\sigma)$ is $+1$ if σ is even and -1 if σ is odd. It is easily seen, since the identity $(= \sigma\sigma^{-1})$ permutation is even, that $sgn(\sigma) = sgn(\sigma^{-1})$. The **determinant** of an $n \times n$ matrix is defined by

$$det(\mathbf{A}) \equiv \sum_{\sigma} sgn(\sigma) a_{1\sigma(1)} \dots a_{n\sigma(n)},$$

where the summation is taken over all possible permutations of $\{1, 2, \dots, n\}$. It is easily seen that determinant of the unit matrix is $+1$. We collect some of the important properties of the determinant in the following

Theorem 2.2.8 *i.* $det(\mathbf{A}) = det(\mathbf{A}^T)$

ii. Let

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{A}_2 \end{bmatrix}, \mathbf{A}' = \begin{bmatrix} \mathbf{a}'_1 \\ \mathbf{A}_2 \end{bmatrix}, \mathbf{A}" = \begin{bmatrix} \mathbf{a}_1 + \mathbf{a}'_1 \\ \mathbf{A}_2 \end{bmatrix}.$$

Then $det(\mathbf{A}") = det(\mathbf{A}) + det(\mathbf{A}')$.

iii. If $\mathbf{A}$ *has two identical rows, or has two identical columns then* $det(\mathbf{A}) = 0$.

iv. If $\mathbf{E}$ *is an elementary matrix* $det(\mathbf{EA}) = det(\mathbf{E})det(\mathbf{A})$. *Since every invertible matrix can be factored into elementary matrices, it follows that* $det(\mathbf{AB}) = det(\mathbf{A})det(\mathbf{B})$, *for every pair of* $n \times n$ *matrices* $\mathbf{A}, \mathbf{B}$.

v. $det(\mathbf{A}) \neq 0$ *iff* $\mathbf{A}$ *is invertible.*

Problem 2.1 Size of a basis: *Prove*

i. *Theorem 2.2.1*

ii. *If $\mathcal{V}_1$ is a subspace of vector space $\mathcal{V}_2$, $\dim \mathcal{V}_1 \leq \dim \mathcal{V}_2$.*

iii. *If $\mathcal{V}_1 \subseteq \mathcal{V}_2$ and $\dim \mathcal{V}_1 = \dim \mathcal{V}_2$ then $\mathcal{V}_1 = \mathcal{V}_2$.*

iv. *an $m \times n$ matrix with $m > n$ cannot have linearly independent rows.*

v. *any vector in a vector space $\mathcal{V}$ can be written uniquely as a linear combination of the vectors in a basis of $\mathcal{V}$.*

Problem 2.2 Ways of interpreting the matrix product: *Define product of matrices in the usual way i.e.* $\mathbf{C} = \mathbf{AB}$ *is equivalent to* $\mathbf{C}_{ij} = \sum_k a_{ik} b_{kj}$. *Now show that it can be thought of as follows*

i. *Columns of* $\mathbf{C}$ *are linear combinations of columns of* $\mathbf{A}$ *using entries of columns of* $\mathbf{B}$ *as coefficients.*

ii. *rows of* $\mathbf{C}$ *are linear combinations of rows of* $\mathbf{B}$ *using entries of rows of* $\mathbf{A}$ *as coefficients.*

Problem 2.3 Properties of matrix product: *Prove, when* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ *are matrices and the products are defined*

i. $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$

ii. $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

Problem 2.4 Partitioning rules: *Prove*

i. *the partitioning rules.*

ii.

$$\begin{bmatrix} \mathbf{A}_{11} \cdots & \mathbf{A}_{1k} \\ \vdots & \vdots \\ \mathbf{A}_{r1} \cdots & \mathbf{A}_{rk} \end{bmatrix}^T = \begin{bmatrix} \mathbf{A}_{11}^T \cdots & \mathbf{A}_{r1}^T \\ \vdots & \vdots \\ \mathbf{A}_{1k}^T \cdots & \mathbf{A}_{rk}^T \end{bmatrix}$$

Problem 2.5 Solution space and column dependence structure: *Prove theorem 2.2.2 and Corollary 2.2.1.*

Problem 2.6 Algorithm for computing RRE: *Give an algorithm for converting any rectangular matrix into the RRE form. Give an upper bound for the number of arithmetical steps in your algorithm.*

Problem 2.7 Uniqueness of the RRE matrix: *Show that no RRE matrix is row equivalent to a distinct RRE matrix. Hence prove that every matrix is row equivalent to a unique matrix in the RRE form.*

Problem 2.8 RRE of special matrices:

i. *If* $\mathbf{A}$ *is a matrix with linearly independent columns what is its RRE form? If in addition* $\mathbf{A}$ *is square what is its RRE form?*

ii. *If* $\mathbf{A}, \mathbf{B}$ *are square such that* $\mathbf{AB} = \mathbf{I}$ *show that* $\mathbf{BA} = \mathbf{I}$.

iii. *Prove Theorem 2.2.6*

Problem 2.9 Existence and nature of solution for linear equations: *Consider the equation* $\mathbf{Ax} = \mathbf{b}$.

i. *Show that it has a solution*

  (a) *iff* $r(\mathbf{A}) = r(\mathbf{A}|\mathbf{b})$.

  (b) *iff whenever* $\lambda^T\mathbf{A} = \mathbf{0}, \lambda^T\mathbf{b}$ *is also zero.*

ii. *Show that a vector is a solution of the above equation iff it can be written in the form* $\mathbf{x}_o + \mathbf{x}_p$ *where* $\mathbf{x}_p$ *is a particular solution of the equation while* $\mathbf{x}_o$ *is a vector in the null space of* $\mathbf{A}$ *(i.e. a solution to the linear equation with* $\mathbf{b}$ *set equal to zero).*

iii. **Motivation for the matrix product:** *Why is the matrix product defined as in Problem 2.2? (In the above equation suppose we make the substitution* $\mathbf{x} = \mathbf{By}$. *What would the linear equation in terms of* $\mathbf{y}$ *be?)*

iv. **Linear dependence and logical consequence:** *The above equation may be regarded as a set of linear equations (one for each row of* $\mathbf{A}$*) each of which in turn could be thought of as a statement. Show that a linear equation is a logical consequence of others iff it is* **linearly** *dependent on the others.*

Problem 2.10 Positive definite matrices:

i. *Construct an example where* $\mathbf{A}$, $\mathbf{B}$ *are invertible but their sum is not.*

ii. *A matrix* $\mathbf{K}$ *is* **positive semidefinite (positive definite)** *iff* $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0 \ \ \forall \mathbf{x} \neq 0$ *(*$\mathbf{x}^T\mathbf{K}\mathbf{x} > 0 \ \ \forall \mathbf{x} \neq 0$*). Show that*

   (a) *a matrix is invertible if it is positive definite;*

   (b) *sum of two positive semidefinite matrices (positive definite matrices) is positive semidefinite (positive definite);*

   (c) *if* $\mathbf{K}$ *is a positive definite matrix,then* $\mathbf{A}\mathbf{K}\mathbf{A}^T$ *is positive semidefinite and if, further, rows of* $\mathbf{A}$ *are linearly independent, then* $\mathbf{A}\mathbf{K}\mathbf{A}^T$ *is positive definite;*

   (d) *inverse of a symmetric positive definite matrix is also symmetric positive definite.*

Problem 2.11 Projection of a vector on a vector space: *Let* $\mathbf{x}$ *be a vector on* S *and let* $\mathcal{V}$ *be a vector space on* S*. Show that* $\mathbf{x}$ *can be uniquely decomposed as* $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$*, where* $\mathbf{x}_1 \in \mathcal{V}$ *and* $\mathbf{x}_2 \in \mathcal{V}^\perp$*.The vector* $\mathbf{x}_1$ *is called the* **projection** *of* $\mathbf{x}$ *on* $\mathcal{V}$ **along** $\mathcal{V}^\perp$*.*

Problem 2.12 Parity of a Permutation: *Show that if a permutation can be obtained by composing an odd number of transpositions it cannot also be obtained by composing an even number of transpositions.*

Problem 2.13 Graph of a permutation: *Define the graph* $\mathcal{G}_\sigma$ *of a permutation* σ *on* $\{1, 2, \cdots n\}$ *as follows:* $V(\mathcal{G}_\sigma) \equiv \{1, 2, \cdots, n\}$*; draw an edge with an arrow from* i *to* j *iff* $\sigma(i) = j$*.*

i. *Show that every vertex in this graph has precisely one arrow coming in and one going out. Hence, conclude that each connected component is a directed circuit.*

ii. *Show that if* $\mathcal{G}_\sigma$ *has an odd (even) number of even length circuits then* σ *is odd (even).*

Problem 2.14 Properties of the determinant: *Prove Theorem 2.2.8.*

Problem 2.15 Equivalence of definitions of a determinant: *Show that the usual definition of a determinant by expanding along a row or column is equivalent to the definition using permutations.*

Problem 2.16 Laplace expansion of the determinant: *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix. Show that*

$$det(\mathbf{A}) = \sum sgn(\sigma)\ det\left(\mathbf{A}\begin{pmatrix} r_1, & \cdots & , r_k \\ i_1, & \cdots & , i_k \end{pmatrix}\right) det\left(\mathbf{A}\begin{pmatrix} r_{k+1}, & \cdots & , r_m \\ i_{k+1}, & \cdots & , i_m \end{pmatrix}\right),$$

($\mathbf{A}\begin{pmatrix} d_1, & \cdots & , d_p \\ i_1, & \cdots & , i_p \end{pmatrix}$ *is the* $p \times p$ *matrix whose* (s,t) *entry is the* (d_s, i_t) *entry of* $\mathbf{A}$*), where the summation is over all subsets* $\{r_1, \cdots, r_k\}$ *of* $\{1, \cdots, n\}$
and $\sigma \equiv \begin{pmatrix} r_1, \cdots, r_k, r_{k+1} \cdots r_n \\ i_1, \cdots, i_k, i_{k+1} \cdots i_n \end{pmatrix}$ *i.e.,* $\sigma(r_j) \equiv i_j, j = 1, \cdots, n.$

Problem 2.17 Binet Cauchy Theorem: *Let* $\mathbf{A}$ *be an* $m \times n$ *and* $\mathbf{B}$ *an* $n \times m$ *matrix with* $m \leq n$*. If an* $m \times m$ *submatrix of* $\mathbf{A}$ *is composed of columns* $i_1, \cdots, i_m$*, the* **corresponding** $m \times m$ *submatrix of* $\mathbf{B}$ *is the one with rows* $i_1, \cdots, i_m$*. Prove the Binet Cauchy Theorem:* $det(\mathbf{AB}) =$ $\sum$ *product of determinants of corresponding* $m \times m$ *submatrices of* $\mathbf{A}$ *and* $\mathbf{B}$*.*

2.3 Linear Inequality Systems

2.3.1 The Kuhn-Fourier Theorem

In this section we summarize basic results on inequality systems which we need later on in the book. Proofs are mostly omitted. They may be found in standard references such as [Stoer+Witzgall70] and [Schrijver86]. This section follows the former reference.

A **linear inequality system** is a set of constraints of the following kind on the vector $\mathbf{x} \in \Re^n$.

$$\left.\begin{array}{rcl} \mathbf{Ax} & = & \mathbf{a}_o \\ \mathbf{Bx} & > & \mathbf{b}_o \\ \mathbf{Cx} & \geq & \mathbf{c}_o \end{array}\right\} \qquad (I)$$

Here, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are matrices, $\mathbf{a}_o, \mathbf{b}_o, \mathbf{c}_o$ are column vectors with appropriate number of rows. We say $\mathbf{x}_1 > \mathbf{x}_2 (\mathbf{x}_1 \geq \mathbf{x}_2)$ iff each component of $\mathbf{x}_1$ is greater than (greater than or equal to) the corresponding component of $\mathbf{x}_2$.

A **solution** of an inequality system is a vector which satisfies all the inequality constraints of the system. A constraint which is satisfied by every solution of an inequality system is said to be a **consequence** of the system. In particular, we are concerned with constraints of the kind $\mathbf{d}^T\mathbf{x} = \mathbf{d}_o$ or $> \mathbf{d}_o$ or $\geq \mathbf{d}_o$. A **legal linear combination** of the system (I) is obtained by linearly combining the equations and inequalities with real coefficients - α_i for the linear equations, and non-negative real coefficients β_j, γ_k for the '$>$' linear inequalities and '$\geq$' linear inequalities respectively. The resulting constraint would be a linear equation iff β_j, γ_k are all zero. It would be a '$>$' inequality iff at least one of the β_j's is nonzero. It would be a '$\geq$' inequality iff all of β_j are zero but at least one of the γ_k is nonzero. A legal linear combination is thus a consequence of the system. A legal linear combination, with at least one of the $\alpha_i, \beta_j, \gamma_k$ nonzero, that results in the LHS becoming zero is called a **legal linear dependence** of the system. Another important way of deriving consequence relations is by **weakening**. This means to weaken '$=$' to '$\geq$' and '$>$' to '$\geq$' and also in the case of '$>$' and '$\geq$' to lower the right side value.

Example 2.3.1 *Consider the system of linear inequalities:*

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ 2x_1 + x_2 &= 4 \\ x_1 + x_2 &> 1 \\ 2x_1 + 3x_2 &> 2 \\ x_1 + 5x_2 &\geq 2 \\ -x_1 - 2x_2 &\geq 4. \end{aligned}$$

The legal linear combination corresponding to $\alpha_1 = 1, \alpha_2 = 1, \beta_1 = 0, \beta_2 = 0, \gamma_1 = 0, \gamma_2 = 0$ is
$3x_1 + 3x_2 = 7$;
that corresponding to $\alpha_1 = 1, \alpha_2 = 0, \beta_1 = 1, \beta_2 = 0, \gamma_1 = 1, \gamma_2 = 0$ is
$3x_1 + 8x_2 > 6$;
that corresponding to $\alpha_1 = 1, \alpha_2 = 0, \beta_1 = 0, \beta_2 = 0, \gamma_1 = 1, \gamma_2 = 0$ is
$2x_1 + 7x_2 \geq 5$.
The legal linear combination corresponding to $\alpha_1 = 1, \alpha_2 = 0, \beta_1 = 0, \beta_2 = 0, \gamma_1 = 0, \gamma_2 = 1$ is the zero relation
$0x_1 + 0x_2 \geq 7$.

Thus in this case, the system has a **legal linear dependence** that is a **contradiction**.

We can now state the fundamental theorem of Kuhn and Fourier [Fourier1826], [Kuhn56].

Theorem 2.3.1 (Kuhn-Fourier Theorem) *A linear inequality system has a solution iff no legal linear dependence is a contradiction.*

Sketch of the Proof of Theorem 2.3.1: First reduce the linear equations to the RRE form. If a row arises with zero coefficients but with nonzero right side at this stage, we have a legal linear dependence that is a contradiction. Otherwise express some of the variables in terms of the others. This substitution is now carried out also in the inequalities. So henceforth, without loss of generality, we may assume that we have only inequalities. If we prove the theorem for such a reduced system, it can be extended to one which has equalities also.

Suppose each variable has either zero coefficient or the same sign in all the inequalities of the system and further, if there are inequalities with zero coefficients they are not contradictory.

In this case it is easy to see that the system has a solution whether the coefficients of a particular variable are all zero or otherwise. If all the coefficients are zero we are done - the theorem is clearly true. If not, it is not possible to get a legal linear dependence without using zero coefficients. So the theorem is again true in this case.

We now present an elimination procedure which terminates at the above mentioned situation.
Let the inequalities be numbered $(1), \cdots, (r), (r+1), \cdots, (k)$. Let x_n be present with coefficient $+1$ in the inequalities $(1), \cdots, (r)$ and with coefficient -1 in the inequalities $(r+1), \cdots, (k)$. We create $r(k-r)$ inequalities without the variable x_n by adding each of the first r inequalities to each of the last $(k-r)$ inequalities. Note that if both the inequalities are of the $(\geq)$ kind, the addition would result in another of the $(\geq)$ kind and if one of them is of the $(>)$ kind, the addition would result in another of the $(>)$ kind.

If the original system has a solution, it is clear that the reduced system also has one. On the other hand, if the reduced system has a solution $(x'_1, \cdots, x'_{n-1})$ it is possible to find a value x'_n of x_n such that $(x'_1, \cdots, x'_{n-1}, x'_n)$ is a solution of the original system. We indicate how,

below.
Let the inequalities added be

$$a_{i1}x_1 + \cdots + x_n \geq b_i$$

$$a_{j1}x_1 + \cdots - x_n > b_j$$

(The cases where both are ($\geq$), both are ($>$) or first inequality ($>$) and second ($\geq$) are similar.) The pair of inequalities can be written equivalently as

$$a_{j1}x_1 + \cdots + a_{j(n-1)}x_{n-1} - b_j > x_n \geq b_i - a_{i1}x_1 - \cdots - a_{i(n-1)}x_{n-1} \qquad (*)$$

The extreme left of the above inequality ($*$) is always derived from the inequalities $(r+1)$ to (k) while the extreme right is always derived from the (1) to (r) inequalities. When $x'_1, \cdots, x'_{n-1}$ is substituted in the above inequality, it would be satisfied for every pair of inequalities, from $(j+1)$ to (k) on the extreme left and (1) to (j) on the extreme right. After substitution, let the least of the extreme left term be reached for inequality (p) and let the highest of the extreme right term be reached for inequality (q). Since

$$a_{p1}x'_1 + \cdots + a_{p(n-1)}x'_{n-1} - b_p > b_q - a_{q1}x'_1 - \cdots - a_{q(n-1)}x'_{n-1}$$

(this inequality results when (p) and (q) are added), we can find a value x'_n of x_n which lies between left and right sides of the above inequality. Clearly $(x'_1, \cdots, x'_n)$ is a solution of the original system.

If this procedure were repeated, we would reach a system where there are inequalities with all the coefficients of zero value and where the signs of the coefficients of a variable are all the same in all the inequalities. If some of the inequalities which have all zero coefficients are contradictory there is no solution possible and the theorem is true. If none of such inequalities are contradictory the solution always exists as mentioned before and there can be no legal linear combination that is contradictory. Thus once again the theorem is true.

□

As an immediate consequence we can prove the celebrated 'Farkas Lemma'.

Theorem 2.3.2 (Farkas Lemma) *The homogeneous system*

$$\mathbf{A}\ \mathbf{x} \leq 0$$

has the consequence

$$\mathbf{d}^T\mathbf{x} \leq 0$$

iff the row vector $\mathbf{d}^T$ *is a nonnegative linear combination of the rows of* $\mathbf{A}$.

Proof : By Kuhn-Fourier Theorem (Theorem 2.3.1), the system

$$\mathbf{A}^T\mathbf{y} = \mathbf{d}$$

$$\mathbf{y} \geq \mathbf{0}$$

has a solution iff

'$\mathbf{x}^T\mathbf{A}^T + \beta^T\mathbf{I} = \mathbf{0}, \beta^T \geq \mathbf{0}$' implies '$\mathbf{x}^T\mathbf{d} \leq \mathbf{0}$';
i.e., iff '$\mathbf{Ax} \leq \mathbf{0}$' implies '$\mathbf{d}^T\mathbf{x} \leq \mathbf{0}$.'

□

The analogue of 'vector spaces' for inequality systems is '**cones**'. A **cone** is a collection of vectors closed under addition and non-negative linear combination. It is easily verified that the solution set of $\mathbf{Ax} \geq \mathbf{0}$ is a cone. Such cones are called **polyhedral**. We say vectors $\mathbf{x}$, $\mathbf{y}$ (on the same set S) are **polar** iff $< \mathbf{x}, \mathbf{y} >$ (i.e., their dot product) is nonpositive. If $\mathcal{K}$ is a collection of vectors, the **polar of** $\mathcal{K}$, denoted by $\mathcal{K}^p$ is the collection of vectors polar to every vector in $\mathcal{K}$. Thus, Farkas Lemma states:
'Let $\mathcal{C}$ be the polyhedral cone defined by $\mathbf{Ax} \leq \mathbf{0}$. A vector $\mathbf{d}$ belongs to $\mathcal{C}^p$ iff $\mathbf{d}^T$ is a nonnegative linear combination of the rows of $\mathbf{A}$.'

2.3.2 Linear Programming

Let $\mathcal{S}$ be a linear inequality system with '$\leq$' and '$=$' constraints ('$\geq$' and '$=$' constraints). The **linear programming problem** or **linear program** is to finda solution $\mathbf{x}$ of $\mathcal{S}$ which maximizes a given linear function $\mathbf{c}^T\mathbf{x}$ (minimizes a given linear function $\mathbf{c}^T\mathbf{x}$). The linear function to be optimized is called the **objective function**. A solution of $\mathcal{S}$ is called a **feasible solution**, while a solution which optimizes $\mathbf{c}^T\mathbf{x}$ is called an **optimal solution**, of the linear programming problem.

The **value** of a feasible solution is the value of the objective function on it.

The following linear programming problems are said to be **duals** of each other

Primal program

$$\text{Maximize} \qquad \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2$$

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \end{pmatrix} \begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{matrix} = \mathbf{b}_1$$

$$\begin{pmatrix} \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{matrix} \leq \mathbf{b}_2$$

$$\mathbf{x}_2 \geq 0$$

Dual program

$$\text{Minimize} \qquad \mathbf{b}_1^T\mathbf{y}_1 + \mathbf{b}_2^T\mathbf{y}_2$$

$$\begin{pmatrix} \mathbf{A}_{11}^T & \mathbf{A}_{21}^T \end{pmatrix} \begin{matrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{matrix} = \mathbf{c}_1$$

$$\begin{pmatrix} \mathbf{A}_{12}^T & \mathbf{A}_{22}^T \end{pmatrix} \begin{matrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{matrix} \geq \mathbf{c}_2$$

$$\mathbf{y}_2 \geq 0.$$

We now present the duality theorem of linear programming [von Neumann47], [Gale+Kuhn+Tucker51].

Theorem 2.3.3 *For dual pairs of linear programs the following statements hold:*

i. The value of each feasible solution of the minimization program is greater than or equal to the value of each feasible solution of the maximization program;

ii. if both programs have feasible solutions then both have optimal solutions and the optimal values are equal;

iii. if one program has an optimal solution then so does the other.

The usual proof uses Farkas Lemma, or more conveniently, Kuhn-Fourier Theorem. We only sketch it.
Sketch of Proof: Part (i) follows by the solution of Exercise 2.1. Now we write down the inequalities of the primal and dual programs and another '$\leq$' inequality which is the **opposite** of the inequality in part (i). Part (ii) would be proved if this system of inequalities has a solution. We assume it has no solution and derive a contradiction by using Kuhn-Fourier Theorem.

□

Exercise 2.1 *Prove part (i) of Theorem 2.3.3.*

A very useful corollary of Theorem 2.3.3 is the following:

Corollary 2.3.1 *(***Complementary Slackness***)*
Let
$\left\{ \begin{array}{c} \max \mathbf{c}^T\mathbf{x} \\ \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{array} \right\}$ *and* $\left\{ \begin{array}{c} \min \mathbf{b}^T\mathbf{y} \\ \mathbf{A}^T\mathbf{y} \geq \mathbf{c} \end{array} \right\}$
be dual linear programs. Let $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ *be optimal solutions to the respective programs. Then,*

i. $\hat{x}_i > 0$ *implies* $(\mathbf{A}^T)_i\hat{\mathbf{y}} = c_i$,

ii. $(\mathbf{A}^T)_i\hat{\mathbf{y}} > c_i$ *implies* $\hat{\mathbf{x}}_i = 0$.

Proof : We have by part (ii) of Theorem 2.3.3 $\mathbf{c}^T\hat{\mathbf{x}} = \hat{\mathbf{y}}^T\mathbf{b}$, equivalently

$$\mathbf{c}^T\hat{\mathbf{x}} - \hat{\mathbf{y}}^T\mathbf{A}\hat{\mathbf{x}} = (\mathbf{c}^T - \hat{\mathbf{y}}^T\mathbf{A})\hat{\mathbf{x}} = 0.$$

The result now follows since $(\mathbf{c}^T - \hat{\mathbf{y}}^T\mathbf{A}) \geq 0$ and $\hat{\mathbf{x}} \geq \mathbf{0}$.

□

2.4 Solutions of Exercises

E 2.1: We use the linear programs given in the definition of dual linear programs. We have

$$\begin{aligned}\begin{pmatrix} \mathbf{b}_1^T & \mathbf{b}_2^T \end{pmatrix} \begin{matrix}\mathbf{y}_1 \\ \mathbf{y}_2\end{matrix} &\geq \left(\begin{pmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T \end{pmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}^T \right) \begin{matrix}\mathbf{y}_1 \\ \mathbf{y}_2\end{matrix} \\ &\geq \begin{pmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T \end{pmatrix} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix}.\end{aligned}$$

2.5 Solutions of Problems

Most of these problems can be found as standard results in undergraduate texts on linear algebra (see for instance [Hoffman+Kunze72]). We only give the solution to the last two problems. Here we follow [MacDuffee33], [Gantmacher59] respectively.

P 2.16: We state the following simple lemma without proof.

Lemma 2.5.1 *If $\alpha_1, \cdots, \alpha_t$ are permutations of $\{1, \cdots, n\}$ then $sgn(\alpha_1\alpha_2\cdots\alpha_t) = (sgn(\alpha_1))(sgn(\alpha_2))\cdots(sgn(\alpha_t))$ (where $\alpha_i\alpha_j$ denotes composition of permutations α_i, α_j).*

We have

$$\sum sgn(\sigma)\ det\left(\mathbf{A}\begin{pmatrix} r_1, & \cdots & , r_k \\ i_1, & \cdots & , i_k \end{pmatrix}\right) det\left(\mathbf{A}\begin{pmatrix} r_{k+1}, & \cdots & , r_m \\ i_{k+1}, & \cdots & , i_m \end{pmatrix}\right) =$$
$$\sum sgn(\sigma)(\sum sgn(\alpha)(a_{r_1\alpha(i_1)}\cdots a_{r_k\alpha(i_k)}))(\sum sgn(\beta)(a_{r_{k+1}\beta(i_{k+1})}\cdots a_{r_n\beta(i_n)})),$$

where α, β are permutations on the sets $\{i_1, \cdots, i_k\}$, $\{i_{k+1}, \cdots, i_n\}$ respectively. Let α' agree with α over $\{i_1, \cdots, i_k\}$ and over $\{i_{k+1}, \cdots, i_n\}$, with the identity permutation. Let β' agree with β over $\{i_{k+1}, \cdots, i_n\}$ and with the identity permutation over $\{i_1, \cdots, i_k\}$. So

$$\begin{aligned} LHS &= \sum sgn(\sigma)sgn(\alpha')sgn(\beta')(a_{r_1\alpha(i_1)}\cdots a_{r_k\alpha(i_k)}a_{r_{k+1}\beta(i_{k+1})}\cdots a_{r_n\beta(i_n)}) \\ &= \sum sgn(\beta'\alpha'\sigma)(a_{r_1\alpha\sigma(r_1)}\cdots a_{r_k\alpha\sigma(r_k)}a_{r_{k+1}\beta\sigma(r_{k+1})}\cdots a_{r_n\beta\sigma(r_n)}) \\ &= \sum sgn(\mu)(a_{r_1\mu(r_1)}\cdots a_{r_k\mu(r_k)}a_{r_{k+1}\mu(r_{k+1})}\cdots a_{r_n\mu(r_n)}), \end{aligned}$$

where $\mu \equiv \beta'\alpha'\sigma$. Since the RHS is the usual definition of the determinant of $\mathbf{A}$, the proof is complete.

P 2.17: Let a_{ij}, b_{ij} denote respectively the $(i,j)^{th}$ entry of $\mathbf{A}, \mathbf{B}$. Then the matrix

$$\mathbf{AB} = \begin{bmatrix} \sum_{i_1=1}^n a_{1i_1} b_{i_1 1} & \cdots & \sum_{i_m=1}^n a_{1i_m} b_{i_m m} \\ \vdots & & \vdots \\ \sum_{i_1=1}^n a_{mi_1} b_{i_1 1} & \cdots & \sum_{i_m=1}^n a_{mi_m} b_{i_m m} \end{bmatrix}.$$

Now each column of $\mathbf{AB}$ can be thought of as the sum of n appropriate columns - for instance the transpose of the first column is made up of rows - a typical one being $(a_{1i_1} b_{i_1 1}, \cdots, a_{mi_1} b_{i_1 1})$. Using Theorem 2.2.8

$$\begin{aligned} det(\mathbf{AB}) &= \sum_{i_1,\cdots,i_m} det\left(\begin{bmatrix} a_{1i_1} b_{i_1 1} & \cdots & a_{1i_m} b_{i_m m} \\ \vdots & & \vdots \\ a_{mi_1} b_{i_1 1} & \cdots & a_{mi_m} b_{i_m m} \end{bmatrix} \right) \\ &= \sum (b_{i_1 1} \cdots b_{i_m m}) \ det\left(\mathbf{A} \begin{pmatrix} 1, & \cdots & , m \\ i_1, & \cdots & , i_m \end{pmatrix} \right), \end{aligned}$$

where $\mathbf{A}\begin{pmatrix} 1, & \cdots & , m \\ i_1, & \cdots & , i_m \end{pmatrix}$ is the $m \times m$ matrix which has the first m rows of $\mathbf{A}$ in the same order as in $\mathbf{A}$ but whose j^{th} column is the i_j^{th} column of $\mathbf{A}$. So, again by Theorem 2.2.8,

$$\det(\mathbf{AB}) = \sum_{k_1,\cdots,k_m} det\left(\mathbf{A} \begin{pmatrix} 1, & \cdots & , m \\ k_1, & \cdots & , k_m \end{pmatrix} \right) (sgn(\sigma)) \ b_{\sigma(k_1)1} \cdots b_{\sigma(k_m)m},$$

where $k_1 < \cdots < k_m, \{k_1, \cdots, k_m\} = \{i_1, \cdots, i_m\}$ and σ is the permutation $\begin{pmatrix} k_1, & \cdots & , k_m \\ i_1, & \cdots & , i_m \end{pmatrix}$, i.e.,

$$\sigma(k_j) = i_j.$$

So,

$$det(\mathbf{AB}) = \sum_{\substack{k_1, \cdots, k_m \\ k_1 < \cdots < k_m}} det\left(\mathbf{A} \begin{pmatrix} 1, & \cdots & , m \\ k_1, & \cdots & , k_m \end{pmatrix} \right) det\left(\mathbf{B} \begin{pmatrix} k_1, & \cdots & , k_m \\ 1, & \cdots & , m \end{pmatrix} \right).$$

Chapter 3

Graphs

3.1 Introduction

We give definitions of graphs and related notions below. Graphs should be visualized as points joined by lines with or without arrows rather than be thought of as formal objects. We would not hesitate to use informal language in proofs.

3.2 Graphs: Basic Notions

3.2.1 Graphs and Subgraphs

A graph $\mathcal{G}$ is a triple $(V(\mathcal{G}), E(\mathcal{G}), i_{\mathcal{G}})$ where $V(\mathcal{G})$ is a finite set of **vertices**, $E(\mathcal{G})$ is a finite set of **edges** and $i_{\mathcal{G}}$ is an **incidence function** which associates with each edge a pair of vertices, not necessarily distinct, called its **end points** or **end vertices** (i.e., $i_{\mathcal{G}} : E(\mathcal{G}) \rightarrow$ collection of subsets of $V(\mathcal{G})$ of cardinality 2 or 1).
Vertices are also referred to as **nodes** or **junctions** while edges are referred to also as **arcs** or **branches**.
We note

i. an edge may have a single end point - such edges are called **selfloops**.

ii. a vertex may have no edges incident on it - such vertices are said to be **isolated**.

iii. the graph may be in several pieces.

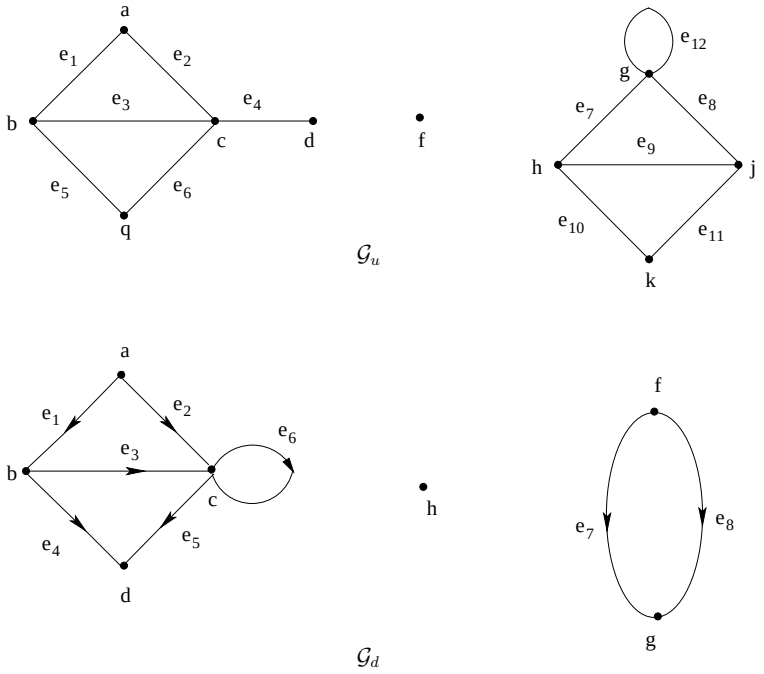


Figure 3.1: Undirected and Directed Graphs

Figure 3.1 shows a typical graph $\mathcal{G}_u$.

A **directed** graph $\mathcal{G}$ is a triple $(V(\mathcal{G}), E(\mathcal{G}), a_{\mathcal{G}})$ where $V(\mathcal{G}), E(\mathcal{G})$ are the vertex set and the edge set respectively and $a_{\mathcal{G}}$ associates with each edge an ordered pair of vertices not necessarily distinct (i.e., $a_{\mathcal{G}} : E(\mathcal{G}) \rightarrow V(\mathcal{G}) \times V(\mathcal{G})$). The first element of the ordered pair is the **positive end point** or **tail** of the arrow and the second element is the **negative end point** or **head** of the arrow. For selfloops, positive and negative endpoints are the same. Directed graphs are usually drawn as graphs with arrows in the edges. In Figure 3.1, $\mathcal{G}_d$ is a directed graph.

We say a vertex v and an edge e are **incident** on each other iff v is an end point of e. If e has end points u, v we say that u, v are **adjacent**

to each other. Two edges e_1, e_2 are **adjacent** if they have a common end point. The **degree** of a vertex is the number of edges incident on it with selfloops counted twice.
A graph $\mathcal{G}_s$ is a **subgraph** of $\mathcal{G}$ iff $\mathcal{G}_s$ is a graph, $V(\mathcal{G}_s) \subseteq V(\mathcal{G}), E(\mathcal{G}_s) \subseteq E(\mathcal{G})$, and the endpoints of an edge in $\mathcal{G}_s$ are the same as its end points in $\mathcal{G}$.
Subgraph $\mathcal{G}_s$ is a **proper subgraph** of $\mathcal{G}$ iff it is a subgraph of $\mathcal{G}$ but not identical to it. The **subgraph** of $\mathcal{G}$ on V_1 is that subgraph of $\mathcal{G}$ which has V_1 as its vertex set and the set of edges of $\mathcal{G}$ with both end points in V_1 as the edge set. The **subgraph** of $\mathcal{G}$ on E_1 has $E_1 \subseteq E(\mathcal{G})$ as the edge set and the endpoints of edges in E_1 as the vertex set. If $\mathcal{G}$ is a directed graph the edges of a subgraph would retain the directions they had in $\mathcal{G}$ (i.e., they would have positive and negative end points as in $\mathcal{G}$).

Exercise 3.1 *(k) In any graph with atleast two nodes and no parallel edges (edges with the same end points) or selfloops show that the degree of some two vertices must be equal.*

Exercise 3.2 *(k) Show that*

i. the sum of the degrees of vertices of any graph is equal to twice the number of edges of the graph;

ii. the number of odd degree vertices in any graph must be even.

3.2.2 Connectedness

A **vertex edge alternating sequence** (**alternating sequence** for short) of a graph $\mathcal{G}$ is a sequence in which

i. vertices and edges of $\mathcal{G}$ alternate,

ii. the first and last elements are vertices and

iii. whenever a vertex and an edge occur as adjacent elements they are incident on each other in the graph.

Example: For the graph $\mathcal{G}_u$ in Figure 3.1, $(a, e_1, b, e_3, c, e_6, q, e_6, c, e_4, d)$ is an alternating sequence.
A **path** is a graph all of whose edges and vertices can be arranged in an alternating sequence without repetitions.
It can be seen that the degree of precisely two of the vertices of the path is one and the degree of all other vertices (if any) is two. The two vertices of degree one must appear at either end of any alternating sequence containing all nodes and edges of the path without repetition. They are called **terminal nodes**. The path is said to be **between** its terminal nodes. It is clear that there are only two such alternating sequences that we can associate with a path. Each is the reverse of the other. The two alternating sequences associated with the path in Figure 3.2 are (v_1, e_1, v_2, e_2, v_3, e_3, v_4) and (v_4, e_3, v_3, e_2, v_2, e_1, v_1).

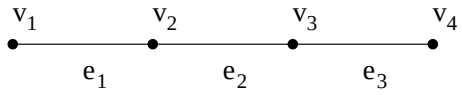


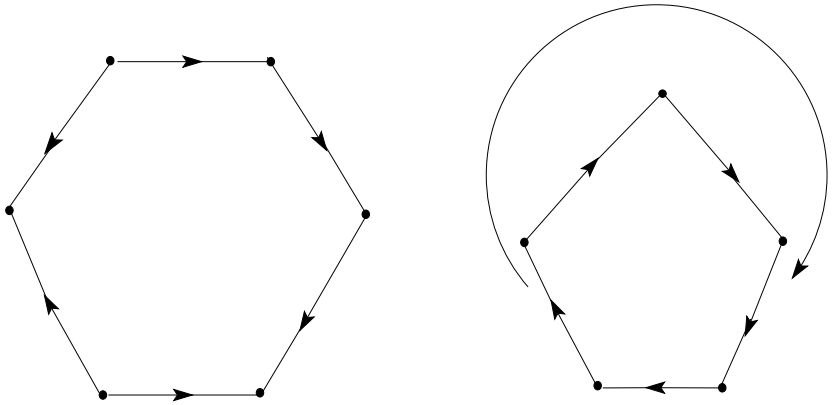
Figure 3.2: A Path Graph

We say 'go along the path from v_i to v_j' instead of 'construct the alternating sequence without repetitions having v_i as the first element and v_j as the last element'. Such sequences are constructed by considering the alternating sequence associated with the path in which v_i precedes v_j and taking the subsequence starting with v_i and ending with v_j.
A directed graph may be a path if it satisfies the above conditions. However, the term **strongly directed path** is used if the edges can be arranged in a sequence so that the negative end point of each edge, except the last is the positive end point of the succeeding edge.
A graph is said to be **connected** iff for any given pair of distinct vertices there exists a path subgraph between them. The path graph in Figure 3.2 is connected while the graph $\mathcal{G}_u$ in Figure 3.1 is disconnected.
A **connected component** of a graph $\mathcal{G}$ is a connected subgraph of $\mathcal{G}$ that is not a proper subgraph of any connected subgraph of $\mathcal{G}$ (i.e., it is a maximal connected subgraph). Connected components correspond to 'pieces' of a disconnected graph.

Exercise 3.3 *(k) Let $\mathcal{G}$ be a connected graph. Show that there is a vertex such that if the vertex and all edges incident on it are removed the remaining graph is still connected.*

3.2.3 Circuits and Cutsets

A connected graph with each vertex having degree two is called a **circuit graph** or a **polygon graph**. ($\mathcal{G}_L$ in Figure 3.3 is a circuit graph). If $\mathcal{G}'$ is a circuit subgraph of $\mathcal{G}$ then $E(\mathcal{G}')$ is a **circuit** of $\mathcal{G}$. A single edged circuit is called a **selfloop**.



$\mathcal{G}_L$ $\mathcal{G}_{L_D}$

Figure 3.3: A Circuit Graph and a Strongly Directed Circuit Graph

Each of the following is a characteristic property of circuit graphs (i.e., each can be used to define the notion).
We omit the routine proofs.

i. A circuit graph has precisely two paths between any two of its vertices.

ii. If we start from any vertex v of a circuit graph and follow any path (i.e., follow an edge, reach an adjacent vertex, go along a new edge incident on that vertex and so on) the first vertex to be repeated would be v. Also during the traversal we would have encountered all vertices and edges of the circuit graph.

iii. Deletion of any edge (leaving the end points in place) of a circuit graph reduces it to a path.

Exercise 3.4 *Construct*

i. a graph with all vertices of degree 2 *that is not a circuit graph,*

ii. a non circuit graph which is made up of a path and an additional edge,

iii. a graph which has no circuits,

iv. a graph which has every edge as a circuit.

Exercise 3.5 *Prove*

Lemma 3.2.1 *(k) Deletion of an edge (leaving end points in place) of a circuit subgraph does not increase the number of connected components in the graph.*

Exercise 3.6 *Prove*

Theorem 3.2.1 *(k) A graph contains a circuit if it contains two distinct paths between some two of its vertices.*

Exercise 3.7 *Prove*

Theorem 3.2.2 *(k) A graph contains a circuit if every one of its vertices has degree* ≥ 2.

A set $T \subseteq E(\mathcal{G})$ is a **crossing edge set** of $\mathcal{G}$ if $V(\mathcal{G})$ can be partitioned into sets V_1, V_2 such that $T = \{e : e$ has an end point in V_1 and in $V_2\}$. (In Figure 3.4, C is a crossing edge set). We will call V_1, V_2 the **end vertex sets** of T. Observe that while end vertex sets uniquely determine a crossing edge set there may be more than one pair of end vertex sets consistent with a given crossing edge set. A crossing edge set that is minimal (i.e., does not properly contain another crossing edge set) is called a **cutset** or a **bond**. A single edged cutset is a **coloop**.

Exercise 3.8 *Construct a graph which has (a) no cutsets (b) every edge as a cutset.*

Exercise 3.9 *Construct a crossing edge set that is not a cutset (see Figure 3.4).*

Exercise 3.10 *(k) Show that a cutset is a minimal set of edges with the property that when it is deleted leaving endpoints in place the number of components of the graph increases.*

Exercise 3.11 *Short (i.e., fuse end points of an edge and remove the edge) all branches of a graph except a cutset. How does the resulting graph look?*

Exercise 3.12 *Prove*

Theorem 3.2.3 *(k) A crossing edge set T is a cutset iff it satisfies the following:*

i. If the graph has more than one component then T must meet the edges of only one component and

ii. if the end vertex sets of T are V_1, V_2 in that component, then the subgraphs on V_1 and V_2 must be connected.

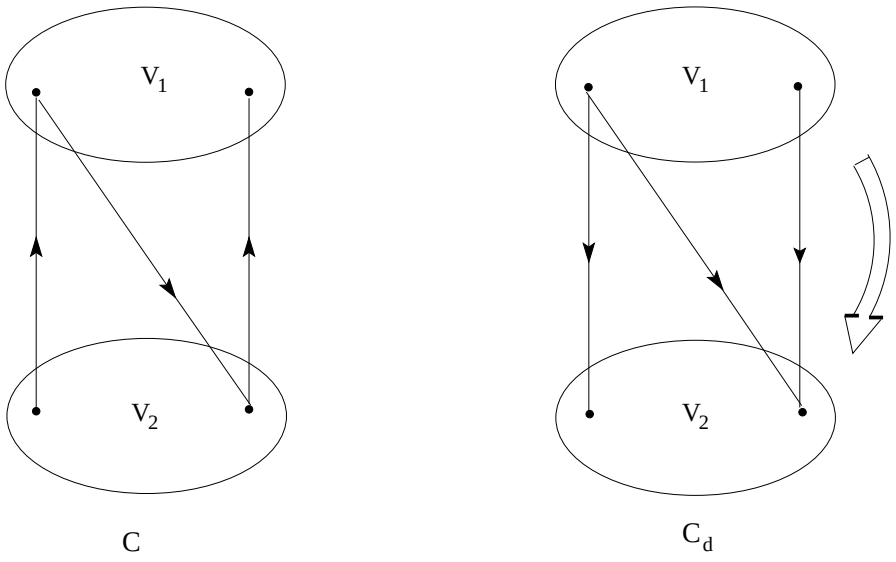


Figure 3.4: A Crossing Edge Set and a Strongly Directed Crossing Edge Set

3.2.4 Trees and Forests

A graph that contains no circuits is called a **forest graph** (see graphs $\mathcal{G}_t$ and $\mathcal{G}_f$ in Figure 3.5). A connected forest graph is also called a **tree graph** (see graph $\mathcal{G}_t$ in Figure 3.5).

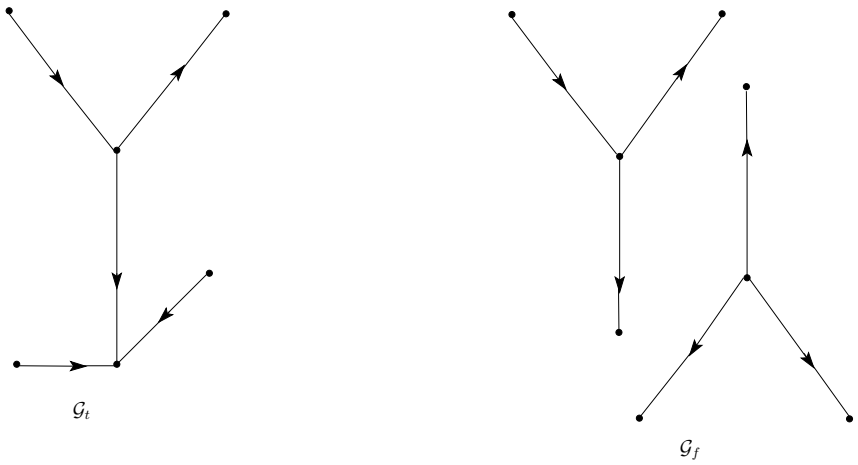


Figure 3.5: A Tree Graph $\mathcal{G}_t$ and a Forest Graph $\mathcal{G}_f$

A **forest** of a graph $\mathcal{G}$ is the set of edges of a forest subgraph of $\mathcal{G}$ that has $V(\mathcal{G})$ as its vertex set and has as many connected components as $\mathcal{G}$ has. A forest of a connected graph $\mathcal{G}$ is also called a **tree** of $\mathcal{G}$. The complement relative to $E(\mathcal{G})$ of a forest (tree) is a **coforest (cotree)** of $\mathcal{G}$. The number of edges in a forest (coforest) of $\mathcal{G}$ is its **rank (nullity)**. Theorem 3.2.4 assures us that this notion is well defined.

Exercise 3.13 *(k) Show that a tree graph on two or more nodes has*

i. precisely one path between any two of its vertices

ii. at least two vertices of degree one.

Exercise 3.14 *Prove*

Theorem 3.2.4 *(k) A tree graph on n nodes has $(n-1)$ branches. Any connected graph on n nodes with $(n-1)$ edges is a tree graph.*

Corollary 3.2.1 *The forest subgraph on n nodes and p components has $(n-p)$ edges.*

Exercise 3.15 *Prove*

Theorem 3.2.5 *(k) A subset of edges of a graph is a forest (coforest) iff it is a maximal subset not containing any circuit (cutset).*

Exercise 3.16 *(k) Show that every forest (coforest) of a graph $\mathcal{G}$ intersects every cutset (circuit) of $\mathcal{G}$.*

Exercise 3.17 *Prove*

Lemma 3.2.2 *(k) A tree graph splits into two tree graphs if an edge is opened (deleted leaving its end points in place).*

Exercise 3.18 *(k) Show that a tree graph yields another tree graph if an edge is shorted (removed after fusing its end points).*

Exercise 3.19 *Prove*

Theorem 3.2.6 *(k) Let f be a forest of a graph $\mathcal{G}$ and let e be an edge of $\mathcal{G}$ outside f. Then $e \cup f$ contains only one circuit of $\mathcal{G}$.*

Exercise 3.20 *Prove*

Theorem 3.2.7 *(k) Let $\overline{f}$ be a coforest of a graph $\mathcal{G}$ and let e be an edge of $\mathcal{G}$ outside $\overline{f}$ (i.e., $e \in f$). Then $e \cup \overline{f}$ contains only one cutset of $\mathcal{G}$ (i.e., only one cutset of $\mathcal{G}$ intersects f in e).*

Exercise 3.21 *(k) Show that every circuit is an f-circuit with respect to some forest (i.e., intersects some coforest in a single edge).*

Exercise 3.22 *(k) Show that every cutset is an f-cutset with respect to some forest (i.e., intersects some forest in a single edge).*

Exercise 3.23 *(k) Show that shorting an edge in a cutset of a graph does not reduce the nullity of the graph.*

3.2.5 Strongly Directedness

The definitions we have used thus far hold also in the case of directed graphs. The subgraphs in each case retain the original orientation for the edges. However, the prefix 'strongly directed' in each case implies a stronger condition. We have already spoken of the strongly directed path. A strongly directed circuit graph has its edges arranged in a sequence so that the negative end point of each edge is the positive

end point of the succeeding edge and the positive end point of the last edge is the negative end point of the first (see $\mathcal{G}_{L_d}$ in Figure 3.3). The set of edges of such a graph would be a **strongly directed circuit**.

A **strongly directed crossing edge set** would have the positive end points of all its edges set in the same end vertex set (see C_d in Figure 3.4).

In this book we will invariably assume that the graph is directed but our circuit subgraphs, paths etc. although they are directed graphs, will, unless otherwise stated, not be strongly directed. When it is clear from the context the prefix 'directed' will be omitted when we speak of a graph. For simplicity we would write directed path, directed circuit, directed crossing edge set instead of strongly directed path etc.

Exercise 3.24 *Prove:*
(Minty) Any edge of a directed graph is either in a directed circuit or in a directed cutset but not both.

(For solution see Theorem 3.4.7).

3.2.6 Fundamental Circuits and Cutsets

Let f be a forest of $\mathcal{G}$ and let $e \notin f$. It can be shown (Theorem 3.2.6) that there is a unique circuit contained in $e \cup f$. This circuit is called the **fundamental circuit (f - circuit) of e with respect to f** and is denoted by $L(e, f)$. Let $e_t \in f$. It can be shown (Theorem 3.2.7) that there is a unique cutset contained in $e_t \cup \bar{f}$. This cutset is called the **fundamental cutset of e_t with respect to f** and is denoted by $B(e_t, f)$.

Remark: The f-circuit $L(e, f)$ is obtained by adding e to the unique path in the forest subgraph on f between the end points of e. For the subgraph on f, the edge e_t is a crossing edge set with end vertex sets say V_1, V_2. Then the f-cutset $B(e_t, f)$ is the crossing edge set of $\mathcal{G}$ with end vertex sets V_1, V_2.

3.2.7 Orientation

Let $\mathcal{G}$ be a directed graph. We associate **orientations** with circuit subgraphs and crossing edge sets as follows:

An **orientation** of a circuit subgraph is an alternating sequence of its vertices and edges, without repetitions except for the first vertex being also the last (note that each edge is incident on the preceding and succeeding vertices). Two orientations are **equivalent** if one can be obtained by a **cyclic** shift of the other. Diagrammatically an orientation may be represented by a circular arrow. It is easily seen that there can be at most two orientations for a circuit graph. (A single edge circuit subgraph has only one). These are obtained from each other by reversing the sequence. When there are two non equivalent orientations we call them **opposite** to each other. We say that an edge of the circuit subgraph **agrees** with the **orientation** if its positive end point immediately precedes itself in the orientation (or in an equivalent orientation). Otherwise it is opposite to the orientation.
The orientation associated with a circuit subgraph would also be called the **orientation** of the **circuit**.

Example: For the circuit subgraph of Figure 3.6 the orientations $(n_1, e,\ n_6,\ e_6,\ n_5,\ e_5,\ n_4,\ e_4,\ n_3,\ e_3,\ n_2,\ e_2,\ n_1)$, and $(n_6,\ e_6,\ n_5,\ e_5,\ n_4,\ e_4,\ n_3,\ e_3,\ n_2,\ e_2,\ n_1, e, n_6)$ are equivalent. This is the orientation shown in the figure. It is opposite to the orientation $(n_1, e_2,\ n_2,\ e_3,\ n_3,\ e_4,\ n_4,\ e_5,\ n_5,\ e_6,\ n_6, e, n_1)$. The edge e **agrees** with this latter orientation and is **opposite** to the former orientation.

An **orientation** of a crossing edge set is an ordering of its end vertex sets V_1, V_2 as (V_1, V_2) or as (V_2, V_1). An edge e in the crossing edge set with positive end point in V_1 and negative end point in V_2 **agrees** with the orientation (V_1, V_2) and is **opposite** to the orientation (V_2, V_1). In Figure 3.6 the orientation of the crossing edge set is (V_1, V_2).

Theorem 3.2.8 *(k) Let f be a forest of a directed graph $\mathcal{G}$. Let $e_t \in f$ and let $e_c \in \overline{f}$. Let the orientation of $L(e_c, f)$ and $B(e_t, f)$ agree with e_c, e_t, respectively. Then $L(e_c, f) \cap B(e_t, f) = \emptyset$ or $\{e_c, e_t\}$.*
Further when the intersection is nonvoid e_t agrees with (opposes) the orientation of $L(e_c, f)$ iff e_c opposes (agrees with) the orientation of $B(e_t, f)$.

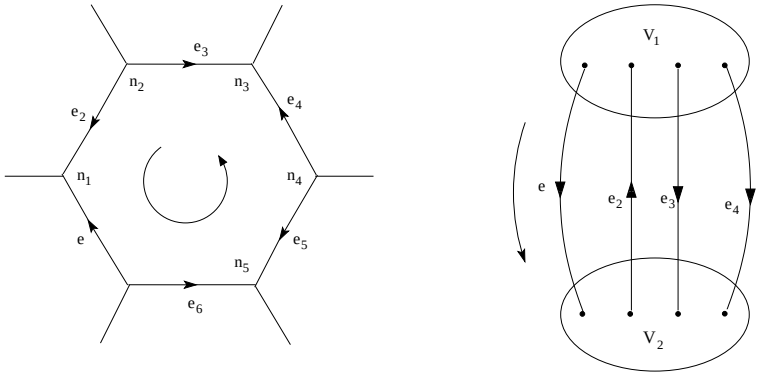


Figure 3.6: Circuit subgraph and Crossing Edge Set with Orientations

Proof : We confine ourselves to the case where $\mathcal{G}$ is connected since even if it is disconnected we could concentrate on the component where e_t is present.
If $B(e_t, f)$ is deleted from $\mathcal{G}$, two connected subgraphs $\mathcal{G}_1, \mathcal{G}_2$ result whose vertex sets are the end vertex sets V_1, V_2, respectively of $B(e_t, f)$. Now e_c could have both end points in V_1, both end points in V_2, or one end point in V_1 and another in V_2. In the former two cases $L(e_c, f) \cap B(e_t, f) = \emptyset$. In the last case $L(e_c, f)$ must contain e_t. For, the path in the tree subgraph on f between the endpoints of e_c must use e_t since that is the only edge in f with one endpoint in V_1 and the other in V_2. Now $L(e_c, f)$ contains only one edge, namely e_c from $\overline{f}$ and $B(e_t, f)$ contains only one edge, namely e_t from f. Hence in the third case

$$L(e_c, f) \cap B(e_t, f) = \{e_c, e_t\}.$$

Let us next assume that the intersection is nonvoid. Suppose that e_c has its positive end point a in V_1 and negative end point b in V_2. Let $(b, \cdots, e_t, \cdots, a, e_c, b)$ be an orientation of the circuit. It is clear that e_t would agree with this orientation if V_2 contains its positive end point and V_1 its negative end point (see Figure 3.7). But in that case e_c would oppose the orientation of $B(e_t, f)$ (which is (V_2, V_1), taken to agree with the orientation of e_t). The other cases can be handled similarly.

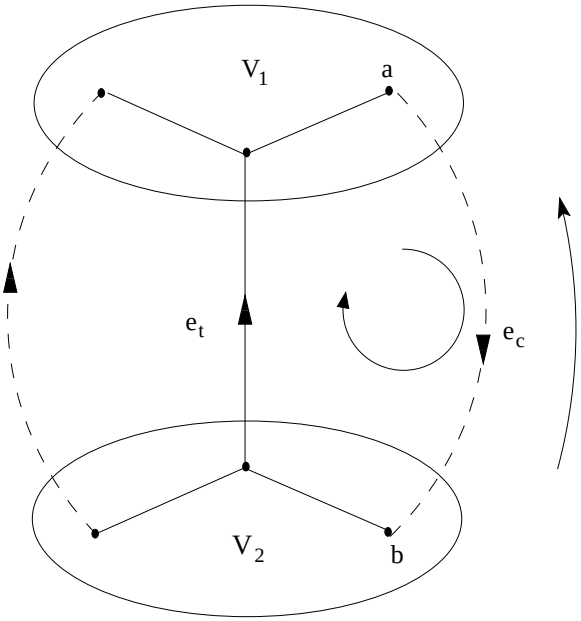


Figure 3.7: Relation between f-circuit and f-cutset

3.2.8 Isomorphism

Let $\mathcal{G}_1 \equiv (V_1, E_1, i_1)$, $\mathcal{G}_2 \equiv (V_2, E_2, i_2)$, be two graphs. We say $\mathcal{G}_1$, $\mathcal{G}_2$ are **identical** iff $V_1 = V_2, E_1 = E_2$ and $i_1 = i_2$. However, graphs could be treated as essentially the same even if they satisfy weaker conditions. We say $\mathcal{G}_1$, $\mathcal{G}_2$ are **isomorphic** to each other and denote it by (abusing notation) $\mathcal{G}_1 = \mathcal{G}_2$ iff there exist bijections $\eta : V_1 \to V_2$ and $\epsilon : E_1 \to E_2$ s.t. any edge e has end points a, b in $\mathcal{G}_1$ iff $\epsilon(e)$ has endpoints $\eta(a), \eta(b)$. If $\mathcal{G}_1, \mathcal{G}_2$ are directed graphs then we would further require that an end point a of e, in $\mathcal{G}_1$, is positive (negative) iff $\eta(a)$ is the positive (negative) endpoint of $\epsilon(e)$. When we write $\mathcal{G}_1 = \mathcal{G}_2$ usually the bijections would be clear from the context. However, when two graphs are isomorphic there would be many **isomorphisms** ((η, ϵ) pairs) between them.

The graphs $\mathcal{G}, \mathcal{G}'$ in Figure 3.8 are isomorphic. The node and edge bijections are specified by the ('). Clearly there is at least one other (η, ϵ) pair between the graphs.

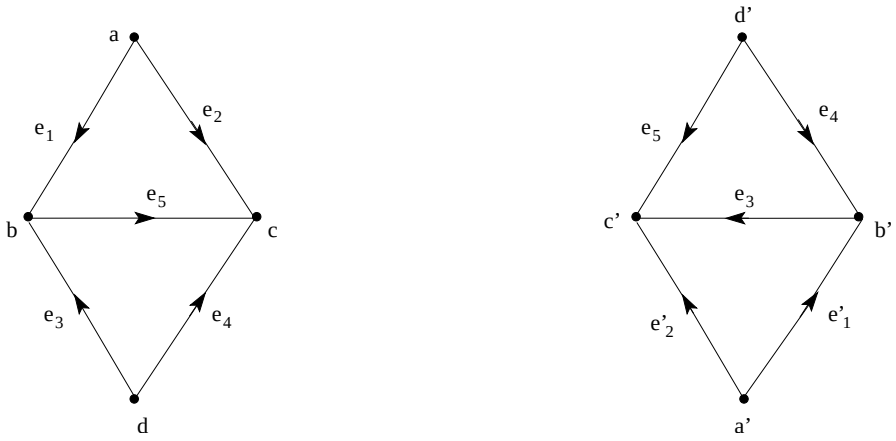
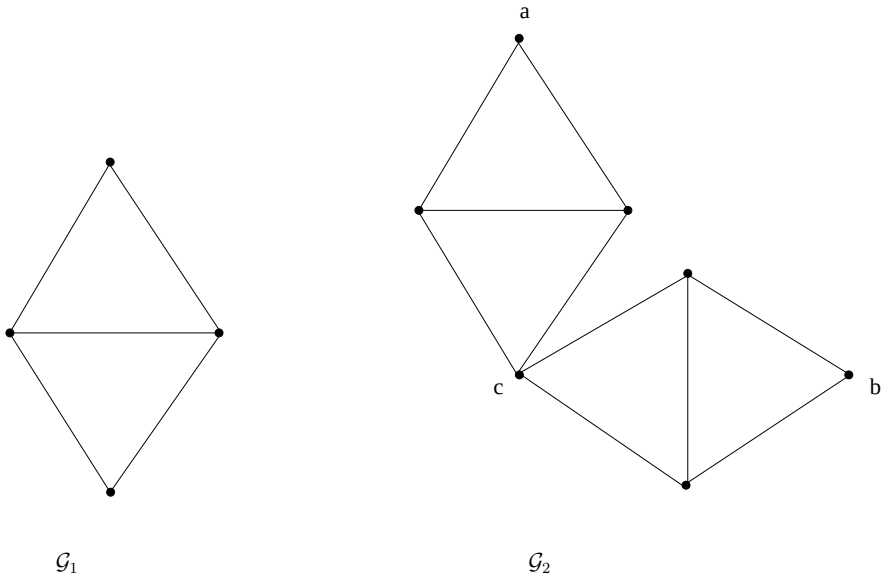


Figure 3.8: Isomorphic Directed Graphs

3.2.9 Cyclically connectedness

A graph $\mathcal{G}$ is said to be **cyclically connected** iff given any pair of vertices there is a circuit subgraph containing them.



$\mathcal{G}_1$ $\mathcal{G}_2$

Figure 3.9: Cyclically Connected and Cyclically Disconnected Graphs

Example: The graph $\mathcal{G}_1$ in Figure 3.9 is cyclically connected while $\mathcal{G}_2$ of the same figure is not cyclically connected since no circuit subgraph contains both nodes a and b.
Whenever a connected graph is not cyclically connected there would be two vertices a, b through which no circuit subgraph passes. If a, b are not joined by an edge there would be a vertex c such that every path between a and b passes through c. We then say c is a **cut vertex** or **hinge**. The graph $\mathcal{G}_2$ of Figure 3.9 has c as a cut vertex.
It can be shown that a graph is cyclically connected iff any pair of edges can be included in the same circuit.

In any graph it can be shown that if edges e_1, e_2 and e_2, e_3 belong to circuits C_{12}, C_{23}, then there exists a circuit $C_{13} \subseteq C_{12} \cup C_{23}$ s.t. $e_1, e_3 \in C_{13}$. It follows that the edges of a graph can be partitioned into blocks such that within each block every pair of distinct edges can be included in some circuit and edges belonging to different blocks cannot be included in the same circuit (each coloop would form a block by itself). We will call such a block an **elementary separator** of the graph. Unions of such blocks will be called **separators**. The subgraphs on elementary separators will be called **2-connected components**.(Note that a coloop is a 2-connected component by itself). If two 2-connected components intersect they would do so at a single vertex which would be a cut vertex. If two graphs have a single common vertex, we would say that they are put together by **hinging**.

3.3 Graphs and Vector Spaces

There are several natural 'electrical' vectors that one may associate with the vertex and edge sets of a directed graph $\mathcal{G}$.

e.g. i. potential vectors on the vertex set,
ii. current vectors on the edge set,
iii. voltage (potential difference) vectors on the edge set.

Our concern will be with the latter two examples. We need a few preliminary definitions. Henceforth, unless otherwise specified, by **graph** we mean **directed graph**.

The Incidence Matrix

The **incidence matrix A** of a graph $\mathcal{G}$ is defined as follows:
A has one row for each node and one column for each edge.

$$\begin{aligned}\mathbf{A}(i,j) \quad = \quad & +1(-1) \text{ if edge } j \text{ has its arrow leaving (entering) node } i. \\ & 0 \text{ if edge } j \text{ is not incident on node } i \\ & \quad \text{or if edge } j \text{ is a selfloop.}\end{aligned}$$

Example: The incidence matrix of the directed graph $\mathcal{G}_d$ in Figure 3.1 is

$$\mathbf{A} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ \hline f \\ g \\ \hline h \end{array} \begin{array}{c} \begin{array}{cccccc|cc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \left[\begin{array}{cccccc|cc} +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \tag{3.1}$$

Note that the selfloop e_6 is represented by a zero column. This is essential for mathematical convenience. The resulting loss of information (as to which node it is incident at) is electrically unimportant. The isolated node h corresponds to a zero row. Since the graph is disconnected the columns and rows can be ordered so that the block diagonal nature of the incidence matrix is evident.

Exercise 3.25 *(k) Prove:*
A matrix $\mathbf{K}$ *is the incidence matrix of some graph* $\mathcal{G}$ *iff it is a* $0, \pm 1$ *matrix and has either zero columns or columns with one* $+1$ *and one* -1 *and remaining entries* 0*.*

Exercise 3.26 *(k) Prove:*
The sum of the rows of $\mathbf{A}$ *is* $\mathbf{0}$*. Hence the rank of* $\mathbf{A}$ *is less than or equal to the number of its rows minus* 1*.*

Exercise 3.27 *(k) Prove:*
If the graph is disconnected the sum of the rows of $\mathbf{A}$ *corresponding to any component would add up to* $\mathbf{0}$*. Hence, the rank of* $\mathbf{A}$ *is less than or equal to the number of its rows less the number of components (=* $r(\mathcal{G})$*).*

Exercise 3.28 *(k) Prove:*
If $\mathbf{f} = \lambda^{\mathbf{T}}\mathbf{A}$*, then* $\mathbf{f}(e_i) = \lambda(a) - \lambda(b)$ *where* a *is the positive end point of* e_i *and* b*, its negative end point. Thus if* λ *represents a potential vector with* $\lambda(n)$ *denoting the potential at* n *then* $\mathbf{f}$ *represents the corresponding potential difference vector.*

Exercise 3.29 *Construct incidence matrices of various types of graphs e.g. connected, disconnected, tree, circuit, complete graph K_n (every pair of n vertices*
joined by an edge), path.

Exercise 3.30 *Show that the transpose of the incidence matrix of a circuit graph, in which all edges are directed along the orientation of the circuit, is a matrix of the same kind.*

Exercise 3.31 *(k) Show that an incidence matrix remains an incidence matrix under the following operations:*

i. deletion of a subset of the columns,

ii. replacing some rows by their sum.

3.3.1 The Circuit and Crossing Edge Vectors

A **circuit vector** of a graph $\mathcal{G}$ is a vector $\mathbf{f}$ on $E(\mathcal{G})$ corresponding to a circuit of $\mathcal{G}$ with a specified orientation:

$$\begin{aligned} f(e_i) &= +1(-1) \text{ if } e_i \text{ is in the circuit and agrees} \\ &\quad \text{with (opposes) the orientation of the circuit.} \\ &= 0 \text{ if } e_i \text{ is not in the circuit.} \end{aligned}$$

Example: The circuit vector associated with the circuit subgraph in Figure 3.6

$$\begin{array}{c} \quad e \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \\ f = \left[\begin{array}{cccccccccc} -1 & +1 & -1 & +1 & -1 & +1 & 0 & \dots & 0 \end{array} \right] \end{array} \tag{3.2}$$

Exercise 3.32 *(k) Compare a circuit vector with a row of the incidence matrix. Prove:*
A row of the incidence matrix and a circuit vector will

i. have no nonzero entries common if the corresponding node is not present in the circuit subgraph, or

ii. have exactly two nonzero entries common if the node is present in the circuit subgraph. These entries would be ± 1. One of these entries would have opposite sign in the incidence matrix row and the circuit vector and the other entry would be the same in both.

Exercise 3.33 *Prove*

Theorem 3.3.1 *(k) Every circuit vector of a graph $\mathcal{G}$ is orthogonal to every row of the incidence matrix of $\mathcal{G}$.*

(This follows immediately from the statement of the previous exercise). A **crossing edge vector** of a graph $\mathcal{G}$ is a vector $\mathbf{f}$ on $E(\mathcal{G})$ corresponding to a crossing edge set with a specified orientation (V_1, V_2):

$\mathbf{f}(e_i)$ = +1(-1) if e_i is in the crossing edge set and agrees with (opposes) the orientation (V_1, V_2).
= 0 if e_i is not in the crossing edge set.

If the crossing edge set is a cutset then the corresponding vector is a **cutset vector**.

Example: The crossing edge vector associated with the crossing edge set of Figure 3.6 is

$$\begin{matrix} & e & e_2 & e_3 & e_4 & & & \\ f = \Big[& +1 & -1 & +1 & +1 & 0 & \cdots & 0 \Big]. \end{matrix} \tag{3.3}$$

Exercise 3.34 *Prove*

Theorem 3.3.2 *(k) The crossing edge vector corresponding to the crossing edge set of orientation (V_1, V_2) is obtained by summing the rows of the incidence matrix corresponding to the nodes in V_1.*

Hence, a crossing edge vector of $\mathcal{G}$ is a voltage vector and is orthogonal to every circuit vector of $\mathcal{G}$. (This can also be proved directly).

Exercise 3.35 *(k) When is a row of the incidence matrix also a cutset vector? Can a cutset be a circuit? Can a cutset vector be a circuit vector?*

Exercise 3.36 *(k)* **RRE of an Incidence Matrix:**
Give a simple rule for finding the RRE of an incidence matrix.

3.3.2 Voltage and Current Vectors

For a graph $\mathcal{G}$ a **current vector i** is a vector on $E(\mathcal{G})$ that is orthogonal to the rows of the incidence matrix of $\mathcal{G}$, **equivalently**, that satisfies Kirchhoff's current equations (KCE): $\mathbf{Ax} = \mathbf{0}$ [Kirchhoff1847].

A **voltage vector v** of $\mathcal{G}$ is a vector on $E(\mathcal{G})$ that is linearly dependent on the rows of the incidence matrix of $\mathcal{G}$ i.e.
$\mathbf{v^T} = \lambda^\mathbf{T}\mathbf{A}$ for some vector λ.
The vector λ assigns a value to each node of $\mathcal{G}$ and is called a **potential vector**. We say **v** is **derived** from the node potential vector λ.
Voltage vectors and current vectors form vector spaces denoted by $\mathcal{V}_v(\mathcal{G}), \mathcal{V}_i(\mathcal{G})$, and called voltage space of $\mathcal{G}$ and current space of $\mathcal{G}$ respectively.

Exercise 3.37 *Prove*
*(***Tellegen's Theorem (weak form)***) Any voltage vector of $\mathcal{G}$ is orthogonal to every current vector of $\mathcal{G}$.*

Remark: When the graph is disconnected with components $\mathcal{G}_1 \ldots \mathcal{G}_p$ it is clear that both the current and voltage space can be written in the form $\oplus_{i=1}^{p}\mathcal{V}(\mathcal{G}_i)$. However, in order to write the space in this decomposed form it is not necessary that the $\mathcal{G}_i$ be disconnected. All that is required is that there be no circuit containing edges from different $\mathcal{G}_i$ (see the discussion on separators). We say that graphs $\mathcal{G}_1, \mathcal{G}_2$ are **2-isomorphic** and denote it by $\mathcal{G}_1 \cong \mathcal{G}_2$ iff there exists a bijection $\in$: $E(\mathcal{G}_1) \to E(\mathcal{G}_2)$ through which an edge in $\mathcal{G}_1$ can be identified with an edge in $\mathcal{G}_2$ so that $\mathcal{V}_v(\mathcal{G}_1) = \mathcal{V}_v(\mathcal{G}_2)$.
Whitney [Whitney33c] has shown that two 2-isomorphic graphs can be made isomorphic through the repeated use of the following operations:

i. Decompose the graphs into their 2-connected components.

ii. Divide one of the graphs into two subgraphs $\mathcal{G}'$ and $\mathcal{G}$" which have precisely two vertices, say a and b, in common. Split the nodes into a_1, a_2 and b_1, b_2 so that the two subgraphs are now disconnected with a_1, b_1, belonging to $\mathcal{G}'$ and a_2, b_2 to $\mathcal{G}$". Let $\mathcal{G}'_e$ be the graph obtained from $\mathcal{G}'$ by adding an edge e between a_1, b_1. Now reverse all arrows of edges of $\mathcal{G}'$ which lie in the 2-connected component containing e in $\mathcal{G}'_e$ and attach a_1 to b_2 and a_2 to b_1.

If **c** is a circuit vector corresponding to the circuit C with an orientation then the **Kirchhoff's Voltage Equation** (KVE) [Kirchhoff1847] corresponding to C is

$$\mathbf{c}^T\mathbf{x} = 0$$

We have the following basic characterization of voltage vectors:

Theorem 3.3.3 *(k) A vector* $\mathbf{v}$ *on* $E(\mathcal{G})$ *is a voltage vector iff it satisfies KVE corresponding to each circuit with an orientation.*

Proof : By Theorem 3.3.1 we know that a circuit vector is orthogonal to every row of the incidence matrix. Hence, a circuit vector is orthogonal to any vector that is linearly dependent on the rows of the incidence matrix i.e. orthogonal to a voltage vector. Hence, every voltage vector satisfies KVE corresponding to any circuit with orientation. Now let $\mathbf{v}$ be a vector that satisfies KVE corresponding to every circuit with an orientation. We will construct a potential vector λ s.t. $\lambda^{\mathbf{T}}\mathbf{A} = \mathbf{v}^{\mathbf{T}}$. Take any node d as the datum node, i.e., $\lambda(d) \equiv 0$. Suppose $\lambda(a)$ is already defined and edge e has a as the positive (negative) end and b as the opposite end. Then we take $\lambda(b) \equiv \lambda(a) - v(e) (\lambda(b) \equiv \lambda(a) + v(e))$. In this manner every node in the same connected component is assigned a λ value. A node that is reachable from d by two different paths will not be assigned two different values as otherwise we can find a circuit with orientation for which KVE is violated. Repeating this procedure for each component yields a λ vector s.t. $\lambda^T\mathbf{A} = \mathbf{v}^{\mathbf{T}}$.

□

3.3.3 Voltage and Current Vector Spaces and Tellegen's Theorem

In this subsection, we compute the rank of $\mathcal{V}_v(\mathcal{G})$ and $\mathcal{V}_i(\mathcal{G})$ and prove that the spaces are complementary orthogonal (**Tellegen's Theorem (strong form)**).

Theorem 3.3.4 *(k) Let* G *be a graph on* n *nodes with* p *connected components. Then*

i. Any set of $(n-p)$ *rows of* $\mathbf{A}$ *which omits one row per component of* $\mathcal{G}$*, is a basis of* $\mathcal{V}_v(\mathcal{G})$.

ii. $r(\mathcal{V}_v(\mathcal{G})) = n - p$

Proof :
If $\mathcal{G}$ is made up of p connected components, by (if necessary) rearranging the rows and columns of $\mathbf{A}$ it can be put in the block diagonal form with p blocks. Hence, any union of linearly independent vectors from different $\mathbf{A}_i$ would be linearly independent. We need to show that dropping any row of $\mathbf{A}_i$ results in a set of linearly independent vectors. So let us, without loss of generality, assume that $\mathcal{G}$ is connected and select any $(n-1)$ rows of $\mathbf{A}$. Suppose these are linearly dependent. Then there is a non trivial linear combination of these rows which is a zero vector. From this set of rows we omit all the rows which are being multiplied by zeros. The remaining set of rows is nonvoid. Consider the corresponding set of vertices say V_1. This set does not contain all vertices of the graph. Since the graph is connected there must be an edge e with one end point in V_1 and the other outside. The submatrix of $\mathbf{A}$ with rows V_1 has only one nonzero entry in the column e. Hence, by multiplying these rows by nonzero scalars and adding we cannot get a zero row. This contradiction shows that any $(n-1)$ rows of $\mathbf{A}$ must be linearly independent. Since the sum of rows of $\mathbf{A}$ is a zero vector, dropping one row of $\mathbf{A}$ results in a basis of $\mathcal{V}_v(\mathcal{G})$ when $\mathcal{G}$ is connected and hence any set of $(n-p)$ rows of $\mathbf{A}$ which omits one row per component of $\mathcal{G}$ is a basis of $\mathcal{V}_v(\mathcal{G})$. Hence, $r(\mathcal{V}_v(\mathcal{G})) = n - p$.

□

A **reduced incidence matrix** $\mathbf{A}_r$ of a graph $\mathcal{G}$ is obtained by omitting one row belonging to each component of $\mathcal{G}$.
We know by Theorem 3.3.4 that the reduced incidence matrix is a representative matrix for $\mathcal{V}_v(\mathcal{G})$. A standard representative matrix for $\mathcal{V}_v(\mathcal{G})$ may be built as described below.

3.3.4 Fundamental cutset matrix of a forest f

We know by Theorem 3.2.7 that there is a unique cutset of a graph $\mathcal{G}$ that intersects a forest f in an edge e. This we have called the fundamental cutset of e with respect to f and denoted it by $B(e, f)$. We assign this cutset an orientation agreeing with that of e. Let $e_1, e_2, \ldots, e_r$

be the edges in the forest f and let $\mathbf{v}_1, \ldots, \mathbf{v}_r$ be the corresponding cutset vectors. A matrix which has $\mathbf{v}_1, \ldots, \mathbf{v}_r$ as rows is called the **fundamental cutset matrix** $\mathbf{Q}_f$ of f. This matrix is unique within permutation of rows and columns. By reordering rows and columns, if required, this matrix can be cast in the form

$$\begin{array}{cc} \bar{f} & f \end{array}$$

$$\mathbf{Q}_f \equiv \left[\begin{array}{cc} \mathbf{Q}_{11} & \mathbf{I} \end{array} \right] \tag{3.4}$$

It is clear that $\mathbf{Q}_f$ has $\mid f \mid (= (n-p))$ rows which are linearly independent. Since a cutset vector is linearly dependent on the rows of the incidence matrix $\mathbf{A}$ (Theorem 3.3.2) and $r(\mathbf{A}) = n - p$, it follows that $\mathbf{Q}_f$ is a standard representative matrix for $\mathcal{V}_v(\mathcal{G})$.

Example: Consider the graph of Figure 3.10.

Let $f \equiv \{e_3\ e_4\ e_5\ e_6\ e_7\}$ and let $\bar{f} = \{e_1\ \ e_2\}$.

$$\mathbf{Q}_f = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{3.5}$$

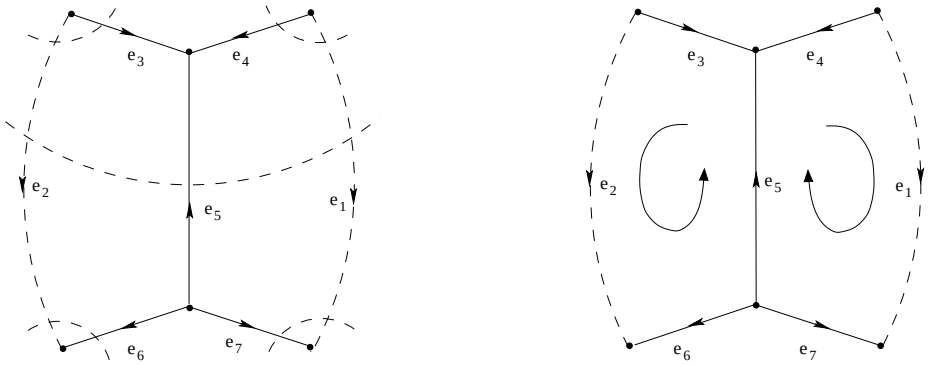


Figure 3.10: f-cutsets and f-circuits

3.3.5 Fundamental circuit matrix of a forest f

We have already seen that addition of an edge e to a forest f creates a unique circuit which we have called the fundamental circuit of e with respect to f denoted by $L(e, f)$. As before we assign this circuit an orientation agreeing with that of e. Let $e_1, \cdots, e_\nu$ be edges in the coforest $\bar{f}$. Let $\mathbf{c}_1, \cdots, \mathbf{c}_\nu$ be the corresponding circuit vectors. A matrix with these vectors as rows is called the **fundamental circuit matrix** $\mathbf{B}_f$ of f. This matrix is unique within permutation of rows and columns. By reordering rows and columns, if required, this matrix can be cast in the form

$$\mathbf{B}_f \equiv \begin{array}{c} \begin{matrix} \bar{f} & f \end{matrix} \\ [\mathbf{I} \quad \mathbf{B}_{12}] \end{array}$$

It is clear that $\mathbf{B}_f$ has $\mid \bar{f} \mid$ rows which are linearly independent. Since a circuit vector is orthogonal to all the rows of the incidence matrix, it must be a current vector. Thus rows of $\mathbf{B}_f$ are current vectors.
Example: Consider the graph in Figure 3.10. Here $f \equiv \{e_3,\ e_4,\ e_5,\ e_6,\ e_7\}$ and $\bar{f} \equiv \{e_1,\ e_2\}$.

$$\mathbf{B}_f = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{matrix} \\ \left[\begin{array}{ccccccc} 1 & 0 & 0 & -1 & +1 & 0 & -1 \\ 0 & 1 & -1 & 0 & +1 & -1 & 0 \end{array} \right] \end{array}. \tag{3.6}$$

Theorem 3.3.5 *(k) Let $\mathcal{G}$ be a graph on e edges, n nodes and p connected components. Then*
(a) $r(\mathcal{V}_i(\mathcal{G})) = e - n + p$
(b) (**Tellegen's Theorem (strong form)***) $(\mathcal{V}_v(\mathcal{G}))^\perp = \mathcal{V}_i(\mathcal{G})$.*

Proof : The rows of a fundamental circuit matrix are current vectors and $e - n + p$ in number. Hence, $r(\mathcal{V}_i(\mathcal{G})) \geq e - n + p$.
On the other hand every voltage vector is orthogonal to every current vector since a voltage vector is linearly dependent on the rows of $\mathbf{A}$ while a current vector is orthogonal to these rows. Thus, $(\mathcal{V}_v(\mathcal{G}))^\perp \supseteq \mathcal{V}_i(\mathcal{G})$.
By Theorem 2.2.5, $r(\mathcal{V}_v(\mathcal{G})) + r(\mathcal{V}_v(\mathcal{G}))^\perp = e$
We have already seen that $r(\mathcal{V}_v(\mathcal{G})) = n - p$. Hence $r(\mathcal{V}_v(\mathcal{G}))^\perp = e-n+p$ and $r(\mathcal{V}_i(\mathcal{G})) \leq e-n+p$. We conclude that $r(\mathcal{V}_i(\mathcal{G})) = e-n+p$ and thus $\mathcal{V}_i(\mathcal{G}) = (\mathcal{V}_v(\mathcal{G}))^\perp$.

□

Corollary 3.3.1 *(k) The rows of an f-circuit matrix of a graph $\mathcal{G}$ form a basis for the current space of $\mathcal{G}$.*

Exercise 3.38 *(k) Examine which potential vectors correspond to a zero voltage vector.*

Exercise 3.39 *Consider the column space $\mathcal{C}(\mathbf{A})$ of $\mathbf{A}$. Show that $(\mathcal{C}(\mathbf{A}))^\perp$ is one dimensional if the graph is connected. Hence show that $r(\mathbf{A}) = n - 1$.*

Exercise 3.40 *(k) The following is another proof for '$r(\mathbf{A}) = n-1$ if the graph is connected'. If the graph is connected $r(\mathbf{A}) \leq n - 1$ since the sum of the rows is zero. But $\mathbf{Q}_f$ has $n-1$ independent rows which are linear combinations of rows of $\mathbf{A}$. Hence $r(\mathbf{A}) = n - 1$.*

Exercise 3.41 *An* **elementary vector** *of a vector space is a nonzero vector with minimal support (subset on which it takes nonzero values). Prove*

Theorem 3.3.6 *(k) The circuit vector (cutset vector) is an elementary current vector (elementary voltage vector) and every elementary current vector (elementary voltage vector) is a scalar multiple of a circuit vector (cutset vector).*

Exercise 3.42 *Prove*

Theorem 3.3.7 *(k) A set of columns of* $\mathbf{A}$ *is linearly independent iff the corresponding edges of the graph do not contain a circuit. A set of columns of* $\mathbf{B}_f$ *is linearly independent iff the corresponding edges of the graph do not contain a cutset.*

Exercise 3.43 *(k) Every standard representative matrix of* $\mathcal{V}_v(\mathcal{G})$ *(standard representative matrix of* $\mathcal{V}_i(\mathcal{G})$*) is a fundamental cutset (fundamental circuit) matrix of* $\mathcal{G}$*.*

Exercise 3.44 An alternative proof of the strong form of Tellegen's Theorem:
(k) Let $\mathbf{B}_f$, $\mathbf{Q}_f$ *be the f-circuit and f-cutset matrix with respect to the same forest. Prove:*

i. $\mathbf{B}_f^T\mathbf{Q}_f = \mathbf{0}$

ii. If $\mathbf{B}_f = [\mathbf{I}\ \mathbf{B_{12}}]$ *then* $\mathbf{Q}_f = [-\mathbf{B_{12}^T}\ \mathbf{I}]$*. (Note that this implies Theorem 3.2.8).*

iii. Rows of $\mathbf{B}_f$, $\mathbf{Q}_f$ *are current vectors (voltage vectors). Their ranks add upto* $e(=|\ E(\mathcal{G})\ |)$*. Hence,* $(\mathcal{V}_i(\mathcal{G}))^{\perp} = \mathcal{V}_v(\mathcal{G})$*.*

Exercise 3.45 *(k) Prove*

Theorem 3.3.8 *(k) The maximum number of independent KVE for a graph is* $r(\mathcal{V}_i(\mathcal{G}))$ $(= e - n + p)$*.*

3.4 Basic Operations on Graphs and Vector Spaces

In this section, we discuss basic operations on graphs (directed and undirected) which correspond to open circuiting some edges and short circuiting some others. These operations are related to two vector

space operations: restriction and contraction. Since real vector spaces are associated primarily with directed graphs, henceforth we deal only with such graphs, but, omit the adjective 'directed'.

3.4.1 Restriction and Contraction of Graphs

Let $\mathcal{G}$ be a graph on the set of edges E and let $T \subseteq E$.

Definition 3.4.1 *The graph* $\mathcal{G}$**open**$(\mathbf{E-T})$ *is the subgraph of* $\mathcal{G}$ *with* T *as the edge set and* $V(\mathcal{G})$ *as the vertex set. Thus* $\mathcal{G}open(E-T)$ *is obtained by removing (deleting) edges in* $E-T$ *leaving their end points in place.*
The **restriction** *of* $\mathcal{G}$ *to* T*, denoted by* $\mathcal{G}\cdot\mathbf{T}$*, is the subgraph of* $\mathcal{G}$ *obtained by deleting isolated vertices from* $\mathcal{G}open(E-T)$*. Thus,* $\mathcal{G}\cdot T$ *is the subgraph of* $\mathcal{G}$ *on* T*.*
If $\mathcal{G}$ *is directed,* $\mathcal{G}open(E-T), \mathcal{G}\cdot T$*, would be directed with edges retaining the original directions they had in* $\mathcal{G}$*.*

Definition 3.4.2 *The graph* $\mathcal{G}$**short** $(\mathbf{E-T})$ *is built by first building* $\mathcal{G}openT$*. Let* $V_1, \cdots, V_k$ *be the vertex sets of the connected components of* $\mathcal{G}openT$*. The set* $\{V_1, \cdots, V_k\}$ *is the vertex set and* T *is the edge set of* $\mathcal{G}short\ (E-T)$*. An edge* $e \in T$ *would have* V_i, V_j *as its end points in* $\mathcal{G}short\ (E-T)$ *iff the end points of* e *in* $\mathcal{G}$ *lie in* V_i, V_j*. If* $\mathcal{G}$ *is directed,* V_i, V_j *would be the positive and negative endpoints of* e *in* $\mathcal{G}short\ (E-T)$ *provided the positive and negative end points of* e *in* $\mathcal{G}$ *lie in* V_i*,* V_j *respectively.*
(Thus, $\mathcal{G}short\ (E-T)$ *is obtained from* $\mathcal{G}$ *by short circuiting the edges in* $(E-T)$ *(fusing their end points) and removing them).*
The **contraction** *of* $\mathcal{G}$ *to* T*, denoted by* $\mathcal{G}\times T$*, is obtained from* $\mathcal{G}short\ (E-T)$ *by deleting the isolated vertices of the latter.*

Example: Consider the graph $\mathcal{G}$ of Figure 3.11.
Let $T = \{e_1, e_6, e_{11}\}$. The graph $\mathcal{G}openT$ is shown in the figure. Graph $\mathcal{G}\cdot(E-T)$ is obtained by omitting isolated vertex v_1 from $\mathcal{G}openT$. Graph $\mathcal{G}short\ (E-T)$ is also shown in the same figure. Graph $\mathcal{G}\times T$ is obtained by omitting the isolated vertex $\{\ v_8, v_9\ \}$ from $\mathcal{G}short\ (E-T)$.

We denote $(\mathcal{G}\times T_1)\cdot T_2, T_2 \subseteq T_1 \subseteq E(\mathcal{G})$ by $\mathcal{G}\times T_1\cdot T_2$ and $(\mathcal{G}\cdot T_1)\times T_2$. $T_2 \subseteq T_1 \subseteq E(\mathcal{G})$ by $\mathcal{G}\cdot T_1 \times T_2$. Graphs denoted by such expressions are called **minors** of $\mathcal{G}$. It can be seen that when a set $A \subseteq E(\mathcal{G})$ is

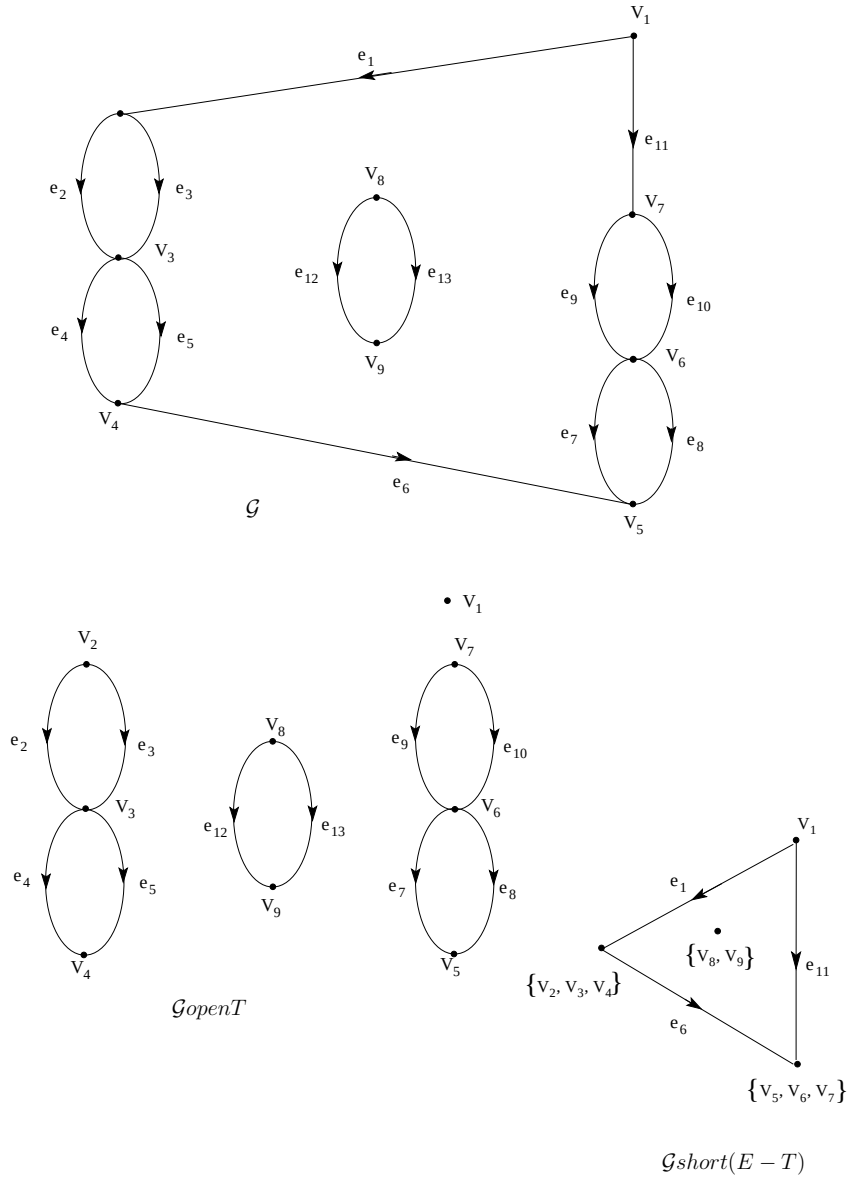


Figure 3.11: Minors of a Graph

being shorted and a disjoint set $B \subseteq E(\mathcal{G})$, is being opened then the final graph does not depend on the order in which these operations are carried out but only on the sets A and B. Now $\mathcal{G} \times T(\mathcal{G} \cdot T)$ differs from $\mathcal{G}short\ (E-T)$ $(\mathcal{G}open(E-T))$ only in that the isolated vertices are omitted. We thus have the following theorem where equality refers to isomorphism.

Theorem 3.4.1 *(k) Let $\mathcal{G}$ be a graph with $T_2 \subseteq T_1 \subseteq E(G)$. Then*

i. $\mathcal{G} \times T_1 \times T_2 = \mathcal{G} \times T_2$,

ii. $\mathcal{G} \cdot T_1 \cdot T_2 = \mathcal{G} \cdot T_2$,

iii. $\mathcal{G} \times T_1 \cdot T_2 = \mathcal{G} \cdot (E - (T_1 - T_2)) \times T_2$.

Proof : The theorem is immediate when we note that both graphs are obtained by shorting and opening the same sets. In (i) $E - T_2$ is shorted while in (ii) $E - T_2$ is opened. In (iii) $E - T_1$ is shorted and $T_1 - T_2$ is opened.

□

Exercise 3.46 *(k)* **Simplification of Expression for minors:** *Show that any minor of the form $\mathcal{G} \times T_1 \cdot T_2 \times T_3 \ldots T_n, T_1 \supseteq T_2 \supseteq \ldots \supseteq T_n$*
(the graph being obtained by starting from $\mathcal{G}$ and performing the operations from left to right in succession), can be simplified to a minor of the form
$\mathcal{G} \cdot T' \times T_n$ *or* $\mathcal{G} \times T' \cdot T_n$.

Exercise 3.47 *Train yourself to visualize $\mathcal{G}_1 \equiv \mathcal{G}short\ (E-T)$ (Put components of $\mathcal{G}openT$ inside surfaces which then become nodes of $\mathcal{G}_1$). How many components does it have? When would a branch of $\mathcal{G}$ become a selfloop of $\mathcal{G}_1$? When would a circuit free set of branches of $\mathcal{G}$ become dependent in $\mathcal{G}_1$?*

Exercise 3.48 Circuits of minors: *Prove*

Lemma 3.4.1 *(k)*

i. A subset C of T is a circuit of $\mathcal{G} \cdot T$ iff C is a circuit of $\mathcal{G}$.

ii. A subset C of T is circuit of $\mathcal{G} \times T$ iff C is a minimal intersection of circuits of $\mathcal{G}$ with T (equivalently,iff C is an intersection of a circuit of $\mathcal{G}$ with T but no proper subset of C is such an intersection).

Exercise 3.49 *(k)* **Cutsets of minors:** *Prove*

Lemma 3.4.2 *(k)*

i. A subset B of T is a cutset of $\mathcal{G} \cdot T$ iff it is a minimal intersection of cutsets of $\mathcal{G}$ with T.

ii. A subset B of T is a cutset of $\mathcal{G} \times T$ iff it is a cutset of $\mathcal{G}$.

3.4.2 Restriction and Contraction of Vector Spaces

We now describe operations on vector spaces which are analogous to the operations of opening and shorting edges in a graph.
Let $\mathcal{V}$ be a vector space on S and let $T \subseteq S$.

Definition 3.4.3 *The* **restriction** *of $\mathcal{V}$ to T, denoted by $\mathcal{V}.T$, is the collection of vectors $\mathbf{f}_T$ where $\mathbf{f}_T$ is the restriction of some vector $\mathbf{f}$ of $\mathcal{V}$ to T.*
The **contraction** *of $\mathcal{V}$ to T, denoted by $\mathcal{V} \times T$, is the collection of vectors $\mathbf{f'}_T$ where $\mathbf{f'}_T$ is the restriction to T of some vector $\mathbf{f}$ of $\mathcal{V}$ such that $\mathbf{f}/(S-T) = \mathbf{0}$.*

It is easily seen that $\mathcal{V} \cdot T$, $\mathcal{V} \times T$ are vector spaces.

As in the case of graphs we denote $(\mathcal{V} \times T_1) \cdot T_2$ by $\mathcal{V} \times T_1 \cdot T_2$. Such expressions denote vector spaces which are called **minors** of $\mathcal{V}$. To bring out the analogy between graph minor and vector space minor operations we say we 'open' T when we restrict $\mathcal{V}$ to $(S-T)$ and say we 'short' T when we contract $\mathcal{V}$ to $(S-T)$.
It turns out that the order in which we open and short disjoint sets of elements is unimportant. More formally we have

Theorem 3.4.2 *(k) Let $T_2 \subseteq T_1 \subseteq S$. Then*

i. $\mathcal{V} \cdot T_1 \cdot T_2 = \mathcal{V} \cdot T_2$,

ii. $\mathcal{V} \times T_1 \times T_2 = \mathcal{V} \times T_2$,

iii. $\mathcal{V} \times T_1 \cdot T_2 = \mathcal{V} \cdot (S - (T_1 - T_2)) \times T_2$.

Proof of (iii): We show that a vector in the LHS belongs to a vector in the RHS.
Let $\mathbf{f}_{T_2} \in \mathcal{V} \times T_1 \cdot T_2$.
Then there exists a vector $\mathbf{f}_{T_1} \in \mathcal{V} \times T_1$ such that $\mathbf{f}_{T_1}/T_2 = \mathbf{f}_{T_2}$ and a vector $\mathbf{f} \in \mathcal{V}$ with $\mathbf{f}/(S - T_1) = \mathbf{0}$ such that $\mathbf{f}/T_1 = \mathbf{f}_{T_1}$.
Now let $\mathbf{f}'$ denote $\mathbf{f}/(S - (T_1 - T_2))$.

Clearly $\mathbf{f}' \in \mathcal{V} \cdot (S - (T_1 - T_2))$. Now $\mathbf{f}'/(S - T_1) = \mathbf{0}$.
Hence, $\mathbf{f}'/T_2 \in \mathcal{V} \cdot (S - (T_1 - T_2)) \times T_2$.
Thus, $\mathcal{V} \times T_1 \cdot T_2 \subseteq \mathcal{V} \cdot (S - (T_1 - T_2)) \times T_2$.
The reverse containment is similarly proved.

□

Remark: To see the proof of the above theorem quickly, observe that a typical vector of both LHS and RHS is obtained by restricting a vector of $\mathcal{V}$, that takes zero value on $S - T_1$, to T_2.

Exercise 3.50 *(k) Prove:*
Any minor of the form $\mathcal{V} \times T_1 \cdot T_2 \times T_3 \ldots T_n, T_1 \supseteq T_2 \supseteq \ldots \supseteq T_n$, *can be simplified to a minor of the form*

$$\mathcal{V} \cdot T' \times T_n \text{ or } \mathcal{V} \times T' \cdot T_n.$$

3.4.3 Vector Space Duality

We now relate the minors of $\mathcal{V}$ to the minors of $\mathcal{V}^\perp$. We remind the reader that $\hat{\mathcal{V}}^\perp$, the complementary orthogonal space of $\hat{\mathcal{V}}$ is defined to be on the same set as $\hat{\mathcal{V}}$. In the following results we see that the contraction (restriction) of a vector space corresponds to the restriction (contraction) of the orthogonal complement. We say that contraction and restriction are **(orthogonal) duals** of each other.

Theorem 3.4.3 *(k) Let* $\mathcal{V}$ *be a vector space on* S *and let* $T \subseteq S$. *Then,*

i. $(\mathcal{V} \cdot T)^\perp = \mathcal{V}^\perp \times T$.

ii. $(\mathcal{V} \times T)^\perp = \mathcal{V}^\perp \cdot T$.

Proof :
i. Let $\mathbf{g}_T \in (\mathcal{V} \cdot T)^\perp$. For any $\mathbf{f}$ on S let $\mathbf{f}_T$ denote $\mathbf{f}/T$. Now if $\mathbf{f} \in \mathcal{V}$, then $\mathbf{f}_T \in \mathcal{V} \cdot T$ and $< \mathbf{g}_T, \mathbf{f}_T > = 0$.
Let $\mathbf{g}$ on S be defined by $\mathbf{g}/T \equiv \mathbf{g}_T$, $\mathbf{g}/S - T \equiv \mathbf{0}$. If $\mathbf{f} \in \mathcal{V}$ we have

$$\begin{aligned} < \mathbf{f}, \mathbf{g} > &= < \mathbf{f}_T, \mathbf{g}_T > + < \mathbf{f}_{S-T}, \mathbf{g}_{S-T} > \\ &= 0 + < \mathbf{f}_{S-T}, \mathbf{0}_{S-T} > \\ &= 0. \end{aligned}$$

Thus $\mathbf{g} \in \mathcal{V}^{\perp}$ and therefore, $\mathbf{g}_T \in \mathcal{V}^{\perp} \times T$.Hence, $(\mathcal{V} \ . \ T)^{\perp} \subseteq \mathcal{V}^{\perp} \times T$.
Next let $\mathbf{g}_T \in \mathcal{V}^{\perp} \times T$.
Then there exists $\mathbf{g} \in \mathcal{V}^{\perp}$ s.t. $\mathbf{g}/S - T = \mathbf{0}$ and $\mathbf{g}/T = \mathbf{g}_T$.
Let $\mathbf{f}_T \in \mathcal{V} \cdot T$. There exists $\mathbf{f} \in \mathcal{V}$ s.t. $\mathbf{f}/T = \mathbf{f}_T$.
Now $0 =< \mathbf{f}, \mathbf{g} >=< \mathbf{f}_T, \mathbf{g}_T > + < \mathbf{f}_{S-T}, \mathbf{0}_{S-T} > =< \mathbf{f}_T, \mathbf{g}_T >$.
Hence, $\mathbf{g}_T \in (\mathcal{V} \cdot T)^{\perp}$.
We conclude that
$\mathcal{V}^{\perp} \times T \subseteq (\mathcal{V} \cdot T)^{\perp}$. This proves that $(\mathcal{V} \cdot T)^{\perp} = \mathcal{V}^{\perp} \times T$.

ii. We have $(\mathcal{V}^{\perp} \cdot T)^{\perp} = (\mathcal{V}^{\perp})^{\perp} \times T$.
By Theorem 2.2.5
$((\mathcal{V}^{\perp} \cdot T)^{\perp})^{\perp} = \mathcal{V}^{\perp} \cdot T$ and $(\mathcal{V}^{\perp})^{\perp} = \mathcal{V}$. Hence, $\mathcal{V}^{\perp} \cdot T = (\mathcal{V} \times T)^{\perp}$.

□

The following corollary is immediate.

Corollary 3.4.1 *(k)* $(\mathcal{V} \times P \cdot T)^{\perp} = \mathcal{V}^{\perp} \cdot P \times T, T \subseteq P \subseteq S$.

3.4.4 Relation between Graph Minors and Vector Space Minors

We now show that the analogy between vector space minors and graph minors is more substantial than hitherto indicated - in fact the minors of voltage and current spaces of a graph correspond to appropriate graph minors.

Theorem 3.4.4 *(k) Let $\mathcal{G}$ be a graph with edge set E. Let $T \subseteq E$. Then*

i. $\mathcal{V}_v(\mathcal{G} \cdot T) = (\mathcal{V}_v(\mathcal{G})) \cdot T$

ii. $\mathcal{V}_v(\mathcal{G} \times T) = (\mathcal{V}_v(\mathcal{G})) \times T$

Proof : We remind the reader that by definition a voltage vector $\mathbf{v}$ is a linear combination of the rows of the incidence matrix, the coefficients of the linear combination being given by the entries in a potential vector λ. We say $\mathbf{v}$ is derived from λ.
i. Let $\mathbf{v}_T \in \mathcal{V}_v(\mathcal{G} \cdot T)$
Now $\mathcal{V}_v(\mathcal{G} \cdot T) = \mathcal{V}_v(\mathcal{G}open(E - T))$.

Thus, $\mathbf{v}_T \in \mathcal{V}_v(\mathcal{G}open(E-T))$. The graph $\mathcal{G}open(E-T)$ has the same vertex set as $\mathcal{G}$ but the edges of $(E-T)$ have been removed.
Let $\mathbf{v}_T$ be derived from the potential vector λ of $\mathcal{G}open(E-T)$. Now for any edge $e \in T$, $\mathbf{v}_T(e) = \lambda(a) - \lambda(b)$, where a, b are the positive and negative end points of e. However, λ is also a potential vector of $\mathcal{G}$. Let the voltage vector $\mathbf{v}$ of $\mathcal{G}$ be derived from λ. For the edge $e \in T$, we have, as before, $\mathbf{v}(e) = \lambda(a) - \lambda(b)$. Thus, $\mathbf{v}_T = \mathbf{v}/T$ and therefore, $\mathbf{v}_T \in (\mathcal{V}_v(\mathcal{G})) \cdot T$. Hence $\mathcal{V}_v(\mathcal{G} \cdot T) \subseteq (\mathcal{V}_v(\mathcal{G})) \cdot T$.
The reverse containment is proved similarly.

ii. Let $\mathbf{v}_T \in \mathcal{V}_v(\mathcal{G} \times T)$. Now $\mathcal{V}_v(\mathcal{G} \times T) = \mathcal{V}_v(\mathcal{G}short\ (E-T))$.
Thus, $\mathbf{v}_T \in \mathcal{V}_v(\mathcal{G}short\ (E-T))$.
The vertex set of $\mathcal{G}short\ (E-T)$ is the set $\{V_1, V_2, \ldots V_n\}$ where V_i is the vertex set of the i^{th} component of $\mathcal{G}openT$. Let $\mathbf{v}_T$ be derived from the potential vector $\hat{\lambda}$ in $\mathcal{G}short\ (E-T)$. The vector $\hat{\lambda}$ assigns to each of the V_i the value $\hat{\lambda}(V_i)$. Now define a potential vector λ on the nodes of $\mathcal{G}$ as follows: $\lambda(n) \equiv \hat{\lambda}(V_i), n \in V_i$. Since $\{V_1, \ldots V_k\}$ is a partition of $V(\mathcal{G})$, it is clear that λ is well defined. Let $\mathbf{v}$ be the voltage vector derived from λ in $\mathcal{G}$. Whenever $e \in E-T$ we must have $\mathbf{v}(e) = 0$ since both end points must belong to the same V_i.
Next, whenever $e \in T$ we have $\mathbf{v}(e) = \lambda(a) - \lambda(b)$ where a is the positive end point of e and b, the negative endpoint. Let $a \in V_a$, $b \in V_b$,where $V_a, V_b \in V(\mathcal{G}short\ (E-T))$. Then the positive endpoint of e in $\mathcal{G}short\ (E-T)$ is V_a and the negative end point, V_b.
By definition $\lambda(a) - \lambda(b) = \hat{\lambda}(V_a) - \hat{\lambda}(V_b)$.Thus $\mathbf{v}/T = \mathbf{v}_T$. Hence, $\mathbf{v}_T \in (\mathcal{V}_v(\mathcal{G})) \times T$. Thus, $\mathcal{V}_v(\mathcal{G} \times T) \subseteq (\mathcal{V}_v(\mathcal{G})) \times T$. The reverse containment is proved similarly.

□

Using duality we can now prove

Theorem 3.4.5 *(k) Let $\mathcal{G}$ be a directed graph on edge set E. Let $T \subseteq E$. Then,*

i. $\mathcal{V}_i(\mathcal{G}\ .\ T) = (\mathcal{V}_i(\mathcal{G})) \times T$.

ii. $\mathcal{V}_i(\mathcal{G} \times T) = (\mathcal{V}_i(\mathcal{G})) \cdot T$.

Proof :
i. $\mathcal{V}_i(\mathcal{G}\ .\ T) = (\mathcal{V}_v(\mathcal{G}\ .\ T))^{\perp}$ by the strong form of Tellegen's Theorem.

By Theorem 3.4.4, $\mathcal{V}_v(\mathcal{G} \,.\, T) = (\mathcal{V}_v(\mathcal{G})) \cdot T$.
Hence,

$$\begin{aligned} \mathcal{V}_i(\mathcal{G} \,.\, T) &= ((\mathcal{V}_v(\mathcal{G})) \cdot T)^{\perp} \\ &= (\mathcal{V}_v(\mathcal{G}))^{\perp} \times T \\ &= \mathcal{V}_i(\mathcal{G}) \times T. \end{aligned}$$

ii. The proof is similar.

□

Exercise 3.51 *(k) For a connected directed graph $\mathcal{G}$ on node set $\{v_1, \ldots, v_k\}$ if currents $J_1, J_2 \ldots, J_k$ enter nodes $v_1, v_2, \ldots, v_k$ show that there exists a vector* $\mathbf{i}$ *on $E(\mathcal{G})$, s.t.* $\mathbf{Ai} = \mathbf{J}$ *iff $\Sigma J_i = 0$.*

Exercise 3.52 *Prove Theorem 3.4.5 directly. (Hint: the result of the preceding exercise would be useful in extending a current vector of $\mathcal{G} \times T$ to a current vector of $\mathcal{G}$).*

3.4.5 Representative Matrices of Minors

As defined earlier, the **representative matrix R** of a vector space $\mathcal{V}$ on S has the vectors of a basis of $\mathcal{V}$ as its rows. Often the choice of a suitable representative matrix would give us special advantages. We describe how to construct a representative matrix which contains representative matrices of $\mathcal{V} \,.\, T$ and $\mathcal{V} \times (S - T)$ as its submatrices. We say in such a case that $\mathcal{V} \,.\, T$ and $\mathcal{V} \times (S - T)$ become 'visible' in $\mathbf{R}$.

Theorem 3.4.6 *(k) Let $\mathcal{V}$ be a vector space on S. Let $T \subseteq S$. Let* $\mathbf{R}$ *be a representative matrix as shown below*

$$\begin{matrix} & T & S-T \end{matrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{TT} & \mathbf{R}_{T2} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \tag{3.7}$$

where the rows of $\mathbf{R}_{TT}$ are linearly independent. Then $\mathbf{R}_{TT}$ is a representative matrix for $\mathcal{V} \,.\, T$ and $\mathbf{R}_{22}$, a representative matrix for $\mathcal{V} \times (S - T)$.

Proof : The rows of $\mathbf{R}_{TT}$ are restrictions of vectors on S to T. Hence, any linear combination of these rows will yield a vector of $\mathcal{V}\ .\ T$. If $\mathbf{f}_T$ is any vector in $\mathcal{V}\ .\ T$ there exists a vector $\mathbf{f}$ in $\mathcal{V}$ s.t. $\mathbf{f}/T = \mathbf{f}_T$. Now $\mathbf{f}$ is a linear combination of the rows of $\mathbf{R}$. Hence, $\mathbf{f}/T(= \mathbf{f}_T)$ is a linear combination of the rows of $\mathbf{R}_{TT}$. Further it is given that the rows of $\mathbf{R}_{TT}$ are linearly independent. It follows that $\mathbf{R}_{TT}$ is a representative matrix of $\mathcal{V}\ .\ T$.
It is clear from the structure of $\mathbf{R}$ (the zero in the second set of rows) that any linear combination of the rows of $\mathbf{R}_{22}$ belongs to $\mathcal{V} \times (S - T)$. Further if $\mathbf{f}$ is any vector in $\mathcal{V}$ s.t. $\mathbf{f}/T = \mathbf{0}$ then $\mathbf{f}$ must be a linear combination only of the second set of rows of $\mathbf{R}$. For, if the first set of rows are involved in the linear combination, since rows of $\mathbf{R}_{TT}$ are linearly independent, $\mathbf{f}/T$ cannot be zero. We conclude that if $\mathbf{f}/(S-T)$ is a vector in $\mathcal{V} \times (S - T)$, it is linearly dependent on the rows of $\mathbf{R}_{22}$. Now rows of $\mathbf{R}$ are linearly independent. We conclude that $\mathbf{R}_{22}$ is a representative matrix of $\mathcal{V} \times T$.

□

Remark: To build a representative matrix of $\mathcal{V}$ with the form as in Theorem 3.4.6, we start from any representative matrix of $\mathcal{V}$ and perform row operations on it so that under the columns T we have a matrix in the RRE form.

The following corollary is immediate

Corollary 3.4.2 *(k)*

$$r(\mathcal{V}) = r(\mathcal{V}\ .\ T) + r(\mathcal{V} \times (S - T))\ , T \subseteq S$$

Corollary 3.4.3 *(k) Let $\mathcal{G}$ be a graph on E. Then*

$$r(\mathcal{G}) = r(\mathcal{G}\ .\ T) + r(\mathcal{G} \times (E - T))\ , \forall\ T \subseteq E$$

Proof : We observe that $r(\mathcal{G})$ = number of edges in a forest of $\mathcal{G} = r(\mathcal{V}_v(\mathcal{G}))$. The result follows by Theorem 3.4.4.

□

In the representative matrix of Theorem 3.4.6 the submatrix $\mathbf{R}_{T2}$ contains information about how $T, S - T$ are linked by $\mathcal{V}$. If $\mathbf{R}_{T2}$ is a zero matrix then it is clear that $\mathcal{V} = \mathcal{V}_T \oplus \mathcal{V}_{S-T}$ where $\mathcal{V}_T, \mathcal{V}_{S-T}$ are vector spaces on $T, S - T$.

Definition 3.4.4 *A subset T of S is a* **separator** *of $\mathcal{V}$ iff $\mathcal{V} \times T = \mathcal{V} \,.\, T$.*

It is immediate that if T is a separator so is $(S - T)$. Thus, we might say that $T, (S - T)$ are decoupled in this case. Now by definition $\mathcal{V} \,.\, T \supseteq \mathcal{V} \times T$. Hence, equality of the spaces follows if their dimensions are the same. Hence, T is a separator iff $r(\mathcal{V} \times T) = r(\mathcal{V} \,.\, T)$.
The connectivity of $\mathcal{V}$ at T is denoted by $\xi(T)$ and defined as follows:

$$\xi(T) \equiv r(\mathcal{V} \,.\, T) - r(\mathcal{V} \times T)$$

It is easily seen that $\xi(T) = \xi(S - T)$. Further, this number is zero if T is a separator.

Exercise 3.53 *(k)*

i. Let

$$\begin{array}{c} \quad T_1 \quad T_2 \quad T_3 \\ \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \mathbf{R}_{13} \\ \mathbf{R}_{21} & \mathbf{0} & \mathbf{R}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{33} \end{bmatrix} \end{array} \tag{3.8}$$

Rows of $\mathbf{R}_{12}$ and $\begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{R}_{21} \end{bmatrix}$ are given to be linearly independent. Show that $\mathbf{R}_{33}$ is a representative matrix of $\mathcal{V} \times T_3$, $\mathbf{R}_{12}$ of $\mathcal{V} \,.\, T_2$, $\mathbf{R}_{21}$ of $\mathcal{V} \,.\, (T_1 \cup T_2) \times T_1$ as well as $\mathcal{V} \times (T_1 \cup T_3)$. T_1 (and hence these spaces must be the same).

ii. How would $\mathbf{R}$ look if $\mathcal{V} \,.\, (T_1 \cup T_2)$ has T_1, T_2 as separators?

Exercise 3.54 *(k) Let*

$$\begin{array}{c} \quad T_1 \quad T_2 \\ \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \\ \mathbf{0} & \mathbf{R}_{33} \end{bmatrix}. \end{array} \tag{3.9}$$

Suppose rows of

$\begin{pmatrix} \mathbf{R}_{11} \\ \mathbf{R}_{21} \end{pmatrix}, \begin{pmatrix} \mathbf{R}_{22} \\ \mathbf{R}_{33} \end{pmatrix}$, *are linearly independent. Show that the number of rows of $\mathbf{R}_{22} = r(\mathcal{V} \,.\, T_2) - r(\mathcal{V} \times T_2)$ $(= r(\mathcal{V} \,.\, T_1) - r(\mathcal{V} \times T_1))$.*

Exercise 3.55 *(k) Prove:*
Let $\xi'(\cdot)$ be the $\xi(\cdot)$ function for $\mathcal{V}^{\perp}$. Then $\xi'(T) = \xi(T)$, $\forall\ T \subseteq S$.

Exercise 3.56 *(k) Show that the union of a forest of $\mathcal{G} \times T$ and a forest of $\mathcal{G}\ .\ (E-T)$ is a forest of $\mathcal{G}$. Hence, (Corollary 3.4.3) $r(\mathcal{G} \times T) + r(\mathcal{G}\ .\ (E-T)) = r(\mathcal{G})$.*

Exercise 3.57 *(k) Prove:*
$\nu(\mathcal{G}\ .\ T) + \nu(\mathcal{G} \times (S-T) = \nu(\mathcal{G})$.

Exercise 3.58 *(k) Prove:*
Let $\mathcal{G}$ be a graph on E. Then $T \subseteq E$ is a **separator** *of $\mathcal{G}$ (i.e., no circuit intersects both T and $E-T$ (Subsection 3.2.9) iff T is a separator of $\mathcal{V}_v(\mathcal{G})$. Hence, T is a separator of $\mathcal{G}$ iff $r(\mathcal{G}\ .\ T) = r(\mathcal{G} \times T)$.*

Exercise 3.59 *Let T be a separator of $\mathcal{G}$. Let $\mathcal{G}\ .\ T, \mathcal{G}\ .\ (E-T)$ have α_1, α_2 forests respectively, β_1, β_2 circuits respectively and γ_1, γ_2 cutsets respectively. How many forests, coforests, circuits and cutsets does $\mathcal{G}$ have?*

3.4.6 Minty's Theorem

Tellegen's Theorem is generally regarded as the most fundamental result in Electrical Network Theory. There is however, another fundamental result which can be proved to be formally equivalent to Tellegen's Theorem [Narayanan85c] and whose utility is comparable to the latter. This is Minty's Theorem (strong form) [Minty60], which we state and prove below.

Theorem 3.4.7 *(***Minty's Theorem (strong form)***) Let $\mathcal{G}$ be a directed graph.*
Let $E(\mathcal{G})$ be partitioned into red,blue and green edges. Let e be a green edge.
Then e **either** *belongs to a circuit containing only blue and green edges with all green edges of the same direction with respect to the orientation of the circuit* **or** *e belongs to a cutset containing only red and green edges with all green edges of the same direction with respect to the orientation of the cutset but* **not both**.

Proof: We first prove the weak form:

'in a graph each edge is present in a directed circuit or in a directed cutset but not both'

Proof of weak form: We claim that a directed circuit and a directed cutset of the same graph cannot intersect. For, suppose otherwise. Let the directed cutset have the orientation (V_1, V_2). The directed circuit subgraph must necessarily have vertices in V_1 as well as in V_2 in order that the intersection be nonvoid. But if we traverse the circuit subgraph starting from the node in V_1 we would at some stage crossover into V_2 by an edge e_{12} and later return to V_1 by an edge e_{21}. Now e_{12}, e_{21} have the same orientation with respect to the circuit which means that if one of them has positive end point in V_1 and negative end point in V_2 the other must have the positive and negative end points in V_2, V_1, respectively. But this contradicts the fact that they both belong to the same directed cutset with orientation (V_1, V_2).

Next we show that any edge e must belong either to a directed circuit or to a directed cutset. To see this, start from the negative end point n_2 of the edge and reach as many nodes of the graph as possible through directed paths. If through one of these paths we reach the positive end point n_1 of e we can complete the directed circuit using e. Suppose n_1 is not reachable through directed paths from n_2. Let the set of all nodes reachable by directed paths from n_2 be enclosed in a surface. This surface cannot contain n_1 and has at least one edge, namely e with one end inside the surface and one outside. It is clear that all such edges must be directed into the surface as otherwise the surface can be enlarged by including more reachable nodes. This collection of edges is a directed crossing edge set and contains a directed cutset which has e as a member (see Exercise 3.60). This completes the proof of the weak form.

Proof of strong form: We open the red edges r and short the blue edges b to obtain from $\mathcal{G}$, the graph $\mathcal{G}_g$ on the green edge set g ,i.e., $\mathcal{G}_g = \mathcal{G} \times (E(\mathcal{G}) - b) \cdot g$. In this graph the weak form holds. Suppose the edge e is part of a directed cutset in $\mathcal{G}_g$. Then this is still a directed cutset containing only green edges in $\mathcal{G} \cdot (E(\mathcal{G}) - r)$. (By Lemma 3.4.2, a set $C \subseteq T \subseteq E(\mathcal{G})$ is a cutset of $\mathcal{G} \times T$ iff it is a cutset of $\mathcal{G}$). It would be a part of a red and green cutset in $\mathcal{G}$ when red edges are introduced between existing nodes. On the other hand, suppose the edge e is part of a directed circuit in $\mathcal{G}_g$. Then this is still a directed

circuit containing only green edges in $\mathcal{G} \times (E(\mathcal{G}) - b)$. (By Lemma 3.4.1, a set $C \subseteq T \subseteq E(\mathcal{G})$ is a circuit of $\mathcal{G} \cdot T$ iff it is a circuit of $\mathcal{G}$). It would be a part of a blue and green circuit in $\mathcal{G}$ when blue edges are introduced by splitting existing nodes.
Thus, the strong form is proved.

□

Exercise 3.60 *(k) Let e be a member of a directed crossing edge set C. Show that there exists a directed cutset C_1 s.t. $e \in C_1 \subseteq C$.*

Exercise 3.61 *(k)* **A Generalization:** *Prove:*
Let $\mathcal{V}$ be a vector space on S over the real field and let $e \in S$. Then e is in the support of a nonzero nonnegative vector $\mathbf{f}$ in $\mathcal{V}$ or in the support of a nonzero nonnegative vector $\mathbf{g}$ in $\mathcal{V}^\perp$ but not in both.

Exercise 3.62 *(k)* **Partition into strongly connected components:** *Prove:*
The edges of a directed graph can be partitioned into two sets - those that can be included in directed circuits and those which can be included in directed cutsets.

i. *Hence show that*
the vertex set of a directed graph can be partitioned into blocks so that any pair of vertices in each block are reachable from each other; partial order can be imposed on the blocks s.t. $B_i \geq B_j$ iff a vertex of B_j can be reached from a vertex of B_i.

ii. *Give a good algorithm for building the partition as well as the partial order.*

3.5 Problems

Problems on Graphs

Problem 3.1 *(k) If a graph has no odd degree vertices,then it is possible to start from any vertex and travel along all edges without repeating any edge and to return to the starting vertex. (Repetition of nodes is allowed).*

Problem 3.2 *(k) Any graph on 6 nodes has either 3 nodes which are pairwise adjacent or 3 nodes which are pairwise non-adjacent.*

Problem 3.3 *(k) A graph is made up of parallel but oppositely directed edges only. Let $T, E-T$ be a partition of the edges of $\mathcal{G}$ such that*

i. *if $e \in T$ then the parallel oppositely directed edge $e' \in T$.*

ii. *it is possible to remove from each parallel pair of edges in $T(E-T)$ one of the edges so that the graph is still strongly connected.*

Show that it is possible to remove one edge from each parallel pair of edges in $\mathcal{G}$ so that the graph remains strongly connected.

Problem 3.4 *(k) We denote by $\mathcal{K}_n$ the graph on n nodes with a single edge between every pair of nodes and by $\mathcal{K}_{m,n}$ the bipartite graph (i.e.,no edges between left vertices and no edges between right vertices) on m left vertices and n right vertices, with edges between every pair of right and left vertices.*

i. *How many edges do $\mathcal{K}_n, \mathcal{K}_{m,n}$ have?*

ii. *Show that every circuit of $\mathcal{K}_{m,n}$ has an even number of edges.*

iii. *Show that $\mathcal{K}_n$ has n^{n-2} trees.*

iv. *A vertex colouring is an assignment of colours to vertices of the graph so that no two of them which have the same colour are adjacent. What is the minimum number of colours required for $\mathcal{K}_n, \mathcal{K}_{m,n}$?*

Problems on Circuits

Problem 3.5 *[Whitney35]* **Circuit Matroid:** *Show that the collection $\mathcal{C}$ of circuits of a graph satisfy the* **matroid circuit axioms***:*

i. *If $C_1, C_2 \in \mathcal{C}$ then C_1 cannot properly contain C_2.*

ii. *If $e_c \in C_1 \cap C_2, e_d \in C_1 - C_2$, then there exists $C_3 \in \mathcal{C}$ and $C_3 \subseteq C_1 \cup C_2$ s.t. $e_c \notin C_3$ but e_d does.*

Problem 3.6 *(k)* **Circuit Characterization:**

i. *A subset of edges C is a circuit of a graph iff it is a minimal set of edges not intersecting any cutset in a single branch.*

ii. *Same as (i) except 'single branch' is replaced by 'odd number of branches'.*

iii. *C is a circuit of a graph iff it is a minimal set of branches not contained in any forest (intersecting every coforest).*

Problem 3.7 *(k)* **Cyclically Connected in terms of Edges:** *A graph in which any two vertices can be included in a circuit subgraph is said to be cyclically connected. In such a graph any two edges can also be so included.*

Problem 3.8 *(k)* **Cut Vertex:** *A graph with no coloops is cyclically connected iff it has no cut vertex (a vertex whose removal along with its incident edges disconnects the graph).*

Problems on Cutsets

Problem 3.9 *(k)* **Cutset Matroid:** *Show that the collection of cutsets of a graph satisfies the circuit axioms of a matroid.*

Problem 3.10 *(k)* **Cutset Characterization:**

i. *A subset of edges C is a cutset of a graph iff it is a minimal set of edges not intersecting any circuit in a single edge (in an odd number of edges).*

ii. *C is a cutset of a graph iff it is a minimal set of branches not contained in any coforest (intersecting every forest).*

Problem 3.11 *(k) Show that every crossing edge set is a disjoint union of cutsets.*

Problem 3.12 *(k)* **Cyclically Connected in terms of Edges in Cutsets:** *In a cyclically connected graph any two edges can be included in a cutset.*

Problems on Graphs and Vector Spaces

Problem 3.13 *(k) Show directly that KCE of a tree graph has only the trivial solution. What is the structure for which KVE has only the trivial solution?*

Problem 3.14 Rank of Incidence Matrix of a Tree Graph:
Give three proofs for 'rank of incidence matrix of a tree graph = number of edges of the graph' using

i. the determinant of a reduced incidence matrix

ii. current injection

iii. by assuming branches to be voltage sources and evaluating node potentials.

Problem 3.15 *(k)* **Nontrivial KCE Solution and Coforest:**
Prove directly that the support of every nonzero solution to KCE meets every coforest. Hence, the rows of an f-circuit matrix of $\mathcal{G}$ span $\mathcal{V}_i(\mathcal{G})$. Hence, $r(\mathcal{V}_i(\mathcal{G})) = e - (v - p)$.

Problem 3.16 *(k)* **Nontivial KVE Solution and Forest:**
Prove directly that the support of every nonzero solution to KVE meets every forest. Hence, the rows of an f-cutset matrix of $\mathcal{G}$ span $\mathcal{V}_v(\mathcal{G})$. Hence, $r(\mathcal{V}_v(\mathcal{G})) = (v - p)$.

Problem 3.17 *(k)* **Determinants of Submatrices of Incidence Matrix:**
The determinant of every submatrix of the incidence matrix $\mathbf{A}$ is 0, ± 1. Hence, this property also holds for every $\mathbf{Q}_f$ and $\mathbf{B}_f$.

Problem 3.18 Interpreting Current Equations:
Let $\mathbf{A}$ be an incidence matrix.

i. Find one solution to $\mathbf{Ax} = \mathbf{b}$, if it exists, by inspection (giving a current injection interpretation).

ii. Find one solution to $\mathbf{A^T y} = \mathbf{v}$ by inspection (using voltage sources as branches).

Problem 3.19 *i. Let* $\mathbf{A}$ *be the incidence matrix of* $\mathcal{G}$*. If* $\mathbf{Ax} = \mathbf{b}$ *is equivalent to* $\mathbf{Q}_f\mathbf{x} = \hat{\mathbf{b}}$*, relate* $\hat{\mathbf{b}}$ *to* $\mathbf{b}$*. Using current injection give a simple rule for obtaining* $\hat{\mathbf{b}}$ *from* $\mathbf{b}$*.*

ii. If $\mathbf{Q}_{f_1}\mathbf{x} = \mathbf{b}_1$*, and* $\mathbf{Q}_{f_2}\mathbf{x} = \mathbf{b}_2$ *are equivalent give a simple rule for obtaining* $\mathbf{b}_1$ *from* $\mathbf{b}_2$*.*

iii. If $\mathbf{B}_{f_1}\mathbf{y} = \mathbf{d}_1$*, and* $\mathbf{B}_{f_2}\mathbf{y} = \mathbf{d}_2$ *are equivalent give a simple rule for obtaining* $\mathbf{d}_1$ *from* $\mathbf{d}_2$*.*

Problem 3.20 *If two circuit (cutset) vectors figure in the same f-circuit (f-cutset) matrix show that the signs of the overlapping portion fully agree or fully oppose. So overlapping f-circuits (f-cutsets) fully agree or fully oppose in their orientations.*

Problem 3.21 *(k) Give simple rules for computing* $\mathbf{AA}^T, \mathbf{B}_f\mathbf{B}_f^T, \mathbf{Q}_f\mathbf{Q}_f^T$*. Show that the number of nonzero entries of* $\mathbf{AA}^T$ *is* $2e+n$ *if the graph has no parallel edges. Show that* $\mathbf{B}_f\mathbf{B}_f^T, \mathbf{Q}_f\mathbf{Q}_f^T$ *may not have any zero entries. Hence observe that nodal analysis is preferable to fundamental loop analysis and fundamental cutset analysis from the point of view of using Gaussian elimination.*
(Consider the case where a single edge lies in every circuit (cutset) corresponding to rows of $\mathbf{B}_f(\mathbf{Q}_f)$*).*

Problem 3.22 *Under what conditions can two circuit (cutset) vectors of a given graph be a part of the same f-circuit (f-cutset) matrix?*

Problem 3.23 *(k) Construct good algorithms for building f-circuit and f-cutset vectors for a given forest (use dfs or bfs described in Subsections 3.6.1, 3.6.2). Compute the complexity.*

Problem 3.24 Special Technique for Building a Representative Matrix of $\mathcal{V}_i(\mathcal{G})$**:**
Prove that the following algorithm works for building a representative matrix of $\mathcal{V}_i(\mathcal{G})$*:*
Let $\mathcal{G}_1$ *be a subgraph of* $\mathcal{G}$*,*
$\mathcal{G}_2$ *be a subgraph of* $\mathcal{G}$ *s.t.* $E(\mathcal{G}_1) \cap E(\mathcal{G}_2)$ *is a forest of* $\mathcal{G}_1$*,*
$\vdots$
$\mathcal{G}_k$ *be a subgraph of* $\mathcal{G}$ *s.t.* $E(\mathcal{G}_k) \cap \left[\bigcup_{i=1}^{k-1} E(\mathcal{G}_i)\right]$ *is a forest of the subgraph*
$\mathcal{G} \cdot \left(\bigcup_{i=1}^{k-1} E(\mathcal{G}_i)\right)$ *and let* $\bigcup E(\mathcal{G}_i) = E(\mathcal{G})$*.*

Build representative matrices $\mathbf{R}_j$ *for* $\mathcal{V}_i(\mathcal{G}_j), j = 1, 2, \cdots k$. *Extend the rows of* $\mathbf{R}_j$ *to size* $E(\mathcal{G})$ *by padding with* 0*s. Call the resulting matrix* $\hat{\mathbf{R}}_j$. *Then* $\mathbf{R}$ *is a representative matrix for* $\mathcal{V}_i(\mathcal{G})$, *where*

$$\mathbf{R} = \begin{bmatrix} \hat{\mathbf{R}}_1 \\ \vdots \\ \hat{\mathbf{R}}_k \end{bmatrix}.$$

Problem 3.25 Equivalence of Minty's and Tellegen's Theorems:
Prove that Minty's Theorem (strong form) and Tellegen's Theorem (strong form) are formally equivalent.

Problems on Basic Operations of Graphs

Problem 3.26 *(k) Let* $\mathcal{G}$ *be graph. Let* $K \subseteq E(\mathcal{G})$. *Then*

i. K *is a forest of* $\mathcal{G} \cdot T$ *iff it is a maximal intersection of forests of* $\mathcal{G}$ *with* T.

ii. K *is a forest of* $\mathcal{G} \times T$ *iff it is a minimal intersection of forests of* $\mathcal{G}$ *with* T.

iii. K *is a forest of* $\mathcal{G} \times T$ *iff* $K \cup$ *(a forest of* $\mathcal{G} \cdot (S - T)$*) is a forest of* $\mathcal{G}$.

iv. K *is a coforest of* $\mathcal{G} \cdot T$ *iff* $K \cup$ *(a coforest of* $\mathcal{G} \times (S - T)$*) is a coforest of* $\mathcal{G}$.

Problem 3.27 Relation between Forests Built According to Priority and Graph Minors: *Let* $A_1, \cdots A_n$ *be pairwise disjoint subsets of* $\mathcal{G}$.

i. A forest f *of* $\mathcal{G}$ *contains edges from these sets in the same priority iff it is the union of forests from* $\mathcal{G} \cdot A_1$, $\mathcal{G} \cdot (A_1 \cup A_2) \times A_2$, $\mathcal{G} \cdot (A_1 \cup A_2 \cup A_3) \times A_3, \cdots \mathcal{G} \times A_n$.

ii. Suppose the graph has only such forests what can you conclude?

iii. What can you conclude if the priority sequence $A_i, i = 1, \cdots n$ and $A_{\sigma(i)} i = 1, \cdots n$ for every permutation σ of $1, \cdots n$ yield the same forests?

Problem 3.28 *(k) Show how to build an f-circuit (f-cutset) matrix of $\mathcal{G}$ in which f-circuit (f-cutset) matrices of $\mathcal{G} \cdot T$ and $\mathcal{G} \times (E - T)$ become 'visible' (appear as submatrices). Let $T_2 \subseteq T_1 \subseteq E(\mathcal{G})$. Repeat the above so that the corresponding matrix of $\mathcal{G} \times T_1 \cdot T_2$ is 'visible'.*

Problem 3.29 *(k) Suppose in an electrical network on graph $\mathcal{G}$ the subset T is composed of current (voltage) sources. How will you check that they do not violate KCL (KVL)?*

3.6 Graph Algorithms

In this section we sketch some of the basic graph algorithms which we take for granted in the remaining part of the book. The algorithms we consider are

- construction of trees and forests of various kinds for the graph (bfs, dfs, minimum spanning)
- finding the connected components of the graph
- construction of the shortest path between two vertices of the graph
- construction of restrictions and contractions of the graph
- bipartite graph based algorithms such as for dealing with partitions
- flow maximization in networks

The account in this section is very brief and informal. For more details the readers are referred to [Aho+Hopcroft+Ullman74] [Kozen92] [Cormen+Leiserson+Rivest90].

For each of the above algorithms we compute or mention the 'asymptotic worst case complexity' of the algorithm. Our interest is primarily

in computing an upper bound for the worst case running time of the algorithm and sometimes also for the worst case storage space required for the algorithm. A memory unit, for us, contains a single elementary symbol (a number - integer or floating point, or an alphabet). Accessing or modifying such a location would be assumed to cost unit time. Operations such as comparison, addition, multiplication and division are all assumed to cost unit time. Here as well as in the rest of the book we use the 'big Oh' notation:
Let $f, g : \mathcal{N}^p \rightarrow \mathcal{N}$ where $\mathcal{N}$ denotes the set of nonnegative integers and p is a positive integer. We say f **is** $O(g)$ iff there exists a positive integer k s.t. $f(\mathbf{n}) \leq kg(\mathbf{n})$ for all $\mathbf{n}$ outside a finite subset of $\mathcal{N}^p$.

The time and space complexity of an algorithm to solve the problem (the number of elementary steps it takes and the number of bits of memory it requires) would be computed in terms of the **size of the problem instance**. The size normally refers to the number of bits (within independently specified multiplying constant) required to represent the instance of the problem in a computer. It could be specified in terms of several parameters. For example, in the case of a directed graph with capacitated edges the size would be in terms of number of vertices, number of edges and the maximum number of bits required to represent the capacity of an edge. In general, the **size of a set** would be its cardinality while the **size of a number** would be the number of bits required to represent it. Thus, if n is a positive integer, $\log n$ would be its size – the base being any convenient positive integer. All the algorithms we study in this book are polynomial time (and space) algorithms, i.e., their worst case complexity can be written in the form $O(f(n_1, \cdots, n_p))$ where $f(\cdot)$ is a polynomial in the n_i. Further, in almost all cases, the polynomials would have low degree (≤ 5).

Very rarely we have used words such as NP-complete and NP-Hard. Informally, a problem is in P if the 'answer to it' (i.e., the answer to every one of its instances) can be **computed** in polynomial time (i.e., in time polynomial in the size of the instance) and is in NP if the correctness of the candidate answer to every instance of it can be **verified** in polynomial time. It is clear that P $\subseteq$ NP. However, although it is widely believed that P $\neq$ NP, a proof for this statement has not been obtained so far. An **NP-Hard** problem is one which has the prop-

erty that if its answer can be **computed** in polynomial time, then we can infer that the answer to every problem in NP can be computed in polynomial time. An NP-Hard problem need not necessarily be in NP. If it is in NP, then it is said to be **NP-complete**. The reader interested in formal definitions as well as in additional details is referred to [Garey+Johnson79], [Van Leeuwen90].

Exercise 3.63 *A decision problem is one for which the answer is (***yes** *or* **no***). Convert the problem 'find the shortest path between* v_1 *and* v_2 *in a graph' into a 'short' sequence of decision problems.*

For most of our algorithms elementary **data structures** such as arrays, stacks, queues are adequate. Where more sophisticated data structures (such as Fibonacci Heaps) are used, we mention them by name and their specific property (such as time for retrieval, time for insertion etc.) that is needed in the context. Details are skipped and may be found in [Kozen92].

Storing a graph: A graph can be stored in the form of a sequence whose i^{th} (composite) element contains the information about the i^{th} edge (names of end points; if edge is directed the names of positive and negative end points). This sequence can be converted into another whose i^{th} element contains the information about the i^{th} node (names of incident edges, their other end points; if the graph is directed, the names of out-directed and in-directed edges and their other end points.) We will assume that we can retrieve incidence information about the i^{th} edge in $O(1)$ time and about the (i^{th} node) in $O(degree\ of\ node\ i)$ time. The conversion from one kind of representation to the other can clearly be done in $O(m+n)$ time where m is the number of edges and n is the number of vertices.

Sorting and Searching: For sorting a set of indexed elements in order of increasing indices, there are available, algorithms of complexity $O(n \log n)$, where n is the number of elements [Aho+Hopcroft+Ullman74]. We use such algorithms without naming them. In such a sorted list of elements to search for a given indexed element takes $O(\log n)$ steps by using **binary search**.

3.6.1 Breadth First Search

A **breadth first search (bfs) tree or forest** for the given graph $\mathcal{G}$ is built as follows:
Start from any vertex v_o and **scan** edges incident on it.
Select these edges and put the vertices $v_1, v_2 \cdots v_{ko}$ which are adjacent to v_o in a **queue** in the order in which the edges between them and v_o were scanned.
Mark v_o as belonging to component 1 and level 0. Mark $v_1, \cdots, v_{ko}$, as belonging to component 1 and level 1 and as **children** of v_o. Mark the vertex v_o additionally as a **parent** of its children (against each of its children).
Suppose at any stage we have the queue $v_{i1}, \cdots, v_{ik}$ and a set M_i of marked vertices.
Start from the left end (first) of the queue, scan the edges incident on it and select those edges whose other ends are unmarked. If a selected edge is between v_{ij} and the unmarked vertex v_{um} then the former (latter) is the **parent (child)** of the latter (former).
Put the children of v_{i1} in the queue after v_{ik} and delete v_{i1} from the queue.
Mark these vertices as belonging to the level next to that of v_{i1} and to the same component as v_{i1} and as children of v_{i1} (against v_{i1}). Mark the vertex v_{i1} as a parent of its children (against its children).
Continue.
When the graph is disconnected it can happen that the queue is empty but all vertices have not yet been marked. In this case continue the algorithm by picking an unmarked vertex.
Mark it as of level 0 but as of component number one more than that of the previous vertex. Continue.
STOP when all vertices of the graph have been marked.

At the conclusion of the above algorithm we have a breadth first search forest made up of the selected edges and a partition of the vertex set of the graph whose blocks are the vertex sets of the components of the graph. The starting vertices in each component are called **roots**. The level number of each vertex gives its **distance** from the root (taking the length of each edge to be one). The path in the forest from a given vertex in a component to the root in the component is obtained by travelling from the vertex to its parent and so on back to the root.

In a directed graph a bfs starting from any vertex would yield all vertices reachable from it through directed paths. In this case, while processing a vertex, one selects only the outward directed edges.

The **complexity of the** bfs **algorithm** is $O(m+n)$ where m is the number of edges and n is the number of vertices. (Each edge is 'touched' atmost twice. Each vertex other than the root is touched when an edge incident on it is touched or when it is a new root. Except where the root formation is involved the labour involved in touching a vertex can always be absorbed in that of touching an edge. Each touching involves a fixed number of operations).

The **complexity** of computing **all the reachable vertices** from a given vertex or a set of vertices of a directed graph through bfs is clearly also $O(m+n)$.

3.6.2 Depth First Search

A **depth first search** (dfs) **tree or forest** for the given graph $\mathcal{G}$ is built as follows:
Start from any vertex v_o and **scan** the edges incident on it.
Select the first nonselfloop edge. Let v_1 be its other end point. Put v_o, v_1 in a **stack**. (A **stack** is a sequence of data elements in which the last (i.e., latest) element would be processed first). Mark v_o as belonging to component 1 and as having dfs number 0, v_1 as belonging to component 1 and as having dfs number 1. Mark v_o as the parent of v_1 (against v_1) and v_1 as a child of v_o (against v_o).
Suppose at any stage, we have the stack $v_{i1}, \cdots, v_{ik}$ and a set M_i of marked vertices.
Start from the top of the stack, i.e., from v_{ik} and scan the edges incident on it. Let e be the first edge whose other end point v_{i+1} is unmarked. Select e. Mark v_{i+1} as of dfs number one more than that of the highest dfs number of a vertex in M_i and of component number same as that of v_{ik}. Mark (against v_{i+1}) v_{ik} as its parent and (against v_{ik}) v_{i+1} as one of its children. Add v_{i+1} to the top of the stack and repeat the process.
Suppose v_{ik} has no edges incident whose other end points are unmarked. Then delete v_{ik} from the stack (so that $v_{i(k-1)}$ goes to the

top of the stack).
Continue.
STOP when all vertices in the graph have been marked.
When the graph is disconnected it can happen that the stack is empty but all vertices have not yet been marked. In this case continue the algorithm by picking an unmarked vertex. Give it a dfs number 0 but component number one more than that of the previous vertex.

At the conclusion of the above algorithm we have a depth first search forest made up of the selected edges and a partition of the vertex set of the graph whose blocks are the vertex sets of the components of the graph. The starting vertices in each component are called roots. The path in the forest from a given vertex in a component to the root in the component is obtained by travelling from the vertex to its parent and so on back to the root. The **time complexity of the** dfs **algorithm** can be seen to be $O(m+n)$ where m is the number of edges and n, the number of vertices in the graph.

Exercise 3.64 *(k) Let e be an edge outside a dfs tree of the graph. Let v_1, v_2 be the end points of e with dfs numbering a, b respectively. If $b > a$ show that v_1 is necessarily an ancestor of v_2 (ancestor $\equiv$ parent's parent's ... parent).*

The dfs tree can be used to detect 2-connected components of the graph in $O(m+n)$ time [Aho+Hopcroft+Ullman74]. It can be used to construct the planar embedding of a planar graph in $O(n)$ time [Hopcroft+Tarjan74], [Kozen92]. There is a directed version of the dfs tree using which a directed graph can be decomposed into strongly connected components (maximal subsets of vertices which are mutually reachable by directed paths). Using the directed dfs tree this can be done in $O(m+n)$ time [Aho+Hopcroft+Ullman74].

Fundamental circuits: Let t be a forest of graph $\mathcal{G}$ and let $e \in (E(\mathcal{G}) - t)$. To construct $L(e,t)$ we may proceed as follows: Do a dfs of $\mathcal{G}\ .\ t$ starting from any of its vertices. This would give a dfs number to every vertex in $\mathcal{G}\ .\ t$.
Let v_1, v_2 be the end points of e. From v_1, v_2 proceed towards the root by moving from child to parent until you meet the first common ancestor v_3 of v_1 and v_2. This can be done as follows: Suppose v_1 has a higher dfs number than v_2. Move from v_1 to root until you reach the first v_1' whose dfs number is less or equal to that of v_2. Now repeat

the procedure with v_2, v_1' and so on alternately until the first common vertex is reached. This would be v_3. Then $L(e,t) \equiv \{e\} \cup \{$ edges in paths from v_1 to v_3 and v_2 to $v_3\}$.
To build the circuit vector corresponding to $L(e,t)$ proceed as follows: Let v_1 be the positive end point and v_2, the negative end point of e. The path from v_2 to v_1 in the tree is the path from v_2 to v_3 followed by the path from v_3 to v_1. The circuit vector has value $+1$ at e, 0 outside $L(e,t)$ and $+1$ (-1) at e_j, if it is along (against) the path from v_2 to v_1 in the tree. Complexity of building the $L(e,t)$ is $O(\mid L(e,t) \mid)$ and that of building all the $L(e_i,t)$ is $O(\sum \mid L(e,t) \mid)$.

Exercise 3.65 *How would you build the f-circuit for a bfs tree?*

3.6.3 Minimum Spanning Tree

We are given a connected undirected graph $\mathcal{G}$ with real weights $(w(\cdot))$ on its edges. The problem is to find a spanning tree of least total weight (total weight = sum of weights of edges in the tree). We give **Prim's algorithm** for this purpose:
Choose an arbitrary vertex v_o. Among the edges incident on v_o select one of least weight.
Suppose at some stage, X is the set of edges selected and $V(X)$, the set of their end points. If $V(X) \neq V(\mathcal{G})$, select an edge e of least weight among those which have only one end point in $V(X)$.
Now replace X by $X \cup e$ and repeat.
Stop when $V(X) = V(\mathcal{G})$.
The selected edges constitute a minimum spanning tree.

Exercise 3.66 *Justify Prim's algorithm for minimum spanning tree.*

Complexity: Let n be the number of vertices and m, the number of edges of the graph. The algorithm has n stages. At each stage we have to find the minimum weight edge among the set of edges with one end point in $V(X)$. Such edges cannot be more than m in number. So finding the minimum is $O(m)$ and the overall complexity is $O(mn)$. However, this complexity can be drastically improved if we store the vertices in $(V(\mathcal{G}) - V(X))$ in a Fibonacci Heap. This data structure permits the extraction of the minimum valued element in $O(\log n)$ amortized time (where n is the number of elements in the

heap), changing the value of an element in $O(1)$ amortized time and deleting the minimum element in $O(\log n)$ amortized time. (Loosely, an operation being of amortized time $O(f(n))$ implies that, if the entire running of the algorithm involves performing the operation k times, then the time for performing these operations is $O(kf(n))$.

For each vertex v in $(V(\mathcal{G}) - V(X))$ the **value** is the minimum of the weights of the edges connecting it to $V(X)$. To pick a vertex of least value we have to use $O(\log n)$ amortized time. Suppose v has been added to $V(X)$ and X replaced by $X \cup e$, where e has v as one its ends. Now the value of a vertex v' in $(V(\mathcal{G}) - (V(X) \cup e))$ has to be updated only if there is an edge between v and v'. Throughout the algorithm this updating has to be done only once per edge and each such operation takes $O(1)$ amortized time. So overall the updating takes $O(m)$ time. The extraction of the minimum valued element takes $O(n \log n)$ time over all the n stages. At each stage the minimum element has to be deleted from the heap. This takes $O(\log n)$ amortized time and $O(n \log n)$ time overall. Hence, the running time of the algorithm is $O(m + n \log n)$. (Note that the above analysis shows that, without the use of the Heap, the complexity of Prim's algorithm is $O(n^2)$).

3.6.4 Shortest Paths from a Single Vertex

We are given a graph $\mathcal{G}$, without parallel edges, in which each edge e has a nonnegative **length** $l(v_1, v_2)$, where v_1, v_2 are the end points of e. If $v_1 = v_2$, then $l(v_1, v_2) \equiv 0$. The **length of a path** is defined to be the sum of the lengths of the edges in the path.
The problem is to find shortest paths from a given vertex (called the source) to every vertex in the same connected component of the graph.
We give **Dijkstra's Algorithm** for this problem.
Start from the source vertex v_o and assign to each adjacent vertex v_i, a **current distance** $d_c(v_i) \equiv l(v_o, v_i)$. **Mark**, against each v_i, the vertex v_o as its foster parent. (We will call v_i, the foster children of v_o).
Let v_1 be the adjacent vertex to v_o with the least value of $d_c(v_i)$. Declare the **final distance of** v_1, $d_f(v_1) \equiv d_c(v_1)$. Mark, against v_1, the vertex v_o as its parent. (We will call v_1, a child of v_o).
(At this stage we have **processed** v_o and **marked** its adjacent vertices).

Assign a current distance ∞ to each unmarked vertex.
Suppose $X \subseteq V(\mathcal{G})$ denotes the processed set of vertices at some stage.
For each neighbour $v_j \in (V(\mathcal{G}) - X)$ of last added vertex v_k,
Check if $d_c(v_j) > d_f(v_k) + l(v_k, v_j)$.
If Yes, then
Mark, against v_j, the vertex v_k as its foster parent
(deleting any earlier mark, if present). (We will call v_j, a foster child of v_k).
Set $d_c(v_j) \equiv d_f(v_k) + l(v_k, v_j)$.
Find a vertex $v_q \in (V(\mathcal{G}) - X)$ with the least current distance $d_c(v_q)$.
Declare v_q to have been processed and its final distance $d_f(v_q)$ from v_o to be $d_c(v_q)$. Mark, against v_q, its foster parent u_q as its parent (we will call v_q a child of u_q).
Add v_q to X. Repeat the procedure with $X \cup v_q$ in place of X.
STOP when all vertices in the connected component of v_o are processed.

To find a shortest path from a processed vertex v_j to v_o, we travel back from v_j to its parent and so on, from child to parent, until we reach v_o.

Justification: To justify the above algorithm, we need to show that the shortest distance from v_o to v_q (the vertex with the least current distance in $V(\mathcal{G}) - X$) is indeed $d_f(v_q)$. First, we observe that a finite $d_c(v)$, and therefore $d_f(v)$, for any vertex v is the length of some path from v_o to v. By induction, we may assume that for every vertex v_{in} in X, $d_f(v_{in})$ = length of the shortest path from v_o to v_{in}. Note that this is justified when $X = \{v_o\}$. Suppose $d_f(v_q)$ is greater than the length of a shortest path $P(v_o, v_q)$ from v_o to v_q. Let $P(v_o, v_q)$ leave X for the first time at v_3 and let the next vertex be $v_{out} \in (V(\mathcal{G}) - X)$.
If $v_{out} = v_q$, we must have
$d_f(v_q) \leq d_f(v_3) + l(v_3, v_q)$ = length of $P(v_o, v_q)$.
This is a contradiction. So $v_{out} \neq v_q$. Now
$d_c(v_{out}) \leq (d_f(v_3) + l(v_3, v_{out})) \leq$ length of $P(v_o, v_q)$.
Hence, $d_c(v_{out}) < d_c(v_q) = d_f(v_q)$, which contradicts the definition of v_q. We conclude that $d_f(v_q)$ must be the length of the shortest path from v_o to v_q.

Complexity: Let n be the number of vertices and m, the number of edges of the graph. This algorithm has n stages. At each stage we have

to compute $d_c(v_j)$ for vertices v_j adjacent to the last added vertex. This computation cannot exceed $O(m)$ over all the stages. Further at each stage we have to find the minimum of $d_c(v_i)$ for each v_i in $(V(\mathcal{G}) - X)$. This is $O(n)$. So we have an overall complexity of $O(n^2 + m)$. Now $m \leq n^2$. So the time complexity reduces to $O(n^2)$.
We note that the complexity of this algorithm reduces to $O(m+n \log n)$ if the elements in $V(\mathcal{G}) - X$ are stored in a Fibonacci Heap (see [Kozen92]).

3.6.5 Restrictions and Contractions of Graphs

Let $\mathcal{G}$ be a graph and let $T \subseteq E(\mathcal{G})$. To build $\mathcal{G} \,.\, T$, we merely pick out the edge - end point list corresponding to T. This has complexity $O(\mid T \mid)$. (Note that the edges of T still bear their original index as in the sequence of edges of $\mathcal{G}$).
To build $\mathcal{G} \times T$ we first build $\mathcal{G}openT$. The graph $\mathcal{G}openT$ has $\mathcal{G} \,.\, (E(\mathcal{G}) - T)$ + remaining vertices of $\mathcal{G}$ as isolated vertices. Next we find the connected components of $\mathcal{G}openT$. Let the vertex sets of the components be $X_1, \cdots, X_k$. For each X_i, whenever $v \in X_i$, mark it as belonging to X_i (some one vertex of X_i can represent X_i). Changing the names of endpoints amounts to directing a pointer from vertices to the X_i that they belong to. Now in the edge - end point list of T, for each edge e, if v_1, v_2 are its (positive and negative) endpoints, and if $v_1 \in X_i$, $v_2 \in X_j$, then replace v_1 by X_i and v_2 by X_j ($\mathcal{G}shortT$ has vertex set $(X_1, \cdots, X_k)$).
The complexity of building $\mathcal{G}openT$ is $O(n+ \mid E - T \mid)$, where n is the number of vertices of $\mathcal{G}$, that of finding its components is $O(n+ \mid E-T \mid)$ (using dfs say). Changing the names of endpoints amounts to directing a pointer from vertices to the X_i that they belong to. This has already been done. So the overall complexity is $O(n + m)$ where $m = \mid E(\mathcal{G}) \mid$.

Elsewhere, we describe methods of network analysis (by decomposition) which require the construction of the graphs $\mathcal{G} \,.\, E_1, \cdots, \mathcal{G} \,.\, E_k$ or $\mathcal{G} \times E_1, \cdots, \mathcal{G} \times E_k$, where $\{E_1, \cdots, E_k\}$ is a partition of $E(\mathcal{G})$. The complexity of building $\oplus_i \mathcal{G} \,.\, E_i$ is clearly $O(n + m)$, while that of building $\oplus_i \mathcal{G} \times E_i$ is $O(k(n + m))$.

3.6.6 Hypergraphs represented by Bipartite Graphs

Hypergraphs are becoming increasingly important for modeling many engineering situations. By definition, a **hypergraph** $\mathcal{H}$ is a pair $(V(\mathcal{H}), E(\mathcal{H}))$, where $V(\mathcal{H})$ is the set of **vertices** of $\mathcal{H}$ and $E(\mathcal{H})$, a family of subsets of $V(\mathcal{H})$ called the **hyperedges** of $\mathcal{H}$. (We remind the reader that in a family, the same member subset could be repeated with distinct indices yielding distinct members of the family). The reader would observe that undirected graphs are a special case of hypergraphs (with the hyperedges having cardinality 1 or 2). The most convenient way of representing a hypergraph is through a **bipartite graph** $B \equiv (V_L, V_R, E)$ - a graph which has a **left vertex set** V_L, a (disjoint) **right vertex set** V_R and the set of edges E each having one end in V_L and the other in V_R. We could represent $\mathcal{H}$ by $B_{\mathcal{H}} \equiv (V_L, V_R, E)$ identifying $V(\mathcal{H})$ with $V_L, E(\mathcal{H})$ with V_R with an edge in the bipartite graph between $v \in V_L$ and $e \in V_R$ iff v is a member of the hyperedge e of $\mathcal{H}$.

We can define **connectedness** for $\mathcal{H}$ in a manner similar to the way the notion is defined for graphs. $\mathcal{H}$ is **connected** iff for any pair of vertices v_1, v_f there exists an **alternating sequence** $v_1, e_1, v_2, e_2, \cdots, e_f, v_f$, where the v_i are vertices and e_i, edges s.t. each edge has both the preceding and succeeding vertices as members. It is easily seen that $\mathcal{H}$ is connected iff $B_{\mathcal{H}}$ is connected. Hence, checking connectedness of $\mathcal{H}$ can be done in $O(\mid V_L \mid + \mid V_R \mid + \mid E \mid)$ time. Since everything about a hypergraph is captured by a bipartite graph we confine our attention to bipartite graphs in this book. The reader interested in 'standard' hypergraph theory is referred to [Berge73].

3.6.7 Preorders and Partial Orders

A preorder is an ordered pair $(P, \preceq)$ where P is a set and ' $\preceq$' is a binary relation on P that satisfies the following:
$x \preceq x, \forall\, x \in P$;
$x \preceq y, y \preceq z \Rightarrow x \preceq z\ , \forall\, x, y, z \in P$.
We can take the elements of P to be vertices and join x and y by an edge directed from y to x if $x \preceq y$. Let $\mathcal{G}_p$ be the resulting directed graph on the vertex set P. Then the vertex sets of the strongly con-

nected components of $\mathcal{G}_P$ are the equivalence classes of the preorder (x, y belong to an equivalence class iff $x \preceq y$ and $y \preceq x$).
Let $\mathcal{P}$ be the collection of equivalence classes. If $X_1, X_2 \in \mathcal{P}$, we define $X_1 \leq X_2$ iff in the graph $\mathcal{G}_p$, a vertex in X_1 can be reached from a vertex in X_2. It is easily seen that this defines a **partial order** ($X_i \leq X_i; X_i \leq X_j$ and $X_j \leq X_i$ iff $X_i = X_j; X_i \leq X_j, X_j \leq X_k \Rightarrow X_i \leq X_k$). This partial order $(\mathcal{P}, \leq)$ is said to be **induced by** $(P, \preceq)$. By using a directed dfs forest on the graph $\mathcal{G}_P$ representing the preorder $(P, \preceq)$ we can get a graph representation of the induced partial order in time $O(m + n)$ where m is the number of edges and n is the number of vertices in $\mathcal{G}_P$ [Aho+Hopcroft+Ullman74].

A partial order can be represented more economically by using a Hasse Diagram. Here a directed edge goes from a vertex y to a vertex x iff y **covers** x, i.e., $x \leq y, x \neq y$ and there is no z s.t. $z \neq x$ and $z \neq y$ and $x \leq z \leq y$. An **ideal** I of $(\mathcal{P}, \leq)$ is a collection of elements of $\mathcal{P}$ with the property that if $x \in I$ and $y \leq x$ then $y \in I$. The **principal ideal** I_x in $(\mathcal{P}, \leq)$ of an element $x \in \mathcal{P}$ is the collection of all elements $y \in \mathcal{P}$ s.t. $y \leq x$. Clearly an ideal is the union of the principal ideals of its elements. A **dual ideal** I^d is a subset of $\mathcal{P}$ with the property that if $x \in I$ and $x \leq z$ then $z \in I^d$. **Ideals and dual ideals of preorders** are defined similarly. The **dual** of a partial order $(\mathcal{P}, \leq)$ is the partial order $(\mathcal{P}, \geq)$, where $x \geq y$ iff $y \leq x$. We define the dual of a preorder in the same manner. We use $\leq$ and $\geq$ interchangeably (writing $y \leq x$ or $x \geq y$) while speaking of a partial order or a preorder.

Preorders and partial orders are used repeatedly in this book (see for instance Chapter 10).

Lattices

Let $(\mathcal{P}, \leq)$ be a partial order. An **upper bound** of $e_1, e_2 \in \mathcal{P}$ is an element $e_3 \in \mathcal{P}$ s.t. $e_1 \leq e_3$ and $e_2 \leq e_3$.
A **lower bound** of e_1 and e_2 would be an element $e_4 \in \mathcal{P}$ s.t. $e_4 \leq e_1$ and $e_4 \leq e_2$.
A **least upper bound** (l.u.b.) of e_1, e_2 would be an upper bound e^u s.t. whenever e_3 is an upper bound of e_1, e_2 we have $e_3 \geq e^u$. A **greatest lower bound** (g.l.b.) of e_1, e_2 would be a lower bound e_l s.t. whenever e_4 is a lower bound of e_1, e_2 we have $e_4 \leq e_l$. It is easy

to see that if l.u.b. (g.l.b.) of e_1, e_2 exists, then it must be unique. We denote the l.u.b. of e_1, e_2 by $e_1 \vee e_2$ and call it the **join** of e_1 and e_2. The g.l.b. of e_1, e_2 is denoted by $e_1 \wedge e_2$ and called the **meet** of e_1 and e_2. If every pair of elements in $\mathcal{P}$ has a g.l.b. and an l.u.b. we say that $(\mathcal{P}, \leq)$ is a **lattice**. A **lattice** can be defined independently of a partial order taking two operations '$\vee$' and '$\wedge$' as primitives satisfying the properties given below:
(idempotency) $x \vee x = x \quad \forall\, x \in \mathcal{P}$; $x \wedge x = x \quad \forall\, x \in \mathcal{P}$.
(commutativity) $x \vee y = y \vee x \quad \forall\, x, y \in \mathcal{P}$; $x \wedge y = y \wedge x \quad \forall\, x, y \in \mathcal{P}$.
(associativity) $(x \vee y) \vee z = x \vee (y \vee z) \quad \forall\, x, y, z \in \mathcal{P}$.
$(x \wedge y) \wedge z = x \wedge (y \wedge z) \quad \forall\, x, y, z \in \mathcal{P}$.
(absorption) $x \wedge (x \vee y) = x \vee (x \wedge y) = x \quad \forall\, x, y \in \mathcal{P}$.
The reader may verify that these properties are indeed satisfied by g.l.b. and l.u.b. operations if we start from a partial order.
A lattice that satisfies the following additional property is called a **distributive lattice**.
(distributivity) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \quad \forall\, x, y, z \in \mathcal{P}$;
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \quad \forall\, x, y, z \in \mathcal{P}$.
e.g. The collection of all subsets of a given set with union as the join operation and intersection as the meet operation is a distributive lattice. (For a comprehensive treatment of lattice theory see [Birkhoff67]).

Exercise 3.67 *Show that the collection of ideals of a partial order form a distributive lattice under union and intersection.*

3.6.8 Partitions

Let S be a finite set. A collection $\{S_1, \cdots, S_k\}$ of nonvoid subsets of S is a **partition** of S iff $\bigcup_i S_i = S$ and $S_i \cap S_j = \emptyset$ whenever i, j are distinct. If $\{S_1, \cdots, S_k\}$ is a partition of S then the S_i are referred to as its **blocks**.
Let $\mathcal{P}_S$ denote the collection of all partitions of S. We may define a partial order $(\mathcal{P}, \leq)$ on $\mathcal{P}_S$ as follows: Let $\Pi_1, \Pi_2 \in \mathcal{P}_S$. Then $\Pi_1 \leq \Pi_2$ (equivalently $\Pi_2 \geq \Pi_1$) iff each block of Π_1 is contained in a block of Π_2. We say Π_1 is **finer than** Π_2 or Π_2 is **coarser than** Π_1. If Π_a, Π_b are two partitions of S, the **join** of Π_a and Π_b, denoted by $\Pi_a \vee \Pi_b$, is the finest partition of S that is coarser than both Π_a and Π_b and the **meet** of Π_a and Π_b denoted by $\Pi_a \wedge \Pi_b$, is the coarsest partition of S

that is finer than both Π_a and Π_b. It can be seen that these notions are well defined: To obtain the meet of Π_a and Π_b we take the intersection of each block of Π_a with each block of Π_b and throw away the empty intersections. Observe that any element of S lies in precisely one such intersection. Clearly, the resulting partition Π_{ab} is finer than both Π_a and Π_b. Suppose Π_c is finer than both Π_a and Π_b. Let N_c be a block of Π_c. Then N_c is contained in some block N_a of Π_a and some block N_b of Π_b. So $N_c \subseteq N_a \cap N_b$ and hence N_c is contained in some block of Π_{ab}. This proves that Π_{ab} is the meet of Π_a and Π_b and therefore, that the 'meet' is well defined. Next let Π, Π' be two partitions coarser than Π_a and Π_b. It is then easy to see that $\Pi \wedge \Pi'$ is also coarser than Π_a and Π_b. Hence there is a unique finest partition of S coarser than Π_a and Π_b. Thus, the 'join' is well defined.

Storing partitions: We can store a partition by marking against an element of S, the name of the block to which it belongs.

Building $\Pi_a \wedge \Pi_b$: When Π_a, Π_b are stored, each element of S would have against it two names - a block of Π_a and a block of Π_b; a pair of names of intersecting blocks of Π_a, Π_b can be taken to be the name of a block of $\Pi_a \wedge \Pi_b$. Thus forming $\Pi_a \wedge \Pi_b$ from Π_a, Π_b is $O(\mid S \mid)$.
Building $\Pi_a \vee \Pi_b$: We first build a bipartite graph B with blocks of Π_a as V_L, blocks of Π_b as V_R with an edge between $N_a \in V_L$ and $N_b \in V_R$ iff $N_a \cap N_b \neq \emptyset$. It can be seen that this bipartite graph can be built in $O(\mid S \mid)$ time (For each element of S, check which blocks of Π_a, Π_b it belongs to). We find the connected components of this bipartite graph. This can be done in $O(m+n)$ time where m is the number of edges and n, the number of vertices in the bipartite graph. But both m and n do not exceed $\mid S \mid$. So $O(m+n) = O(\mid S \mid)$. Now we collect blocks of Π_a (or Π_b) belonging to the same connected component of B. Their union would make up a block of $\Pi_a \vee \Pi_b$. (For, this block is a union of some blocks K of Π_a as well as a union of some blocks of Π_b. Union of any proper subset of blocks of K would cut some block of Π_b). This involves changing the name marked against an element $u \in S$ - instead of say N_a, it would be N_v, which is the name of the connected component of B in which N_a is a vertex. Thus, building $\Pi_a \vee \Pi_b$ is $O(\mid S \mid)$.

3.6.9 The Max-Flow Problem

In this subsection we outline the max-flow problem and a simple solution for it. We also indicate the directions in which more sophisticated solutions lie. In subsequent chapters we use max-flow repeatedly to model various minimization problems. Other than the flexibility in modeling that it offers, the practical advantage of using the concept of max-flow lies in the availability of efficient algorithms.

Let $\mathcal{G}$ be a directed graph. The **flow graph** (or flow network) $F(\mathcal{G})$ is the tuple $(\mathcal{G}, \mathbf{c}$, s,t) where $c : E(\mathcal{G}) \rightarrow \Re_+$ is a real nonnegative **capacity** function on the edges of $\mathcal{G}$ and s and t are two vertices of $\mathcal{G}$ named **source** and **sink**, respectively. A **flow f** associated with $F(\mathcal{G})$ is a vector on $E(\mathcal{G})$ satisfying the following conditions:

i. **f** satisfies KCE at all nodes except s and t, i.e., at each vertex v other than s, t, the **net outward flow**

$$\sum_i f(e_{outi}) - \sum_j f(e_{inj}) = 0$$

where $e_{outi}(e_{inj})$ are the edges incident at v and directed out of (directed into) v.

ii. the net outward flow at s is nonnegative, and at t, is non-positive.

iii. $0 \leq f(e) \leq c(e) \quad \forall\ e \in E(\mathcal{G})$.

(Often a flow is defined to be a vector satisfying (i) and (ii) above while a **feasible flow** would be one that satisfies all three conditions). An edge e with $f(e) = c(e)$ is said to be **saturated** with respect to **f**. The **value** of the flow **f**, denoted by $|\ \mathbf{f}\ |$, is the net outward flow at s. A flow of maximum value is called a **max-flow**. An **s,t-cut** (**cut** for short) is an ordered pair (A, B), where A, B are disjoint complementary subsets of $V(\mathcal{G})$ s.t. $s \in A$ and $t \in B$. The **capacity** of the cut (A, B), denoted by $c(A, B)$ is the sum of the capacities of edges with positive end in A and negative end in B. A cut of minimum capacity is called a **min-cut**. The **flow across** (A, B) denoted by $f(A, B)$, is the sum of the flows in the 'forward' edges going from A to B minus the sum of the flows in the 'backward' edges going from B to A.

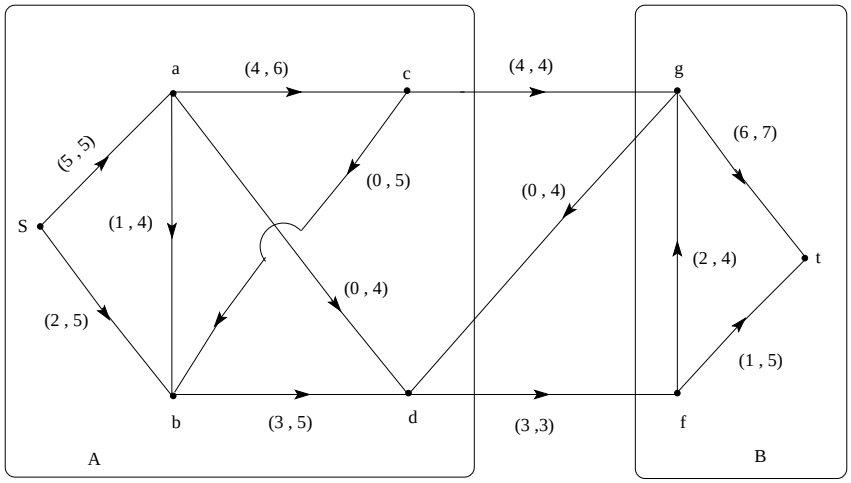


Figure 3.12: A Flow Graph with a max-flow and a min-cut

Example: Figure 3.12 shows a flow graph. Alongside each directed edge is an ordered pair with the second component indicating the capacity of the edge. A feasible flow $\mathbf{f}$ is defined on this flow graph with $f(e)$ being the first component of the ordered pair alongside e. The reader may verify that the net flow leaving any node other than the source s and the sink t is zero. At s there is a net positive outward flow ($= 7$) and at t there is a net negative outward flow ($= -7$). Let $A \equiv \{s, a, b, c, d\}$ and let $B \equiv \{g, f, t\}$. Then (A, B) is an s, t cut. It can be verified that $f(A, B) = 4 + 3 - 0 = 7$. Observe that the forward edges (c, g) and (d, f) of the cut (A, B) are saturated while the backward edge (g, d) carries zero flow. It is clear that in the present case $f(A, B) = c(A, B)$. From the arguments given below it would follow that the given flow has the maximum value, i.e, is a max-flow and that the cut (A, B) is a min-cut, i.e., has minimum capacity.

Clearly the flow across an s,t-cut (A, B) cannot exceed the capacity of (A, B), i.e., $f(A, B) \leq c(A, B)$. Let (A, B) be an s,t-cut. If we add the outward flows at all nodes inside A we would get the value $f(A, B)$ (flow of each edge with both ends within A is added once with a $(+)$ sign and another time with a $(-)$ sign and hence cancels) as well as $\mid \mathbf{f} \mid$ (at all nodes other than s the net outward flow is zero). We conclude that $\mid \mathbf{f} \mid = f(A, B)$.

Let $\mathbf{f}$ be a flow in $F(\mathcal{G})$. Let P be a path oriented from s to t. Suppose it is possible to change the flow in the edges of P, without violating capacity constraints, as follows: the flow in each edge e of P is increased (decreased) by $\delta > 0$ if e supports (opposes) the orientation of P.
Such a path is called an **augmenting** path for the flow $\mathbf{f}$. Observe that this process does not disturb the KCE at any node except s, t. At s, the net outward flow goes up by δ, while at t, the net inward flow goes up by δ. Thus, if $\mathbf{f}'$ is the modified flow, $\mid \mathbf{f}' \mid = \mid \mathbf{f} \mid + \delta$. This is the essential idea behind flow maximization algorithms.

It is convenient to describe max-flow algorithms and related results in terms of the **residual graph** $\mathcal{G}_f$ associated with the flow $\mathbf{f}$. The graph $\mathcal{G}_f$ has the vertex set $V(\mathcal{G})$. Whenever $e \in E(\mathcal{G})$ and $f(e) < c(e)$, $\mathcal{G}_f$ has an edge e_+ between the same end points and in the same direction as e; and if $0 < f(e)$, $\mathcal{G}_f$ has an edge e_- in the opposite direction as e. Note that both e_+ and e_- may be present in $\mathcal{G}_f$. The edge e_+ has the **residual capacity** $r_f(e_+) \equiv c(e) - f(e)$
and the edge e_- has the **residual capacity** $r_f(e_-) \equiv f(e)$.
We note that a directed path P from s to t in the residual graph $\mathcal{G}_f$ corresponds to an augmenting path in $F(\mathcal{G})$ with respect to $\mathbf{f}$. Henceforth we would call such a path P in $\mathcal{G}_f$ also, an **augmenting path** of $\mathbf{f}$. The maximum amount by which the flow can be increased using this augmenting path is clearly the minimum of the residual capacities of the edges of P. This value we would call the **bottle neck capacity** of P.

We now present a simple algorithm for flow maximization. This algorithm is due to Edmonds and Karp [Edmonds+Karp72].

ALGORITHM 3.1 Algorithm Max-Flow
INPUT *A flow graph* $F(\mathcal{G}) \equiv (\mathcal{G}, \mathbf{c}, s, t)$.

OUTPUT*(i) A maximum valued flow* $\mathbf{f}_{max}$ *for* $F(\mathcal{G})$.
(ii) A min-cut (A, B) *s.t.* $\mid \mathbf{f}_{max} \mid \equiv c(A, B)$.

Initialize *Let* $\mathbf{f}$ *be any flow of* $F(\mathcal{G})$ *(* $\mathbf{f}$ *could be the zero flow for instance).*