

FACES OF MATCHING POLYHEDRA

by

William R. Pulleyblank

A Thesis

Submitted in Partial Fulfilment
of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

at the

UNIVERSITY OF WATERLOO

Waterloo, Ontario

Faculty of Mathematics

Department of Combinatorics and Optimization

March, 1973

The University of Waterloo requires the signature
of all persons using this thesis. Please sign
below, and give address and date.

14-9-77

F. Zorlin, 205 Brantford Ave., Kitchener

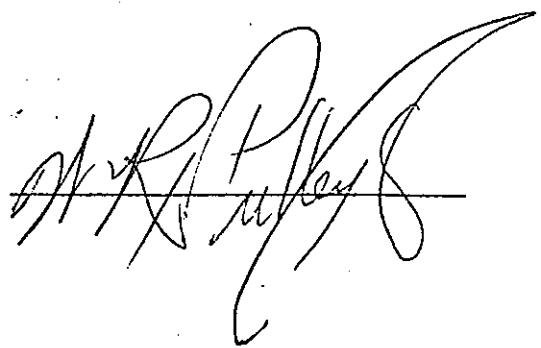
(C) WILLIAM R. PULLEYBLANK

1973.

I hereby declare that I am the sole author of this report.

I authorize the University of Waterloo to lend it to other institutions or individuals for the purpose of scholarly research.

Signature

A handwritten signature in black ink, appearing to read "M.R. Pyle". The signature is written in a cursive style with a horizontal line underneath it.

To

Janet

Abstract

Let $G = (V, E, \psi)$ be a finite loopless graph, let $b = (b_i : i \in V)$ be a vector of positive integers. A feasible matching is a vector $x = (x_j : j \in E)$ of nonnegative integers such that for each node i of G , the sum of the x_j over the edges j of G incident with i is no greater than b_i . The matching polyhedron $P(G, b)$ is the convex hull of the set of feasible matchings.

In Chapter 3 we describe a version of Edmonds' blossom algorithm which solves the problem of maximizing $c \cdot x$ over $P(G, b)$ where $c = (c_j : j \in E)$ is an arbitrary real vector. This algorithm proves a theorem of Edmonds which gives a set of linear inequalities sufficient to define $P(G, b)$.

In Chapter 4 we prescribe the unique subset of these inequalities which are necessary to define $P(G, b)$, that is, we characterize the facets of $P(G, b)$. We also characterize the vertices of $P(G, b)$, thus describing the structure possessed by the members of the minimal set X of feasible matchings of G such that for any real vector $c = (c_j : j \in E)$, $c \cdot x$ is maximized over $P(G, b)$ by a member of X .

In Chapter 5 we present a generalization of the blossom algorithm which solves the problem: maximize $c \cdot x$ over a face F of $P(G, b)$ for any real vector $c = (c_j : j \in E)$. In other words, we find a feasible matching x of G which satisfies the constraints obtained by replacing an arbitrary subset of the inequalities which define $P(G, b)$ by

equations and which maximizes $c \cdot x$ subject to this restriction. We also describe an application of this algorithm to matching problems having a hierarchy of objective functions, so called "multi-optimization" problems.

In Chapter 6 we show how the blossom algorithm can be combined with relatively simple initialization algorithms to give an algorithm which solves the following postoptimality problem. Given that we know a matching $x^0 \in P(G, b)$ which maximizes $c \cdot x$ over $P(G, b)$, we wish to utilize x^0 to find a feasible matching $x' \in P(G, b')$ which maximizes $c \cdot x$ over $P(G, b')$, where $b'_i = (b'_i : i \in V)$ is a vector of positive integers and $c = (c_j : j \in E)$ is an arbitrary real vector.

In Chapter 7 we describe a computer implementation of the blossom algorithm described herein.

ACKNOWLEDGEMENTS

It is difficult to express my enormous debt of gratitude to Professor Jack Edmonds who encouraged me to come to Waterloo, introduced me to the subjects discussed in this thesis and provided help, inspiration and encouragement in quantities that tended to grow exponentially with the size of the thesis. Many of the factors which made the University of Waterloo an ideal place to do graduate work are directly attributable to him. I value very highly all aspects of our association.

I also wish to extend my appreciation to Professor Ellis L. Johnson for the time he spent with me discussing matching theory and computer implementations of algorithms of the sort described herein.

Financial support was provided by the National Research Council of Canada and my wife Janet, who also contributed unlimited moral support.

I wish to thank Mrs. Wendy Johnson for the extremely fast and uncannily accurate typing of the thesis and to Mrs. Elaine Fitzgerald who assisted with the final stages of the typing.

This work was done while on an educational leave of absence from I.B.M. Canada Ltd.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	
ACKNOWLEDGEMENTS	
CHAPTER 1. Introduction and Foundations	1.1
1.1 Introduction	1.1
1.2 Set Theory and General Notation	1.13
1.3 Graph Theory	1.14
1.4 Linear Algebra	1.18
1.5 Linear Programming	1.20
1.6 Integer Programming and Good Algorithms	1.24
CHAPTER 2. Basic Polyhedral Theory	2.1
2.1 Polyhedra and their Faces	2.1
2.2 Dimension and a First Facet Characterization	2.6
2.3 Second Facet Characterization	2.13
2.4 Vertices of Polyhedra	2.22
CHAPTER 3. The Matching Problem and the Blossom Algorithm	3.1
3.1 The Matching Problem	3.1
3.2 Nested Families of Sets	3.8
3.3 Blossoms, Shrinking and Shrinkable Families	3.10
3.4 The Matching Polyhedron	3.19
3.5 Linear Programming Formulation	3.21
3.6 Alternating Forests	3.23
3.7 Hungarian Forests	3.27
3.8 The Blossom Algorithm	3.36
3.9 Efficiency of the Blossom Algorithm	3.58

	<u>Page</u>
3.10 Min-Max Theorems and Discreteness of the Dual Solution	3.60
CHAPTER 4	
Facets and Vertices of Matching Polyhedra	4.1
4.1 Dimension of $P(G, b)$ and Nonnegatively Facets	4.2
4.2 Degree Constraint Facets	4.4
4.3 Blossom Facets	4.9
4.4 b -critical Graphs	4.29
4.5 Vertices of Matching Polyhedra	4.39
CHAPTER 5	
Optimizing over Faces of $P(G, b)$	5.1
5.1 The Faces of $P(G, b)$	5.2
5.2 A Preconditioning Algorithm	5.9
5.3 Pseudo Hungarian Forests	5.12
5.4 The Face Optimization Algorithm (Phase II)	5.22
5.5 The "Big-M" Method	5.38
5.6 Multi-Optimization in Matching Problems	5.41
CHAPTER 6	
A Post-Optimality Problem	6.1
6.1 Obtaining a Starting Solution	6.2
6.2 The Post-Optimality Algorithm	6.9
6.3 Obtaining a Nested Family	6.17
CHAPTER 7	
A Computer Implementation of the Blossom Algorithm	7.1
7.1 Storage of the Graph	7.1
7.2 Tree Handling	7.5
7.3 Blossoms, Shrinking and Pseudonodes	7.9
7.4 Parameters Passed and Returned	7.14

7.5 The Main Procedure	7.18
7.6 Experimental Results	7.24

APPENDIX

REFERENCES

Chapter 1

Introduction and Foundations

1.1. Introduction

Let $G = (V, E, \psi)$ be a finite loopless graph, where V is the set of nodes of G , E is the set of edges of G and ψ is the incidence function of G which maps E into the set of all two element subsets of V . For each $i \in V$, let b_i be a positive integer. A feasible matching is a vector $x = (x_j : j \in E)$ of nonnegative integers such that for each node i of G , the sum of the x_j over the edges j of G incident with i is no greater than b_i . The matching polyhedron $P(G, b)$ is the bounded polyhedron containing all feasible matchings of G and all of whose vertices are feasible matchings of G . (In other words, $P(G, b)$ is the convex hull of the set of feasible matchings.) In this thesis we examine several different aspects of the faces of $P(G, b)$.

The later sections of Chapter 1 consist of a summary of the basic results from various fields of mathematics which are assumed to be known, we also introduce all our basic notation and terminology.

In Chapter 2 we develop the general polyhedral theory used in later chapters. This topic is developed from the point of view of studying systems of linear inequalities. The facets of a polyhedron are the faces of the polyhedron which have dimension one less than the dimension of the polyhedron itself. In characterizing the facets of matching polyhedra in Chapter 4 we make extensive use of (2.2.15),

which states that a proper face F of a polyhedron P of dimension d is a facet of P if and only if F contains $d + 1$ affinely independent elements. In Theorems (2.3.25), (2.3.30), (2.3.31), (2.3.32) and (2.3.34) we discuss the connection between the facets of a polyhedron and a minimal set of inequalities necessary to define the polyhedron. We show in (2.3.32) that if P is a polyhedron of full dimension, then the facets of P determine, up to multiplication by a positive constant, the minimal subset of inequalities needed to define P . Since matching polyhedra are of full dimension, this is the case in which we are interested.

We discuss the vertices of polyhedra in the last section of Chapter 2 and prove three fundamental results. First (Theorem 2.4.1)), the vertices of a polyhedron P are precisely those elements $v \in P$ for which there is some linear objective function c such that v is the unique member of P maximizing $c \cdot x$ over P . Second (Theorem (2.4.5)), if P is a bounded polyhedron then for any linear objective function c , there is a vertex v of P which maximizes $c \cdot x$ over P . Third (Theorem (2.4.10)), any nonempty bounded polyhedron is equal to the convex hull of its vertices.

Chapter 2 is largely expository, however the point of view taken in this chapter is somewhat different from standard references on polyhedra (Grünbaum [G1], Rockafellar [R1] and Stoer, Witzgall [S1]) and tends to emphasize the relationship between polyhedra and linear programming.

In Chapter 3 we describe a version of the so called blossom algorithm (Edmonds [E1], [E2], [E3], [E4]). This algorithm finds a matching $x^0 \in P(G, b)$ which maximizes $c \cdot x$ over $P(G, b)$. In fact the algorithm described solves a somewhat more general problem, it maximizes $c \cdot x$ over a face F of $P(G, b)$ obtained by requiring the sum of the x_j on the edges j incident with node i to be exactly equal to b_i for all nodes i belonging to some subset W of V .

For any node $i \in V$ we let $\delta(i)$ denote the set of edges of G incident with i . For any $S \subseteq V$ we let $\gamma(S)$ denote the set of edges of G having both ends in S . For any $J \subseteq E$ we let $x(J)$ denote $\sum_{j \in J} x_j$ and for any $W \subseteq V$ we let $b(W)$ denote $\sum_{i \in W} b_i$. The feasible matchings of G are the integer solutions of the linear system

$$(1.1.1) \quad x_j \geq 0 \quad \text{for all } j \in E,$$

$$(1.1.2) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V.$$

Clearly if we let P be the polyhedron defined by (1.1.1) and (1.1.2) then $P \supseteq P(G, b)$. In fact, if G is bipartite or if b_i is even for all $i \in V$ then $P = P(G, b)$. However in general there are vertices of P which are not vertices of $P(G, b)$ and thus have fractional components. Consequently there are generally some linear objective functions which when maximized over P , attain their maximum for a member x of P having fractional components. It can be seen that if x is a noninteger vertex of P then every component of x

is either integer or half integer valued and the edges j for which x_j are half integer valued form the edge sets of node disjoint odd polygons.

The blossom algorithm proves a theorem of Edmonds, that

$P(G, b) = \{(x_j \in \mathbb{R} : j \in E) : x \text{ satisfies (1.1.1) and (1.1.2)}\}$

$$(1.1.3) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q$$

where $Q = \{S \subseteq V : b(S) \text{ is odd, } |S| \geq 3\}$ and $q_S = 1/2(b(S)-1)$ for all $S \in Q$. It is not difficult to see that every feasible matching of G satisfies the constraints (1.1.3); it is more difficult to see that this set of constraints is sufficient to define $P(G, b)$, that is, that all vertices of the polyhedron defined by (1.1.1)-(1.1.3) are integer valued.

The blossom algorithm makes use of the weak duality theorem of linear programming and the principle of complementary slackness to prove the optimality of the matching which it finds. For any linear objective function c it produces an integer solution x^0 to the linear program: maximize $c \cdot x$ subject to x satisfying (1.1.1)-(1.1.3). It also produces a solution y^0 to the dual linear program and shows that x^0 and y^0 satisfy the complementary slackness conditions for optimality. Thus, where d is the objective function of the dual linear program, $c \cdot x^0 = d \cdot y^0$. By the weak duality theorem of linear programming, any solution x of (1.1.1)-(1.1.3) must satisfy $c \cdot x \leq d \cdot y^0$, therefore x^0 is an optimal solution to the linear program: maximize

$c \cdot x$ subject to (1.1.1)-(1.1.3). Since every feasible matching x of G satisfies (1.1.1)-(1.1.3) it follows that x^0 is the optimal matching we require.

From this it easily follows that $P(G, b)$ is the solution set of (1.1.1)-(1.1.3), for if v is any vertex of the polyhedron defined by (1.1.1)-(1.1.3) then there is some linear objective function maximized over that polyhedron only by v . But we have seen that every linear objective function is maximized by an integer solution of (1.1.1)-(1.1.3), hence all the vertices of this polyhedron are feasible matchings.

The set of inequalities (1.1.3) is generally far larger than is necessary to define $P(G, b)$; as was mentioned if G is bipartite then none of them are necessary. In view of the structure of the vertices of P , the solution set of (1.1.1) and (1.1.2), it has been surmised that all of the constraints (1.1.3) which are really necessary are those for which S is the node set of an odd polygon. Unfortunately, these are generally not enough; if we just add these inequalities to our linear system (1.1.1)-(1.1.2) then we usually introduce new fractional vertices having a more complex structure than those possessed by P . In Chapter 4 of this thesis, by considering the structure of G and the value of b , we prescribe the minimal subset of the inequalities (1.1.3) which must be added to (1.1.1) and (1.1.2) to obtain $P(G, b)$.

Since $P(G, b)$ is of full dimension there is a direct correspondence between the facets of $P(G, b)$ and the

inequalities necessary to define $P(G, b)$, namely

$\{x \in P(G, b) : ax = \alpha\}$ is a facet of $P(G, b)$ if and only if the inequality $ax \leq \alpha$ (or a positive multiple of $ax \leq \alpha$) is necessary to define $P(G, b)$. Thus in Chapter 4 when we characterize the facets of $P(G, b)$ we are in fact prescribing which of the inequalities (1.1.1)-(1.1.3) are necessary to define $P(G, b)$. We prove

Theorem (4.1.2). For every $j \in E$, $\{(x_j : j \in E) \in P(G, b) : x_j = 0\}$ is a facet of $P(G, b)$.

In other words all the constraints (1.1.1) are essential for defining $P(G, b)$.

However, some of the constraints (1.1.2) are not necessary. For any $i \in V$ we let $N(i)$ be the set of nodes of G adjacent to i . If v, w are nodes of G such that $N(v) = \{w\}$, $N(w) = \{v\}$ and $b_w = b_v$ then we call the connected component of G spanned by $\{v, w\}$ a balanced edge.

Theorem (4.2.1). For any $i \in V$, $\{(x_j : j \in E) \in P(G, b) : x(\delta(i)) = b_i\}$ is a facet of $P(G, b)$ if and only if i is a node of a balanced edge

or

$b(N(i)) > b_i$ and if $b(N(i)) = b_i + 1$ then $\gamma(N(i)) = \emptyset$.

A salient feature of the blossom algorithm is the "shrinking" process applied to certain subgraphs of G , effectively reducing the size of the problem under consideration. It is implicit in the blossom algorithm that the set Q in (1.1.3)

can be replaced by the set $Q^0 \equiv \{S \subseteq V : G[S] \text{ is shrinkable}\}$ where $G[S]$ is the subgraph of G induced by S , that is $G[S] \equiv (S, \gamma(S), \psi|_{\gamma(S)})$. We prove that all we need add is a connectivity condition to the condition of shrinkability and we have the essential inequalities of the sort (1.1.3).

Theorem (4.3.46). For any $S \subseteq V$ such that $G[S]$ is shrinkable, $\{x \in P(G, b) : x(\gamma(S)) = q_S\}$ is a facet of $P(G, b)$ if and only if $G[S]$ contains no cutnode v for which $b_v = 1$.

The necessity of our conditions of both Theorem (4.2.1) and Theorem (4.3.46) is proved by constructing $|E|$ affinely independent feasible matchings of G which belong to the facet of $P(G, b)$. We define a near perfect matching of G deficient at $v \in V$ to be a matching x of G which satisfies

$$x(\delta(i)) = b_i \text{ for all } i \in V - \{v\},$$

$$x(\delta(v)) = b_v - 1.$$

A feasible matching x of G will satisfy $x(\gamma(S)) = q_S$ if and only if \bar{x} , the restriction of x to $\gamma(S)$, is a near perfect matching of $G[S]$. Thus when constructing feasible matchings of G which satisfy $x(\gamma(S)) = q_S$, our first step is to be able to construct a large number of near perfect matchings of $G[S]$.

We say that G is b -critical if for every node v of G there is a near perfect matching of G which is deficient at v . These $|V|$ near perfect matchings can be seen to be

linearly independent, but we usually require a much larger set of linearly independent near perfect matchings. However we show that if a graph G is b-critical and contains no cutnode v for which $b_v = 1$ then G has as many linearly independent near perfect matchings as it has edges. This we prove by showing (Theorem (4.4.2)) that a graph G is b-critical if and only if G is shrinkable. We also prove that these conditions are equivalent to G being connected, $b(V)$ being odd and the empty set being the only subset of V which violates Tutte's condition (3.10.34) for the existence of a perfect matching.

Thus we obtain two more facet characterization theorems (4.4.15), (4.4.17). In particular we have the following.

Theorem. For any $S \subseteq V$ such that $b(S)$ is odd and $|S| \geq 3$, $F \equiv \{x \in P(G, b) : x(\gamma(S)) = q_S\}$ is a facet of $P(G, b)$ if and only if

$G[S]$ is b-critical and contains no cutnode v such that $b_v = 1$

or

F is a facet of the sort described in Theorem (4.2.1).

As a result of this theorem we can see very easily that if G is bipartite then none of the inequalities (1.1.3) need be added to define $P(G, b)$, for let S be any subset of V such that $b(S)$ is odd and $|S| \geq 3$. Then there must be a part T of $G[S]$ for which $b(T) < 1/2 b(S)$. Obviously we cannot construct a near perfect matching of

$G[S]$ deficient at a node v belonging to T and consequently $G[S]$ cannot be b-critical.

There is a close relationship between polyhedron theory and min-max theorems; whenever we know a set of linear inequalities sufficient to define a polyhedron, linear programming duality immediately provides us with a min-max theorem and we have already discussed how we use a min-max theorem proved by the blossom algorithm to establish the matching polyhedron. We discuss the min-max theorem proved by the blossom algorithm in Section 3.10 and show how it implies theorems of Berge [B2] and Tutte [T1], [T2], [T3].

When we know the facets of a polyhedron, we are able to obtain a "best possible" min-max theorem. In Theorems (4.4.20) we describe such a theorem. We also show how the min-max theorems proved by the blossom algorithm can be combined with our characterization of b-critical graphs to obtain strengthenings of Tutte's theorems, in particular, we derive the following theorem concerning the existence of perfect 1-matchings (matchings x which satisfy $x(\delta(i)) = 1$ for all $i \in V$).

Theorem (4.4.22) $G = (V, E, \psi)$ has a perfect 1-matching if and only if for every $X \subseteq V$ such that $G[V - X]$ consists of 1-critical components, the number of components of $G[V - X]$ is no greater than $|X|$.

In Theorem (4.5.3) we characterize the vertices of $P(G, b)$ and show that every matching produced by the blossom

algorithm is a vertex of $P(G, b)$. Since the vertex set of $P(G, b)$ is the smallest subset X of $P(G, b)$ such that for any linear function c , $c \cdot x$ is maximized over $P(G, b)$ by a member of X , this shows that the blossom algorithm makes use of as small a subset of $P(G, b)$ as possible when solving matching problems. As we saw in Chapter 2, every member of a bounded polyhedron can be expressed as a convex combination of its vertices, in (4.5.21) we describe an algorithm which will express any feasible matching of G which is not a vertex of $P(G, b)$ as a convex combination of two other members of $P(G, b)$. We also describe how this algorithm can be used to express any $x \in P(G, b)$ as a convex combination of a subset of the vertices of $P(G, b)$.

In Chapter 5 we consider the problem of maximizing $c \cdot x$ over any face F of $P(G, b)$ where $c = (c_j : j \in E)$ is an arbitrary real vector. That is, we are given sets $J \subseteq E$, $W \subseteq V$ and $N \subseteq Q$ and we wish to maximize $c \cdot x$ over all $x = (x_j : j \in E) \in P(G, b)$ which satisfy

$$(1.1.4) \quad x_j = 0 \quad \text{for all } j \in J,$$

$$(1.1.5) \quad x(\delta(i)) = b_i \quad \text{for all } i \in W,$$

$$(1.1.6) \quad x(\gamma(S)) = q_S \quad \text{for all } S \in N.$$

For any $J \subseteq E$, $W \subseteq V$ and $N \subseteq Q$ we let $F(J, W, N) \equiv \{(x_j : j \in E) \in P(G, b) : x \text{ satisfies (1.1.4)-(1.1.6)}\}$. The algorithm proposed to solve this problem consists of two parts. The first part described in Section 5.2 is a preconditioning process which finds sets $J' \subseteq E$, $W' \subseteq V$

and $N' \subseteq Q$ such that $F(J, W, N) = F(J', W', N')$ and N' has the property that for any $S, T \in N'$ such that $S \cap T \neq \emptyset$, either $S \subseteq T$ or $T \subseteq S$. (We call such a family of sets a nested family of sets.) The second part of the algorithm described in Section 5.4, can then be used to solve the equivalent problem. The algorithm is a generalization of the blossom algorithm of Chapter 3 and an upper bound on the amount of work performed by this algorithm in solving a problem maximize $c \cdot x$ over $F(J', W', N') \subseteq P(G, b)$ is of the same order as the amount of work performed by the blossom algorithm in solving $c \cdot x$ over $P(G, b)$.

In Section 5.5 we describe how this problem of maximizing $c \cdot x$ over a face F of $P(G, b)$ can be reduced to the problem of maximizing a new objective function c' over $P(G, b)$. This so called "Big-M" method is attractive theoretically, but in practice the number of significant digits in the components of c' tends to increase rather rapidly and so this method does have limitations as a practical method.

In Section 5.6 we discuss multi-optimization matching problems, matching problems in which we have a sequence c_1, c_2, \dots, c_k of objective functions and wish to solve the following problem. Let $X_0 \equiv P(G, b)$ and for each $i \in \{1, 2, \dots, k\}$ let

$$X_i \equiv \{x \in X_{i-1} : c^i x \text{ is maximized over } X_{i-1}\}.$$

We wish to find a matching $x^* \in X_k$. We show how the face optimization algorithm of this chapter can be used to solve

this sort of problem and various generalizations of this problem.

In Chapter 6 we discuss a post optimality problem. We assume that we know a matching $x^0 \in P(G, b)$ which maximizes $c \cdot x$ over $P(G, b)$ and we wish to find a matching $x^* \in P(G, b')$ which maximizes $c \cdot x$ over $P(G, b')$ where $b' = (b'_i : i \in V)$ is a vector of positive integers. Since the parameters G and c of our original problem are unchanged in the new problem, we would hope that we could make use of x^0 so as to be able to solve the new problem more quickly than by simply reapplying the blossom algorithm.

In this chapter we describe a relatively simple initialization procedure which can be combined with the blossom algorithm when we know x^0 and an optimal dual solution y^0 to the original problem, so that an upper bound on the amount of work performed in finding x^* depends upon the value of $|b - b'|$ in essentially the same way as the upper bound on the amount of work performed by the blossom algorithm depended on the value of b .

Finally, in Chapter 7, we discuss a computer implementation of the blossom algorithm and describe some experimental results.

1.2 Set Theory and General Notation

We use the symbol " \equiv " to indicate a definition and reserve the symbol " $=$ " for denoting the equality of two objects.

If X and Y are sets we denote the union and intersection of X and Y by $X \cup Y$ and $X \cap Y$ respectively. We let $X - Y$ denote the set theoretic difference, that is

$$X - Y \equiv \{x \in X : x \notin Y\}.$$

We denote the empty set by \emptyset . Expressions involving \cup , \cap , $-$ should be evaluated from left to right, thus

$$X \cup Y \cap Z - V$$

should be taken to be

$$((X \cup Y) \cap Z) - V.$$

If R is a set of sets, we will let

$$\cup(R) \equiv \bigcup_{X \in R} X$$

and

$$\cap(R) \equiv \bigcap_{X \in R} X.$$

We let $|X|$ denote the cardinality of X .

We let \mathbb{R} denote the set of real numbers. For any $X \subseteq \mathbb{R}$ we let

$$\max X \equiv \max_{x \in X} x$$

and

$$\min X \equiv \min_{x \in X} x.$$

where $X = (x_i : i \in I)$ is an indexed set of members of \mathbb{R} , we let

$$\Sigma X \equiv \sum_{i \in I} x_i.$$

For any $x \in \mathbb{R}$, $[x]$ denotes the largest integer no greater than x . $[x]$ is sometimes called the floor of x or the integer part of x .

We use $X \subseteq Y$ to denote "X is a subset of Y" and we use $X \subset Y$ to denote "X is a proper subset of Y" (thus $X \neq Y$).

If ψ is a function mapping a set X into a set Y , then for any $S \subseteq X$ we let $\psi|S$ denote the restriction of ψ to S . That is $\bar{\psi} = \psi|S$ is the function mapping S into Y defined by

$$\bar{\psi}(s) \equiv \psi(s) \text{ for all } s \in S.$$

We always use the words maximal and minimal in the sense of set inclusion. Thus if R is a family of sets we say that X is a maximal member of R if there is no $Y \in R$ such that $Y \supseteq X$. Similarly X is a minimal member of R if there is no $Y \in R$ such that $Y \subseteq X$.

We denote the cartesian product of two sets X and Y by $X \times Y$. Thus

$$X \times Y \equiv \{(x, y) : x \in X, y \in Y\}.$$

1.3 Graph Theory.

Standard references on graph theory are Berge [B3], Busacker and Saaty [B5] and Harary [H2]. For our purpose a graph G is an ordered triple (V, E, ψ) where V and E

are finite sets and ψ is a function mapping E into the set of two element subsets of V . The members of V are called nodes, the members of E are called edges, and ψ is called the incidence function. We say that $j \in E$ meets $v \in V$ or j and v are incident if $v \in \psi(j)$. We say that $v, w \in V$ are adjacent if there is $j \in E$ such that $\psi(j) = \{v, w\}$. If $\{v, w\} = \psi(j)$ then v and w are called the ends of j . If H is any graph we let $V(H)$, $E(H)$ and ψ_H denote the node set, edge set and incidence function of H respectively.

(1.3.1) A track τ in $G = (V, E, \psi)$ from v_0 to v_n is a sequence

$$v_0, j_1, v_1, j_2, v_2, \dots, j_n, v_n \text{ for some } n \geq 0$$

such that

$$v_i \in V \text{ for } i \in \{0, 1, \dots, n\},$$

$$j_i \in E \text{ for } i \in \{1, 2, \dots, n\},$$

$$\psi(j_i) = \{v_{i-1}, v_i\} \text{ for } i \in \{1, 2, \dots, n\}.$$

We call n the length of τ , we say that τ is odd or even according as the length of τ is even or odd. We let $E(\tau)$ denote $\{j_i : i \in \{1, 2, \dots, n\}\}$ and $V(\tau)$ denote $\{v_i : i \in \{0, 1, \dots, n\}\}$. For any $j_i \in E(\tau)$ we call j_i an even edge of τ if i is even and an odd edge of τ if i is odd. Edges occurring more than once in τ may be both even and odd.

A track τ induces an ordering on the nodes in $V(\tau)$ and edges in $E(\tau)$. Thus for any $P \subseteq V(\tau)$ we say that v is the first node in $V(\tau) \cap P$ if $s = \min\{i \in \{0, 1, 2, \dots, n\} : v_i \in P\}$ and $v = v_s$. We define last node and first and

last edge analogously.

(1.3.2) A path is a track π of length n for which $|V(\pi)| = n + 1$. In other words, no node occurs more than once.

A path π is said to be maximal with a given property if no other path having that property has π as a subsequence. (Obviously there is no such thing as a maximal track.)

A graph $G = (V, E, \psi)$ is said to be connected if for every $\{v, w\} \in V$ there is a path (track) π in G joining v to w .

A graph H is said to be a subgraph of $G = (V, E, \psi)$ if $V(H) \subseteq V$, $E(H) \subseteq E$ and $\psi_H = \psi|_{E(H)}$. In this case we say that G contains H . A maximal connected subgraph of G is called a component of G .

The distance between nodes v and w belonging to the same component of G is defined to be the length of the shortest path joining v and w .

Let $G = (V, E, \psi)$ be any graph. For any $S \subseteq V$ we let $\delta_G(S)$ denote the coboundary of S , that is

$$(1.3.3) \quad \delta_G(S) \equiv \{j \in E: |S \cap \psi(j)| = 1\}.$$

When S consists of a single element v , then we abbreviate $\delta_G(\{v\})$ by $\delta_G(v)$. For any $v \in V$ we call $|\delta_G(v)|$ the valence of v . For any $S \subseteq V$ we let $\gamma_G(S)$ denote the set of edges of G having both ends in S , thus

$$(1.3.4) \quad \gamma_G(S) \equiv \{j \in E: \psi(j) \subseteq S\}.$$

We abbreviate δ_G and γ_G by δ and γ respectively.

(1.3.5) Let $S \subseteq V$. We let $G[S]$ denote the graph $(S, \gamma(S), \psi|_{\gamma(S)})$. We call $G[S]$ the subgraph of G induced by S .

(1.3.6) A Polygon is a connected graph P such that $|\delta_P(v)| = 2$ for all $v \in V(P)$. If $|E(P)|$ is even then we say that P is an even polygon, otherwise we call P an odd polygon.

(1.3.6a) Let P be a polygon and let $w \in V(P)$. Let τ be a track in P from w to w such that $V(\tau) = V(P)$, $E(\tau) = E(P)$ and the length of τ is as small as possible with this property. We call τ a track from w to w induced by P. Intuitively, τ is the track obtained by travelling once around the polygon P , starting at w .

(1.3.7) A graph $G = (V, E, \psi)$ is bipartite if V can be partitioned into $V_1 \cup V_2$ and $E = \delta(V_1) = \delta(V_2)$. Any $S \subseteq V$ such that $\delta(S) = E$ and $\gamma(S) = \emptyset$ is called a part of G .

(1.3.8) Theorem. (König [K1] p. 170) G is bipartite if and only if G contains no odd polygon.

(1.3.9) A cutnode v of $G = (V, E, \psi)$ is a node $v \in V$ such that $G[V - \{v\}]$ has more components than G . G is nonseparable if G is connected and has no cutnode. A block is a maximal nonseparable subgraph of G . It is easily seen that

(1.3.10) every polygon of G is a subgraph of a

block of G ,

that is, no polygon can have edges from different blocks.

An isthmus of G is an edge $j \in E$ such that $(V, E - \{j\}, \psi|_{E - \{j\}})$ has more components than G .

(1.3.11) A forest is a graph which contains no polygons, a tree is a connected forest. A tree T is said to be trivial if $|V(T)| \leq 1$. The following results are well known.

(1.3.12) Theorem. Every nontrivial tree has at least two nodes of valence 1.

(1.3.13) Theorem. If T is a tree then $|E(T)| = |V(T)| - 1$.

1.4 Linear Algebra.

Let J be a finite set. We let $\mathbb{R}^J \equiv \{(x_j : j \in J) : x_j \in \mathbb{R} \text{ for all } j \in J\}$. We let 0 denote the vector which is zero in every component.

(1.4.1) A set $X \subseteq \mathbb{R}^J$ is said to be linearly independent if whenever $\sum_{x \in X} \alpha_x x = 0$ for some $(\alpha_x \in \mathbb{R} : x \in X)$ we have $\alpha_x = 0$ for all $x \in X$. Otherwise X is linearly dependent.

(1.4.2) Let $X \subseteq \mathbb{R}^J$. A basis of X is a maximal linearly independent subset of X . The following result is well known.

(1.4.3) Theorem. (Birkhoff & MacLane [B4], Ch. 7, §4). All bases of $X \subseteq \mathbb{R}^J$ have the same cardinality called the rank of X , and the rank of X is no greater than $|J|$.

(1.4.4) If $x, y \in \mathbb{R}^J$ we let $x \cdot y$ or xy denote $\sum\{x_j \cdot y_j : j \in J\}$.

(1.4.5) The null space of $X \subseteq \mathbb{R}^J$ is defined to be $\{y \in \mathbb{R}^J : y \cdot x = 0 \text{ for all } x \in X\}$. We define the nullity of X to be the rank of the null space of X . The following is a basic result.

(1.4.6) Theorem. (Birkhoff & MacLane [B4], Ch. VIII, Theorem 11). For any $X \subseteq \mathbb{R}^J$, the rank of X plus the nullity of X equals $|J|$.

(1.4.7) If $x, y \in \mathbb{R}^J$, we say $x \leq y$ if $x_j \leq y_j$ for all $j \in J$. We say $x < y$ if $x_j < y_j$ for all $j \in J$.

(1.4.8) Let I, J be finite sets. If $A \subseteq \mathbb{R}^{I \times J}$ is the matrix $(a_{ij} \in \mathbb{R} : i \in I, j \in J)$ then for any $S \subseteq I$ we let A_S denote $(a_{ij} : i \in S, j \in J)$. Similarly if $b = (b_i : i \in I) \in \mathbb{R}^I$, we denote $(b_i : i \in S)$ by b_S . If S is a single element v we abbreviate $A_{\{v\}}$ by A_v . If $x = (x_j : j \in J) \in \mathbb{R}^J$ we define the product Ax to be the vector $y = (y_i : i \in I) \in \mathbb{R}^I$ where $y_i = A_i \cdot x$ for all $i \in I$.

We define the transpose of A , denoted by A^T to be the

matrix $(a'_{ji} : j \in J, i \in I) \in \mathbb{R}^{J \times I}$ where $a'_{ji} = a_{ij}$ for all $i \in I, j \in J$.

(1.4.9) By the rank of A and nullity of A (written $\text{rank}(A)$, $\text{nullity}(A)$) we mean the rank and nullity respectively of $\{A_i : i \in I\}$ as defined in (1.4.3) and (1.4.5).

We call $\{A_i : i \in I\}$ the rows of A; and $\{(a_{ij} : i \in I) : j \in J\}$ the columns of A.

1.5 Linear Programming

Let I, J be finite sets, let $H \subseteq I$ and let $K \subseteq J$. Let $A \in \mathbb{R}^{I \times J}$, $b \in \mathbb{R}^I$ and $c \in \mathbb{R}^J$. A (primal) linear programming problem is

$$(1.5.1) \quad \text{maximize } c \cdot x$$

for $x \in \mathbb{R}^J$ satisfying

$$(1.5.2) \quad x_K \geq 0,$$

$$(1.5.3) \quad x_{J-K} \text{ unrestricted in sign,}$$

$$(1.5.4) \quad A_H x \leq b_H,$$

$$(1.5.5) \quad A_{I-H} x = b_{I-H}.$$

The dual linear program (Dantzig [D1] p. 126) is the linear program

$$(1.5.6) \quad \text{minimize } b \cdot y$$

for $y \in \mathbb{R}^I$ satisfying

$$(1.5.7) \quad y_H \geq 0,$$

$$(1.5.8) \quad y_{I-H} \text{ unrestricted in sign,}$$

$$(1.5.9) \quad A_K^T y \geq c_K,$$

$$(1.5.10) \quad A_{J-K}^T y = c_{J-K}.$$

Texts on linear programming generally show how a problem of the form (1.5.1)-(1.5.5) or (1.5.6)-(1.5.10) can be reduced to a problem in which $K = J$ and $H = \emptyset$ or $H = I$. (e.g. Dantzig [D1] p. 85-89). The following theorems are then usually proved for problems in these canonical forms. These results can be easily extended to apply to linear programs in the forms (1.5.1)-(1.5.5) or (1.5.6)-(1.5.10).

A vector $x \in \mathbb{R}^J$ satisfying (1.5.2)-(1.5.5) is called a feasible solution to the primal problem. A vector $y \in \mathbb{R}^I$ which satisfies (1.5.7)-(1.5.10) is called a feasible dual solution.

A feasible primal solution x^0 which maximizes $c \cdot x$ for all feasible primal solutions is called an optimal primal solution; an optimal dual solution is defined analogously.

The following is a fundamental theorem of linear programming (See Dantzig [D1] p. 120 Theorem 1).

(1.5.11) Theorem. For any linear programming problem exactly one of the following situations occurs.

- i) There exists no feasible solution.
- ii) For any $\alpha \in \mathbb{R}$ there is a feasible solution x such that $c \cdot x > \alpha$.
- iii) There is an optimal feasible solution.

The following theorems give the relationship between the values of $c \cdot x$ and $b \cdot y$ for primal and dual feasible solutions.

(1.5.12) Weak L.P. Duality Theorem (Dantzig [D1] p. 130)

If x is a feasible primal solution and y is a feasible dual solution then $c \cdot x \leq b \cdot y$.

(1.5.13) Corollary. If for any $\alpha \in \mathbb{R}$ there is a feasible dual solution y such that $b \cdot y \leq \alpha$ then there is no feasible primal solution.

(1.5.14) Strong L.P. Duality Theorem (Dantzig [D1] p. 129 Theorem 1, p. 134, Theorems 2, 3).

If there is a feasible primal solution and an upper bound $c \cdot x$ over for all feasible primal solutions x then there is an optimal primal solution x^0 and an optimal dual solution y^0 and $c \cdot x^0 = b \cdot y^0$.

(1.5.15) Corollary (Farkas' Lemma) (Dantzig [D1] p. 137, Theorem 6.)

Let $A \in \mathbb{R}^{I \times J}$, $b \in \mathbb{R}^I$. There exists $x \in \mathbb{R}^J$ such that $x \geq 0$ and $Ax = b$ if and only if there is no $y \in \mathbb{R}^I$ such that $A^T y \leq 0$ and $b \cdot y > 0$.

The following theorem is used extensively in later chapters. It is the tool used to prove optimality of the solutions produced by the matching algorithms.

↳ (1.5.16) Complementary Slackness Theorem

(Dantzig [D1] p. 135, 136).

A feasible solution x^0 to (1.5.2)-(1.5.5) and a feasible solution y^0 to (1.5.7)-(1.5.10) are optimal if and only if

$$(1.5.17) \frac{x_j^0 > 0 \text{ implies } A_j^T y^0 = c_j}{\text{for all}} \quad \underline{j \in K},$$

$$(1.5.18) \frac{y_i^0 > 0 \text{ implies } A_i x^0 = b_i}{\text{for all}} \quad \underline{i \in H}.$$

Proof. For any feasible solution x to (1.5.2)-(1.5.5) and any feasible solution y to (1.5.7)-(1.5.10) we define

$$(1.5.19) \begin{aligned} f(x, y) &\equiv x \cdot (A^T y - c) + y \cdot (b - Ax) \\ &= x_K (A_K^T y - c_K) + y_H (b_H - A_H x) \end{aligned}$$

$$(1.5.20) \quad = \sum_{j \in K} x_j (A_j^T y - c_j) + \sum_{i \in H} y_i (b_i - A_i x)$$

by (1.5.5) and (1.5.10). By (1.5.2), (1.5.4), (1.5.7) and (1.5.9) every term in (1.5.20) is the product of nonnegative factors so

$$(1.5.21) \quad f(x, y) \geq 0.$$

Moreover,

(1.5.22) $f(x, y) = 0$ if and only if one factor in each term of (1.5.20) is zero.

Simplifying (1.5.19) gives

$$(1.5.23) \quad f(x, y) = b \cdot y - c \cdot x.$$

(Note that (1.5.21) and (1.5.23) together prove (1.5.12)).

If x^0 and y^0 satisfy (1.5.17) and (1.5.18) then by (1.5.22) $f(x^0, y^0) = 0$. Therefore, by (1.5.21) and (1.5.23) x^0 and y^0 are optimal solutions.

If x^0 and y^0 are optimal solutions then by (1.5.13) (Strong L.P. Duality) $b \cdot y^0 = c \cdot x^0$ so by (1.5.23), $f(x^0, y^0) = 0$. Therefore by (1.5.22), x^0 and y^0 must satisfy (1.5.17) and (1.5.18). \square

Notice that the sufficiency of (1.5.17) and (1.5.18) were easily proved, however we required the strong duality theorem of linear programming to prove their necessity. In the applications we make use of complementary slackness in proving optimality of the matchings produced by the blossom algorithm and the face optimization algorithm, all we require is the sufficiency of (1.5.17) and (1.5.18) for the algorithm in fact produces solutions x^0 and y^0 satisfying (1.5.17) and (1.5.18).

1.6 Integer Programming and Good Algorithms.

When studying algorithms it is often desirable to be able to establish an upper bound on the amount of work performed by the algorithm as a function of the size of the problem. An elementary step of an algorithm is any step performed by the algorithm which does not depend on the size of the problem, for example adding two numbers, comparing two numbers, seeing whether an edge of a graph meets a node of a graph. Thus an algorithm will, in solving a problem, perform a certain number of elementary steps. If there is

some constant K such that the number of these elementary steps which can be performed in solving a problem P whose size is measured by the parameters r_1, r_2, \dots, r_n is no greater than $K \cdot f(r_1, r_2, \dots, r_n)$ where f is some function of r_1, r_2, \dots, r_n then we say that an upper bound on the amount of work performed by the algorithm is of the order $f(r_1, r_2, \dots, r_n)$.

In this thesis, when discussing bounds on algorithms, we make a "fixed-word" assumption, namely that the time required to perform arithmetic operations (addition, subtraction, division by two) on two numbers is independent of the number of digits in the numbers. This is the way in which most large computers operate, the number of significant digits to be considered becomes a constraint as to whether or not a problem is solvable rather than a factor in the time taken to solve the problem.

Following the terminology of Edmonds [E1] we call an algorithm "good" if there is an upper bound on the amount of work performed by the algorithm that is of the order $p(r_1, r_2, \dots, r_n)$ where $p(r_1, r_2, \dots, r_n)$ is a polynomial function of r_1, r_2, \dots, r_n .

Consider the problem (1.5.1)-(1.5.5) with the added restriction

$$(1.6.1) \quad x_j \text{ is integer valued for all } j \in J.$$

Such a problem is called an integer programming problem. Although it does not have a polynomial bound, the famous Simplex Algorithm of Dantzig, does provide a practical method of solving reasonably large linear programming problems.

CHAPTER 2

Basic Polyhedral Theory

In this chapter we define polyhedra and develop some of their basic properties which are used in later chapters. In particular we prove two theorems characterizing the facets of a polyhedron which are used extensively in Chapter 4.

This treatment of the subject, suggested by J. Edmonds, is most similar to that of Stoer, Witzgall [S1]. Other standard references are Grünbaum [G1] and Rockafellar [R1]. The advantage of our approach for present purposes is that it tends to emphasize the relationship between polyhedral theory and linear programming and it is in fact this relationship which prompts our interest in special classes of polyhedra.

2.1 Polyhedra and their Faces

Let I and J be finite sets, let

$$A = (a_{ij} : i \in I, j \in J) \in \mathbb{R}^{I \times J} \text{ and let } b = (b_i : i \in I) \in \mathbb{R}^I.$$

We call the set of linear inequalities $Ax \leq b$ a linear system and define a Polyhedron to be the solution set of any linear system. We define the polyhedron

$$P(A, b) \equiv \{x \in \mathbb{R}^J : Ax \leq b\}.$$

We take A , b , I and J to be defined as above throughout the rest of this chapter.

If there is $i \in I$ such that $A_i = 0$ then either $b_i < 0$ in which case $P(A, b) = \emptyset$ or else $b_i \geq 0$ and

$P(A, b) = P(A_{I-\{i\}}, b_{I-\{i\}})$. Therefore we will henceforth assume that $A_i \neq 0$ for all $i \in I$ (that is, the matrix A has no zero rows).

If K is a finite set, $A' \in \mathbb{R}^{K \times J}$ and $b' \in \mathbb{R}^K$ then

$$P \equiv \{x \in \mathbb{R}^J : Ax \leq b, A'x = b'\}$$

is the same set as

$$Q \equiv \{x \in \mathbb{R}^J : Ax \leq b, A'x \leq b', (-A')x \leq -b'\}.$$

Since Q is a polyhedron, we have

(2.1.1) any $P \subseteq \mathbb{R}^J$ which is the solution set of a finite system of linear inequalities and linear equations is a polyhedron.

For any $I' \subseteq I$ we define

$$(2.1.2) f(I') \equiv \{x \in P(A, b) : A_{I'}, x = b_{I'}\}.$$

By (2.1.1) $f(I')$ is a polyhedron and is called a face of $P(A, b)$. The fact that the faces of $P(A, b)$ depend on the polyhedron, not the linear system $Ax \leq b$ is shown in (2.1.5). The empty set is also taken to be a face of every polyhedron.

It is clear that

(2.1.3) every face of a face of a polyhedron P is itself a face of P ,
also,

(2.1.4) the intersection of any collection of faces of a polyhedron P is itself a face of P ; if $I^k \subseteq I$ for $k \in K$ we have $\bigcap_{k \in K} f(I^k) = f(\cup_{k \in K} I^k)$.

There is associated with every linear system $Ax \leq b$ a unique maximal set $I^0 \subseteq I$ for which $P(A, b) = f(I^0)$ (since for any $t \in I$, either there exists $x^t \in P(A, b)$ such that $A_t x^t < b_t$ in which case $t \notin I^0$ or no such x^t exists and $t \in I^0$). We call I^0 the equality set of $Ax \leq b$. We say that I^1 is the equality set of a face F of $P(A, b)$ if I^1 is the maximal subset of I such that $F = f(I^1)$.

It is easily seen that there are many different sets of linear inequalities which define the same polyhedron. However the faces of the polyhedron depend only upon the polyhedron itself and not upon the choice of inequalities. This we now prove by showing that a nonempty subset F of a polyhedron P is a face of P if and only if there is some linear function c which is maximized over P by precisely the members of F .

(2.1.5) Theorem. $F \subseteq P(A, b)$ is a nonempty face of $P(A, b)$ if and only if

(2.1.6) there is $c \in \mathbb{R}^J$ and $\alpha \in \mathbb{R}$ such that
 $cx = \alpha$ for all $x \in F$ and $cx < \alpha$ for all $x \in P(A, b) - F$.

Proof. First we prove the necessity of (2.1.6), let F be a nonempty face of $P(A, b)$, let I^0 be the equality

set of F . Then for each $x \in P(A, b) - F$ there is some $t(x) \in I^0$ such that

$$(2.1.7) \quad A_{t(x)}x < b_{t(x)} .$$

If $I^0 = \emptyset$ we take $c_j \equiv 0$ for all $j \in J$, otherwise take $c_j \equiv \sum(a_{ij} : i \in I^0)$ for all $j \in J$.

For any $x \in F$,

$$\sum(c_j x_j : j \in J) = \sum(a_{ij} x_j : i \in I^0, j \in J) = \sum(b_i : i \in I^0)$$

since I^0 is the equality set of F . For any $x \in P(A, b) - F$ we have

$$\begin{aligned} \sum(c_j x_j : j \in J) &= \sum(a_{ij} x_j : i \in I^0 - \{t(x)\}, j \in J) + \sum(a_{t(x)j} x_j : j \in J) \\ &< \sum(b_i : i \in I^0) \text{ by (2.1.7).} \end{aligned}$$

Thus if we take $\alpha \equiv \sum(b_i : i \in I^0)$, α and c so defined satisfy (2.1.6).

We now prove the sufficiency. Let F be a nonempty subset of P , let c and α be as in (2.1.6). Then the linear program

$$\text{maximize } c \cdot x$$

for

$$Ax \leq b$$

has an upper bound. So by the strong linear programming duality theorem (1.5.14) there is an optimal solution $y^0 (y^0 : i \in I)$ to the dual linear program

$$\text{minimize } b \cdot y$$

$$y \geq 0$$

$$A^T y = c .$$

By complementary slackness (1.5.16) a solution x to $Ax \leq b$ maximizes cx if and only if $A_i x = b_i$ for all $i \in I$ such that $y_i^0 \neq 0$. Thus $F = f(\{i \in I: y_i^0 \neq 0\})$ and the proof is complete. \square

We obtain the following result by combining (2.1.6) and (1.5.10).

(2.1.8) Theorem. Let $c \in \mathbb{R}^J$. If there is $\alpha \in \mathbb{R}$ such that $c \cdot x \leq \alpha$ for all x belonging to a nonempty polyhedron $P(A, b)$ then there is a face F of $P(A, b)$ such that x^0 maximizes $c \cdot x$ for $x \in P(A, b)$ if and only if $x^0 \in F$.

Proof. Since $P(A, b) \neq \phi$ and since $c \cdot x \leq \alpha$ for all $x \in P(A, b)$ it follows from (1.5.11) that there is $x^0 \in P(A, b)$ such that $c \cdot x^0 = \max\{c \cdot x: x \in P(A, b)\}$. Let $F \equiv \{x \in P(A, b): c \cdot x = c \cdot x^0\}$. By (2.1.5) F is a face of $P(A, b)$. \square

Let I^0 be the equality set of $Ax \leq b$. We call $x \in P(A, b)$ an interior point of $P(A, b)$ if $A_{I-I^0} x < b_{I-I^0}$.

(2.1.9) Proposition. Every nonempty polyhedron has an interior point.

Proof. Suppose I^0 is the equality set of $Ax \leq b$ and $P(A, b) \neq \phi$. If $I^0 = I$ then any $x \in P(A, b)$ is trivially an interior point. Otherwise for each $t \in I - I^0$ there must be $x^t \in P(A, b)$ such that

$$\underset{I^0}{A^0} \underset{I^0}{x^t} = \underset{I^0}{b}$$

$$(2.1.10) \quad \underset{I^t}{A_t} \underset{I^t}{x^t} < \underset{I^t}{b_t}$$

$$\underset{I^t}{A_t} \underset{I^t}{x^t} \leq \underset{I^t}{b_t}$$

where $I^t = I - I^0 - \{t\}$ for otherwise t would be in the equality set of $P(A, b)$. Let

$$\bar{x} \in \Sigma(x^t : t \in I - I^0) / |I - I^0|.$$

It follows immediately from (2.1.10) that

$$\underset{I^0}{A^0} \bar{x} = \underset{I^0}{b},$$

$$\underset{I-I^0}{A} \bar{x} < \underset{I-I^0}{b}$$

so \bar{x} is an interior point of $P(A, b)$ as required. \square

2.2 Dimension and a First Facet Characterization

Let $Ax \leq b$ have equality set I^0 . If $P(A, b) = \emptyset$ then we define the dimension of $P(A, b)$ to be -1 .

Otherwise we define the dimension of $P(A, b)$ to be

$$|J| - \text{rank } (\underset{I^0}{A}) .$$

We show in (2.2.14) that dimension depends only on the polyhedron not on the linear system which defines the polyhedron.

We denote the dimension of a polyhedron P by $\dim(P)$. It follows from (1.4.9) and (1.4.3) that if $P \neq \emptyset$, $\dim(P) \geq 0$.

Clearly every polyhedron P is a face of itself

called an improper face. All other faces including the empty face, are called proper faces.

If $\dim(P(A, b)) = |J|$, that is if $P(A, b) \neq \emptyset$ and $\text{rank}(A_{I^0}) = 0$ where I^0 is the equality set of $Ax \leq b$, then we say that $P(A, b)$ is of full dimension.

First we show that the dimension of every proper face of a polyhedron P is less than $\dim(P)$.

(2.2.1) Proposition. Let F be a proper face of $P(A, b)$. Then $\dim(F) \leq \dim(P(A, b)) - 1$.

Proof. Since $P(A, b)$ has a proper face, $P(A, b)$ is nonempty. If $F = \emptyset$ then the result is trivial. Assume $F \neq \emptyset$, let I^0 be the equality set of $Ax \leq b$, let I' be the equality set of F . Then $I^0 \subseteq I'$ and $\text{rank}(A_{I^0}) \leq \text{rank}(A_{I'})$. Suppose

$$(2.2.2) \quad \text{rank}(A_{I^0}) = \text{rank}(A_{I'}) .$$

Then a row basis of A_{I^0} is a row basis of $A_{I'}$, hence for any $t \in I' - I^0$, A_t is a linear combination of rows of A_{I^0} . If b_i is not equal to the same linear combination of the components of b_{I^0} then $F = \emptyset$, contradictory to our assumption. Otherwise, for any $x \in \mathbb{R}^J$ satisfying $A_{I^0}x = b_{I^0}$ we also have $A_t x = b_t$ so $t \in I^0$, contradictory to the choice of t . Hence (2.2.2) must be false,

$$\text{rank}(A_{I^0}) + 1 \leq \text{rank}(A_{I'})$$

and the result now follows from the definition of dimension. \square

Let $\{x^k : k \in K\} \subseteq \mathbb{R}^J$. We say that $x^k : k \in K$ are affinely independent if for any $(\alpha_k \in \mathbb{R} : k \in K)$ such that

$$\sum (\alpha_k x^k : k \in K) = 0$$

and

$$\sum (\alpha_k : k \in K) = 0$$

we have $\alpha_k = 0$ for all $k \in K$. If $x^k : k \in K$ are not affinely independent then we say that they are affinely dependent.

Let $\{x^k : k \in K\} \subseteq \mathbb{R}^J$. We say that $x \in \mathbb{R}^J$ is an affine combination of $\{x^k : k \in K\}$ if there exist $\alpha_k \in \mathbb{R}$ for $k \in K$ such that

$$x = \sum (\alpha_k x^k : k \in K)$$

and

$$\sum (\alpha_k : k \in K) = 1$$

The following is an immediate consequence of these definitions

(2.2.3) Proposition. The vectors $x^k \in \mathbb{R}^J : k \in K$ are affinely independent if and only if no x^h for $h \in K$ is an affine combination of $\{x^k : k \in K - \{h\}\}$.

The following proposition relates affine independence to linear independence.

(2.2.4) Proposition. The vectors $x^k \in \mathbb{R}^J : k \in K$ are affinely independent if and only if for any $h \in K$, the vectors $x^k - x^h : k \in K - \{h\}$ are linearly independent.

Proof. Suppose $x^k : k \in K$ are affinely independent, let $h \in K$ and let $K' \equiv K - \{h\}$. Let $(\alpha_k \in \mathbb{R} : k \in K')$ be such that

$$\sum (\alpha_k (x^k - x^h) : k \in K') = 0$$

Then

$$-\sum (\alpha_k : k \in K') x^h + \sum (\alpha_k x^k : k \in K') = 0$$

and

$$-\sum (\alpha_k : k \in K') + \sum (\alpha_k : k \in K') = 0$$

so since $x^k : k \in K$ are affinely independent we must have $\alpha_k = 0$ for all $k \in K$ and the vectors $x^k - x^h : k \in K - \{h\}$ are linearly independent.

Conversely, suppose that for $h \in K$ the vectors $x^k - x^h : k \in K' \equiv K - \{h\}$ are linearly independent. Let $(\alpha_k \in \mathbb{R} : k \in K)$ be such that

$$(2.2.5) \quad \sum (\alpha_k x^k : k \in K) = 0$$

$$(2.2.6) \quad \sum (\alpha_k : k \in K) = 0.$$

Then by (2.2.6) $\alpha_h = -\sum (\alpha_k : k \in K')$ so (2.2.5) implies

$$-\sum (\alpha_k x^h : k \in K') + \sum (\alpha_k x^k : k \in K') = 0 \text{ or}$$

$\sum (\alpha_k (x^k - x^h) : k \in K') = 0$. Since $(x^k - x^h) : k \in K'$ are linearly independent we have $\alpha_k = 0$ for all $k \in K'$. Hence, by (2.2.6), $\alpha_h = 0$ and so $x^k : k \in K$ are affinely independent and the proof is complete. \square

Note that affine independence is implied by linear independence and affine dependence implies linear dependence.

For $V \subseteq \mathbb{R}^J$ we define the affine rank of V to be the cardinality of a largest affinely independent subset of V . In view of (2.2.4) and (1.4.3),

(2.2.7) the affine rank of $V \subseteq \mathbb{R}^J$ is no greater than $|J| + 1$.

We now prove a theorem which relates the affine rank of a polyhedron to its dimension and thus shows that the dimension of a polyhedron is determined irrespective of the linear system.

(2.2.8) Lemma. If $\dim(P(A, b)) = k$ then $P(A, b)$ contains $k + 1$ affinely independent elements.

Proof. If $k = -1$ then $P(A, b) = \emptyset$ and the result is trivial. Otherwise $k \geq 0$ and $P(A, b) \neq \emptyset$. Let I^0 be the equality set of $Ax \leq b$. By (2.19) $P(A, b)$ has an interior point x which satisfies

$$(2.2.9) \quad A_{I^0}x = b_{I^0},$$

$$(2.2.10) \quad A_{I-I^0}x < b_{I-I^0}.$$

If $k = 0$ then $\{x\}$ is the set of affinely independent elements we require. Suppose $k \geq 1$. Since $\dim(P(A, b)) = k$, $\text{rank}(A_{I^0}) = |J| - k$. Therefore by (1.4.6) $\text{nullity}(A_{I^0}) = k$. Hence there are k linearly independent vectors $y^1, y^2, \dots, y^k \in \mathbb{R}^J$ such that

$$(2.2.11) \quad A_{I^0}^{y^i} = b_{I^0} \quad \text{for } i \in \{1, 2, \dots, k\}.$$

Let $t \in \{1, 2, \dots, k\}$. In view of (2.2.10) there is

$\epsilon_t > 0$ such that

$$A_{I-I^0}(x + \epsilon_t y^t) \leq b_{I-I^0}$$

$$\text{since } A_{I-I^0}(x + \epsilon_t y^t) = A_{I-I^0}x + \epsilon_t (A_{I-I^0}y^t).$$

$$\begin{aligned} \text{Then } A_{I^0}(x + \epsilon_t y^t) &= A_{I^0}x + \epsilon_t A_{I^0}y^t \\ &= b_{I^0} \end{aligned}$$

by (2.2.9) and (2.2.11). Thus the vectors $x, x + \epsilon_1 y^1, x + \epsilon_2 y^2, \dots, x + \epsilon_k y^k$ all belong to $P(A, b)$. Moreover, since y^1, y^2, \dots, y^k are linearly independent and since $\epsilon_t > 0$ for all $t \in \{1, 2, \dots, k\}$, $\epsilon_1 y^1, \epsilon_2 y^2, \dots, \epsilon_k y^k$ are linearly independent. Hence by (2.2.4) $x, x + \epsilon_1 y^1, \dots, x + \epsilon_k y^k$ are affinely independent and the proof is complete. \square

(2.2.12) Lemma. If $P(A, b)$ contains $k+1$ affinely independent members then $\dim(P(A, b)) \geq k$.

Proof. If $k \leq 0$ the result is trivial, assume $k \geq 1$. Let x^0, x^1, \dots, x^k be affinely independent members of $P(A, b)$. Then if I^0 is the equality set of $Ax \leq b$ we have

$$(2.2.13) \quad A_{I^0}^{x^i} = b_{I^0} \quad \text{for } i \in \{0, 1, \dots, k\}.$$

By (2.2.4) the vectors $x^1 - x^0, x^2 - x^0, \dots, x^k - x^0$

are linearly independent. Moreover by (2.2.13)

$$A_{I^0}(x^i - x^0) = A_{I^0}x^i - A_{I^0}x^0$$

$$= b_{I^0} - b_{I^0} = 0$$

for $i \in \{1, 2, \dots, k\}$. Hence $\text{nullity}(A_{I^0}) \geq k$ and so $\text{rank}(A_{I^0}) \leq |J| - k$. Thus $\dim(F) = |J| - \text{rank}(A_{I^0}) \geq k$. \square

We can now combine these two lemmas to obtain the following theorem.

(2.2.14) Theorem. The dimension of $P(A, b)$ is one less than the affine rank of $P(A, b)$.

We showed (2.1.5) that the faces of a polyhedron P are independent of the choice of inequalities used to represent P . A consequence of (2.2.14) is that the dimension of a polyhedron is also independent of the choice of inequalities since the affine rank does not depend on the set of inequalities used to define the polyhedron.

If F is a face of $P(A, b)$ and $\dim(F) = \dim(P(A, b)) - 1$ then F is called a facet of $P(A, b)$.

In Chapter 4 we make extensive use of the following corollary of (2.2.14).

(2.2.15) Corollary. If F is a proper face of a polyhedron P of dimension d then F is a facet of P if and only if F contains d affinely independent elements.

Proof. The result is a combination of (2.2.1) and (2.2.14). \square

2.3 Second Facet Characterization

We prove in this section that the facets of a polyhedron P are precisely the maximal proper faces of P . We also show that the facets of P correspond in a certain sense to a minimal collection of inequalities required to define P . We then discuss the specialization of this theorem to the case in which P is of full dimension as this is the situation which we study in chapter 4.

(2.3.1) Theorem. Let $P \in P(A, b)$ be nonempty and let I^0 be the equality set of $Ax \leq b$. Let $I' \subseteq I - I^0$. Let $P' \in P(A_{I-I'}, b_{I-I'})$. Then $P \neq P'$ if and only if $I' \cup I^0$ contains the equality set of a nonempty proper face of P .

Proof. Clearly $P \subseteq P'$, suppose there is some $y \in P' - P$. Then for some nonempty $K \subseteq I'$ we have

$$(2.3.2) \quad A_{I-K}y \leq b_{I-K}$$

and

$$(2.3.3) \quad A_K y > b_K .$$

By (2.1.9) P has an interior point w , that is, w satisfies

$$(2.3.4) \quad A_{I-I^0}w < b_{I-I^0} ,$$

$$(2.3.5) \quad A_{I^0}w = b_{I^0} .$$

Therefore we can choose $\lambda \in \mathbb{R}$ satisfying $0 < \lambda < 1$ such that if we let $z \equiv \lambda w + (1 - \lambda)y$ then for some nonempty $T \subseteq K$

$$(2.3.6) \quad A_T z = b_T ,$$

$$(2.3.7) \quad A_{I-I^0-T} z < b_{I-I^0-T} ,$$

$$(2.3.8) \quad A_{I^0} z \leq b_{I^0} .$$

(Take $\lambda \equiv \max\{(A_i y - b_i) / (A_i(y - w)): i \in K\}$ and let T be the set of $i \in K$ which attain this maximum).

By (2.3.6) - (2.3.8) $z \in P$ so

$$(2.3.9) \quad A_{I^0} z = b_{I^0} .$$

By (2.3.6) and (2.3.9) $z \in f(I^0 \cup T)$ and by (2.3.7), $I^0 \cup T$ is the equality set of this face. This proves the necessity of our condition, since $I^0 \cup T \subseteq I^0 \cup K \subseteq I^0 \cup I'$.

Conversely, suppose that $I^0 \cup I'$ contains the equality set of a nonempty proper face F of P . Let K be the equality set of F . Note that $I^0 \subset K \subseteq I^0 \cup I'$. By (2.1.9) F has an interior point y , that is, y satisfies

$$(2.3.10) \quad A_K y = b_K$$

$$(2.3.11) \quad A_{I-K} y < b_{I-K} .$$

Similarly P has an interior point w , that is an element w satisfying

$$(2.3.12) \quad A_{I^0} w = b_{I^0} ,$$

$$(2.3.13) \quad A_{I-I^0}^0 w < b_{I-I^0}.$$

For any $\epsilon > 0$ let $z(\epsilon) \equiv (1 + \epsilon)y - \epsilon w$. Then

$$(2.3.14) \quad A_{I^0}^0 z(\epsilon) = b_{I^0} \text{ for any } \epsilon \in \mathbb{R} \text{ by}$$

(2.3.10) and (2.3.12).

$$(2.3.15) \quad A_{K-I^0}^0 z(\epsilon) = b_{K-I^0} + \epsilon(A_{K-I^0}^0 y - A_{K-I^0}^0 w)$$

$$> b_{K-I^0} \text{ for any } \epsilon > 0$$

by (2.3.11) and (2.3.13).

$$A_{I-K} z(\epsilon) = A_{I-K} y + \epsilon \cdot A_{K-I^0}^0 (y - w)$$

so in view of (2.3.11) if we choose $\bar{\epsilon} > 0$ sufficiently small we will have

$$(2.3.16) \quad A_{I-K} z(\bar{\epsilon}) \leq b_{I-K}.$$

Since $K \supset I^0$, by (2.3.15) $z(\bar{\epsilon}) \notin P$. Since $I - K \subseteq I - I'$, by (2.3.14) and (2.3.16) $z(\bar{\epsilon}) \in P' = P(A_{I-I'}, b_{I-I'})$.

That is $P \neq P'$ and the proof is complete. \square

We are now in a position to prove the following theorem equating the facets of a polyhedron to its maximal proper faces.

(2.3.17) Theorem. $F \neq \phi$ is a facet of $P(A, b)$

if and only if F is a maximal proper face of $P(A, b)$.

Proof. Suppose $F \neq \phi$ is a maximal proper face of

$P(A, b)$. Then by (2.2.1)

$$(2.3.18) \quad \dim(F) \leq \dim(P(A, b)) - 1.$$

Let I^0 be the equality set of $Ax \leq b$, let I' be the equality set of F . Let $i \in I' - I^0$ and let $K \subseteq I' - I^0 - \{i\}$.

If $K \cup I^0 (= I' - \{i\})$ contained the equality set of a proper face F' of $P(A, b)$ then $F \subset F'$ contradicting the maximality of F . Thus by (2.3.1),

$$P(A, b) = P(A_{I-K}, b_{I-K})$$

and I^0 is the equality set of $A_{I-K}x \leq b_{I-K}$. The equality set of F in $P(A_{I-K}, b_{I-K})$ is $I^0 \cup \{i\}$ so since $\text{rank}(A_{I^0 \cup \{i\}}) \leq \text{rank}(A_{I^0}) + 1$ we have

$$(2.3.19) \quad \dim(F) \geq \dim(P(A, b)) - 1.$$

Combining (2.3.18) and (2.3.19) we see that F is a facet of $P(A, b)$.

Conversely, suppose that $F \neq \emptyset$ is a facet of $P(A, b)$.

Then

$$(2.3.20) \quad \dim(F) = \dim(P(A, b)) - 1.$$

Suppose that there is a face F' of $P(A, b)$ such that $F \subset F' \subset P(A, b)$. By (2.2.1)

$$(2.3.21) \quad \dim(F') \leq \dim(P(A, b)) - 1.$$

By (2.1.3) F is a face of F' and since we assume $F \subset F'$, F is a proper face of F' . Thus by (2.2.1),

$$(2.3.22) \quad \dim(F) \leq \dim(F') - 1.$$

Combining (2.3.21) and (2.3.22) we have

$$\dim(F) \leq \dim(P(A, b)) - 2$$

a contradiction to (2.3.20) which proves the theorem. \square

It should be noted that the hypothesis $F \neq \phi$ is indeed necessary in (2.3.17) as is shown by the following example.

Let $P = \{(x_1, x_2) \in \mathbb{R}^{1,2} : x_1 + x_2 = 1\}$. Then $\dim(P) = 1$ and ϕ is the only proper face of P . But $\dim(\phi) = -1$ so ϕ is not a facet of P . This also illustrates that there do exist polyhedra having no facets.

(2.3.23) Corollary. Let P be a polyhedron, let $d = \dim(P)$. Let $F \neq \phi$ be a face of P of dimension $k < d$. Then there are faces $F_{k+1}, F_{k+2}, \dots, F_{d-1}$ of P such that

$$F \subset F_{k+1} \subset F_{k+2} \subset \dots \subset F_{d-1} \subset P$$

$$\dim(F_j) = j \text{ for } j \in \{k+1, k+2, \dots, d-1\}$$

Proof. We prove by induction on $d - k$. If $d - k = 1$ then there is nothing to prove. Suppose the result is true when $d - k < t \geq 2$ and assume $d = k + t$. Let F_{d-1} be a maximal proper face of P containing F , that is

$$F \subset F_{d-1} \subset P.$$

Then $F_{d-1} \neq \phi$ so by (2.3.17) $\dim(F_d) = d - 1$. Since $(d - 1) - k < t$ there are by our induction hypothesis faces $F_{k+1}, F_{k+2}, \dots, F_{d-2}$ of F_{d-1} such that

$$F \subset F_{k+1} \subset \dots \subset F_{d-2} \subset F_{d-1}$$

and $\dim(F_j) = j$ for $j \in \{k+1, k+2, \dots, d-2\}$. By (2.1.3) F_j is a face of P for $j \in \{k+1, k+2, \dots, d-2\}$ so the result follows. \square

Given the polyhedron $P(A, b)$ we may wish to find a set $I^* \subset I$ such that $P(A_{I^*}, b_{I^*}) = P(A, b)$ and I^* is minimal with this property. The next theorem characterizes such sets. First we observe the following fact.

(2.3.24) Proposition. Let F_1, F_2, \dots, F_k be the facets of $P(A, b)$, let I^0 be the equality set of $Ax \leq b$ and let I^i be the equality set of F_i for $i \in \{1, 2, \dots, k\}$. Then $I^i \cap I^j = I^0$ for all distinct $i, j \in \{1, 2, \dots, k\}$.

Proof. Let i, j be distinct members of $\{1, 2, \dots, k\}$ and let $K = I^i \cap I^j$. Then $I^0 \subseteq K$. Since $F_i \neq F_j$ and since both are maximal proper faces (by (2.3.17)) there are $x_i \in F_i - F_j$ and $x_j \in F_j - F_i$. Then $x_i, x_j \in f(K)$ so $F_i \neq f(K) \neq F_j$. But $f(K) \supseteq F^i \cup F^j$ so since F^i and F^j are maximal, $f(K) = P(A, b)$ so $K = I^0$, completing the proof. \square

(2.3.25) Theorem. Let $F_i : i \in K$ be the facets of a nonempty polyhedron $P(A, b)$, let I^0 be the equality set of $Ax \leq b$ and let I^i be the equality set of F_i for $i \in K$. Let $I^* \subseteq I$. Then $P(A, b) = P(A_{I^*}, b_{I^*})$ if and only if

$$(2.3.26) \quad \text{rank}(A_{I^0 \cap I^*}) = \text{rank}(A_{I^0})$$

and

$$(2.3.27) \quad (\underline{I^* \cap I^i}) - I^0 \neq \emptyset \text{ for all } i \in K.$$

Proof. Suppose I^* satisfies (2.3.26) and (2.3.27).

Then the rows of $A_{I^0 \cap I^*}$ are a basis of the rows of A_{I^0} .

Hence for any $t \in I^0 - I^*$, A_t must be a linear combination of rows of $A_{I^0 \cap I^*}$ and b_t must be the same linear combination of the rows of $b_{I^0 \cap I^*}$ or we would have

$P(A, b) = \emptyset$. Thus if $x \in \mathbb{R}^J$ satisfies $A_{I^0 \cap I^*}x = b_{I^0 \cap I^*}$ then it also satisfies $A_{I^0}x = b_{I^0}$. Hence

$$(2.3.28) \quad P(\underline{A_{I^*}, b_{I^*}}) = P(\underline{A_{I^0 \cup I^*}, b_{I^0 \cup I^*}}).$$

By (2.3.27), $(I - I^*) \cup I^0$ cannot contain the equality set of a facet of $P(A, b)$ so by (2.3.17) and (2.3.1)

$$(2.3.29) \quad P(\underline{A_{I^0 \cup I^*}, b_{I^0 \cup I^*}}) = P(A, b).$$

Combining (2.3.28) and (2.3.29) proves the sufficiency of (2.3.26) and (2.3.27).

If I^* does not satisfy (2.3.26) then $\dim(P(\underline{A_{I^*}, b_{I^*}})) \geq \dim(P(A, b)) + 1$ so by (2.2.14), $P(\underline{A_{I^*}, b_{I^*}}) \neq P(A, b)$.

If I^* does not satisfy (2.3.27) then $(I - I^*) \cup I^0$ contains the equality set of a proper face of $P(A, b)$ so by (2.3.1), $P(A, b) \neq P(\underline{A_{I^0 \cup I^*}, b_{I^0 \cup I^*}})$. Since $P(A, b) \subseteq P(\underline{A_{I^0 \cup I^*}, b_{I^0 \cup I^*}}) \subseteq P(\underline{A_{I^*}, b_{I^*}})$ the result now follows. \square

If $P(A, b)$ is a polyhedron of full dimension and I^0 is the equality set of $Ax \leq b$ then $\text{rank}(A_{I^0}) = 0$ so since

we assume A has no zero rows, $I^0 = \emptyset$. If I' is the equality set of a facet of $P(A, b)$ then $\text{rank}(A_{I'}) = 1$
so if we define for each $i \in I$

$$p(i) \equiv \{t \in I : A_t = \alpha A_i, b_t = \alpha b_i \text{ for some } \alpha \in \mathbb{R}, \alpha > 0\}$$

then we can easily see that all equality sets of facets are sets of this kind. Moreover for any $i \in I$, for any $t \in p(i)$ we have $f(\{t\}) = f(\{p(i)\})$. Thus (2.3.25) specializes to the following.

(2.3.30) Theorem. Let $P(A, b)$ be a polyhedron of full dimension. Then for any $K \subseteq I$, $P(A, b) = P(A_K, b_K)$ if and only if $K \cap p(i) \neq \emptyset$ for each $i \in I$ such that $f(\{i\})$ is a facet of $P(A, b)$.

(2.3.31) Corollary. Let $P(A, b)$ be of full dimension. Then $K \subseteq I$ is a minimal set such that $P(A, b) = P(A_K, b_K)$ if and only if for each $i \in K$, $f(\{i\})$ is a distinct facet of $P(A, b)$.

We also have the following result.

(2.3.32) Theorem. Let $P(A, b)$ be of full dimension, let $K \subseteq I$ be such that $\{f(i) : i \in K\}$ is the set of facets of $P(A, b)$. Suppose $P(A', b') \supseteq P(A, b)$ where $A' \in \mathbb{R}^{I' \times J}$, $b' \in \mathbb{R}^{I'}$ and I' is a finite set. Then $P(A, b) = P(A', b')$ if and only if

(2.3.33) for each $i \in K$ there are $t \in I'$ and some real $\alpha > 0$ such that $A'_t = \alpha \cdot A_i$ and $b'_t = \alpha b_i$.

Proof. Assume $I' \cap I = \emptyset$, define $\bar{A} \in \mathbb{R}^{(I' \cup I) \times J}$ and $\bar{b} \in \mathbb{R}^{I' \cup I}$ by $\bar{A}_I \equiv A$, $\bar{A}_{I'} \equiv A'$, $\bar{b}_I \equiv b$, $\bar{b}_{I'} \equiv b'$. Suppose $P(A, b) = P(A', b')$. Then $P(\bar{A}, \bar{b}) = P(A', b')$ and $\{f(i) : i \in K\}$ is the set of facets of $P(\bar{A}, \bar{b})$. Hence by (2.3.30) (taking \bar{A}, \bar{b} for A, b and I' for K) we see that (2.3.33) must hold.

Conversely, suppose (2.3.33) holds. By (2.3.30), $P(A, b) = P(A_K, b_K)$. Since $P(A', b') \supseteq P(A, b) = P(A_K, b_K)$, (2.3.33) clearly implies $P(A', b') = P(A_K, b_K) = P(A, b)$ and the proof is complete. \square

(2.3.32) shows that the facets of a full dimensional polyhedron $P(A, b)$ determine up to a positive multiple the minimal set of inequalities of which the polyhedron is the solution set. That is, any set of inequalities defining $P(A, b)$ must contain a positive multiple of $A_i x \leq b_i$ for each i such that $f(\{i\})$ is a facet of $P(A, b)$. (2.3.31) shows that the converse also holds, if $Ax \leq b$ is a minimal set of inequalities defining a full dimensional polyhedron P , then $f(\{i\})$ is a facet of P for each $i \in I$.

This is one of the reasons for our interest in the facets of matching polyhedra. These polyhedra (see section 3.4) can be defined for a graph G by a set of inequalities which generally is far from being minimal. By characterizing the facets of matching polyhedra we are characterizing the minimal sets of inequalities necessary and sufficient to determine these polyhedra.

It may happen (as is the case with matching polyhedra) that $a_{ij} = 0$ or 1 for all $i \in I$ and $j \in J$. Then we

have

$$p(i) = \{t \in I : A_i = A_t \text{ and } b_i = b_t\}$$

and we can simplify (2.3.30) as follows.

(2.3.34) Theorem. Let $P(A, b)$ be of full dimension, suppose $a_{ij} \in \{0, 1\}$ for all $i \in I, j \in J$. Then for any $K \subseteq I$, $P(A, b) = P(A_K, b_K)$ if and only if for each $i \in I$ such that $f(\{i\})$ is a facet of $P(A, b)$ there is $t \in K$ such that $A_i = A_t$ and $b_i = b_t$.

2.4 Vertices of Polyhedra.

In this section we prove results about vertices of polyhedra which indicate their importance to linear programming. We also show that bounded polyhedra are convex combinations of their vertices.

We say that $\hat{x} \in P$ is a vertex of the polyhedron P if $\{\hat{x}\}$ is a face of P and $\dim(\{\hat{x}\}) = 0$.

(2.4.1) Theorem. \hat{x} is a vertex of $P(A, b)$ if and only if there is some $c \in \mathbb{R}^J$ such that \hat{x} is the unique member of P maximizing cx for $x \in P$.

Proof. Any two distinct members of \mathbb{R}^J are easily seen to be affinely independent so $F \subseteq P(A, b)$ is a face of $P(A, b)$ of dimension 0 if and only if F is a face of $P(A, b)$ and $|F| = 1$. By (2.1.5) F is a nonempty face of $P(A, b)$ if and only if there is $c \in \mathbb{R}^J$ such that cx is maximized over $P(A, b)$ by precisely the members of F . The result follows from these two facts. \square

We say that a polyhedron $P \subseteq \mathbb{R}^J$ is bounded if there exist $\ell, u \in \mathbb{R}^J$ such that $\ell \leq x \leq u$ for all $x \in P$.

A bounded polyhedron is commonly called a polytope (see Grunbaum [G1]).

(2.4.2) Theorem. Let $P(A, b)$ be a nonempty bounded polyhedron. Then $P(A, b)$ has a vertex.

Proof. Let I' be the equality set of a nonempty face F of $P(A, b)$ of minimum dimension. If $\dim(F) = 0$ then F consists of a vertex and we are finished. Otherwise if $\dim(F) > 0$ then there are by (2.1.8) an interior point x of F and by (2.2.8) an element $y \in F - \{x\}$. For any $\epsilon \in \mathbb{R}$ let $z(\epsilon) \equiv x + \epsilon \cdot (y - x)$. Then $A_{I'} z(\epsilon) = b_{I'}$, for all $\epsilon \in \mathbb{R}$. If $A_{I-I'}(y - x) \leq 0$ then $z(\epsilon) \notin P(A, b)$ for all $\epsilon \in \mathbb{R}$ such that $\epsilon \geq 0$ which contradicts $P(A, b)$ being bounded. Therefore there is $i \in I - I'$ such that $A_i(y - x) > 0$. Let $\lambda^* = \min\{\frac{b_i - A_i x}{A_i(y-x)} : i \in I - I'\}$ and $A_i(y - x) > 0\}$. Then $z(\lambda^*) \in F$ and there is $i \in I - I'$ such that $A_i z(\lambda^*) = b_i$. Since $x \in F - F(I' \cup \{i\})$, $f(I' \cup \{i\})$ is a proper face of F , since $z(\lambda^*) \in f(I' \cup \{i\})$, $f(I' \cup \{i\}) \neq \emptyset$. By (2.2.1) $\dim(f(I' \cup \{i\})) \leq \dim(F) - 1$ and by (2.1.3) $f(I' \cup \{i\})$ is a face of $P(A, b)$ contradicting our choice of F . Hence $\dim(F) = 0$ and F consists of a vertex of $P(A, b)$. \square

Since any face of a bounded polyhedron is itself a bounded polyhedron, we have the following corollary.

(2.4.3) Corollary. Every nonempty face of a bounded polyhedron contains a vertex.

Observe that by (2.1.5) if $c \in \mathbb{R}^J$ is such that cx has an upper bound for $x \in P(A, b)$, then this upper bound is achieved by precisely, the members of some nonempty face of $P(A, b)$.

By combining this, (2.4.3), and the fact that for any $c \in \mathbb{R}^J$, $c \cdot x$ has an upper bound over a bounded polyhedron we obtain the following.

(2.4.5) Theorem. Let P be a nonempty bounded polyhedron. Then for any $c \in \mathbb{R}^J$, there is a vertex v of P which maximizes $c \cdot x$ over P .

Let K be a finite set, let $\{x^k : k \in K\} \subseteq \mathbb{R}^J$. We say that x is a convex combination of $\{x^k : k \in K\}$ if there is $(\lambda_k : k \in K) \in \mathbb{R}^K$ such that

$$(2.4.6) \quad \lambda_k \geq 0 \quad \text{for all } k \in K,$$

$$(2.4.7) \quad \sum \lambda_k = 1,$$

$$(2.4.8) \quad x = \sum (\lambda_k x^k : k \in K).$$

A set $X \subseteq \mathbb{R}^J$ is convex if every convex combination of every finite subset of X belongs to X .

(2.4.9) Proposition. Polyhedra are convex.

Proof. Let $P(A, b)$ be a polyhedron. If $P(A, b) = \emptyset$ then the result is trivial. If $P(A, b) \neq \emptyset$ let $X = \{x^k : k \in K\}$ be a finite subset of $P(A, b)$ and let x be a convex combination of X . Then there is $(\lambda_k : k \in K) \in \mathbb{R}^K$ satisfying (2.4.6)-(2.4.8). Hence

$$\begin{aligned}
 Ax &= \sum (\lambda_k A x^k : k \in K) \\
 &\leq \sum (\lambda_k : k \in K) b \quad \text{by (2.4.6)} \\
 &= b \quad \text{by (2.4.7)}
 \end{aligned}$$

so $x \in P(A, b)$ and (2.4.9) follows. \square

If $V \subseteq \mathbb{R}^J$ then the convex hull of V is defined to be the set of all $x \in \mathbb{R}^J$ which are convex combinations of finite subsets of V .

(2.4.10) Theorem. If $P(A, b)$ is a nonempty bounded polyhedron then $P(A, b)$ is equal to the convex hull of its set of vertices.

Proof. Let $V = \{v_k : k \in K\}$ be the set of vertices of $P(A, b)$. Let $H(V)$ denote the convex hull of V . It follows from (2.4.9) that $H(V) \subseteq P(A, b)$.

Let $\bar{x} \in P(A, b)$. Then $\bar{x} \in H(V)$ if and only if there exists $\lambda = (\lambda_k : k \in K) \in \mathbb{R}^K$ satisfying (2.4.6), (2.4.7) and

$$\bar{x} = \sum (\lambda_k v_k : k \in K).$$

Suppose no such λ exists. Then by Farkas' Lemma (1.5.15) there are $y \in \mathbb{R}^J$ and $y^0 \in \mathbb{R}$ such that

$$(2.4.11) \quad y \cdot v_k + y_0 \leq 0 \quad \text{for } k \in K$$

$$(2.4.12) \quad y \cdot \bar{x} + y_0 > 0.$$

Since $P(A, b)$ is bounded, by (2.4.5)

there is $\alpha \in \mathbb{R}$ such that $\alpha = \max\{y \cdot x : x \in P(A, b)\}$ and there is $h \in K$ such that $y \cdot v_h = \alpha$. By (2.4.11) $\alpha \leq -y_0$ so since $\bar{x} \in P(A, b)$, $y \cdot \bar{x} \leq \alpha \leq -y_0$ contradictory to (2.4.12). This completes the proof. \square

The number of vertices of a polyhedron is generally much larger than the dimension of the polyhedron. The following theorem due to Carathéodory [C1] shows that if x belongs to the convex hull of $S \subseteq \mathbb{R}^J$ then if r is the affine rank of S , x can be expressed as a convex combination of at most r members of S .

(2.4.13) Carathéodory's Theorem. Let r be the affine rank of $S \subseteq \mathbb{R}^J$, let x be a member of the convex hull of S . Then there is $Y \subseteq S$ such that $|Y| \leq r$ and x is a convex combination of the members of Y .

Proof. See Stoer Witzgall [S1] p. 35.

We combine (2.4.12) with (2.4.10) and (2.2.14) to obtain

(2.4.14) Theorem. Let P be a bounded polyhedron of dimension $d \geq 0$. Then any $x \in P$ can be expressed as a convex combination of a set of at most $d + 1$ vertices of P .

Chapter 3

The Matching Problem and the Blossom Algorithm

In this chapter we describe the matching problem considered here and give a new version of the so-called blossom algorithm for solving this problem. This algorithm, which is used extensively in later chapters, is actually a combination of several other versions of the blossom algorithm. The relationship of this version to other available versions is discussed later, when sufficient terminology has been developed.

3.1 The Matching Problem.

Let V and E be finite sets, let $V^{\leq} \cup V^=$ be a partition of V . Let $c = (c_j : j \in E)$ be an arbitrary real vector, let $b = (b_i : i \in V)$ be a vector of positive integers. Let $A = (a_{ij} : i \in V, j \in E)$ be a matrix of zeros and ones which satisfies

$$(3.1.1) \quad \sum(a_{ij} : i \in V) = 2 \quad \text{for all } j \in E.$$

Then the matching problem under consideration is the following problem.

Find, if one exists, a vector $x = (x_j : j \in E) \in \mathbb{R}^E$ such that x_j is a nonnegative integer for all $j \in E$,

$$\sum(a_{ij}x_j : j \in E) \leq b_i \quad \text{for all } i \in V^{\leq},$$

$$\sum(a_{ij}x_j : j \in E) = b_i \quad \text{for all } i \in V^=.$$

and which maximizes $c \cdot x$ subject to these conditions.

If no such vector exists then we wish to exhibit a structure which will prove that no such vector exists.

The matching problem is, therefore, a special case of the integer programming problem (see section 1.6), the principal restriction being (3.1.1). However whereas all known algorithms for solving general integer programming problems have bounds which are exponential in the size of the input, the blossom algorithm is a method for solving matching problems whose bound is a polynomial function of the size of the input. The description of the algorithm is facilitated by interpreting the problem graphically in the following manner.

Let G be the graph (V, E, ψ) where ψ is defined by

$$\psi(j) = \{i \in V : a_{ij} = 1\} \text{ for all } j \in E.$$

In view of (3.1.1), $|\psi(j)| = 2$ for all $j \in E$. Thus G is a graph without loops having edge set E and node set V . Then the matching problem is

$$(3.1.2) \quad \text{maximize } c \cdot x$$

where

$$(3.1.3) \quad x_j \geq 0 \quad \left. \right\} \text{for all } j \in E$$

$$(3.1.4) \quad x_j \text{ integer valued}$$

$$(3.1.5) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V^{\leq}$$

$$(3.1.6) \quad x(\delta(i)) = b_i \quad \text{for all } i \in V^=$$

(See (1.3.3), (1.3.4) for the definitions of γ , δ). That is, we wish to assign a nonnegative integer x_j to each edge

j of G so that the constraints (3.1.5) and (3.1.6) are satisfied and so that $c \cdot x$ is maximized.

Throughout the remainder of this chapter $G = (V, E, \psi)$ is a graph, $b = (b_i : i \in V)$ is a vector of positive integers called degree constraints, $c = (c_j : j \in E)$ is an arbitrary real vector and $V^{\leq} \cup V^=$ is a partition of V .

The purpose of this chapter is to describe an algorithm, called the blossom algorithm, for solving the problem (3.1.2)-(3.1.6).

It is a version of Edmonds' blossom algorithm. In [E1] and [E3] are versions of the algorithm which solve the problem of maximizing $x(E)$ subject to x satisfying (3.1.3)-(3.1.5) taking $b_i \equiv 1$ for all $i \in V$ and $V^= \equiv \emptyset$.

Another version [E2] solves the more general problem (3.1.2)-(3.1.5) where $b_i \equiv 1$ for all $i \in V$ and $V^= \equiv \emptyset$.

The description of the blossom algorithm in this chapter is based upon a version of the algorithm [E4] which solves the problem (3.1.2)-(3.1.6) taking $V^= \equiv V$ and allowing the b_i to be arbitrary positive integers.

This algorithm has been generalized (Johnson [J1], Edmonds, Johnson [E5] and [E6]) in other directions from those considered in this thesis. In addition a computer implementation of a generalized algorithm is available (Edmonds, Johnson, Lockhart [E7]).

We call any $x \in \mathbb{R}^E$ satisfying (3.1.3) and (3.1.4) a matching. If x also satisfies (3.1.5) and (3.1.6) then x is called a feasible b-matching or simply a feasible matching.

If x is a matching such that $x(\delta(i)) = b_i$ for all $i \in S \subseteq V$ then we say that x is a perfect matching of S ; if $S = V$ then we may simply call x a perfect matching of G . For any matching x and any node i we define the deficiency of x at i to be $b_i - x(\delta(i))$. If x has a positive deficiency at i then we say that x is deficient at i . If x is deficient at i then sometimes we call i a deficient node relative to x . Thus x is a perfect matching of $S \subseteq V$ if S contains no deficient nodes relative to x . In Chapter 4 we will study extensively matchings having a deficiency of 1 at some node of G and having a deficiency of 0 every other node, the so-called near perfect matchings.

If $b_i = 1$ for all $i \in V$ then if x is a feasible matching, $M = \{j \in E: x_j = 1\}$ is a set of edges of G meeting each node of G at most once and each node of V exactly once. This special case has received a great deal of attention and often is the starting point for studies of matching theory (e.g. Berge [B2], Edmonds [E1], [E2], [E3], Tutte [T2]). We call this problem the 1-matching problem and call such a vector x a feasible 1-matching. Several of our theorems of chapter 4 are particularly interesting for the case of 1-matchings.

(3.1.7) Proposition. Let x be a matching of G which satisfies

(3.1.8) $x(\delta(i)) \leq b_i$ for all $i \in V$.

Then for any $S \subseteq V$ such that $b(S)$ is odd,

$$(3.1.9) \quad \underline{x(\gamma(S))} \leq \frac{b(S) - 1}{2} .$$

Proof. By (3.1.8) $\sum(x(\delta(i)) : i \in S) \leq b(S)$ and since $\sum(x(\delta(i)) : i \in S) = 2x(\gamma(S)) + x(\delta(S))$ it follows that

$$2x(\gamma(S)) \leq b(S) - x(\delta(S)) \leq b(S) .$$

Since $x(\gamma(S))$ is integer valued and $b(S)$ is odd it follows that

$$2x(\gamma(S)) \leq b(S) - 1$$

and (3.1.9) is immediate. \square

The sets $S \subseteq V$ for which $b(S)$ is odd play an important role in matching theory where G is not bipartite. For any such set S we define

$$(3.1.10) \quad q_S \equiv (b(S) - 1)/2 .$$

The following are two basic results concerning graphs of particularly simple structure. Notice that in both (3.1.11) and (3.1.16) we neither postulate d nor require x to be integer valued or nonnegative.

(3.1.11) Proposition. For any tree T , for any $d = (d_i : i \in V(T)) \in \mathbb{R}^{V(T)}$, for any $v \in V(T)$ there is a unique $x \in \mathbb{R}^{E(T)}$ such that

$$(3.1.12) \quad \underline{x(\delta_T(i)) = d_i} \text{ for all } i \in V(T) - \{v\} .$$

Proof. We prove by induction on $|V(T)|$. If $|V(T)| = 1$ or 2 the result is trivial. Assume the result

true for trees having fewer than k nodes, for $k \geq 3$ and assume $|V(T)| = k$. By (1.3.12) T has a node t of valence 1 different from v , let $\{j\} = \delta_T(t)$. Clearly

$$(3.1.13) \quad x(\delta_T(t)) = d_t \text{ if and only if } x_j = d_t.$$

Let T' be the tree obtained from T by deleting j and t , let w be the end of j in T' . Define d' by

$$\begin{aligned} d'_i & \text{ for } i \in V(T') - \{w\} \\ d'_i & \quad d_w - d_t \text{ if } i = w. \end{aligned}$$

Since $|V(T')| < k$, by our induction hypothesis

(3.1.14) there is a unique $x' \in \mathbb{R}^{E(T')}$ such that $x'(\delta_{T'}(i)) = d'_i$ for all $i \in V(T') - \{v\}$.

Define $x = (x_h : h \in E(T))$ by

$$\begin{aligned} x'_h & \text{ for } h \in E(T') = E(T) - \{j\}, \\ (3.1.15) \quad x_h & \equiv \\ d_t & \text{ for } h = j. \end{aligned}$$

By (3.1.13)-(3.1.15), x is the unique member of $\mathbb{R}^{E(T)}$ satisfying (3.1.12). \square

(3.1.16) Proposition. Let B be a connected graph containing no even polygon and one odd polygon P . Then for any $d = (d_i : i \in V(B)) \in \mathbb{R}^{V(B)}$ there is a unique $x \in \mathbb{R}^{E(B)}$ such that

$$(3.1.17) \quad x(\delta(i)) = d_i \text{ for all } i \in V(B).$$

Proof. Let $j \in E(P)$, let B' be the graph obtained from

B' by removing j . Then B' is a tree and so is bipartite, let u, v be the ends of j , let V_1 be the part see (1.3.7) of B containing $\{u, v\}$, let V_2 be the other part. Let $d' = (d'_i : i \in V(B))$ be defined by

$$\begin{aligned} d'_i &\equiv d_i & \text{for } i \in V(B) - \{u, v\} \\ d'_i &\equiv d_i - 1/2(d(V_1) - d(V_2)) & \text{for } i \in \{u, v\}. \end{aligned}$$

Then

$$(3.1.18) \quad d'(V_1) = d'(V_2).$$

By (3.1.11) there is a unique $x' \in \mathbb{R}^{E(B')}$ such that $x'(\delta_B, (i)) = d'_i$ for all $i \in V(B) - \{u\}$. By (3.1.18) we have $x'(\delta_B, (u)) = d'_u$ so if we define $x \in \mathbb{R}^{E(B)}$ by

$$\begin{aligned} x'_h &\equiv d'_h & \text{for } h \in E(B') = E(B) - \{j\}, \\ x_h &\equiv 1/2(d(V_1) - d(V_2)) & \text{for } h = j \end{aligned}$$

then x satisfies (3.1.17) as required.

Conversely, suppose $\bar{x} \in \mathbb{R}^{E(B)}$ satisfies (3.1.17). B' is bipartite so we have $\bar{x}(\delta_B, (V_1)) = \bar{x}(\delta_B, (V_2))$. Therefore we must have $\bar{x}_j = 1/2(d(V_1) - d(V_2))$. Therefore $\bar{x}|_{E(B')}$ satisfies $\bar{x}(\delta_B, (i)) = d'_i$ for all $i \in V(B) - \{u\}$ so $\bar{x}|_{E(B')} = x'$ by (3.1.11). Therefore $\bar{x} = x$ proving the uniqueness of x . \square

The following six sections (3.2-3.7) are used to develop the general framework required to describe the blossom algorithm. The algorithm itself is presented in Section 3.8 and in Section 3.9 we compute a bound on the amount of work

required by the algorithm to solve a problem.

3.2 Nested Families of Sets.

Let R be a set of distinct nonempty subsets of V . We say that R is a nested family if for any distinct $S, T \in R$ such that $S \cap T \neq \emptyset$ we have $S \subset T$ or $T \subset S$. An important feature of nested families (of which we make use in establishing upper bounds on the amount of work required by various algorithms) is that they are small compared to the total number of subsets of V .

(3.2.1) Theorem. If R is a nested family of subsets of a nonempty set V then $|R| \leq 2|V| - 1$.

Proof. We prove by induction on $|V|$. If $|V| = 1$ then the result is obvious. Suppose the result is true when $|V| < k$ for some $k \geq 2$ and suppose $|V| = k$. Let R be a nested family of subsets of V for which $|R|$ is as large as possible. Since $S \subseteq V$ for all $S \in R$ we must have $V \in R$ or $R \cup \{V\}$ would be a larger nested family. We must also have

(3.2.2) $\{x\} \in R$ for every $x \in V$,

for if there is $x \in V$ such that $\{x\} \notin R$, then $R \cup \{x\}$ is easily seen to be a larger nested family.

Let v_1, v_2, \dots, v_t be the maximal members of $R - \{V\}$. Since $|V| \geq 2$, since the members of R are distinct and by (3.2.2),

(3.2.3) $t \geq 2$.

For each $i \in \{1, 2, \dots, t\}$ let $R(V_i) = \{S \in R: S \subseteq V_i\}$.

Then $R = \bigcup_{i=1}^t R(V_i) \cup \{V\}$ and $V = \bigcup_{i=1}^t V_i$. By our induction

hypothesis $|R(V_i)| \leq 2|V_i| - 1$ for $i \in \{1, 2, \dots, t\}$.

Since $\bigcup_{i=1}^t R(V_i) \cup \{V\}$ partitions R ,

$$|R| = \sum_{i=1}^t |R(V_i)| + 1$$

$$\leq 2 \sum_{i=1}^t |V_i| - t + 1$$

$$\leq 2|V| - 1$$

by (3.2.3) and since $\bigcup_{i=1}^t V_i$ partitions V . The theorem now

follows by induction. \square

If we prohibit singletons from our nested family then we have the following bound.

(3.2.4) Theorem. Let R be a nested family of subsets of V containing no singletons. Then $|R| \leq |V| - 1$.

Proof. Let R' be the family $R \cup \bigcup_{v \in V} \{v\}$. R' is easily seen to be a nested family, by (3.2.1) $|R'| \leq 2|V| - 1$. Since $|R'| = |R| + |V|$ it follows that $|R| \leq |V| - 1$. \square

If R is a nested family of subsets of V then for each $S \in R$ we let

(3.2.5) $R_S = \{T \in R: T \text{ is a maximal proper subset of } S \text{ belonging to } R\}$

and

$$(3.2.6) \quad V_S = \{v \in S : v \notin u(R_S)\} .$$

We let

$$(3.2.7) \quad n(S) = |R_S| + |V_S| .$$

Thus $n(S)$ is the number of maximal "things" which are combined to form S .

(3.2.8) Theorem. Let R be a nested family of subsets of V for which $n(S) \geq 3$ for all $S \in R$. Then $|R| \leq (|V| - 1)/2$.

Proof. Let $S \in R$. If $|V_S| \geq 2$, then let S' be any two members of V_S . If $|V_S| \leq 1$ then since $n(S) \geq 3$, $|R_S| \geq 2$. In this case let S' be the union of any two members of R_S . Let $R' = R \cup \{S' : S \in R\}$. Then $|R'| = 2|R|$. Moreover R' is a nested family containing no singletons so $|R'| \leq |V| - 1$ by (3.2.4). Therefore $|R| \leq 1/2(|V| - 1)$ and the proof is complete. \square

3.3 Blossoms, Shrinking and Shrinkable Families

One feature of the blossom algorithm is the way it "shrinks" certain subgraphs of a graph to effectively reduce the size of the problem. In this section we define shrinking and describe the sorts of subgraphs which will be shrunk. We also prove some fundamental results concerning shrinkable graphs. The definitions and results of this section are also used in Chapter 4 where we show the close relationship between

5.1.1

shrinkable graphs and facets of the matching polyhedron.

The basic structure used in defining shrinkable graphs is the blossom (the christening feature of the blossom algorithm) which is defined as follows.

A blossom is a connected graph B containing no even polygons, exactly one odd polygon P and for which the degree constraints satisfy the following conditions. Let $v \in V(P)$. By (3.1.16) there is a unique $x \in \mathbb{R}^J$ such that

$$(3.3.1) \quad x(\delta_B(i)) = b_i \text{ for all } i \in V(B) - \{v\}$$

$$(3.3.2) \quad x(\delta_B(v)) = b_v - 1.$$

In order that B be a blossom we require

$$(3.3.3) \quad x_j \text{ be a nonnegative integer for all } j \in E(B),$$

$$(3.3.4) \quad x_j \geq 1 \text{ for all } j \in E(B) - E(P)$$

$$(3.3.5) \quad x_j \geq 1 \text{ for each } j \in E(P) \text{ such that } j \text{ is the first edge in the even length path in } P \text{ from a node } i \in V(P) - \{v\} \text{ to } v.$$

The choice of v is in fact arbitrary, we will show in (3.3.12) that if (3.3.1)-(3.3.5) hold for some $v \in V(P)$ then they also hold for any other choice of $v \in V(P)$.

In order that (3.3.1)-(3.3.3) hold we require

$$(3.3.6) \quad b(V(B)) \text{ is odd for any blossom } B.$$

Since we obtain a tree if we delete any $j \in E(P)$ from B , we have using (1.3.13) that

$$(3.3.7) \quad |V(B)| = |E(B)| \quad \text{for any blossom } B.$$

The graph obtained from B by deleting all edges of P is a forest, each $v \in V(P)$ belongs to a unique (possibly trivial) tree T_v of the forest. These trees are called the petals of the blossom, T_v is the petal rooted at v . The edges belonging to $E(B) - E(P)$ are called the petal edges of the blossom.

(3.3.8) If $v \in V(B)$ has valence 1, or has valence 2 and belongs to $V(P)$ then v is called a terminal node of B .

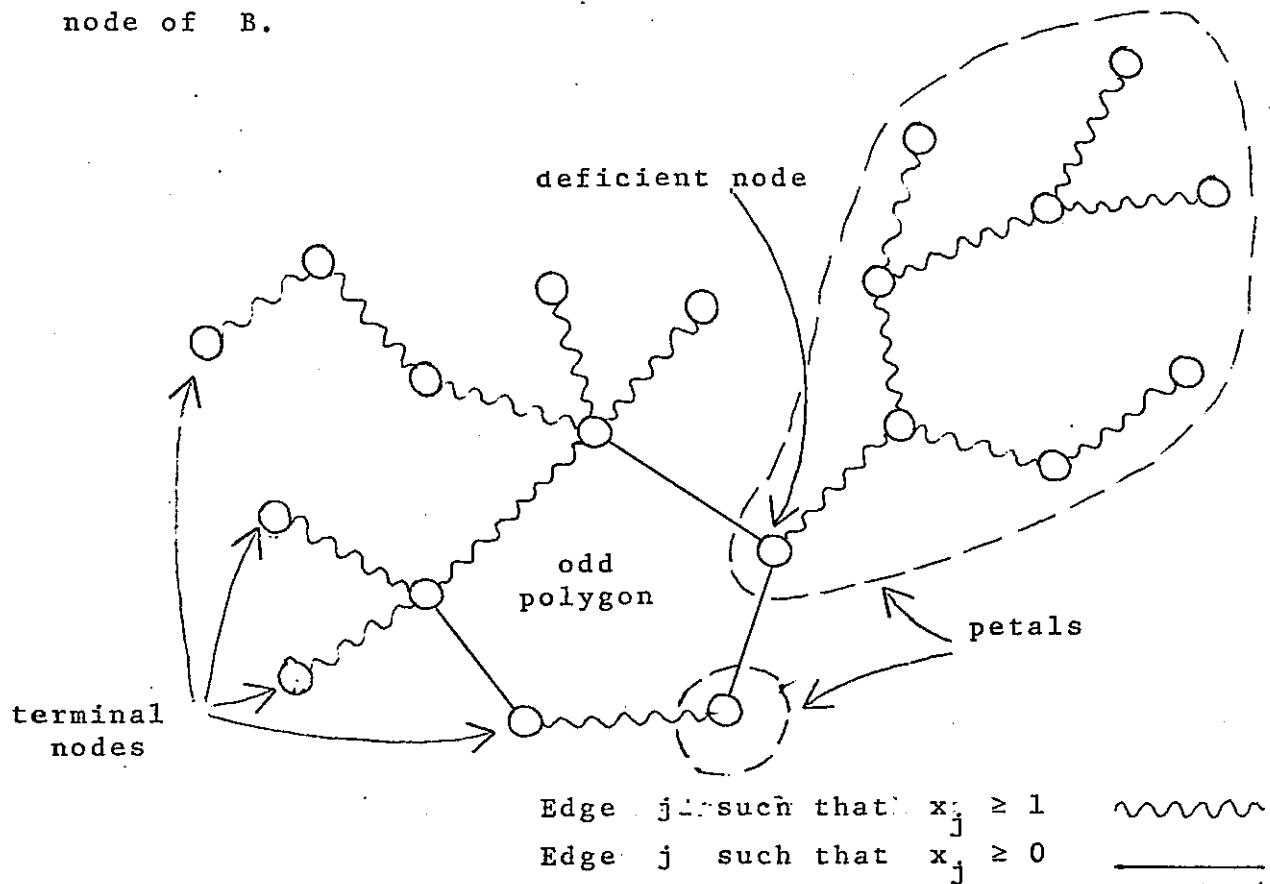


Figure 3.1 Sample Blossom

(3.3.9) Proposition. Let B be a blossom, let $i \in V(B)$ be such that $b_i = 1$. Then i is a terminal node.

Proof. If $i \in V(P)$ then by (3.3.1)-(3.3.5) we must have $b_i \geq 1 + |\delta_B(i) \cap E(T_i)|$. Hence $b_i = 1$ implies $E(T_i) = \emptyset$ and so i is a terminal node. If $i \in V(B) - V(P)$ then by (3.3.4) $b_i \geq |\delta_B(i)|$ so $|\delta_B(i)| = 1$ and i is a terminal node. \square

(3.3.10) Proposition. If $b_i = 1$ for every $i \in V(B)$ then B is a blossom if and only if B is an odd polygon.

Proof. First suppose that B is a blossom. By (3.3.9) every node of B is a terminal node, if any petal T_v contained an edge then v could not be a terminal node, a contradiction. Hence all petals are single nodes and B is an odd polygon.

If B is an odd polygon let $v \in V(B)$ and let τ be a shortest odd length track in B from v to v . If we define $x_j = 0$ for every odd edge of τ and $x_j = 1$ for every even edge of τ then x satisfies (3.3.1)-(3.3.5) so B is a blossom. \square

If B is a graph such that $b(V(B))$ is odd then clearly B can have no perfect matching.

(3.3.11) We define a near perfect matching (abbreviated by np matching) to be a matching x of B such that, for some $v \in V(B)$

$$x(\delta_B(i)) = b_i \text{ for all } i \in V(B) - \{v\},$$

$$x(\delta_B(v)) = b_v - 1.$$

(3.3.12) Proposition. Let B be a blossom containing the odd polygon P . Then for any $i \in V(B)$ there is a np matching x' of B deficient at i . Moreover if $i \in V(P)$ then x' satisfies (3.3.4), (3.3.5) (with x' substituted for x).

Proof. The proof of this proposition actually consists of an algorithm for obtaining such a matching, starting with a np matching x deficient at $v \in V(P)$ satisfying (3.3.1)-(3.3.5).

If $i = v$ then x is the matching we require and we are finished. Otherwise let τ be the shortest track from v to i having even length. Now define x' by

$$(3.3.13) \quad x'_j = \begin{cases} x_j + 1 & \text{if } j \text{ is an odd edge of } \tau, \\ x_j - 1 & \text{if } j \text{ is an even edge of } \tau, \\ x_j & \text{if } j \in E(B) - E(\tau). \end{cases}$$

Clearly x' is integer valued, $x'(\delta_B(s)) = b_s$ for all $s \in V(B) - \{i\}$ and $x'(\delta_B(i)) = b_i - 1$. Moreover if $j \in E(P)$ is an even edge of τ then j is the first edge in an even length path in P from some $w \in V(P) - \{v\}$ to v so by (3.3.5) $x_j \geq 1$ and $x'_j \geq 0$. If $j \in E(B) - E(P)$ is an even edge of τ then by (3.3.4) $x_j \geq 1$ so $x'_j \geq 0$. For any $j \in E(B)$ which is not an even edge of τ we have $x'_j \geq x_j \geq 0$ so $x' \geq 0$ and hence is a np matching deficient at i .

Now suppose $i \in V(P) - \{v\}$. First observe that each $j \in E(P)$ is the first edge in exactly two paths from nodes

of P to i and since P is an odd polygon, both these paths have the same parity. If $j \in E(P) \cap E(\tau)$ then j is the first edge in an even path to i if and only if j is an odd edge of τ so by (3.3.13) $x'_j \geq 1$. If $j \in E(P) - E(\tau)$ is the first edge in an even path to i then it is easily seen that j is the first edge in an even path to v so by (3.3.5) $x'_j = x_j \geq 1$.

Since $i \in V(P)$ implies $E(\tau) \subseteq E(P)$, (3.3.13) and (3.3.4) ensure that $x'_j = x_j \geq 1$ for all $j \in E(B) - E(P)$ and the proof is complete. \square

We now define shrinking. Let $G = (V, E, \psi)$ be a graph let $S \subseteq V$. We say that $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ is the graph obtained from G by shrinking S if

$$\bar{V} = V - S \cup \{S\},$$

$$\bar{E} = E - \gamma(S)$$

$$\begin{aligned} \bar{\psi}(j) = & \quad \psi(j) \quad \text{if } j \in \bar{E} - \delta(S) \\ & \quad \psi(j) - S \cup \{S\} \quad \text{if } j \in \delta(S). \end{aligned}$$

In other words, \bar{G} is the graph obtained from G by contracting all edges of G which have both ends in S and calling the resulting node " S ". We denote \bar{G} by $G \times S$ and call S a pseudonode of \bar{G} (with respect to G). We define the degree constraint $b_S \equiv 1$ for any pseudonode S . We also define

$$(3.3.14) \quad \begin{aligned} \bar{V}^+ &\equiv V^+ - S \cup \{S\} \quad \text{if } S \subseteq V^+, \\ &\quad V^+ - S \quad \text{if } S \not\subseteq V^+. \end{aligned}$$

$$\bar{V}^\leq \equiv \bar{V} - \bar{V}^+.$$

Let R be a nested family of subsets of V (see Section 3.2). For any $S \in R$ we define

$$(3.3.15) \quad R[S] = \{T \in R : T \subset S\}.$$

If $\{S_1, S_2, \dots, S_k\}$ is the set of maximal members of R then we let $G \times R$ denote

$$(3.3.15a) \quad (\dots((G \times S_1) \times S_2) \times \dots) \times S_k.$$

It is easily seen that the ordering of the sets S_1, S_2, \dots, S_k has no effect on $G \times R$.

We say that $G = (V, E, \psi)$ is shrinkable if there is a possible empty nested family R of subsets of V such that

(3.3.16) for every $S \in R$, $G[S] \times R[S]$ is spanned by a blossom B_S ,

$$(3.3.17) \quad b(V(G \times R)) = 1.$$

It is easy to see that

$$(3.3.18) \quad V \in R \text{ is equivalent to (3.3.17) if } R \neq \emptyset.$$

We call R a shrinking family of G . Note that in particular any graph spanned by a blossom is shrinkable. For any $S \subseteq V$ we say that S is shrinkable if $G[S]$ is shrinkable.

If $R = \emptyset$ is a shrinking family of $G = (V, E, \psi)$ then $|V| = 1$, $|E| = \emptyset$ and $b(V) = 1$. We call such a graph degenerate, all other shrinkable graphs are called nondegenerate.

(3.3.19) Proposition. If $G = (V, E, \psi)$ is shrinkable, then $b(V)$ is odd.

Proof. Let R be a shrinking family of G , we prove by induction on $|R|$. If $|R| = 0$ then G is degenerate and the result is trivial. Suppose (3.3.19) holds when G has a shrinking family of fewer than k sets for some $k \geq 1$ and assume $|R| = k$. By (3.3.16) there is a blossom B_V spanning $G \times R[V]$ and by (3.3.6),

$$(3.3.19a) \quad b(V(B_V)) \text{ is odd.}$$

Let S be any maximal member of $R[V]$ and hence a pseudonode of $G \times R[V]$. Then $R[S] \cup \{S\}$ is a shrinking family of $G[S]$ and since $|R[S] \cup \{S\}| < |R|$ we have by induction

$$(3.3.19b) \quad b(S) \text{ is odd.}$$

If W is the set of pseudonodes of $G \times R[V]$ then

$$b(V) = b(V(G \times R[V])) + \sum(\{b(S) : S \in W\} - 1)$$

so since $V(G \times R[V]) = V(B_V)$ we have by (3.3.19a) and (3.3.19b) that $b(V)$ is odd as asserted. \square

If R is a shrinking family of G then for any $S \in R$, $R[S] \cup \{S\}$ is a shrinking family of $G[S]$. Hence we have the following corollary of (3.3.19).

(3.3.20) Corollary. If R is a shrinking family for G then $b(S)$ is odd for all $S \in R$.

(3.3.21) Proposition. Let $G = (V, E, \psi)$ be shrinkable and let R be a shrinking family of G . Then for any $v \in V$ there is a np matching x of G deficient

at v and which satisfies

(3.3.22) $x|\gamma(S)$ is a np matching of $G[S]$ for all $S \in R$.

Proof. We prove by induction on $|R|$. If $|R| = 0$ then G is degenerate and the result is trivial. Assume the result true for graphs having a shrinking family consisting of fewer than k sets for $k \geq 1$ and suppose $|R| = k$. Let v be any node of G . Every maximal $S \in R[V]$ is a pseudonode of the blossom B_V which spans $G \times R[V]$. Let $p \equiv v$ if $v \in V(B_V)$, let $p \equiv S$ if $v \in S$ for some pseudonode S of B_V . By (3.3.12) there is a np matching \tilde{x} of B_V deficient at p . For every pseudonode $T \in V(B_V)$ there is at most one node of T incident with some $j \in E(B_V)$ for which $\tilde{x}_j = 1$ since $b_T = 1$. If such a node $w(T)$ exists, let \bar{x}^T be a np matching of $G[T]$ deficient at $w(T)$ satisfying

(3.3.23) $\bar{x}^T|\gamma(Z)$ is a np matching of $G[Z]$ for every $Z \in R[T]$,

which exists by our induction hypothesis. If no such $w(T)$ exists, then $v \in T$ and we let \bar{x}^T be a np matching of $G[T]$ deficient at v which satisfies (3.3.23). Now define x by

$$\begin{aligned} \tilde{x}_j & \text{ for } j \in E(B_V) \\ x_j &= 0 \quad \text{for } j \in E(G \times R[V]) - E(B_V) \\ & \quad \bar{x}^T \text{ for } j \in \gamma(T), \text{ for } T \in R[V]. \end{aligned}$$

x_j is easily seen to be a np matching of G deficient at v and satisfying (3.3.22), thereby completing the proof. \square

We close this section by noting the following basic property of matchings.

(3.3.24) Proposition. A matching x is a np matching of $G = (V, E, \psi)$ if and only if $x(E) = q_V (=1/2(b(V)-1))$ and $x(\delta(i)) \leq b_i$ for all $i \in V$.

Proof. For any matching x of G ,

$$2x(E) = b(V) - \sum_i (b_i - x(\delta(i))) : i \in V.$$

Thus any np matching x of G satisfies $x(E) = 1/2(b(V)-1)$ (and trivially $x(\delta(i)) \leq b_i$ for all $i \in V$). Conversely any matching x which satisfies $x(E) = q_V$ and $x(\delta(i)) \leq b_i$ for all $i \in V$ must satisfy $x(\delta(i)) = b_i$ for all $i \in V - \{v\}$ and $x(\delta(v)) = b_v - 1$ for some $v \in V$. Thus x is a np matching of G and the result follows. \square

3.4. The Matching Polyhedron.

The matching polyhedron $P(G, b)$ is defined to be the bounded polyhedron in \mathbb{R}^E containing all matchings x of $G = (V, E, \psi)$ which satisfy

$$(3.4.1) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V$$

and for which every vertex is such a matching. (Equivalently, $P(G, b)$ is the convex hull of the set of matchings of G which satisfy (3.4.1).)

Let $Q = \{S \subseteq V : |S| \geq 3 \text{ and } b(S) \text{ is odd}\}$. Edmonds [E3]

has shown that $P(G, b) = \{x \in \mathbb{R}^E : \dots\}$

$$(3.4.2) \quad x_j \geq 0 \quad \text{for all } j \in E,$$

$$(3.4.3) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V$$

$$(3.4.4) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q\}$$

The proof is a consequence of a blossom algorithm similar to the version we are developing here in the following way.

The algorithm shows that for any $c \in \mathbb{R}^E$, $c \cdot x$ is maximized over all x (not necessarily integer valued) satisfying (3.4.2)-(3.4.4) by a matching of G which satisfies (3.4.1). It is implicit in the algorithm that Q can be replaced in (3.4.4) by a subset of itself which is generally much smaller than Q .

Let $Q^0 \equiv \{S \subseteq V : |S| \geq 3 \text{ and } S \text{ is a shrinkable subset of } V\}$. (By (3.3.19) $b(S)$ is odd for each $S \in Q$).

(3.4.5) Theorem.

$$\underline{P(G, b) = P \equiv \{x \in \mathbb{R}^E : \dots\}}$$

$$(3.4.6) \quad \underline{x_j \geq 0 \quad \text{for all } j \in E},$$

$$(3.4.7) \quad \underline{x(\delta(i)) \leq b_i \quad \text{for all } i \in V}$$

$$(3.4.8) \quad \underline{x(\gamma(S)) \leq q_S \quad \text{for every } S \in Q^0}.$$

Proof. It is easily seen that any matching x of G which satisfies (3.4.1) belongs to P , for it satisfies (3.4.6), (3.4.7) by definition and it satisfies (3.4.8) by (3.1.7) and (3.3.19).

We will show by means of the blossom algorithm that for

any $c \in \mathbb{R}^E$, there is a matching x^0 of G satisfying (3.4.1) which maximizes cx over $x \in P$. By (2.4.1) for each vertex v of P there is a vector $c \in \mathbb{R}^E$ such that cx is maximized over $x \in P$ only by v . Hence all vertices of P are matchings satisfying (3.4.1).

(We saw in (2.4.10) that every bounded polyhedron is the convex hull of its set of vertices. Since P contains all matchings of G satisfying (3.4.1) and since all vertices of P are such matchings it follows that P is the convex hull of the matchings of G which satisfy (3.4.1).) \square

When we require matchings satisfying

$$(3.4.9) \quad x(\delta(i)) = b_i \text{ for } i \in V^+ \subseteq V$$

then we are in fact considering a face F of $P(G, b)$. Thus the blossom algorithm presented in this chapter will find (if one exists) a matching $x^0 \in F \subseteq P(G, b)$ maximizing $c \cdot x^0$ over F where F is a face of $P(G, b)$ obtained by requiring (3.4.9) hold. (If $V^+ = \emptyset$ then $F = P(G, b)$). In chapter 5 we study the more general problem of maximizing $c \cdot x$ over any face F of $P(G, b)$.

3.5 Linear Programming Formulation

The following linear program is equivalent to the problem of maximizing cx for $x \in F \subseteq P$ where F is the face of P (defined in (3.4.5)) obtained by requiring (3.4.9) hold.

$$(3.5.1) \quad \text{Maximize } c \cdot x$$

over $x \in \mathbb{R}^E$ which satisfy

$$(3.5.2) \quad x_j \geq 0 \quad \text{for all } j \in E,$$

$$(3.5.3) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V^{\leq} = V - V^=,$$

$$(3.5.4) \quad x(\delta(i)) = b_i \quad \text{for all } i \in V^=,$$

$$(3.5.5) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q^0.$$

For any $j \in E$ let $Q^0(j) = \{S \in Q^0 : j \in \gamma(S)\}$. The dual linear program is

$$(3.5.6) \quad \text{minimize } \sum (b_i y_i : i \in V) + \sum (q_S y_S : S \in Q^0)$$

over $y \in \mathbb{R}^{V \cup Q^0}$ which satisfy

$$(3.5.7) \quad y_S \geq 0 \quad \text{for all } S \in Q^0,$$

$$(3.5.8) \quad y_i \geq 0 \quad \text{for all } i \in V^{\leq}$$

$$(3.5.9) \quad y(\psi(j)) + y(Q^0(j)) \geq c_j \quad \text{for all } j \in E.$$

By complementary slackness (1.5.16) x^0 satisfying (3.5.2)-(3.5.5) and y^0 satisfying (3.5.7)-(3.5.9) are optimal if and only if

$$(3.5.10) \quad x_j^0 > 0 \quad \text{implies} \quad y^0(\psi(j)) + y^0(Q^0(j)) = c_j$$

for any $j \in E$,

$$(3.5.11) \quad y_i^0 > 0 \quad \text{implies} \quad x^0(\delta(i)) = b_i \quad \text{for all } i \in V^{\leq},$$

$$(3.5.12) \quad y_S^0 > 0 \quad \text{implies} \quad x^0(\gamma(S)) = q_S \quad \text{for all } S \in Q^0.$$

The blossom algorithm will actually find a feasible

matching x and a dual solution y satisfying (3.5.7)-(3.5.9) such that x and y satisfy (3.5.10)-(3.5.12) or else will show that no feasible matching exists in a manner described in section 3.7.

We call y a dual solution to the matching problem (3.1.2)-(3.1.6) if y satisfies (3.5.7)-(3.5.9) and an optimal dual solution if y minimizes $\sum(b_i y_i : i \in V) + \sum(q_S y_S : S \in Q^0)$ over all dual solutions.

3.6 Alternating Forests

During the course of the blossom algorithm we construct forests having special properties with respect to a matching. Let T be a tree contained in $G = (V, E, \psi)$, let $r \in V(T)$ be designated as the root of T . There is a unique path $\pi(i)$ in T from r to each $i \in V(T)$. We call i an even node or an odd node of T according as the length of $\pi(i)$ is even or odd. In particular, r is an even node of T . We call $j \in E(T)$ even or odd according as j is the last edge of a path $\pi(i)$ in T to an even or odd node of T (or equivalently, according as j is an even edge or odd edge of any path $\pi(i)$ in T from r to some node $i \in V(T) - \{r\}$ such that $j \in E(\pi(i))$).

Let x be a matching of G . We call T an alternating tree with respect to x (see Figure 3.2) if

$$(3.6.1) \quad x(\delta(r)) < b_r,$$

$$(3.6.2) \quad x(\delta(i)) = b_i \quad \text{for all } i \in V(T) - \{r\},$$

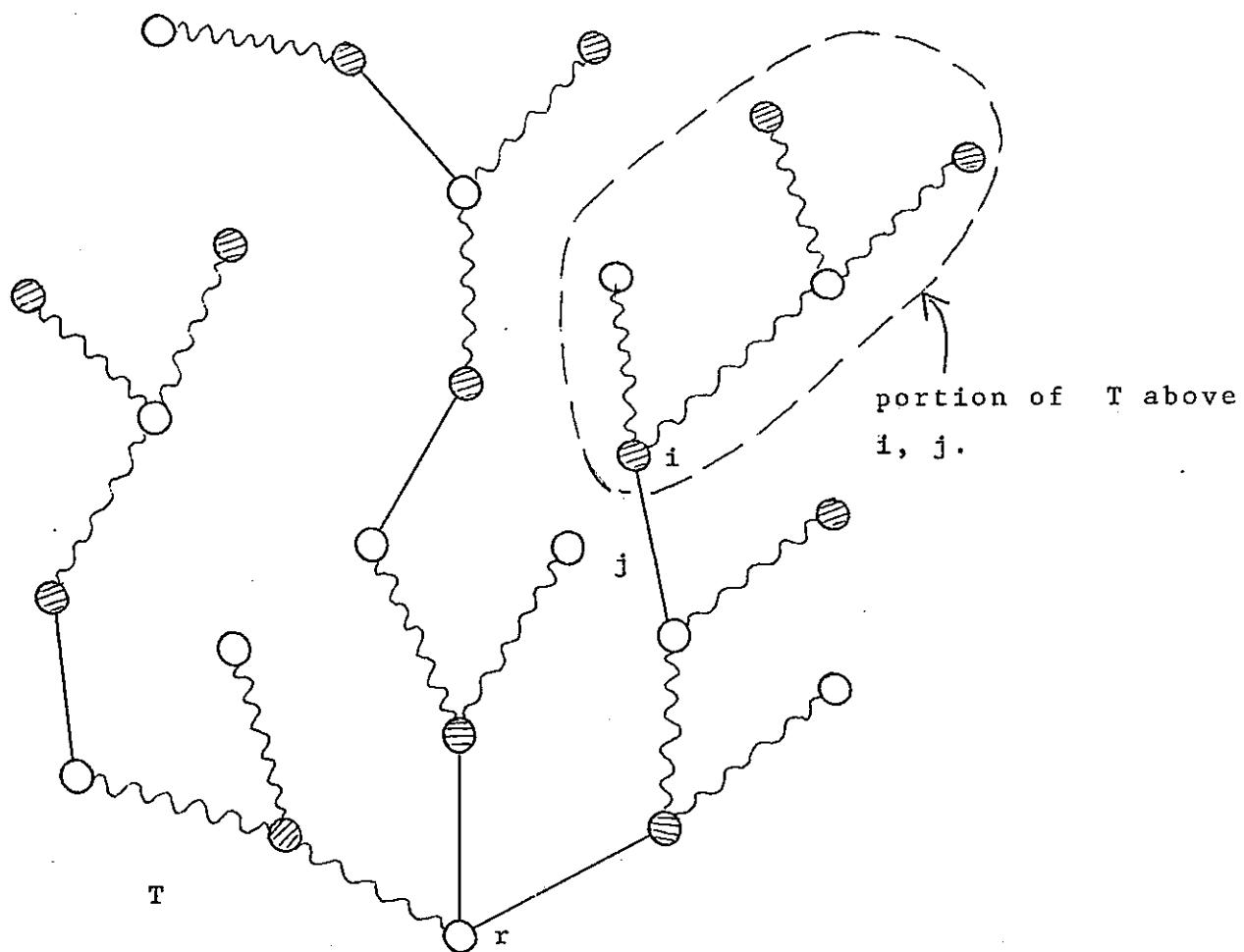


Figure 3.2 Alternating tree

even nodes

odd nodes

edge $j \leftarrow$ such that $x_j \geq 1$

edge j such that $x_j \geq 0$

(3.6.3) if $x_j > 0$ and $\psi(j) \cap V(T) \neq \emptyset$ then

$j \in E(T)$,

(3.6.4) $x_j > 0$ for every even edge j of T .

If we are considering 1-matchings then (3.6.1)-(3.6.4) imply that every even edge j of T has $x_j = 1$ and every odd edge j has $x_j = 0$.

Note that for any $i \in V$, $\{i\}$ is the node set of an alternating tree if $x(\delta(i)) = 0$. We call a nonempty collection of alternating trees an alternating forest.

Let j be an edge of a tree T with root r . If we delete j from T then the resulting graph will consist of two trees, one of which, T' , will not contain r . We call T' the portion of T above j .

Let i be any node of T . If $i = r$ then we say that T is the portion of T above i . Otherwise let k be the first edge of the path in T from i to r and let T' be the portion of T above k . We say that T' is the portion of T above i .

(3.6.5) Proposition. Let T be an alternating tree with respect to the matching x . Let r be the root of T , let I be the set of odd nodes of T and let W be the set of even nodes of T . Then

$$b(W) - (b_r - x(\delta(r))) = b(I).$$

Proof. By (3.6.1) and (3.6.2)

$$(3.6.6) b(W) = \sum_{i \in W} x(\delta(i)) + b_r - x(\delta(r)).$$

Since no edge of T can join two even nodes and by (3.6.3),

$$(3.6.7) \quad \sum(x(\delta(i)) : i \in W) = x(\delta(W) \cap E(T)).$$

By (3.6.2)

$$(3.6.8) \quad b(I) = \sum(x(\delta(i)) : i \in I).$$

Since no edge of T can join two odd nodes and by (3.6.3),

$$(3.6.9) \quad \sum(x(\delta(i)) : i \in I) = x(\delta(I) \cap E(T)).$$

But for any $j \in E(T)$, $j \in \delta(I)$ and $j \in \delta(W)$ so

$$(3.6.10) \quad \delta(I) \cap E(T) = \delta(W) \cap E(T).$$

By (3.6.10), (3.6.9) and (3.6.7) we have

$$\sum(x(\delta(i)) : i \in W) = \sum(x(\delta(i)) : i \in I).$$

Hence (3.6.6) and (3.6.8) combine to give the result. \square

(3.6.11) Corollary. Let F be an alternating forest with respect to the matching x , let K be the set of roots of the trees of F . Let W and I be the sets of even nodes and odd nodes of F respectively.

Then

$$b(W) - \sum(b_r - x(\delta(r)) : r \in K) = b(I).$$

Note that (3.6.1) implies therefore the following.

(3.6.12) Corollary. If W and I are the sets of even and odd nodes of an alternating forest F then

$$\underline{b(W) > b(I)}.$$

3.7. Hungarian Forests.

Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ be the graph obtained from $G = (V, E, \psi)$ by shrinking a (possibly empty) family R of disjoint shrinkable subsets of V . We define

$$(3.7.1) \quad \bar{V}^{\equiv} \equiv (V^{\equiv} \cap \bar{V}) \cup \{S \in R : S \subseteq V^{\equiv}\} .$$

Let F be an alternating forest contained in \bar{G} with respect to a matching \bar{x} of \bar{G} which satisfies

$$(3.7.2) \quad \bar{x}(\delta_{\bar{G}}(i)) \leq b_i \text{ for all } i \in \bar{V}.$$

We call F Hungarian in \bar{G} with respect to \bar{x} if

(3.7.3) no edge of \bar{G} joins two even nodes of F ,

(3.7.4) no edge of \bar{G} joins an even node of F to a node not in F ,

(3.7.5) every odd node of F is a node of G , that is, not a pseudonode of \bar{G} ,

(3.7.6) if $v \in \bar{V}$ is an even node of F then $v \in \bar{V}^{\equiv}$,

(3.7.7) for any $i \in \bar{V}^{\equiv}$, if $\bar{x}(\delta_{\bar{G}}(i)) < b_i$

then i is the root of a tree in F .

Let x be any matching of G which satisfies

(3.7.8) $x(\delta(i)) \leq b_i$ for all $i \in V$.

We define

$$(3.7.9) \quad d(G, V^{\neq}; x) \equiv \sum_{i \in V} (b_i - x(\delta(i))) : i \in V^{\neq}.$$

If M is the set of all matchings of G which satisfy (3.7.8) then we let

$$(3.7.10) \quad D(G, V^{\neq}) \equiv \min\{d(G, V^{\neq}; x) : x \in M\}.$$

Thus $d(G, V^{\neq}; x)$ is a measure of the amount by which x fails to be a feasible matching of G and $D(G, V^{\neq})$ measures how closely we can come to obtaining a feasible matching of G . Clearly

(3.7.11) G has a feasible matching if and only if $D(G, V^{\neq}) = 0$.

Later in this section we show the connection between Hungarian forests and the value of $D(G, V^{\neq})$. We also show in (3.7.36) that knowledge of a Hungarian forest of G enables us to characterize those matchings x of G for which $d(G, V^{\neq}; x) = D(G, V^{\neq})$.

First we prove the following basic result which also indicates the importance of shrinkable sets in the blossom algorithm.

(3.7.12) Proposition. Let R be a family of disjoint shrinkable subsets of V and let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ be the graph obtained from $G = (V, E, \psi)$ by shrinking the members of R . Let \bar{V}^{\neq} be defined as in (3.7.1). Then any matching \bar{x} of \bar{G} satisfying (3.7.2) can be extended to a matching x of G satisfying (3.7.8) such that

$$(3.7.13) \quad d(G, V^{\neq}; x) = d(\bar{G}, \bar{V}^{\neq}; \bar{x}).$$

Proof. For each $S \in R$ we define a node $i(S)$ as follows. If there is some $j \in \delta_{\bar{G}}(S)$ such that $\bar{x}_j = 1$ then let $\{i(S)\} \equiv \psi(j) \cap S$. Otherwise if $S - V^{\neq} \neq \emptyset$, let $i(S)$ be any member of $S - V^{\neq}$. Otherwise let $i(S)$ be any node of S . By (3.3.21) there is a np matching x^S of $G[S]$ deficient at $i(S)$ for every $S \in R$. We define x by

$$x_j \equiv \begin{cases} \bar{x}_j & \text{for } j \in \bar{E}, \\ x_j^S & \text{for } j \in \gamma(S) \text{ for all } S \in R. \end{cases}$$

x is easily seen to satisfy (3.7.8).

For any $v \in \bar{V} - R$ we have $\delta_{\bar{G}}(v) = \delta(v)$ so

$$(3.7.14) \quad b_v - x(\delta(v)) = b_v - \bar{x}(\delta_{\bar{G}}(v)) \quad \text{for all } v \in \bar{V} - R.$$

Let $S \in \bar{V}^{\neq} \cap R$. Then $S \subseteq V^{\neq}$ so

$$\begin{aligned} \sum (b_i - x(\delta(i))) : i \in S \\ &= \sum (b_i - x(\delta(i))) : i \in S - \{i(S)\} + b_{i(S)} - x(\delta(i(S))) \\ &= 0 + (b_{i(S)} - x^S(\delta_{G[S]}(i(S)))) - \bar{x}(\delta_{\bar{G}}(S)) \\ &= 1 - \bar{x}(\delta_{\bar{G}}(S)). \end{aligned}$$

Therefore

$$(3.7.15) \quad \sum (b_i - x(\delta(i))) : i \in S \cap V^{\neq} = b_S - \bar{x}(\delta_{\bar{G}}(S))$$

for all $S \in \bar{V}^{\neq} \cap R$.

Let $S \in R - \bar{V}^{\neq}$. If $i(S) \in V - V^{\neq}$ then

$$\sum (b_i - \bar{x}(\delta(i)) : i \in S \cap V^=) = 0.$$

If $i(S) \in V^=$ then there is $j \in \delta(i(S)) \cap \delta(S)$ such that $\bar{x}_j = 1$. Therefore

$$\begin{aligned} \sum (b_i - \bar{x}(\delta(i)) : i \in S \cap V^=) \\ = b_{i(S)} - \bar{x}_{G[S]}^S(i(S)) - \bar{x}_j \\ = 1 - 1 = 0. \end{aligned}$$

Hence

$$(3.7.16) \quad \sum (b_i - \bar{x}(\delta(i)) : i \in S \cap V^=) = 0 \quad \text{for all } S \in R - \bar{V}^=.$$

Combining (3.7.14)-(3.7.16) gives (3.7.13). \square

(3.7.17) Theorem. Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$, $G = (V, E, \psi)$
 R , $V^=$ and $\bar{V}^=$ be as in (3.7.12). Let F be a Hungarian
forest in \bar{G} with respect to a matching \bar{x} . Let $K \subseteq \bar{V}$ be
the set of roots of trees of F . Then

$$(3.7.18) \quad D(G, V^=) = \sum (b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K).$$

Proof. By (3.7.6) and (3.7.7)

$d(\bar{G}, \bar{V}^=; \bar{x}) = \sum (b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K)$. By (3.7.12) \bar{x} can
be extended to a matching x^0 of G for which
 $d(G, V^=; x) = d(\bar{G}, \bar{V}^=; x^0)$ so

$$(3.7.19) \quad D(G, V^=) \leq \sum (b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K).$$

Now consider the linear program

$$(3.7.20) \text{ maximize } 2x(\gamma(V^=)) + x(\delta(V^=))$$

over $x \in \mathbb{R}^E$ satisfying

$$x \geq 0,$$

$$(3.7.21) \begin{aligned} x(\delta(i)) &\leq b_i & \text{for all } i \in V, \\ x(\gamma(s)) &\leq q_s & \text{for all } s \in Q^0. \end{aligned}$$

By (3.1.7) any matching x of G satisfying (3.7.8) is a feasible solution to this linear program.

The dual linear program is

$$(3.7.22) \text{ minimize } \sum(b_i y_i : i \in V) + \sum(q_s y_s : s \in Q^0)$$

for $y \in \mathbb{R}^{V \cup Q^0}$ satisfying

$$(3.7.23) y_i \geq 0 \quad \text{for all } i \in V \cup Q^0,$$

$$(3.7.24) y(\psi(j)) + y(Q^0(j)) \geq |\psi(j) \cap V^=|$$

for all $j \in E$.

We define a vector y^0 as follows. Let I and W be the sets of odd and even nodes of F respectively.

$$(3.7.25) y_i^0 = \begin{cases} 2 & \text{if } i \in I \cap V^= \\ 1 & \text{if } i \in I - V^= \text{ or if } \\ & i \in V^= - V(F) - \cup(R \cap V(F)), \\ 0 & \text{if } i \in V - V^= - I; \end{cases}$$

$$(3.7.26) y_s^0 = \begin{cases} 2 & \text{if } s \in R \cap W \\ 0 & \text{if } s \in Q^0 - (R \cap W). \end{cases}$$

Now we show that

(3.7.27) y^0 is dual feasible.

If neither end of j is in F or is contained in a pseudonode of F then

$$y^0(\psi(j)) = |\psi(j) \cap V^=|$$

so (3.7.24) is satisfied.

(3.7.28) If exactly one end of j is in F or is contained in a pseudonode of F then by (3.7.4) j must meet an odd node of F so

$$y^0(\psi(j)) = |\psi(j) \cap V^=| + 1$$

and (3.7.24) is satisfied.

If $j \in \gamma(S)$ for some pseudonode S of F then

$$y^0(Q^0(j)) = 2 = |\psi(j) \cap V^=|$$

since by (3.7.6) and (3.7.1), $S \subseteq V^=$. Hence (3.7.24) is satisfied.

If $|\bar{\psi}(j) \cap I| = |\bar{\psi}(j) \cap W| = 1$ then since by (3.7.6) and (3.7.1) $\psi(j) - I \subseteq V^=$ it follows that

$$y^0(\psi(j)) = |\psi(j) \cap V^=|,$$

Thus (3.7.24) is satisfied.

(3.7.29) if $|\bar{\psi}(j) \cap I| = 2$ then

$$y^0(\psi(j)) = |\psi(j) \cap V^=| + 2 \text{ so (3.7.24) is satisfied.}$$

By (3.7.3) this exhausts all cases, so since $y^0 \geq 0$ we have proved (3.7.27).

Now we evaluate the dual objective function for y^0 .

$$\begin{aligned}
 (3.7.30) \quad & \sum(b_i y_i^0 : i \in V) + \sum(q_S y_S^0 : S \in Q^0) \\
 & = b(V^{\bar{E}}) - v(F) - v(R \cap V(F)) + b(I - V^{\bar{E}}) \\
 & + 2b(I \cap V^{\bar{E}}) + 2\sum(q_S : S \in R \cap W).
 \end{aligned}$$

By (3.6.5),

$$\begin{aligned}
 (3.7.31) \quad & b(I - V^{\bar{E}}) + 2b(I \cap V^{\bar{E}}) \\
 & = b(W) + b(I \cap V^{\bar{E}}) - \sum(b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K).
 \end{aligned}$$

By (3.1.10)

$$(3.7.32) \quad 2\sum(q_S : S \in R \cap W) = \sum(b(S) : S \in R \cap W) - b(R \cap W).$$

Substituting (3.7.31) and (3.7.32) into (3.7.30) and simplifying we obtain

$$\begin{aligned}
 & \sum(b_i y_i^0 : i \in V) + \sum(q_S y_S^0 : S \in Q^0) \\
 & = b(V^{\bar{E}}) - \sum(b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K).
 \end{aligned}$$

It therefore follows from the weak L.P. duality theorem (1.5.12) that

$$\begin{aligned}
 (3.7.33) \quad & 2x(\gamma(V^{\bar{E}})) + x(\delta(V^{\bar{E}})) \leq \\
 & b(V^{\bar{E}}) - \sum(b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K)
 \end{aligned}$$

for any feasible solution x to the primal linear program (3.7.21). Since every matching of G which satisfies (3.7.8) is such a feasible solution, and since

$$\begin{aligned}
 (3.7.34) \quad & \sum(b_i - x(\delta(i)) : i \in V^{\bar{E}}) = \\
 & b(V^{\bar{E}}) - (2x(\gamma(V^{\bar{E}})) + x(\delta(V^{\bar{E}})))
 \end{aligned}$$

it follows that

$$(3.7.35) \quad D(G, V^=) \geq \sum_{\bar{G}} (b_i - \bar{x}(\delta_{\bar{G}}(i)) : i \in K).$$

Combining (3.7.19) and (3.7.35) proves the theorem. \square

By using the complementary slackness principle of linear programming we obtain the following characterization of matchings x which minimize $d(G, V^=; x)$.

(3.7.36) Theorem. Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$, $G = (V, E, \psi)$, $R, V^=$ and $\bar{V}^=$ be as in (3.7.12). Then for any matching x of G satisfying (3.7.8) we have $D(G, V^=) = d(G, V^=; x)$ if and only if the following conditions are satisfied.

$$(3.7.37) \quad x(\gamma(S)) = q_S \text{ for all } S \in R \cap V(F).$$

$$(3.7.38) \quad x(\delta(i)) = b_i \text{ for every odd node } i \text{ of } F \text{ and for every } i \in V^= - V(F) - \cup(R \cap V(F)).$$

$$(3.7.39) \quad \text{If } I \text{ and } W \text{ are the sets of odd and even nodes of } F \text{ respectively, then } x_j = 0 \text{ for all } j \in \cup_{i \in I} \delta(i) - \delta_{\bar{G}}(W).$$

Proof. In the proof of (3.7.17) we displayed a matching x^0 satisfying (3.7.8) and a dual solution y^0 such that $2x^0(\gamma(V^=)) + x^0(\delta(V^=)) = \sum_i (b_i y_i : i \in V) + \sum_S (q_S y_S : S \in Q^0)$. Thus y^0 is an optimal solution to the dual linear program (3.7.22)-(3.7.24) so every optimal solution \hat{x} to the primal linear program (3.7.20), (3.7.21) must satisfy the complementary slackness conditions (see (1.5.16)) with respect to y^0 .

Thus by (3.7.25) we must have (3.7.38); by (3.7.26) we require (3.7.37); by (3.7.28) and (3.7.29) we require (3.7.39). Since by (3.7.34) \hat{x} maximizes $2x(\gamma(V^=)) + x(\delta(V^=))$ for x satisfying (3.7.21) if and only if \hat{x} minimizes $d(G, V^=; x)$ for x satisfying (3.7.21) and since we have exhibited a matching x^0 for which $d(G, V^=; x^0) = D(G, V^=)$ the result now follows. \square

If we are considering a matching problem in which $V^= = \phi$ then by (3.7.1) and (3.7.6) there could be no even nodes in a Hungarian forest F in a graph \bar{G} obtained from $G = (V, E, \psi)$ by shrinking some disjoint shrinkable subsets of V . But since every Hungarian forest contains at least one tree rooted at an even node, this means that no Hungarian forest can exist. In other words, Hungarian forests are structures which can arise only when dealing with matching problems in which $V^= \neq \phi$.

The following corollary of (3.7.17) is a necessary condition for a graph G to have a feasible matching.

(3.7.40) Corollary. If G has a feasible matching then no graph \bar{G} obtained from G by shrinking a collection of disjoint shrinkable subsets of V can contain a Hungarian forest.

Proof. If \bar{G} contains a Hungarian forest F with respect to a matching \bar{x} then if K is the set of roots of trees of F , we have

$$\sum_{\bar{G}} (b_i - \bar{x}(\delta_{\bar{G}}(i))) : i \in K > 0.$$

Therefore by (3.7.17), $D(G, V^+) > 0$. Therefore by (3.7.11) G has no feasible matching. \square

In fact, the converse of this corollary is true and will be proved by the blossom algorithm for it will always terminate with either an optimum feasible matching or else with a Hungarian forest.

3.8 The Blossom Algorithm

In this section we describe the blossom algorithm which solves the problem (3.1.2)-(3.1.6). This algorithm is also used in later chapters when we consider more general problems. In Section 3.9 we derive a bound on the amount of work performed by the blossom algorithm in solving a matching problem.

At each stage of the algorithm we have the following things.

(3.8.1) a matching $x = (x_j : j \in E)$,

(3.8.2) a dual solution $y = (y_i : i \in V \cup Q^0)$ which satisfies (3.5.7)-(3.5.9).

Let $G^{\equiv} = (V, E^{\equiv}, \psi|E^{\equiv})$ be the spanning subgraph of G whose edge set consists of all those $j \in E$ satisfying

$$(3.8.3) \quad y(\psi(j)) + y(Q^0(j)) = c_j.$$

G^{\equiv} is called the equality subgraph. The complementary slackness condition (3.5.10) is satisfied by x and y , that is

$$(3.8.4) \quad x_j > 0 \text{ only if } j \in E^=.$$

We also have a nested subfamily R of Q such that

$$(3.8.5) \quad \text{for each } S \in R, H(S) \equiv G^=[S] \times R[S]$$

is spanned by a blossom $B(S)$. (See (3.3.15), (3.3.15a) for the definition of $G^=[S] \times R[S]$.)

Moreover

(3.8.6) $x|_{E(H(S))}$ is a np matching of $H(S)$ deficient at some $i(S)$ belonging to the odd polygon of $B(S)$ and

$$(3.8.7) \quad x_j = 0 \text{ for all } j \in E(H(S)) - E(B(S)).$$

As a result of (3.3.24) a simple induction shows

$$(3.8.8) \quad x(\gamma(S)) = q_S \text{ for all } S \in R.$$

The dual solution y has the property that

$$(3.8.9) \quad y_S > 0 \text{ for } S \in Q^0 \text{ only if } S \in R.$$

Thus x and y satisfy the complementary slackness condition (3.5.12).

Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ be the graph $G^= \times R$. The matching x satisfies

$$(3.8.10) \quad x(\delta(i)) \leq b_i \text{ for all } i \in \bar{V}.$$

(Note that for any $i \in \bar{V}$, $\delta(i) = \delta_{\bar{G}}(i)$.)

We define subsets $\bar{V}^=$ and $\bar{V}^<$ of \bar{V} by

$$(3.8.11) \quad \bar{V}^{\neq} \equiv (V^{\neq} \cap \bar{V}) \cup \{S \in \bar{V}: S \subseteq V^{\neq}\},$$

$$(3.8.12) \quad \bar{V}^{\leq} \equiv \bar{V} - \bar{V}^{\neq}.$$

The matching x also has the following property. Let $G^+(x) = (\bar{V}, E^+(x), \bar{\psi}|E^+(x))$ be the spanning subgraph of \bar{G} whose edges are all those edges of \bar{G} such that $x_j > 0$. Thus $E^+(x) = \{j \in \bar{E}: x_j > 0\}$. Let H be any component of $G^+(x)$. Then

$$(3.8.13) \quad H \text{ contains no even polygon};$$

$$(3.8.14) \quad H \text{ contains at most one odd polygon};$$

(3.8.15) if H contains an odd polygon then
 $x(\delta(i)) = b_i$ for all $i \in V(H)$;

(3.8.16) if H contains no polygons then H has at most one node i for which $x(\delta(i)) < b_i$.

We also have an alternating forest F contained in \bar{G} .

(3.8.17) Each $i \in \bar{V}$ such that $x(\delta(i)) < b_i$ is the root of a tree in F .

F is partitioned into two subforests F^0 and F^1 . F^0 consists of all those trees in F such that the root r belongs to \bar{V}^{\leq} and $y_r = 0$ if $r \in V$ or $y_i = 0$ for some $i \in r$ if $r \in R$. F^1 consists of all other trees of F . It will be seen in the description of the algorithm that as long as there are nodes in F^1 , we do not have the optimum feasible matching we seek and as soon as $V(F^1) = \emptyset$, we implicitly have an optimal solution.

In order that x and y be the optimal solutions we seek, all we need is that they satisfy (3.5.3), (3.5.4) and (3.5.11) for as we showed in (3.1.7), this together with the fact that x is a matching will ensure (3.5.5) is satisfied. We will show in the algorithm that if x and y satisfy the following analogues of (3.5.3), (3.5.4) and (3.5.11) then the required x can be obtained in a straightforward fashion.

$$(3.8.18) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in \bar{V}^{\leq},$$

$$(3.8.19) \quad x(\delta(i)) = b_i \quad \text{for all } i \in \bar{V}^=,$$

$$(3.8.20) \quad y_i > 0 \quad \text{for any } i \in V \cap \bar{V}^{\leq} \quad \text{implies} \\ x(\delta(i)) = b_i,$$

$$(3.8.21) \quad y_i > 0 \quad \text{for all } i \in S \cap V^{\leq} \quad \text{for any} \\ S \in R \cap \bar{V}^{\leq} \quad \text{implies} \quad x(\delta(i)) = b_i.$$

We now define a measure of the amount by which (3.8.18)-(3.8.21) are violated. Let

$$(3.8.22) \quad \Delta(\bar{G}; x, y) \equiv \sum (b_i - x(\delta(i))) : i \in \bar{V}^= \quad \text{or} \\ (i \in \bar{V}^{\leq} \cap V \text{ and } y_i > 0) \quad \text{or} \quad (i \in \bar{V}^{\leq} \cap R \text{ and } y_v > 0 \text{ for} \\ \text{all } v \in i \cap V^{\leq}).$$

It follows from the definition of F^1 that

$$(3.8.23) \quad \Delta(\bar{G}; x, y) = \sum (b_i - x(\delta(i))) : i \text{ is the root} \\ \text{of a tree of } F^1).$$

Clearly $\Delta(\bar{G}; x, y) \geq 0$ for any x satisfying (3.8.10) and $\Delta(\bar{G}; x, y) = 0$ if and only if x and y satisfy (3.8.18)-(3.8.21). In general, one "cycle" of the blossom

algorithm will involve finding a new x' and y' and possibly a new graph \bar{G}' such that $\Delta(\bar{G}'; x', y') \leq \Delta(\bar{G}; x, y) - 1$.

(3.8.24) Initially we may take $x_j \equiv 0$ for all $j \in E$, $y_i \equiv \bar{c} \equiv 1/2 \max\{c_j : j \in E\}$ for $i \in V^=$, $y_i \equiv \max\{0, \bar{c}\}$ for $i \in V^<$ and $R \equiv \emptyset$. Then it is easily seen that all our conditions are satisfied. F will be the spanning forest of G in which every tree consists of a single node.

We now describe the algorithm itself.

Step 1: Scan \bar{E} to find an edge j joining an even node v_1 of F^1 to something other than an odd node of F^1 . If no such edge exists go to Step 8. Otherwise go to Step 2.

Step 2: Let $\{v_2\} \equiv \bar{\psi}(j) - \{v_1\}$.

If v_2 belongs to a component of $G^+(x)$ which is not contained in F then go to Step 3.

If v_2 is an even node of a tree in F which is different from the tree containing v_1 then go to Step 4.

If v_1 and v_2 belong to the same tree of F then go to Step 5.

If v_2 is an odd node of a component of F^0 then go to Step 7.

This exhausts all possibilities for v_2 .

Step 3: Grow Forest F . Let K be the component of $G^+(x)$ containing v_2 . If K contains a polygon then go to Step 3c.

Step 3a (see Figure 3.3): If K contains no polygon,

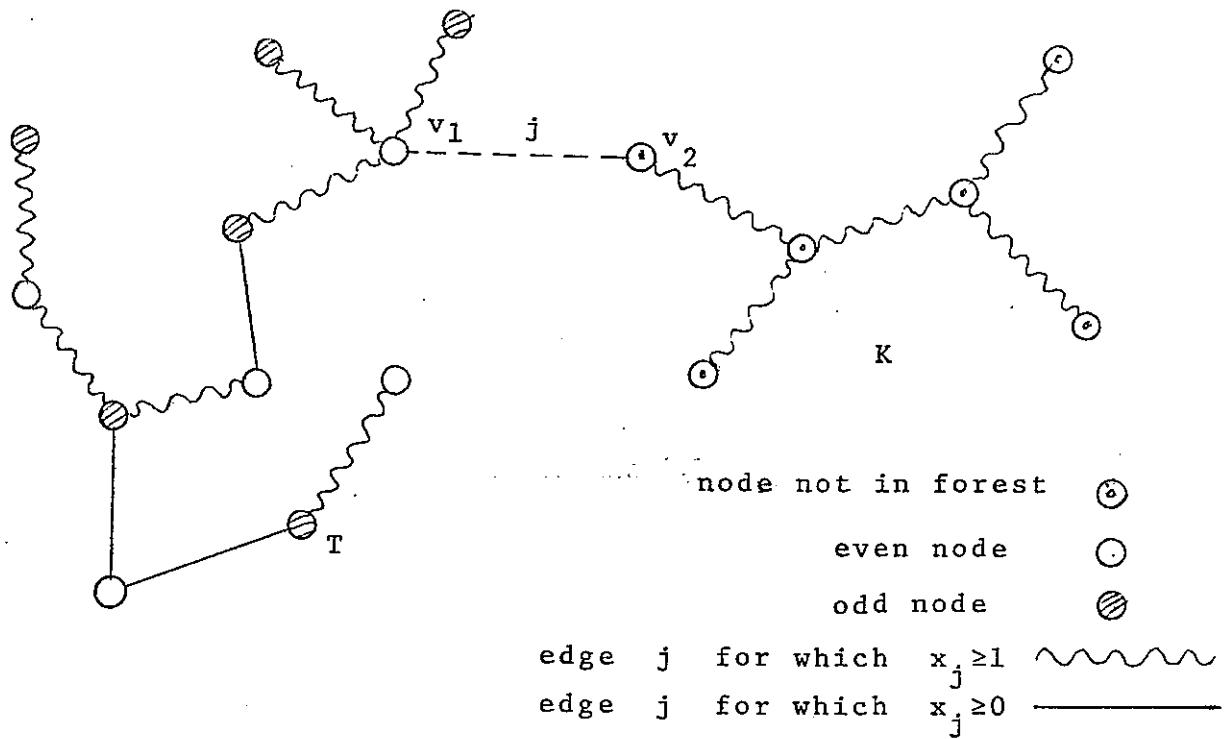


Figure 3.3 Forest Growth

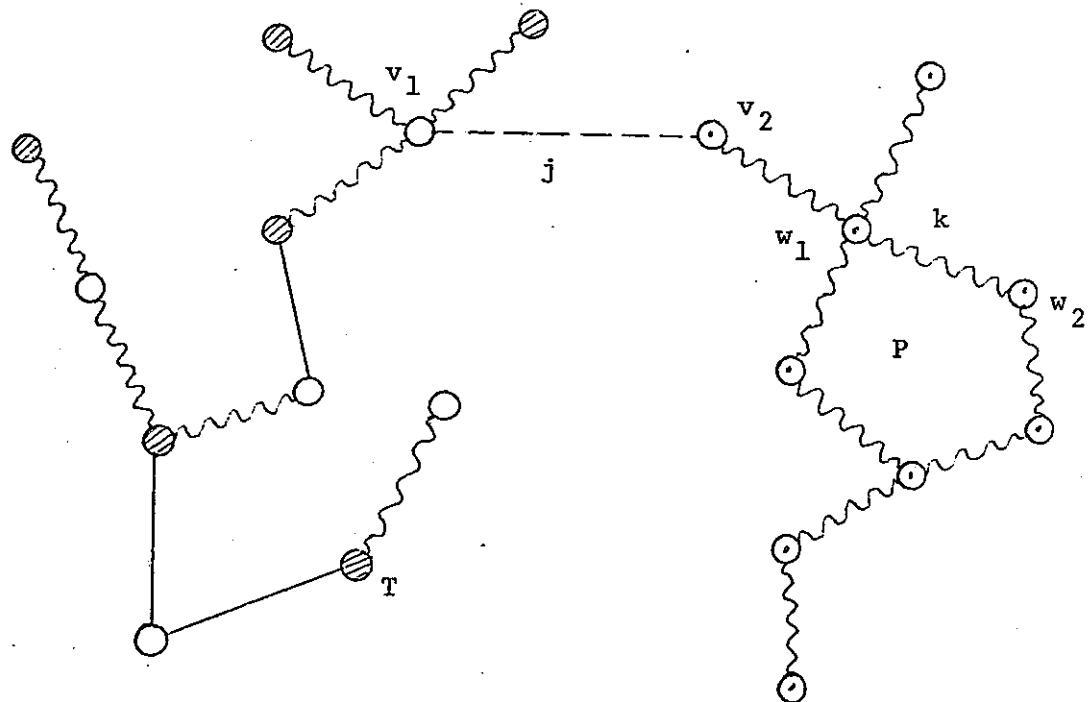


Figure 3.4 Addition of Polygon to Forest

that is, if K is a tree then we grow the alternating tree T containing v_1 by attaching v_2 and K to v_1 by means of the edge j . Since j becomes an odd edge of the new forest F' thereby obtained and by (3.8.17) it is easily seen that (3.6.1)-(3.6.4) are satisfied for F' .

Step 3b: Replace F by F' and go to Step 1.

Step 3c (see Figure 3.4): K contains an odd polygon P . Let w_1 be a node of P which is an odd distance from v_2 in K and for which this distance is as short as possible. Let w_2 be a node of P adjacent to w_1 in P which is no closer to v_2 in P than w_1 . Let k be the edge of P joining w_1 and w_2 . Let K' be the tree obtained from K by removing the edge k . Add K' to the forest by using j as described in Step 3a, thereby obtaining a larger forest F' . Edge k now joins two even nodes of some tree in F' . Replace v_1 , v_2 , j and F by w_1 , w_2 , k and F' respectively and go to Step 5.

Step 4: Augmentation (Two trees) (see Figure 3.5).

Step 4a: Calculation of σ . Let r_2 be the root of the tree T_1 of F^1 containing v_1 and let r_2 be the root of the tree T_2 of F containing v_2 . Let $\sigma_1 \equiv \min\{x_k\}$ where k is an even edge of the path π_1 in T_1 from r_1 to v_1 or let $\sigma_1 \equiv \infty$ if no such edge exists. Let σ_2 and π_2 be analogously defined for T_2 , v_2 and r_2 . By (3.6.4), $\sigma_1, \sigma_2 \geq 1$. Let $\sigma \equiv \min\{\sigma_1, \sigma_2, b_{r_1} - x(\delta(r_1)), b_{r_2} - x(\delta(r_2))\}$. By (3.6.1), $\sigma \geq 1$.

Step 4b: Augmentation. Define x' by

$$x'_k = \sigma \text{ if } k \text{ is an even edge of } \pi_1 \text{ or } \pi_2$$

$$x'_k = \begin{cases} \sigma & \text{if } k \text{ is an odd edge of } \pi_1 \text{ or } \pi_2, \\ & \text{or if } k = j \end{cases}$$

$$x'_k \text{ for all } k \in E - (E(\pi_1) \cup E(\pi_2) \cup \{j\}).$$

Now x' is a matching satisfying (3.8.4), (3.8.6)-(3.8.8),

and (3.8.10) and $\Delta(\bar{G}; x', y) \leq \Delta(\bar{G}; x, y) - 1$ since

$$b_{r_1} - x'(\delta(r_1)) \leq b_{r_2} - x(\delta(r_1)) = 1.$$

Step 4c. Computation of new F. We obtain a new alternating forest in the following way. If $x'(\delta(r_1)) = b_{r_1}$,

then we remove T_1 from F . Similarly if $x'(\delta(r_2)) = b_{r_2}$

then we remove T_2 from F . If k is an even edge of π_1 or π_2 for which $x'_k = 0$ then we remove k and the portion of the tree above it from F . By our choice of σ , at least one of these things must occur. Thus at most one of v_1 and v_2 can be in the new forest F' . If neither are in F' then replace x by x' , F by F' and go to Step 1. If one, say v_1 , belongs to F then perform Step 3a to add the component K of $G^+(x')$ containing v_2 to F'' using the edge j , let F'' be the forest thereby obtained. Replace x by x' , F by F'' and return to Step 1.

Step 5: Augmentation (One tree) (see Figure 3.6)

Step 5a: Calculation of σ and Blossom Test. Let r be the root of the tree T of F^1 containing v_1 and v_2 . Let π_1 be the path in T from r to v_1 and let π_2 be

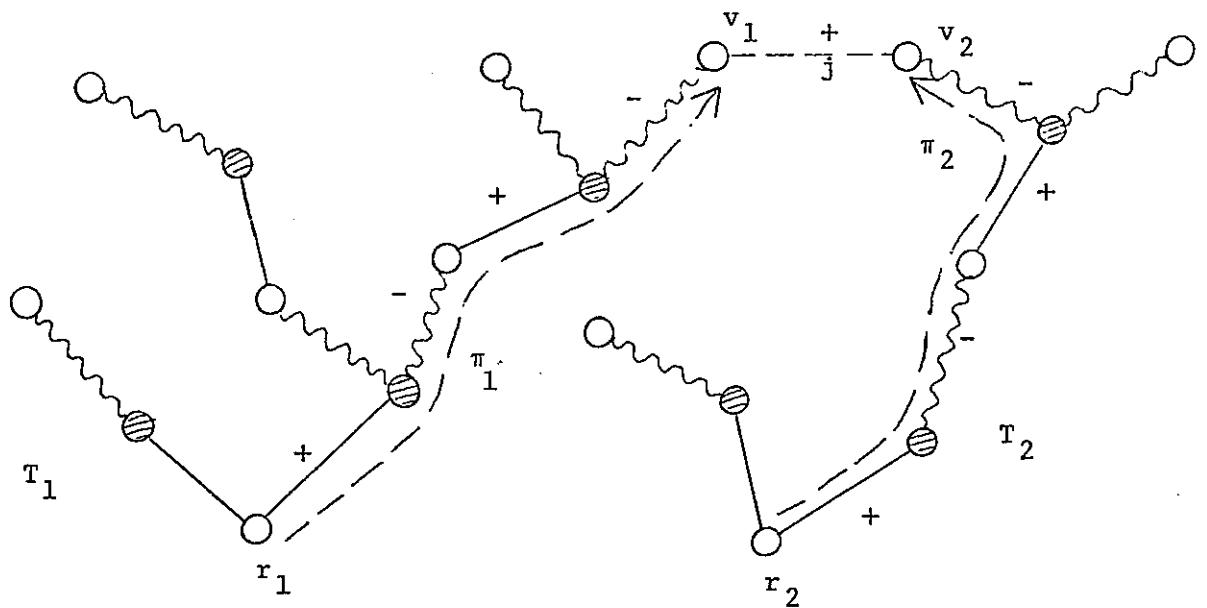


Figure 3.5 Two Tree Augmentation

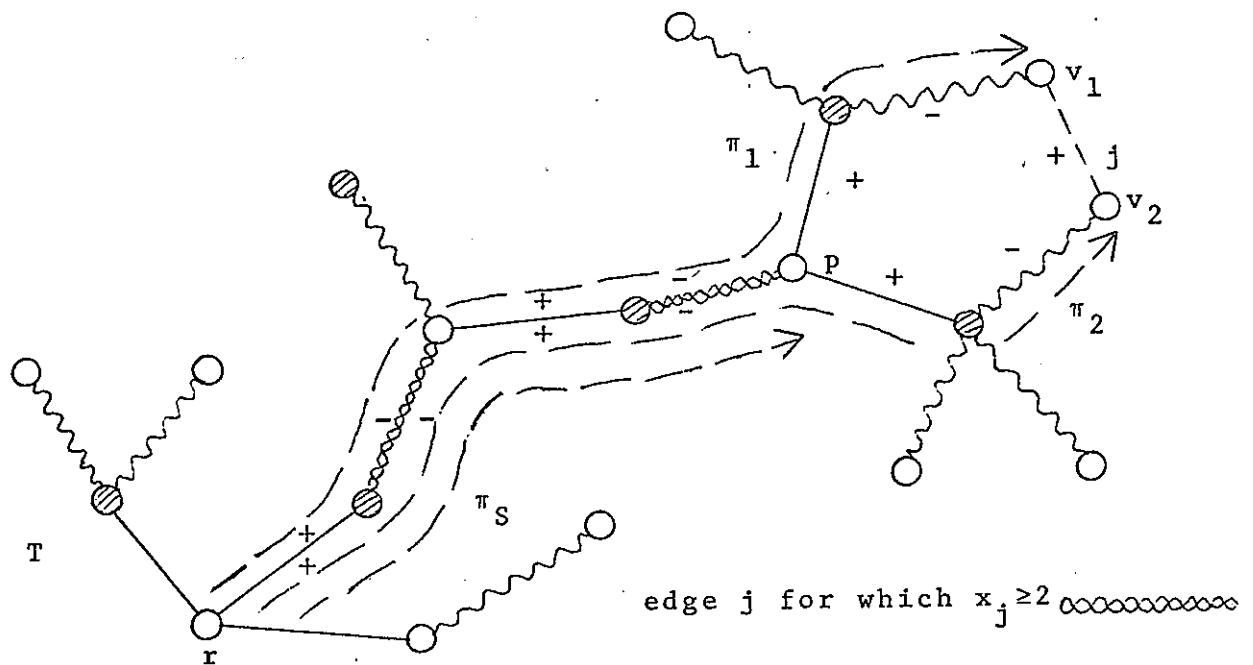


Figure 3.6 One Tree Augmentation

the path in T from r to v_2 . Let π_s be the common position of π_1 and π_2 . π_s is the path in T from r to some node p . (Of course, p may equal r in which case π_s is an empty sequence). Then $E(\pi_1) \cup E(\pi_2) \cup \{j\} - E(\pi_s)$ are the edges of an odd polygon P containing p . ($|E(P)|$ is odd because j joins two even nodes of T .)

Let $\sigma_0 = \min\{x_k : k \text{ is an even edge of } \pi_s\}$, or let $\sigma_0 = \infty$ if no such edge exists. Let

$\sigma_1 = \min\{x_k : k \text{ is an even edge of } \pi_1 \text{ and } k \notin E(\pi_s)\}$. or let $\sigma_1 = \infty$ if no such edge exists. Let σ_2 be defined analogously for π_2 . By (3.6.4), $\sigma_0, \sigma_1, \sigma_2 \geq 1$. Let

$$\sigma = \min\{[1/2 \sigma_0], \sigma_1, \sigma_2, [1/2(b_r - x(\delta(r)))]\}$$

(where for any $\alpha \in \mathbb{R}$, $[\alpha]$ is the largest integer no greater than α). If $\sigma \geq 1$ then go to Step 5b where we augment. Otherwise go to Step 6 where we shrink a portion of \bar{G} .

Step 5b: Augmentation. Define x' as follows.

$$\begin{aligned} x'_k &= 2\sigma \text{ if } k \text{ is an even edge of } \pi_s, \\ x'_k &= 2\sigma \text{ if } k \text{ is an odd edge of } \pi_s, \\ x'_k &= x_k - \sigma \text{ if } k \text{ is an even edge of } \pi_1 \text{ or } \pi_2 \text{ not belonging to } \pi_s, \\ x'_k &= x_k + \sigma \text{ if } k = j \text{ or if } k \text{ is an odd edge of } \pi_1 \text{ or } \pi_2 \text{ not belonging to } \pi_s, \\ x'_k &= x_k \text{ for all } k \in E - (E(\pi_1) \cup E(\pi_2) \cup \{j\}). \end{aligned}$$

We can see by our choice of σ that x' is a matching satisfying (3.8.4), (3.8.6)-(3.8.8), (3.8.10) and $\Delta(\bar{G}; x', y) \leq \Delta(\bar{G}; x, y) - 2$ since $b_r - x'(\delta(r)) \leq b_r - x(\delta(r)) - 2$

and $b_i - x'(\delta(i)) = b_i - x(\delta(i))$ for all $i \in V - \{r\}$.

Step 5c: Computation of new F. Each component H of $G^+(x')$ will satisfy (3.8.13), (3.8.14) and (3.8.16) but need not satisfy (3.8.15). That is there may be a component of $G^+(x')$ containing both a deficient node and an odd polygon. We now analyze the various possibilities.

If $x'(\delta(r)) = b_r$ then let F' be the forest obtained from F by removing T . Since $x'(\delta(i)) = b_i$ for all $i \in V(T)$, each component H of $G^+(x')$ satisfies (3.8.15). F' is an alternating forest. Replace x and F by x' and F' respectively and go to Step 1.

If $x'(\delta(r)) < b_r$ but there are $\ell \in E(\pi_s)$ such that $x'_\ell = 0$, let k be the first such edge in π_s . Let T' be the portion of T above k . Remove T' and k from F thereby obtaining a new alternating forest F' . Since $x'(\delta(i)) = b_i$ for all $i \in V(T')$, each component H of $G^+(x')$ satisfies (3.8.15). Replace x and F by x' and F' and go to Step 1.

If $x'(\delta(r)) < b_r$, $x'_\ell > 0$ for all $\ell \in E(\pi_s)$ but $x'_k = 0$ for some edge k of P , then we remove all such edges k from F thereby obtaining a forest F' . If one end of j , say v_1 , is in F' then the other end v_2 cannot be in F' , adjoin the component H of $G^+(x')$ containing v_2 to F' by means of j , thereby obtaining a new alternating forest F'' . Each component H of $G^+(x')$ satisfies (3.8.15). Replace x and F by x' and F'' and go to Step 1.

Finally, if $x'(\delta(r)) < b_r$ and $x'_\ell > 0$ for all $\ell \in E(\pi_1) \cup E(\pi_2) \cup \{j\}$ then by our choice of σ there is

an even edge k of π_s for which $x'_k = 1$ or $x'(\delta(r)) = b_r - 1$. Replace x by x' and go to Step 6. Note that this is the one case in which there is a component H of $G^+(x')$ violating (3.8.15). This is handled in Step 6.

Step 6: Shrinking Step (see Figure 3.7). We now identify a blossom in \bar{G} . T is the tree of F^1 containing v_1 and v_2 , π_s is the path in T from its root r to the nearest node p of P , the odd polygon formed by adding j to T . Let w be the first even node of π_s such that the path π' in T from w to p contains no even edge k for which $x_k = 1$. (Thus $x_k \geq 2$ for every even edge of π' .) The blossom B consists of P , the subgraph of T induced by π' and any component H of G^+ such that $V(H) \cap V(\pi') \neq \emptyset$ or $V(H) \cap V(P) \neq \emptyset$ except for the even edge of T incident with w if it exists. Let S be the set of all those nodes of G which either belong to $V(B)$ or are contained in pseudonodes of B .

We see that $x(\delta_B(i)) = b_i$ for all $i \in V(B) - \{w\}$ and $x(\delta_B(w)) = b_w - 1$. Thus $x|_{E(B)}$ is a np matching of $\bar{G}[V(B)]$ deficient at w . If $w \notin V(P)$ then we modify our matching so that it will be deficient at a node of P , as this simplifies later discussions. Define x' by

$$\begin{aligned} x'_k &= x_k + 1 \text{ for every odd edge of } \pi' \\ &x'_k - 1 \text{ for every even edge of } \pi'. \end{aligned}$$

If p is an even node of F then let $i(S) \equiv p$. If p is an odd node, let $i(S)$ be an even node of P adjacent to p . Where ℓ is the edge of B joining $i(S)$ and p let

$$x'_\ell \equiv x_\ell - 1.$$

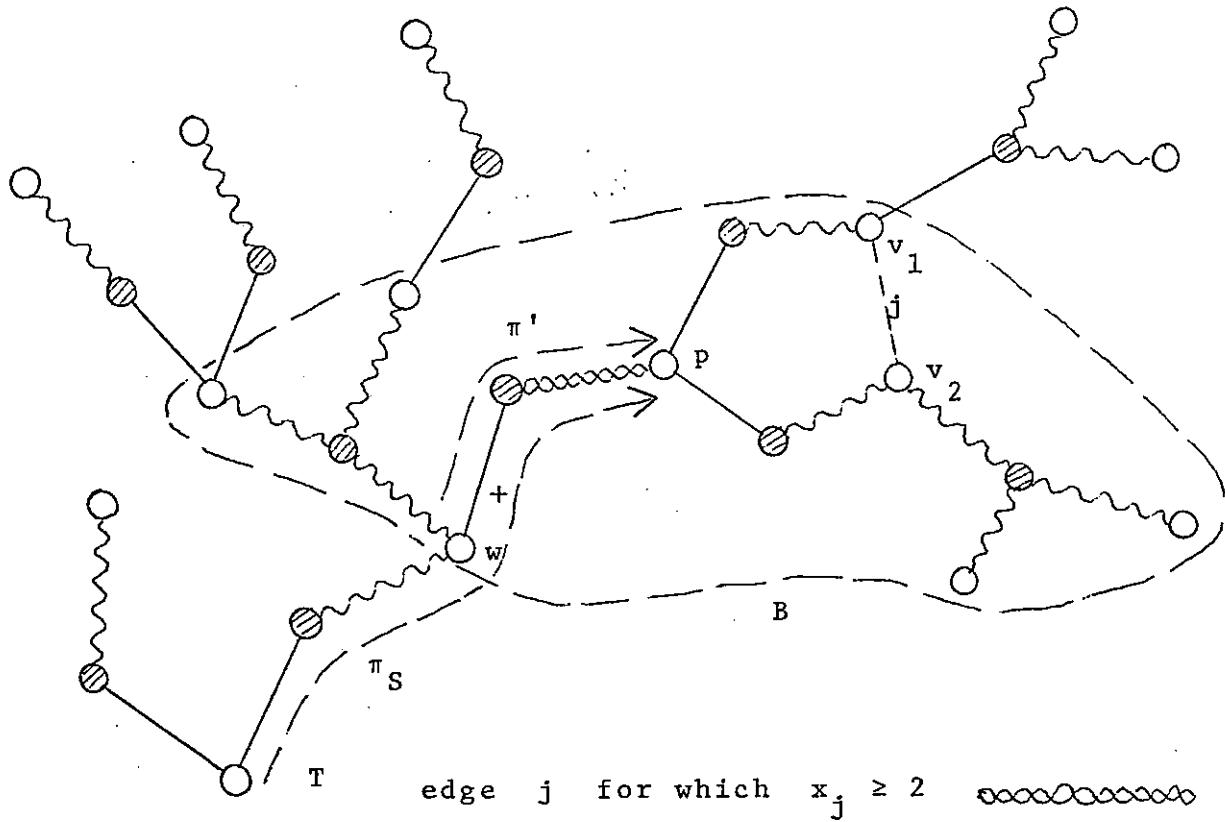


Figure 3.7 Shrinking Step

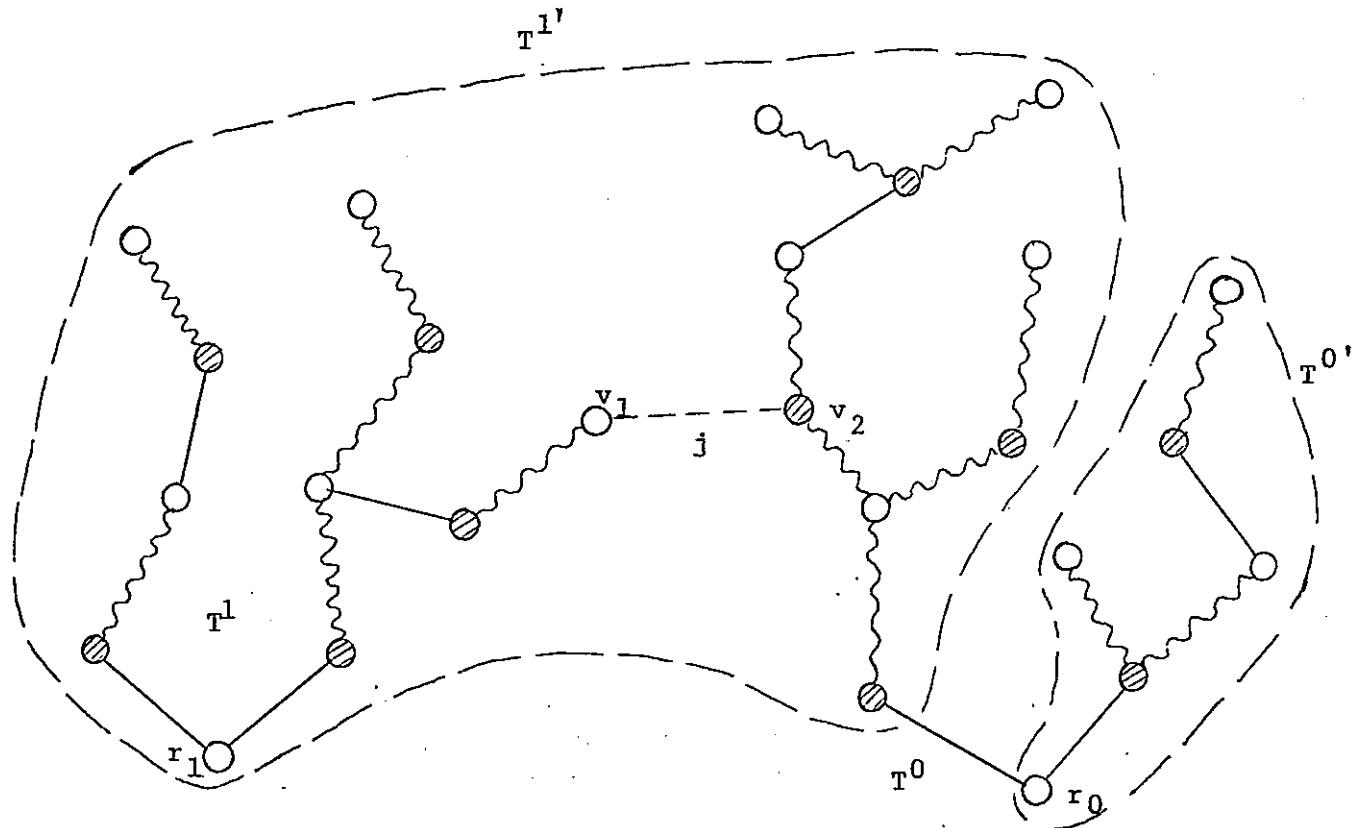


Figure 3.8 Pseudo Forest Growth

For all other edges k of G let $x'_k \equiv x_k$. Now replace x by x' .

$x|E(B)$ satisfies (3.3.1)-(3.3.5) taking $v \in i(S)$.

Let $B(S) \equiv B$. Now if we let $R' \equiv R \cup \{S\}$ we see that R' so defined satisfies (3.8.5)-(3.8.7).

Let $\bar{G}' \equiv G^= \times R'$. Let F' be the forest in \bar{G}' with node set equal to $V(F) \cap V(\bar{G}') \cup \{S\}$ and edge set equal to $E(F) \cap E(\bar{G}')$. Then F' is an alternating forest in \bar{G}' and

(3.8.25) S is an even node of F' .

Let $G'^+(x)$ be defined for \bar{G}' in the same way that $G^+(x)$ was defined for \bar{G} . It is easily seen that every component H of $G'^+(x)$ satisfies (3.8.13)-(3.8.16) since the only component of $G^+(x)$ which could have violated these conditions was the one containing the polygon P and it has been shrunk away.

Note also that $\Delta(\bar{G}'; x, y) \leq \Delta(\bar{G}; x, y)$. Replace \bar{G} , R and F by \bar{G}' , R' and F' respectively and go to Step 1.

Step 7. Grow forest F^1 (Pseudo forest growth). (see Figure 3.8.)

Edge j joins an even node v_1 of a tree T^1 in F^1 to an odd node v_2 of a tree T^0 in F^0 . Let r_0 and r_1 be the roots of T^0 and T^1 respectively. Let \bar{T} be the portion of T^0 above v_2 . We adjoin \bar{T} and the component H of $G^+(x)$ containing x_2 to v_1 by means of the edge j thereby obtaining a larger tree $T^{1'}$. (H is a subgraph of T^0 by (3.6.3).)

If $r_0 \notin V(T^1)$ then replace T^1 by $T^{1'}$ in F thereby obtaining a larger forest $F^{1'}$. Remove \bar{T} , H and any edge of T^0 incident with a node of \bar{T} or H from T^0 , thereby obtaining a smaller tree $T^{0'}$ and a smaller forest $F^{0'}$. Replace F^0, F^1 by $F^{0'}, F^{1'}$ and go to Step 1.

If $r_0 \in V(T^1)$ then remove T^0 from F^0 , let T denote $T^{1'}$ and perform the following step.

Step 7a. (Pseudo Augmentation). Let π be the path in T from r_1 to r_0 . Observe that both r_0 and r_1 are even nodes of T . Let $\sigma_1 = \min\{x_j : j \text{ is an even edge of } \pi\}$. Let $\sigma = \min\{\sigma_1, b_{r_1} - x(\delta(r_1))\}$. Then $\sigma \geq 1$. Let x' be defined by

$$\begin{aligned} x'_k &= \sigma \quad \text{if } k \text{ is an even edge of } \pi \\ x'_k &= x_k + \sigma \quad \text{if } k \text{ is an odd edge of } \pi \\ x'_k &= x_k \quad \text{if } k \notin E(\pi). \end{aligned}$$

Since $b_{r_1} - x'(\delta(r_1)) = b_{r_1} - x(\delta(r_1)) - \sigma$ and $\sigma \geq 1$ it follows from (3.8.23) that $\Delta(\bar{G}; x', y) \leq \Delta(\bar{G}; x, y) - 1$.

If $x'(\delta(r_1)) = b_{r_1}$ then remove T from F^1 thereby obtaining a new forest $F^{1'}$. Reroot T at r_0 and add T to F^0 thereby obtaining a new forest $F^{0'}$. It is easily checked that T rooted at r_0 satisfies (3.6.1)-(3.6.4) with respect to x' .

If $x'(\delta(r_1)) < b_{r_1}$ then by our choice of σ we must have $x'_k = 0$ for some even edge k of π ; let k be the first such edge of π . Let \bar{T} be the portion of T above k . Remove \bar{T} and k from T thereby obtaining a new forest $F^{1'}$. Reroot \bar{T} at r_0 and add it to F^0 thereby obtaining

a new forest F^0' . Again it is easily checked that \bar{T} rooted at r_0 satisfies (3.6.1)-(3.6.4) relative to x' .

Replace x , F^0 and F^1 by x' , F^0' and F^1' respectively and go to Step 1.

Step 8: Termination Test. We now decide whether or not we are ready to go to the final stage of the algorithm. If $V(F^1) = \phi$ then by (3.8.23) $\Delta(\bar{G}; x, y) = 0$ and we go to Step 11, the termination step. Otherwise we go to Step 9 where we will attempt to make a change in the dual variables which will enable further progress.

Step 9: Dual Variable Change.

Step 9a: Calculation of ϵ . Let $E_1 \equiv \{j \in \bar{E}: \text{one member of } \bar{\psi}(j) \text{ is an even node of } F^1 \text{ and the other member of } \bar{\psi}(j) \text{ is not a node of } F^1\}$. If $E_1 = \phi$ then let $\epsilon_1 \equiv \infty$, otherwise let

$$\epsilon_1 \equiv \min\{y(\psi(j)) + y(R(j)) - c_j : j \in E_1\}, \quad \text{where } R(j) \equiv \{S \in R: j \in \gamma(S)\}.$$

Let $E_2 \equiv \{j \in E: \text{both members of } \bar{\psi}(j) \text{ are even nodes of } F^1\}$. If $E_2 = \phi$ then let $\epsilon_2 \equiv \infty$, otherwise let

$$\epsilon_2 \equiv 1/2 \min\{y(\psi(j)) + y(R(j)) - c_j : j \in E_2\}.$$

Let $P \equiv \{S \in R: S \text{ is an odd node of } F^1\}$. If $P = \phi$ then let $\epsilon_3 \equiv \infty$, otherwise let

$$\epsilon_3 \equiv 1/2 \min\{y_S : S \in P\}.$$

Let $Y \equiv \{i \in V^S: i \text{ is an even node of } F^1 \text{ or } i \in S \in R\}$

and S is an even node of F^1 . If $Y = \emptyset$ then let $\epsilon_4 \equiv \infty$, otherwise let

$$\epsilon_4 \equiv \min\{y_i : i \in Y\}.$$

Let $\epsilon \equiv \min\{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$. If $\epsilon = \infty$ then go to Step 10 where we show that there exists no feasible matching. If $\epsilon = 0$ then no dual variable change is necessary so go to Step 9c. Otherwise go to Step 9b where the dual variables are changed.

Step 9b: Change of Dual Variables. Define a new dual solution y' as follows. Let

$$y'_i \equiv \begin{cases} y_i + \epsilon & \text{if } i \in V \text{ is an odd node of } F^1 \text{ or} \\ & \text{belongs to an odd pseudonode of } F^1, \\ y_i - \epsilon & \text{if } i \in V \text{ is an even node of } F^1 \text{ or} \\ & \text{belongs to an even pseudonode of } F^1, \\ y_i & \text{if } i \in V - V(F^1) - u(R \cap V(F^1)). \end{cases}$$

$$y'_S \equiv \begin{cases} y_S + 2\epsilon & \text{if } S \in R \text{ is an even node of } F^1, \\ y_S - 2\epsilon & \text{if } S \in R \text{ is an odd node of } F^1, \\ y_S & \text{if } S \in Q^0 - (R \cap V(F^1)). \end{cases}$$

Because of our choice of ϵ , y' is a feasible dual solution, that is, it satisfies (3.5.7)-(3.5.9). y' also satisfies (3.8.9). Moreover

$$(3.8.26) \quad y'(\psi(j)) + y'(Q^0(j)) = y(\psi(j)) + y(Q(j))$$

for all $j \in E(G^+) \cup E(F) \cup \bigcup_{S \in R} \gamma(S).$

Let $G' =$ be the spanning subgraph of G whose edges are all those $j \in E$ such that $y'(\psi(j)) + y'(Q^0(j)) = c_j$. Let $\bar{G}' \equiv G' \times R$. F is now an alternating forest in \bar{G}' . By (3.8.26) for each $S \in R$, $H(S) = G'[S] \times R[S]$ and $B(S)$ is a blossom spanning $H(S)$ (where $H(S)$ and $B(S)$ are as defined in (3.8.5)).

Step 9c: If $\epsilon \in \{\epsilon_1, \epsilon_2\}$ then there is an edge $j \in E(\bar{G}') - E(\bar{G})$ of the sort we sought in Step 1. Replace y , \bar{G} and G' by y' , \bar{G}' and $G' =$ respectively and go to Step 1 and from there as directed.

Step 9d. If $\epsilon = \epsilon_4$ then let I be the set of nodes $i \in V^L$ such that $y'_i = 0$ and i is either an even node of F^1 or is contained in an even pseudonode of F^1 . Since $\epsilon = \epsilon_4$, $I \neq \emptyset$. For each $i \in I$ let $r(i) \equiv i$ if $i \in V(F^1)$, let $r(i) \equiv S$ if $i \in S \in R \cap V(F^1)$.

For each $i \in I$ such that $r(i)$ is the root of a tree T_i in F^1 , remove T_i from F^1 and add it to F^0 . If any such i exists then we have by (3.8.23) that $\Delta(\bar{G}; x, y') \leq \Delta(\bar{G}; x, y) - 1$; replace y , \bar{G} and G' by y' , \bar{G}' and $G' =$ respectively and go to Step 1.

If there is no $i \in I$ such that $r(i)$ is the root of a tree in F^1 , then choose any $i_0 \in I$, let $r_0 \equiv r(i)$, let T be the tree of F^1 containing r_0 and let r_1 be the root of T . Replace y , \bar{G} , G' by y' , \bar{G}' , $G' =$ respectively and go to Step 7a.

Step 9e. If $\epsilon = \epsilon_3$ then we must expand an odd pseudonode S of F^1 for which $y'_S = 0$. Since $b_S = 1$, by (3.6.2) there

is an edge $j \in \delta_{F_1}(S)$ such that $x_j = 1$. Let $H(S)$ and $B(S)$ be as defined in (3.8.5). Let v be the node of $B(S)$ incident with j . By (3.8.6) we can apply the procedure described in the proof of (3.3.12) to $x|_{E(B(S))}$ and thereby obtain a np matching \bar{x} of $B(S)$ deficient at v .

Let $R' \equiv R - \{S\}$. Since S is a maximal member of R , (3.8.5) is satisfied by R' . Let $\bar{G}'' \equiv G^{\neq} \times R'$. $B(S)$ is a subgraph of \bar{G}'' . Define x' by

$$x'_k \equiv \begin{cases} \bar{x}_k & \text{if } k \in E(B(S)), \\ x_k & \text{if } k \in E - E(B(S)). \end{cases}$$

x' is easily seen to satisfy

$$x'(\delta(i)) \leq b_i \quad \text{for all } i \in V(\bar{G}'').$$

Moreover $\Delta(\bar{G}''; x', y') = \Delta(\bar{G}; x, y)$. Replace j , \bar{G} , G^{\neq} , and R by y' , \bar{G}' , G^{\neq} and R' respectively and go to Step 9f where we determine a new forest F .

Step 9f: If j is an odd edge of F then since by (3.8.4) we have $x_k > 0$ for any even edge of F and since $b_S = 1$ it follows that j is the only edge of F incident with S . Let F' be the subgraph of \bar{G} obtained by replacing the pseudonode S in F with the component K of $G^+(x')$ containing v . Go to Step 9g.

If j is an even edge of F then let ℓ be the unique odd edge of F incident with S . Since S is an odd node of F and $b_S = 1$ these are the only two edges of F incident with S . Let w be the node of $B(S)$ met by ℓ and let π be a track in $B(S)$ from v to w having even

length and for which this length is as small as possible.

Let $\tilde{G}(\pi) = (V(\pi), E(\pi), \tilde{\psi}|E(\pi))$. Let F' be the subgraph of \tilde{G} obtained by replacing the pseudonode S in F with $\tilde{G}(\pi)$ and any component of $G^+(x')$ which contains a node of π .

Step 9g. If F' contains no polygon then it is easily seen that F' is an alternating forest in \tilde{G} ; replace x and F by x' and F' respectively and go to Step 1.

If F' contains a polygon P , then P is the odd polygon of the blossom $B(S)$. Let v_1 be a node of P which is an odd distance from w in $B(S)$ and for which this distance is as small as possible. Let v_2 be a node of P adjacent to v_1 in P and not belonging to the path in P joining w and v_1 . Let j' be the edge of P joining v_1 and v_2 . Remove j' from F' , let F'' be the forest thereby obtained. Now j' joins two even nodes of F'' . Replace F and j by F'' and j' respectively and go to Step 5. At this point F fails to be an alternating forest because j violates (3.6.3) and the component H of $G^+(x)$ containing v_1 may not satisfy (3.8.15). However these situations are automatically corrected in Steps 5 and 6.

Step 10: Hungarian Forest. Since $\epsilon = \infty$ we observe the following. $\epsilon_1 = \infty$ is equivalent to F^1 satisfying (3.7.4). $\epsilon_2 = \infty$ is equivalent to (3.7.3) for F^1 . $\epsilon_3 = \infty$ is equivalent to (3.7.5) for F^1 and $\epsilon_4 = \infty$ is equivalent to (3.7.6) for F^1 . Therefore F^1 is a Hungarian forest so by (3.7.40), G has no feasible matching. By (3.7.17) and (3.8.23),

$D(G, V^{\neq}) = \Delta(\bar{G}; x, y)$. If desired, perform Step 12 so as to "correct" the matching x for edges $j \in \gamma(S)$ for $S \subseteq R$ so that the resulting matching x' will satisfy $x'(\delta(i)) \leq b_i$ for all $i \in V$ and $d(G, V^{\neq}; x') = D(G, V^{\neq})$. We do not bother performing Step 12 in the applications we make of this algorithm in later chapters.

(3.8.27) Finally note that if F_1 is a Hungarian forest, then for any $\epsilon \in \mathbb{R}$ such that $\epsilon \geq 0$ we have that y' as defined in Step 9b is a feasible dual solution also satisfying (3.8.26).

Step 11: Termination with Optimal Solution. Apply Step 12 to "correct" the matching x and then stop, the resulting matching x is the optimal feasible matching we seek and y is an optimum dual solution. Since Step 12 ensured that $\Delta(G; x, y) = 0$ and $x(\delta(i)) \leq b_i$ for all $i \in V$ it follows that (3.5.3) and (3.5.4) and x and y satisfy (3.5.11). Since x is a matching satisfying (3.5.3) and (3.5.4), we also have (3.5.2) and (3.5.5) satisfied. By (3.8.8) and (3.8.9) we know that (3.5.12) is satisfied. By (3.8.2) y satisfies (3.5.7)-(3.5.9). Therefore x is the optimal matching we require and y is an optimal dual solution.

Step 12: Pseudonode Matching Correction. Let $D = \emptyset$. D is the set of members of R for which the matching has been corrected.

Step 12a: If $R = D$ then return to Step 10 or 11 from whence we come.

Step 12b: Let S be a maximal member of $R - D$ and let $D' \equiv D \cup \{S\}$. Let $G' \equiv G^= \times (R - D')$. Then $B(S)$ (as defined in (3.8.5)) is a blossom contained in G' . If $x(\delta(S)) = 0$ then go to Step 12d.

Step 12c: Let $j \in \delta(S)$ be such that $x_j = 1$, let v be the node of $B(S)$ met by j in G' . Apply the procedure described in the proof of (3.3.12) to obtain a np matching \hat{x} of $B(S)$ deficient at v . Let x' be defined by

$$(3.8.28) x'_k \equiv \begin{cases} x_k & \text{for } k \in E - E(B(S)), \\ \hat{x}_k & \text{for } k \in E(B(S)). \end{cases}$$

Then we have

$$(3.8.29) \quad x'(\delta(i)) \leq b_i \quad \text{for all } i \in V(G'),$$

$$(3.8.30) \quad x'(\gamma(T)) = q_T \quad \text{for all } T \in R,$$

$$(3.8.31) \quad \Delta(G'; x', y) = \Delta(\bar{G}; x, y).$$

Replace x , D and \bar{G} by x' , D' and G' respectively.

Return to Step 12a.

Step 12d. Let $v \in S \cap V^<$ be such that $y_v = 0$ if such a node exists, otherwise go to Step 12e. Let $r \equiv v$ if $v \in V(B(S))$, let $r \equiv T$ if $v \in T \in R \cap V(B(S))$. As in Step 12c we apply the procedure described in the proof of (3.3.12) to obtain a np matching \hat{x} of $B(S)$ deficient at r . Let x' be defined as in (3.8.28). Again (3.8.29) and (3.8.30) are immediate and since the only new deficiency we created was at r and since $y_v = 0$, (3.8.31) is satisfied.

Replace x , D and \bar{G} by x' , D' and G' respectively.

Return to Step 12a.

Step 12e: (This step can only be performed if we terminated in Step 10.) In this case $S \in \bar{V}^=$ since S must have been an even node or contained in an even pseudonode of the Hungarian forest F^1 . Therefore the term corresponding to S contributes 1 to $\Delta(\bar{G}; x, y)$. Let v be the node of $B(S)$ at which $x|E(B(S))$ is deficient. If we let $x' \equiv x$ then (3.8.29) and (3.8.30) are satisfied and since the term corresponding to v contributes 1 to $\Delta(G'; x', y)$ we have (3.8.31) satisfied. Go to Step 12a.

3.9 Efficiency of the Blossom Algorithm.

In this section we derive an upper bound on the amount of work done by the blossom algorithm in solving a matching problem. We make a fixed word assumption, that the amount of work required to perform arithmetic (addition, subtraction, division by two) on any numbers encountered in the algorithm is independent of the number of significant digits. Since this is the way in which most large scale computers operate (for reasonably sized numbers) this is a realistic assumption.

(3.9.1) Theorem. An upper bound on the amount of work required by the blossom algorithm to solve a matching problem is of the order

$$\Delta(G; x^0, y^0) \cdot |v| \cdot |E|$$

where x^0 and y^0 are the starting matching and dual solution.

Proof. First we establish an upper bound on the amount of work that can be done by the algorithm without decreasing $\Delta(\bar{G}; x, y)$. Steps 4, 5, 7a and 9d all decrease $\Delta(\bar{G}; x, y)$ by at least one.

In Steps 3 and 7 we grow the forest F^1 . Since $|V(F^1)|$ decreases only after performing one of Steps 4, 5, 7a or 9d, it follows that Steps 3 and 7 can be performed at most $|V|$ times without a decrease in $\Delta(\bar{G}; x, y)$.

In Step 6 we shrink. By (3.8.5) $n(S) \geq 3$ for every $S \in R$ (where $n(S)$ is as defined in (3.2.7)). Thus by (3.2.8) we must have $|R| \leq 1/2(|V| - 1)$ at any point in the algorithm. By (3.8.25) any new S added to R becomes an even node of F^1 . We only expand odd nodes of F^1 (in Step 9e). Thus Step 6 can be performed at most $1/2(|V| - 1)$ times without a decrease in $\Delta(\bar{G}; x, y)$.

In Steps 9e-9g we expand an odd pseudonode of F^1 . This pseudonode must have been in F^1 following the previous augmentation since any pseudonode formed since is an even node of F^1 . Hence Steps 9e-9g can be performed at most $1/2(|V| - 1)$ times without making a change in $\Delta(\bar{G}; x, y)$.

Steps 10, 11, 12 are performed only once in the course of the algorithm. A bound on the amount of work required by these steps is of the order $|V| \cdot |E|$.

Steps 1, 2, 8, 9a, 9b, 9c are performed at most once for each performance of steps 3, 4, 5, 6, 7, 10, 11, 12. A bound on the amount of work performed by each of these can be seen to be of the order $|E|$. The only ones of these steps for which this bound is not obvious are 9a, 9b. However if

we preserve the value of $y(\psi(j)) + y(Q^0(j)) - c_j$ for each $j \in E$ at all times, then it can be seen that this bound is satisfied for these steps.

Finally a bound on the amount of work required for each of Steps 4, 5, 7a or 9d is of the order $|E|$.

Thus a bound on the amount of work that can be done without decreasing $\Delta(\bar{G}; x, y)$ by at least one is of the order $|E| \cdot |V|$ and the theorem follows. \square

(3.9.2) Corollary. If we start with the matching described in (3.8.24) then an upper bound on the amount of work done in solving a matching problem is of the order

$$b(V) \cdot |V| \cdot |E| .$$

Proof. This follows from the fact that if x and y are as defined in (3.8.24) then $\Delta(G; x, y) \leq b(V)$. \square

3.10 Min-Max Theorems and Discreteness of the Dual Solution

Whenever we know a set of linear inequalities sufficient to define a polyhedron P , linear programming duality gives us a min-max theorem concerning any subset of P that contains the vertices. Conversely, we used the blossom algorithm to prove the following min-max theorem which established Theorem (3.4.5).

(3.10.1) Theorem. Let $G = (V, E, \psi)$ be a graph,
let $b = (b_i : i \in V)$ be a vector of positive integers and
let $c = (c_j : j \in E)$ be an arbitrary real vector. Then the
maximum value of $c \cdot x$ for any matching x of G which
satisfies

$$(3.10.2) \quad \underline{x(\delta(i)) \leq b_i} \text{ for all } i \in V$$

is equal to the minimum value of

$$(3.10.3) \quad \underline{\sum(b_i y_i : i \in V) + \sum(q_s y_s : s \in Q^0)}$$

for real $(y_i : i \in V)$ and $(y_s : s \in Q^0)$ satisfying

$$(3.10.4) \quad \underline{y_i \geq 0 \text{ for all } i \in V},$$

$$(3.10.5) \quad \underline{y_s \geq 0 \text{ for all } s \in Q^0}$$

$$(3.10.6) \quad \underline{y(\psi(j)) + y(Q^0(j)) \geq c_j \text{ for all } j \in E.}$$

If the objective function c satisfies certain discreteness properties, then we are able to require certain discreteness properties of the dual variables.

(3.10.7) Theorem. If c_j is integer valued for all $j \in E$ then there is an optimal feasible solution y^0 to the problem of minimizing (3.10.3) subject to (3.10.4)-(3.10.6) which satisfies

(3.10.8) y_i is congruent with $0 \pmod{1/2}$ for all $i \in V$,

(3.10.9) y_s is congruent with $0 \pmod{1}$ for all $s \in Q^0$.

Proof. The problem of minimizing (3.10.3) subject to (3.10.4)-(3.10.6) is the dual linear program to the matching problem maximize cx for $x \in P(G, b)$. We will show that

(3.10.10) if the starting dual solution used by the blossom algorithm is integer valued, then at any point

in the solution of this matching problem the dual solution y will satisfy (3.10), (3.10.9). This we prove by showing that at any point of the algorithm.

(3.10.11) the values of y_i for $i \in V$ belonging to F^1 or contained in a pseudonode of F^1 will be congruent modulo 1.

If the initial dual solution is integer valued, (3.10.8), (3.10.9) and (3.10.11) are obviously satisfied. Now observe that at no point of the algorithm do we add a new tree to F^1 . Moreover at any time we grow a tree in F^1 , all new edges j must belong to the equality subgraph so since c_j is integer valued for all such j , (3.10.8) and (3.10.9) ensure that (3.10.11) will continue to hold.

When computing ϵ so as to make a change of dual variables, (3.10.8), (3.10.9) and (3.10.11) ensure that any of $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ which are finite will be congruent with $0(\text{mod } 1/2)$. Since $V' = \emptyset$, we cannot obtain a Hungarian forest so ϵ is finite and congruent with $0(\text{mod } 1/2)$. Hence y' as defined in Step 9b also satisfies (3.10.8), (3.10.9) and (3.10.11). Thus (3.10.10) is proved and the theorem follows. \square

The following is obtained by combining (3.10.1) and (3.10.7).

(3.10.12) Theorem. If c is integer valued, then the maximum value of cx for any matching x of G satisfying (3.10.2) is equal to the minimum of (3.10.3) subject to (3.10.4)-(3.10.6) and an optimal y can be chosen so as to satisfy (3.10.8), (3.10.9).

In the case that c is further restricted to being 0, 1 valued, we can obtain the following result.

(3.10.13) Theorem. If $c_j \in \{0, 1\}$ for all $j \in E$ then there is an optimal feasible solution y^0 to the problem of minimizing (3.10.3) subject to (3.10.4)-(3.10.6) which satisfies

(3.10.14) $y_i, y_S \in \{0, 1\}$ for all $i \in V$, for all $S \in Q^0$.

Proof. Let $G = (V, E, \psi)$ be a graph for which (3.10.13) fails for some b and such that $|V \cup E|$ is minimum. Clearly $|V| \geq 3$, and we must have $c_j = 1$ for all $j \in E$ since the graph obtained by deleting any edge k for which $c_k = 0$ would still violate (3.10.13).

Suppose G has a perfect matching x^0 . Then the maximum value of $c \cdot x$ over matchings x of G satisfying (3.10.2) is equal to $1/2b(V)$. Choose $v \in V$ and define b' by

$$b'_i = \begin{cases} b_i & \text{for } i \in V - \{v\}, \\ b_i + 1 & \text{for } i = v. \end{cases}$$

Then the maximum of $c \cdot x$ over matchings x of G satisfying (3.10.2) is still $1/2b(V)$. Suppose that y^0 is an optimal dual solution relative to b' which satisfies

(3.10.14). Then $\sum(b'_i y_i^0 : i \in V) + \sum(q_S y_S^0 : S \in Q^0) = 1/2b(V)$.

Hence y^0 is an optimum dual solution relative to b but y^0 satisfies (3.10.14), a contradiction. Hence no optimum

solution relative to b' can satisfy (3.10.14) and since G can have no perfect b' -matching, we assume

(3.10.15) b is chosen so that G has no perfect b -matching.

Let y^0 be an optimum solution relative to b satisfying (3.10.8), (3.10.9). Clearly we have $y_i^0 \in \{0, 1/2, 1\}$ for all $i \in V$ and $y_S^0 \in \{0, 1\}$ for all $S \in Q^0$. Let $W \equiv \{i \in V : y_i^0 = 1/2\}$. If $W = \emptyset$ then y^0 satisfies (3.10.14) and we are finished. If $W = V$ then $\sum b_i y_i^0 : i \in V + \sum q_S y_S^0 : S \in Q^0 \geq 1/2b(V)$ implying G has a perfect matching, contradictory to (3.10.15). Thus we have

$$(3.10.16) \quad \emptyset \neq W \subset V.$$

(3.10.17) For any $j \in \delta(W)$ we must have either $y_v^0 = 1$ where $\{v\} = \psi(j) - W$ or $j \in \gamma(S)$ for some $S \in Q^0$ such that $y_S^0 = 1$.

Otherwise we could have $y^0(\psi(j)) + y^0(Q^0(j)) = 1/2$ contradictory to (3.10.6).

By our minimality assumption for G and (3.10.16) there is an optimal solution y^1 satisfying (3.10.14) to the problem

$$\text{minimize } \sum (b_i y_i : i \in W) + \sum (q_S y_S : S \in Q_W^0)$$

subject to $y_i \geq 0$ for all $i \in W$

$$y_S \geq 0 \text{ for all } S \in Q_W^0$$

$$y(\psi(j)) + y(Q_W^0(j)) \geq 1 \text{ for all } j \in E(G[W])$$

where $Q_W^0 \equiv \{S \in Q^0 : S \subseteq W\}$ and $Q_W^0(j) \equiv \{S \in Q_W^0 : j \in \gamma(S)\}$

for all $j \in J$. If we define y^* by

$$y_i^* = \begin{cases} y_i^0 & \text{for } i \in V - W, \\ y_i^1 & \text{for } i \in W; \end{cases}$$

$$y_s^* = \begin{cases} y_s^0 & \text{for } s \in Q^0 - Q_W^0, \\ y_s^1 & \text{for } s \in Q_W^0 \end{cases}$$

then y^* satisfies (3.10.14) and by (3.10.17), y^* is a feasible solution to the problem of minimizing (3.10.3) subject to (3.10.4)-(3.10.6). Since

$$\sum (b_i y_i^1 : i \in W) + \sum (q_s y_s^1 : s \in Q_W^0)$$

$$\leq \sum (b_i y_i^0 : i \in W) + \sum (q_s y_s^0 : s \in Q_W^0)$$

and since y^0 was optimal it follows that y^* is optimal.

This contradicts the choice of G and completes the proof. \square

Combining (3.10.13) and (3.10.1), we obtain the following.

(3.10.18) Theorem: If $c_j \in \{0, 1\}$ then the maximum value of cx for any matching x of G satisfying (3.10.2) is equal to the minimum of (3.10.3) subject to (3.10.4)-(3.10.6) and a minimum y can be chosen so as to satisfy (3.10.14).

Theorem (3.10.18) can be specialized in the following manner. Let $G = (V, E, \psi)$ be a graph and let $b = (b_i : i \in V)$ be a vector of positive integers. For any $X \subseteq V$ we define

(3.10.19) $C(X) = \{S \subseteq V - X : G[S] \text{ is a component of } G[V - X]\}.$

We partition $C(X)$ as follows.

$$(3.10.20) \quad C_0(X) \equiv \{S \in C(X) : |S| = 1\},$$

$$(3.10.21) \quad C_1(X) \equiv \{S \in C(X) : |S| > 1 \text{ and } b(S) \text{ is odd}\}$$

$$(3.10.22) \quad C_2(X) \equiv \{S \in C(X) : |S| > 1 \text{ and } b(S) \text{ is even}\}.$$

(3.10.23) Theorem. $\max\{x(E) : x \text{ is a matching of } G$

satisfying (3.10.2) $= 1/2(b(V) + \min\{b(X) - |C_1(X)| -$

$b(\cup(C_0(X))) : X \subseteq V$). Moreover

(3.10.23a) there is a set $X^* \subseteq V$ which minimizes
 $b(X) - |C_1(X)| - b(\cup(C_0(X)))$ over $X \subseteq V$ and satisfies
 $C_2(X^*) = \emptyset$ and $C_1(X^*) \subseteq Q^0$.

Proof. Let x be any matching of G which satisfies

(3.10.2). Let $X \subseteq V$. Then for any $S \in C_1(X)$ we have

$x(\gamma(S)) \leq 1/2(b(S) - 1)$ (by (3.1.7)). Therefore

$$(3.10.24) \quad b(\cup(C_1(X))) - |C_1(X)| \geq 2\sum_{S \in C_1(X)} x(\gamma(S)).$$

Let $J \equiv \delta(X) \cap (\delta(\cup(C_0(X)) \cup C_1(X)))$. Then we have

$$(3.10.25) \quad b(X) + b(\cup(C_2(X))) \geq 2x(\gamma(X \cup \cup(C_2(X)))) + x(J).$$

We also have

$$(3.10.26) \quad b(X) \geq x(J).$$

Summing (3.10.24)-(3.10.26) we obtain

$$[b(X) + b(\cup(C_1(X))) + b(\cup(C_2(X)))] + b(X) - |C_1(X)| \geq 2x(E)$$

or

$$b(V) - b(\cup(C_0(X))) - |C_1(X)| + b(X) \geq 2x(E)$$

Therefore

(3.10.27) $\max\{x(E) : x \text{ is a matching of } G$
 $\text{satisfying (3.10.2)}\} \leq 1/2b(V) + 1/2\min\{b(X) - |C_1(X)| -$
 $b(\cup(C_0(X))) : X \subseteq V\}.$

We now show that equality holds.

By (3.10.13) there is a y satisfying (3.10.14) which minimizes (3.10.3) subject to (3.10.4)-(3.10.6) taking $c_j \equiv 1$ for all $j \in E$. Let y^0 be such a solution for which the cardinality of $Z = \{S \in Q^0 : y_S^0 = 1\}$ is as small as possible. Suppose $S, T \in Z$ are such that $S \cap T \neq \emptyset$. If $b(S \cap T) \geq 2$ then if we define y' by

$$y'_i \equiv \begin{cases} y_i^0 & \text{if } i \in V - (S \cup T) \\ y_i^0 + 1/2 & \text{if } i \in S \cup T \end{cases}$$

$$y'_R \equiv \begin{cases} y_R^0 & \text{if } R \in Q^0 - \{S, T\} \\ 0 & \text{if } R \in \{S, T\} \end{cases}$$

it is easily seen that y' is a feasible solution to (3.10.4)-(3.10.6) for which (3.10.3) attains a smaller value than for y^0 , a contradiction to our choice of y^0 . If $b(S \cap T) = 1$, and hence $|S \cap T| = 1$, then $S \cup T \in Q^0$ and if we define y' by

$$\begin{aligned} y'_i &\equiv y_i && \text{for all } i \in V \\ y'_R &\equiv \begin{cases} y_R^0 & \text{if } R \in Q^0 - \{S, T, S \cup T\} \\ 1 & \text{if } R = S \cup T \\ 0 & \text{if } R \in \{S, T\} \end{cases} \end{aligned}$$

then y' is a feasible solution to (3.10.4)-(3.10.6) satisfying (3.10.14) for which the value of (3.10.3) is no greater than that obtained for y^0 . But $|\{R \in Q^0 : y_R' = 1\}| < |Z|$, contradictory to our choice of y^0 . Hence

(3.10.28) the members of Z are pairwise disjoint.

Suppose $y_v^0 = 1$ for some $v \in S \in Z$. Then if we define y' by

$$y_i' = \begin{cases} y_i^0 + 1/2 & \text{if } i \in V - \{v\} \\ y_i^0 = 1 & \text{if } i = v \end{cases}$$

$$y_R' = \begin{cases} y_R^0 & \text{if } R \in Q^0 - \{S\} \\ 0 & \text{if } R = S \end{cases}$$

y' is a feasible solution to (3.10.4)-(3.10.6) which causes (3.10.3) to assume a smaller value than for y^0 , a contradiction. Hence

$$(3.10.29) \quad y_i^0 = 0 \quad \text{for all } i \in S \in Z.$$

Let $X \equiv \{i \in V : y_i^0 = 1\}$. Because of (3.10.29), in order for y^0 to be feasible we require

$$\delta(S) \subseteq \delta(X) \quad \text{for every } S \in Z,$$

$$\delta(i) \subseteq \delta(X) \quad \text{for every } i \in V - \cup(Z)$$

such that $y_i^0 = 0$. Hence $C_0(X) = \{\{i\} \in V - V(Z) : y_i^0 = 0\}$,

$$(3.10.30) \quad C_1(X) = Z$$

$$(3.10.31) \quad C_2(X) = \emptyset.$$

Hence

$$\begin{aligned}
 (3.10.32) \quad & \sum (b_i y_i^0 : i \in V) + \sum (q_S y_S^0 : S \in Q^0) \\
 &= b(X) + \sum (1/2(b(S) - 1) : S \in C_1(X)) + 1/2b(u(C_2(X))) \\
 &= 1/2b(X) + 1/2b(u(C_1(X))) + 1/2b(u(C_2(X))) + \\
 &\quad 1/2b(X) - 1/2|C_1(X)| \\
 &= 1/2b(V) + 1/2b(X) - |C_1(X)| - b(u(C_0(X))).
 \end{aligned}$$

Since by (3.10.18) and our choice of y^0 ,

$$\begin{aligned}
 & \max\{x(E) : x \text{ is a matching of } G \text{ satisfying (3.10.2)}\} \\
 &= \sum (b_i y_i^0 : i \in V) + \sum (q_S y_S^0 : S \in Q)
 \end{aligned}$$

it follows from (3.10.30) that equality holds in (3.10.27). Since $Z \subseteq Q^0$, (3.10.30) and (3.10.31) imply (3.10.23a) completing the proof. \square

Theorem (3.10.23) (excluding (3.10.23a)) reduces to a theorem of Berge [B2] when it is further specialized to 1-matchings.

G has a perfect matching if and only if $\max\{x(E) : x \text{ is a matching of } G \text{ satisfying (3.10.2)}\} = 1/2b(V)$. Therefore, by (3.10.23), G has a perfect matching if and only if

$$\min\{b(X) - |C_1(X)| - b(u(C_0(X))) : X \subseteq V\} = 0.$$

Thus we obtain the fundamental theorem of Tutte.

(3.10.33) Theorem (Tutte [T3]). $G = (V, E, \psi)$ has a perfect matching if and only if for each $X \subseteq V$,

$$(3.10.34) \quad b(X) \geq |C_1(X)| + b(u(C_0(X))).$$

In the case of 1-matchings this reduces to the well known theorem

(3.10.35) Theorem (Tutte [T1]). $G = (V, E, \psi)$
has a perfect 1-matching if and only if for any $X \subseteq V$ the
number of components of $G[V - X]$ having an odd number of
nodes is no greater than $|X|$.

The importance of (3.10.23a) to these theorems is discussed in Section 4.4 (see Theorems (4.4.21) and (4.4.22)).

Chapter 4

Facets and Vertices of Matching Polyhedra

Throughout this chapter we consider a graph $G = (V, E, \psi)$ and we take $b = (b_i : i \in V)$ to be a vector of positive integers. Since isolated nodes, that is nodes v for which $\delta(v) = \emptyset$, are of little interest in matching theory we assume G has no isolated nodes. In section 3.4 we defined the matching polyhedron $P(G, b)$ and proved the theorem of Edmonds that

$$P(G, b) = \{x \in \mathbb{R}^E :$$

$$(4.0.1) \quad x_j \geq 0 \quad \text{for all } j \in E,$$

$$(4.0.2) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V,$$

$$(4.0.3) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q^0},$$

where $Q^0 \equiv \{S \subseteq V : G[S] \text{ is shrinkable}\}$, and $q_S \equiv (1/2)(b(S)-1)$ for any set S such that $b(S)$ is odd. We now characterize the facets and vertices of $P(G, b)$ relating them to the structure of G and the value of b . In particular, for any G and b we prescribe a unique minimal subset of the inequalities (4.0.1)-(4.0.3) of which $P(G, b)$ is the solution set.

The material presented in this chapter does rely to an extent upon the material of Chapter 3. Sections 3.3 and 3.4 are used in characterizing the facets of $P(G, b)$, some of the material of Sections 3.6 and 3.7 is used in showing the equivalence of shrinkable graphs and b -critical graphs. The proof of the vertex characterization is related to the

algorithm itself; in proving the theorem we also show that every matching obtained by the blossom algorithm is a vertex of $P(G, b)$. However we give an additional proof of this portion of the vertex characterization which is developed from basic properties of graph theory and polyhedra theory.

4.1. Dimension of $P(G, b)$ and Nonnegativity Facets

In order to characterize the facets of $P(G, b)$, we first determine its dimension.

(4.1.1) Proposition. $P(G, b)$ is of full dimension.

Proof. Since $P(G, b) \subseteq \mathbb{R}^E$ it follows that $\dim(P(G, b)) \leq |E|$. We show that $\dim(P(G, b)) = |E|$ by exhibiting $|E| + 1$ affinely independent matchings belonging to $P(G, b)$. The result will then follow from (2.2.12).

For each $j \in E$ we define a matching x^j by

$$x_k^j = \begin{cases} 0 & \text{if } k \neq j, \\ 1 & \text{if } k = j. \end{cases}$$

Since $b_i \geq 1$ for all $i \in V$, we have $x^j(\delta(i)) \leq b_i$ for all $i \in V$, for all $j \in E$. Let 0 be the zero vector in \mathbb{R}^E . Then $\{x^j : j \in E\} \cup \{0\} \subseteq P(G, b)$. The set of vectors $\{x^j : j \in E\} \cup \{0\}$ is easily seen to be affinely independent and the result follows. \square

Let $a \in \mathbb{R}^E$, $\alpha \in \mathbb{R}$. We say that the linear inequality $ax \leq \alpha$ gives a facet of $P(G, b)$ if $\{x \in P(G, b) : ax = \alpha\}$ is a facet of $P(G, b)$. In characterizing which of the

inequalities (4.0.1)-(4.0.3) give facets of $P(G, b)$ we use mainly the technique of showing that $ax \leq \alpha$ gives a facet of $P(G, b)$ by displaying $|E|$ affinely independent members x of $P(G, b)$ which satisfy $ax = \alpha$ and then appealing to (4.1.1) and (2.2.15).

(4.1.2) Theorem. For every $j \in E$, $x_j \geq 0$ gives a facet of $P(G, b)$.

Proof. For any $j \in E$ let P_j be the solution set of (4.0.2), (4.0.3) and

$$x_k \geq 0 \text{ for all } k \in E - \{j\}.$$

We define x^j by

$$\begin{aligned} x_k^j &= 0 \text{ for } k \neq j, \\ &= -1 \text{ for } k = j. \end{aligned}$$

Then for each $j \in E$, $x^j \in P^j - P(G, b)$. Therefore by (2.3.30), $x_j \geq 0$ gives a facet of $P(G, b)$. \square

The techniques used in this proof, showing that an inequality gives a facet by showing that if it is omitted we obtain a larger polyhedron, could possibly be used in proving the other facet characterizations of this chapter (theorems (4.2.1) and (4.3.49)). However we find it easier to show that $ax \leq \alpha$ gives a facet of $P(G, b)$ by exhibiting $|E|$ affinely independent members of $P(G, b)$. Theorem (4.1.2) is also easily proved by exhibiting $|E|$ affinely independent matchings of G , each such x satisfying $x_j = 0$. (Take the matchings $0, x^k : k \in E - \{j\}$ defined in the proof

of (4.1.1).)

We call $\{x \in P(G, b) : x_j = 0\}$ a nonnegativity facet of $P(G, b)$ for any $j \in E$.

4.2 Degree Constraint Facets.

In this section we characterize which of the inequalities $x(\delta(i)) \leq b_i$ for $i \in V$ are facets of $P(G, b)$. For each $i \in V$ we let $N(i)$ be the set of nodes of G adjacent to i . Let v and w be nodes of G such that $N(v) = \{w\}$, $N(w) = \{v\}$ and $b_w = b_v$. Then $\{v, w\}$ is the node set of a component H of G containing at least one edge. We call H a balanced edge.

(4.2.1) Theorem. For any $i \in V$, $x(\delta(i)) \leq b_i$ gives a facet of $P(G, b)$ if and only if

(4.2.2) i is a node of a balanced edge

or

(4.2.3) $b(N(i)) > b_i$ and if $b(N(i)) = b_i + 1$ then $\gamma(N(i)) = \emptyset$.

Proof. We first show the necessity of (4.2.2) and (4.2.3). Let i be a node violating (4.2.2) and (4.2.3). We will show that there are inequalities (4.0.1)-(4.0.3) which imply

$$(4.2.4) \quad x(\delta(i)) \leq b_i$$

and which are distinct from (4.2.4). Thus we can remove all copies of (4.2.4) from (4.0.1)-(4.0.3) without changing the solution set and the result follows from (2.3.30) and (4.1.1).

Suppose $b(N(i)) \leq b_i$. Summing the inequalities (4.0.2) for $v \in N(i)$ we obtain

$$x\left(\bigcup_{v \in N(i)} \delta(v)\right) \leq b(N(i))$$

and since $\delta(i) \subseteq \bigcup_{v \in N(i)} \delta(v)$, it follows that (4.0.1) implies

$$x(\delta(i)) \leq b(N(i)) \leq b_i.$$

Moreover if there were $v \in N(i)$ such that $x(\delta(v)) \leq b_v$ and (4.2.4) were the same inequality then $\delta(v) = \delta(i)$ and $b_v = b_i$ so since we do not allow isolated nodes we would have i and v being the nodes of a balanced edge, contradictory to i violating (4.2.2). Hence (4.2.4) is not a facet of $P(G, b)$.

Suppose $b(N(i)) = b_i + 1$ and there is some $j \in \gamma(N(i))$. Let $v \in \psi(j)$ and for each $u \in N(i)$ let $k(u)$ be an edge of G such that $\psi(k(u)) = \{i, u\}$. Let $J \equiv \{k(u) : u \in N(i)\}$ and let the graph B be defined to be $(N(i) \cup \{i\}, J \cup \{j\}, \psi|J \cup \{j\})$. We show that B is a blossom. Clearly B is connected, has no even polygons and exactly one odd polygon. Moreover if we define a matching \bar{x} of B by

$$\bar{x}_{k(u)} \equiv b_u \text{ for all } u \in N(i) - \{v\},$$

$$\bar{x}_{k(v)} \equiv b_{v-1}$$

$$\bar{x}_j \equiv 0$$

we see that \bar{x} is a matching of B satisfying (3.3.1)-(3.3.5) so that B is a blossom. Hence $G[N(i) \cup \{i\}]$ is shrinkable so $N(i) \cup \{i\} \in Q^0$. The inequality (4.0.3) for $N(i) \cup \{i\}$ is

$$(4.2.5) \quad x(\gamma(N(i) \cup \{i\})) \leq \frac{b(N(i)) + b(i) - 1}{2} = b_i$$

so since $\delta(i) \subseteq \gamma(N(i) \cup \{i\}) - \{j\}$, we see that (4.2.5) and (4.0.1) imply (4.2.4). Moreover (4.2.5) is different from (4.2.4). Hence (4.2.4) is not a facet of $P(G, b)$.

Now we prove the sufficiency of (4.2.2) or (4.2.3). Suppose that i is a node of a balanced edge H . For each $h \in \delta(i)$ we define a matching x^h by

$$x_k^h \equiv \begin{cases} b_i & \text{if } k = h, \\ 0 & \text{if } k \in E - \{h\}. \end{cases}$$

Let $j \in \delta(i)$. For each $h \in E - \delta(i)$ we define a matching x^h by

$$x_k^h \equiv \begin{cases} 1 & \text{if } k = h, \\ b_i & \text{if } k = j, \\ 0 & \text{if } k \in E - \{h, j\}. \end{cases}$$

Clearly the set $\{x^h : h \in E\}$ is linearly independent and $x^h(\delta(i)) = b_i$ for all $h \in E$. Since $\{x \in P(G, b) : x(\delta(i)) = b_i\}$ is a proper face of $P(G, b)$ it follows from (2.2.15) that (4.2.4) gives a facet of $P(G, b)$.

Now suppose (4.2.3) is satisfied for $i \in V$. Let K be a minimal subset of $N(i)$ for which $b(K) > b_i$. For each $v \in N(i)$ let $j(v)$ be an edge joining i and v , let $E_K \equiv \{j(v) : v \in K\}$. For every $j \in E_K$, let $\{v(j)\} \equiv \psi(j) \cap K$. Let $\bar{b} = (\bar{b}_j : j \in E_K)$ be defined by $\bar{b}_j \equiv b_{v(j)}$ for all $j \in E_K$. For each $k \in E_K$ we define $d^k = (d_j^k : j \in E_K)$ by

$$d_j^k \equiv \begin{cases} 0 & \text{if } j \in E_K - \{k\} \\ b(K) - b_i & \text{if } j = k. \end{cases}$$

Then $0 < d_k^k \leq \bar{b}$ for all $k \in E_K$ by our choice of K .

For each $k \in E_K$, let $\bar{x}^k \equiv \bar{b} - d_k^k$. Since $\{d_k^k : k \in E_K\}$ is linearly independent, (2.2.4) implies $\{\bar{x}^k : k \in E_K\}$ is affinely independent. Each vector \bar{x}^k can be extended to a matching x^k of G by letting $x_j^k \equiv 0$ for all $j \in E - E_K$.

Then

$$(4.2.6) \quad \{x^k : k \in E_K\} \text{ is affinely independent.}$$

Moreover,

$$(4.2.7) \quad x^k(\delta(v)) \leq b_v \text{ for all } v \in V - \{i\},$$

$$(4.2.8) \quad x^k(\delta(i)) = b_i$$

so $x^k \in P(G, b)$ for each $k \in E_K$.

For each $j \in \delta(i) - E_K$ we define a matching x^j as follows. Let $\{v\} \equiv \psi(j) - \{i\}$. If $v \in K$ then let $k \in E_K$ be chosen such that $x_j^k(v) > 0$ and let x^j be defined by

$$x_\ell^k \quad \text{if } \ell \in E - \{j, j(v)\}$$

$$x_\ell^j \equiv 0 \quad \text{if } \ell = j(v)$$

$$x_{j(v)}^k \quad \text{if } \ell = j.$$

If $v \notin K$ let k be any member of E_K and let $h \in \delta(i)$ be such that $x_h^k > 0$. Let x^j be defined by

$$\begin{aligned} x_\ell^k &\equiv \begin{cases} k & \text{if } \ell \in E - \{j, h\}, \\ 0 & \text{if } \ell = h, \\ 1 & \text{if } \ell = j. \end{cases} \\ x_\ell^j &\equiv \begin{cases} k & \text{if } \ell \in E - \{j, h\}, \\ 0 & \text{if } \ell = h, \\ 1 & \text{if } \ell = j. \end{cases} \end{aligned}$$

In either case, x^j is easily seen to satisfy (4.2.7) and (4.2.8) for every $j \in \delta(i) - K$. Since for each $j \in \delta(i) - E_K$, x^j is the unique matching x so far defined for which $x_j \neq 0$, (4.2.6) implies

$$(4.2.9) \quad \{x^j : j \in \delta(i)\} \text{ is affinely independent.}$$

Finally, for each $j \in E - \delta(i)$ we define a matching x^j as follows.

$$x_j^j \equiv 1$$

$$x_h^j \equiv 0 \text{ for } h \in E - (\delta(i) \cup \{j\})$$

x_h^j is defined for $h \in \delta(i)$ to be sufficiently large that (4.2.7), (4.2.8) are satisfied. This is possible for if $b(N(i)) = b_i + 1$ then by (4.2.3) at most one end of j is in $N(i)$. Therefore defining $x_j^j = 1$ restricts x_k^j to taking on a value one less than $b_{v(k)}$ for at most one edge $k \in \delta(i)$. Hence x^j can be defined as asserted. If $b(N(i)) \geq b_i + 2$ then it is easily seen that after defining $x_j^j = 1$ we can still assign values x_k^j for $k \in \delta(i)$ as required.

For any $j \in E - \delta(i)$, x^j is the only matching x defined for which $x_j \neq 0$. This together with (4.2.9) implies that $\{x^j : j \in E\}$ is affinely independent. Thus we have

defined $|E|$ affinely independent members of $P(G, b)$ each of which satisfies (4.2.8). Moreover $F \equiv \{x \in P(G, b) : x(\delta(i)) = b_i\}$ is a proper face of $P(G, b)$ since $0 \in P(G, b) - F$. Therefore by (2.2.15) it follows that (4.2.4) gives a facet of $P(G, b)$ completing the proof. \square

We call $\{x \in P(G, b) : x(\delta(i)) = b_i\}$ a nonnegativity facet for each $i \in V$ satisfying (4.2.2) or (4.2.3).

In the case of 1-matchings, (4.2.1) specializes to the following

(4.2.10) Theorem. $x(\delta(i)) \leq b_i$ gives a facet of $P(G, 1)$ if and only if

(4.2.11) i is a node of a balanced edge

or

(4.2.12) $|N(i)| > 1$ and if $|N(i)| = 2$ then $\gamma(N(i)) = \emptyset$.

4.3. Blossom Facets.

In this section we give a first characterization of the inequalities $x(\gamma(S)) \leq q_S$ for $S \in Q^0$ which are facets of $P(G, b)$. In fact for each $S \in Q^0$ we give the dimension of

$$F_S \equiv \{x \in P(G, b) : x(\gamma(S)) = q_S\}.$$

These results are obtained by studying shrinkable graphs (as defined in Section 3.3).

In Section 4.4 we give two characterizations of shrinkable graphs and hence two more characterizations of the facets of

this sort.

Recall that a np matching (near perfect) matching of G deficient at $v \in V$ is a matching x of G which satisfies

$$x(\delta(v)) = b_v - 1.$$

$$x(\delta(i)) = b_i \text{ for all } i \in V - \{v\}.$$

The following lemma is useful when proving the independence of matchings.

(4.3.1) Lemma. Let X be a set of np matchings of G and let $x^0 \in X$. If there exist $J(x^0) \subseteq E$ and $d(x^0) \in \mathbb{R}$ such that $x^0(J(x^0)) < d(x^0)$ but $x(J(x^0)) = d(x^0)$ for all $x \in X - \{x^0\}$ then x^0 is not a linear combination of $X - \{x^0\}$.

Proof. Suppose that there are $\alpha_x \in \mathbb{R}$ for $x \in X' \equiv X - \{x^0\}$ such that

$$(4.3.2) \quad x^0 = \sum \alpha_x x : x \in X'.$$

By (3.3.24), $x(E) = 1/2(b(V) - 1)$ for all $x \in X$. Therefore by (4.3.2)

$$x^0(E) = \sum \alpha_x x(E) : x \in X'$$

and hence

$$(4.3.3) \quad \sum \alpha_x : x \in X' = 1.$$

Therefore $\sum \alpha_x x(J(x^0)) : x \in X' = \sum \alpha_x d(x^0) : x \in X' = d(x^0)$ by (4.3.3). Hence (4.3.2) implies that $x^0(J(x^0)) = d(x^0)$, a contradiction which proves the lemma. \square

(4.3.4) We call $v \in V$ a strong cut node of $G = (V, E, \psi)$ relative to b if v is a cutnode of G (see (1.3.9)) and $b_v = 1$. A weak block of G relative to b is a maximal connected subgraph H of G such that $b_v > 1$ for any cutnode v of H . Thus a weak block consists of one or more blocks of G joined by cutnodes v for which $b_v > 1$. Notice that

(4.3.5) the edge sets of the weak blocks of G partition the edges of G .

(4.3.6) We let $\beta(G)$ denote the number of weak blocks of G .

In the case of 1-matchings, strong cutnodes and weak blocks are simply cutnodes and blocks respectively.

(4.3.7) Proposition. G is shrinkable if and only if G is connected and every weak block of G is shrinkable.

Proof. First suppose that G is connected and each weak block of G is shrinkable. We prove that G is shrinkable by induction on $\beta(G)$. If $\beta(G) = 1$ then the result is trivial. Suppose $\beta(G) > 1$ and assume the result is true for graphs having fewer than $\beta(G)$ weak blocks. Let D be a weak block of G , let R_D be a shrinking family for D . Each weak block of $G' \equiv G \times V(D)$ is isomorphic to a weak block of G and so is shrinkable. Moreover G' is connected. Since G is connected, $\beta(G') = \beta(G) - 1$

so by our induction hypothesis G' is shrinkable; let R' be a shrinking family of G' . For each $S \in R'$ we define a set $\zeta(S) \subseteq V$ as follows.

$$\begin{aligned}\zeta(S) &\equiv S \text{ if } V(D) \notin S, \\ &S - \{V(D)\} \cup V(D) \text{ if } V(D) \in S\end{aligned}$$

Let $R \equiv \{\zeta(S): S \in R'\} \cup R_D$. Then R is easily seen to be a shrinking family of G . The sufficiency now follows by induction.

Conversely, suppose that G is shrinkable. Let R be a shrinking family of G . Trivially G is connected. We prove that every weak block of G is shrinkable by induction on $|R|$. If $|R| = 0$, then G consists of a single node v and the result is trivial. Suppose that $|R| \geq 1$ and that the result is true for graphs having shrinking families of fewer than $|R|$ sets. Let S be a minimal member of R . By (3.3.16) $G[S]$ is spanned by a blossom B . By (3.3.9) only terminal nodes of B can be strong cutnodes so B is a subgraph of some weak block D of G . Let $G' \equiv G \times S$. For any $T \in R - \{S\}$ define $\zeta(T) \equiv T$ if $S \cap T = \emptyset$, define $\zeta(T) \equiv T - S \cup \{S\}$ if $S \subset T$ and let $R' = \{\zeta(T): T \in R - \{S\}\}$. R' is a shrinking family of G' and $|R'| = |R| - 1$ so by our induction hypothesis every weak block of G' is shrinkable. Hence every weak block of G different from D is shrinkable. Moreover every weak block of $D \times S$ is shrinkable so as we have already seen, $D \times S$ is shrinkable. Let R'_D be a shrinking family of $D \times S$ and for any $T \in R'_D$ let $\theta(T) \equiv T$ if $S \notin T$, let $\theta(T) \equiv T - \{S\} \cup S$ if $S \in T$.

Then $\{S\} \cup \{\emptyset(T) : T \in R_D'\}$ is a shrinking family of D and the proof now follows by induction. \square

(4.3.8) Proposition. If Z is the set of weak blocks of a connected graph $G = (V, E, \psi)$ then

$$(4.3.9) \quad b(V) - 1 = \sum(b(V(D)) - 1 : D \in Z).$$

Proof. We prove by induction on $|Z|$. If $|Z| = 1$ the result is trivial. Suppose $|Z| > 1$ and (4.3.9) holds for all graphs having fewer than $|Z|$ weak blocks. If every weak block of G contained two or more strong cutnodes then it is easily seen that G would contain a polygon having edges in more than one block, contrary to (1.3.10). Let B be a weak block of G containing exactly one strong cutnode v . Let $G' = G[V - (V(B) - \{v\})]$. Then G' is connected and $Z - \{B\}$ is the set of weak blocks of G' . Therefore by induction

$$b(V(G')) - 1 = \sum(b(V(D)) - 1 : D \in Z - \{B\}).$$

Since $b(V) = b(V(G')) + b(V(B) - \{v\}) = b(V(G')) + b(V(B)) - 1$, (4.3.9) holds and the result follows by induction. \square

(4.3.10) Proposition. Let $G = (V, E, \psi)$ be a shrinkable graph and suppose $x \in P(G, b)$ satisfies $x(E) = 1/2(b(V) - 1)$. Then for any weak block D of G , $x(E(D)) = 1/2(b(V(D)) - 1)$. (Note that x need not be integer valued.)

Proof. Let Z be the set of weak blocks of G . By (4.3.7) each $D \in Z$ is shrinkable so since $x \in P(G, b)$,

x satisfies

$$(4.3.11) \quad x(E(D)) \leq 1/2(b(V(D)) - 1) \quad \text{for all } D \in Z.$$

Therefore, summing for all $D \in Z$ we obtain

$$(4.3.12) \quad x(\bigcup_{D \in Z} E(D)) \leq 1/2 \sum_{D \in Z} (b(V(D)) - 1).$$

By (4.3.5) $E = \bigcup_{D \in Z} E(D)$ so using (4.3.8) we obtain

$$(4.3.13) \quad x(E) \leq 1/2(b(V) - 1).$$

But by hypothesis equality holds in (4.3.13) so equality must hold in (4.3.12) and (4.3.11) which proves the result. \square

(4.3.14) Corollary. If x is a np matching of a shrinkable graph G then for any weak block D of G , $x|_{E(D)}$ is a np matching of D .

Proof. The result follows from combining (4.3.10) and (3.3.24). \square

Now we prove a main result used in characterizing the facets of $P(G, b)$ given by constraints (4.0.3).

(4.3.15) Theorem. If $G = (V, E, \psi)$ is shrinkable then G has $|E| - (\beta(G) - 1)$ linearly independent np matchings.

Proof. Let R be a shrinking family of G ; we prove by induction on $|R|$. If $|R| = 0$ then G is degenerate, $|E| = 0$, $\beta(G) = 1$ and the result is trivial. Suppose $|R| \geq 1$ and the theorem holds for graphs having a shrinking family consisting of fewer than $|R|$ sets.

Let B be a blossom spanning $G \times R[V]$ which exists by (3.3.16). We partition $V(B)$ into $V_1 \cup V_2$ where $V_1 \equiv V(B) \cap V$ and $V_2 \equiv V(B) \cap R$. That is, V_1 is the set of real nodes of B and V_2 is the set of pseudonodes of B .

Let $C \equiv E(G \times R[V]) - E(B)$ and let G' be the graph obtained from G by deleting all the edges in C . Then R is a shrinking family of G' so by (3.3.21) for each $v \in V_1$ there is a np matching \bar{x}^v of G' deficient at v and which satisfies

$$(4.3.16) \quad \bar{x}^v(\gamma(S)) = 1/2(b(S) - 1) \text{ for all } S \in R.$$

For each $v \in V_1$ we define a np matching x^v of G deficient at v by

$$(4.3.17) \quad x_j^v \equiv \begin{cases} \bar{x}_j^v & \text{for } j \in E' \\ 0 & \text{for } j \in C. \end{cases}$$

Let $X_1 = \{x^v : v \in V_1\}$. Since by (4.3.16) each $x \in X$ is a np matching of $G[S]$ for each $S \in V_2$, it follows from (4.3.14) that

(4.3.18) $x|_{E(D)}$ is a np matching of D for every weak block D of $G[S]$ for every $S \in V_2$, for every $x \in X_1$.

For each $S \in V_2$ there are by induction $n(S) \equiv |\gamma(S)| - (\beta(G[S]) - 1)$ linearly independent np matchings $\{\bar{x}^{S,1}, \bar{x}^{S,2}, \dots, \bar{x}^{S,n(S)}\}$ of $G[S]$ since $R[S] \cup \{S\}$

is a shrinking family of $G[S]$ and $|R[S] \cup \{S\}| \leq |R - \{V\}| < |R|$. By (4.3.14),

(4.3.19) $\tilde{x}^{S,i}|_{E(D)}$ is a np matching of D for every weak block D of $G[S]$ for every $i \in \{1, 2, \dots, n(S)\}$.

We extend each to a np matching of G as follows. Let \tilde{x}^S be the np matching of G deficient at S which exists by (3.3.12). For each $T \in V_2 - \{S\}$ let $j(T)$ be the edge of $\tilde{x}(T) \cap E(B)$ such that $\tilde{x}_{j(T)}^S = 1$, let $\{v(T)\} \in \psi(j(T)) \cap T$ and let $\tilde{x}^{T,S}$ be a np matching of $G[T]$ deficient at $v(T)$. By (4.3.14),

(4.3.20) $\tilde{x}^{T,S}|_{E(D)}$ is a np matching of D for every weak block D of $G[T]$.

Now we define $x^{S,i}$ for all $i \in \{1, 2, \dots, n(S)\}$ by

$$\tilde{x}_j^{S,i} \text{ for } j \in \gamma(S),$$

(4.3.21) $x_j^{S,i} = \tilde{x}_j^S \text{ for } j \in E(B),$
 $0 \text{ for } j \in C,$
 $\tilde{x}_j^{T,S} \text{ for } j \in \gamma(T), \text{ for } T \in V_2 - \{S\}.$

Let $X_2 = \{x^{S,i} : i \in \{1, 2, \dots, n(S)\}, S \in V_2\}$. By (4.3.19) and (4.3.20),

(4.3.22) $x|_{E(D)}$ is a np matching of D for every weak block D of $G[T]$ for every $T \in V_2$ for every $x \in X_2$.

Now we show

(4.3.23) $X_1 \cup X_2$ is linearly independent.

Suppose that $\alpha_v \in \mathbb{R}$: $v \in V_1$ and $\alpha_{S,i} \in \mathbb{R}$:

$i \in \{1, 2, \dots, n(S)\}$, $S \in V_2$ are such that

$$(4.3.24) \quad \sum (\alpha_v x^v : v \in V_1)$$

$$+ \sum (\alpha_{S,i} x^{S,i} : i \in \{1, 2, \dots, n(S)\}, S \in V_2) = 0.$$

If we let $\tilde{x}^v \equiv x^v|_{E(B)}$ for each $v \in V_1$ we have

$$\sum (\alpha_v \tilde{x}^v : v \in V_1) + \sum (\tilde{\alpha}_S \tilde{x}^S : S \in V_2) = 0$$

where

$$\tilde{\alpha}_S \equiv \sum (\alpha_{S,i} : i \in \{1, 2, \dots, n(S)\}) \text{ for } S \in V_2.$$

For each $v \in V(B)$, \tilde{x}^v is a np matching of B deficient at v so if we let $J(\tilde{x}^v) \equiv \delta(v) \cap E(B)$ and $d(\tilde{x}^v) \equiv b_v$ for all $v \in V(B)$ then by (4.3.1), $\{\tilde{x}^v : v \in V(B)\}$ is linearly independent so

$$(4.3.25) \quad \alpha_v = 0 \text{ for all } v \in V_1,$$

$$(4.3.26) \quad \tilde{\alpha}_S = 0 \text{ for all } S \in V_2.$$

Now let $S \in V_2$, let $V'_2 \equiv V_2 - \{S\}$. By (4.3.21), (4.3.24) and (4.3.25) we have

$$\sum (\alpha_{S,i} \tilde{x}^{S,i} : i \in \{1, 2, \dots, n(S)\})$$

$$+ \sum (\tilde{\alpha}_T \tilde{x}^{S,T} : T \in V'_2) = 0$$

so by (4.3.26),

$$\sum_{S,i} \bar{x}^{S,i} : i \in \{1, 2, \dots, n(S)\} = 0.$$

But the matchings $\{\bar{x}^{S,i} : i \in \{1, 2, \dots, n(S)\}\}$ are by hypothesis linearly independent so

$$(4.3.27) \quad \alpha_{S,i} = 0 \quad \text{for all } i \in \{1, 2, \dots, n(S)\}.$$

This together with (4.3.25) proves (4.3.23).

Let $k \in C$. We define a np matching \hat{x}^k as follows.

Let v and w be the nodes of B met by k , let \tilde{x}^v be the np matching of B deficient at v . There must be some edge $\ell \in E(B) \cap \delta(w)$ such that $\tilde{x}_\ell^v = 1$, we define a np matching \hat{x}^k of $G \times R[V]$ by

$$(4.3.28) \quad \hat{x}_j^k \equiv \begin{cases} \tilde{x}_j^v & \text{for } j \in E(B) - \{\ell\}, \\ 0 & \text{for } j \in (C - \{k\}) \cup \{\ell\}, \\ 1 & \text{if } j = k. \end{cases}$$

Let $T \in V_2$. If $\hat{x}^k(\delta(T)) = 0$ we let \bar{x}^T be any np matching of $G[T]$. If there is $\ell \in \delta(T)$ such that $\hat{x}_\ell^k = 1$ then let $\{v\} \equiv \psi(\ell) \cap T$ and let \bar{x}^T be a np matching of $G[T]$ deficient at v . Now define \hat{x}^k by

$$(4.3.29) \quad \hat{x}_j^k \equiv \begin{cases} \hat{x}_j^k & \text{for } j \in E(G \times R[V]), \\ \bar{x}_j^T & \text{for } j \in \gamma(T) \text{ for } T \in V_2. \end{cases}$$

Let $X_3 \equiv \{x^k : k \in C\}$. Every $x \in X_3$ is a np matching of G and for any $S \in V_2$, $x|_{\gamma(S)}$ is a np matching of $G[S]$. Therefore by (4.3.14),

(4.3.30) $x|E(D)$ is a np matching of every weak block D of $G[S]$ for every $S \in V_2$ for every $x \in X_3$.

Moreover, by (4.3.17), (4.3.21), (4.3.28) and (4.3.29) for each $k \in C$, x^k is the unique member of $X_1 \cup X_2 \cup X_3$ such that $x_k \neq 0$, so by (4.3.23),

(4.3.31) $X_1 \cup X_2 \cup X_3$ is linearly independent.

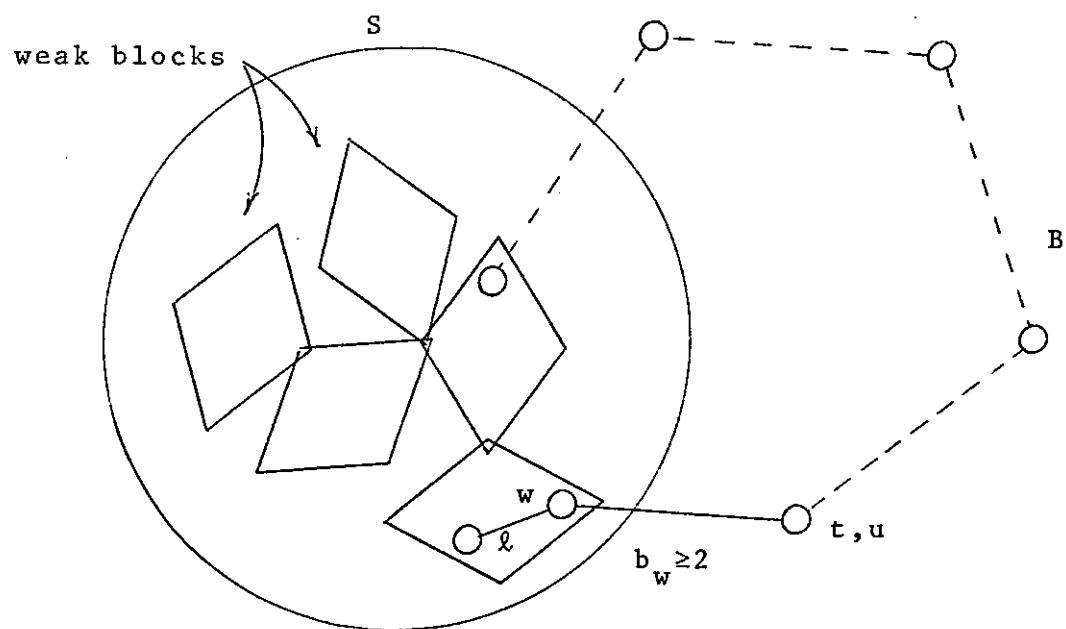
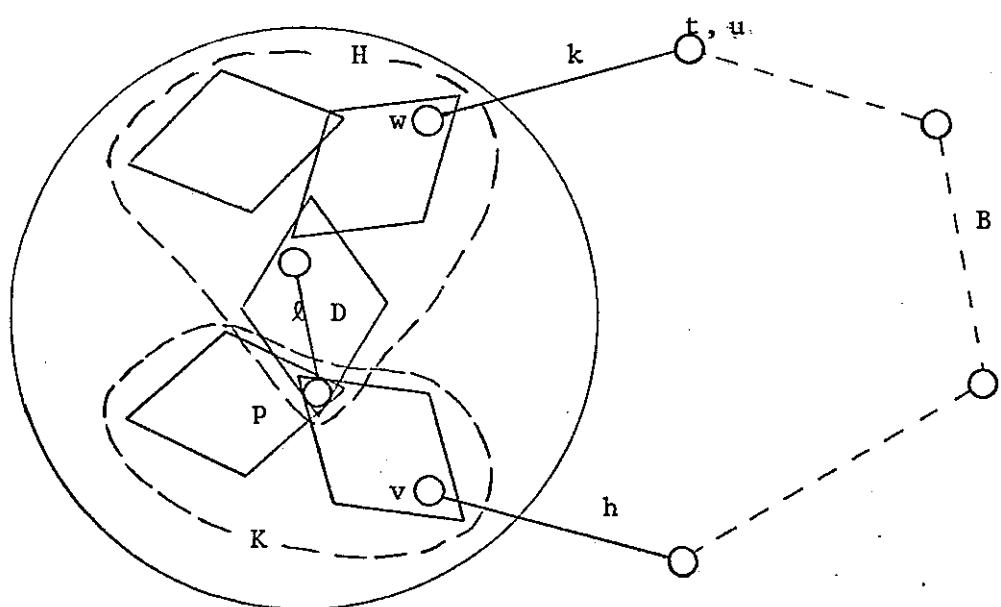
Now let D be a weak block of $G[S]$ such that D is not a weak block of G for some $S \in V_2$. First observe that since $b_S = 1$ by (3.3.9) S must be a terminal node of B and consequently $|\delta_B(S)| \leq 2$. As before we let $G' \equiv (V, E - C, \psi|E - C)$. We distinguish two main cases.

Case 1. D is not a weak block of G' .

Case 1a. An edge h of $\delta_B(S)$ is incident with a node $w \in V(D)$ for which $b_w \geq 2$. (See Figure 4.1).

Since $b_w \geq 2$, w is not a strong cutnode of $G[S]$ and so every edge of $G[S]$ incident with w is an edge of D . Let x^S be a np matching of S deficient at w . Since $b_w \geq 2$ there is some $l \in E(D) \cap \delta(w)$ such that $x_l^S = 1$. Let $\{t\} \equiv \psi(h) - \{S\}$ and let u be the node of $V(B) - \{S\}$ met by h . If $u \in V_1$, then $u = t$, if $u \in V_2$ then $t \in u$. Let \tilde{x} be the np matching of B deficient at u .

(4.3.32) For each $T \in V_2 - \{u\}$ let $j(T)$ be the unique edge j of $\delta_B(T)$ such that $\tilde{x}_j = 1$ and let x^T be a np matching of $G[T]$ deficient at $v(T)$, where $\{v(T)\} \equiv \psi(j(T)) \cap T$.

FIGURE 4.1FIGURE 4.2

If $u \in V_2$ then

(4.3.33) let x^u be a np matching of $G[u]$
deficient at t .

We now define a np matching x^D of G by

$$\begin{aligned} x_j^S & \text{ for } j \in \gamma(S) - \{\ell\}, \\ x_\ell^S & = 1 \text{ if } j = \ell, \\ 0 & \text{ for } j \in C \\ x_j^D & \equiv \tilde{x}_j \text{ for } j \in E(B) - \{h\}, \\ \tilde{x}_h & + 1 \text{ if } j = h \\ x_j^T & \text{ for } j \in \gamma(T) \text{ for } T \in V_2 - \{S\}. \end{aligned}$$

It can be seen that

(4.3.34) $x^D|_{E(A)}$ is a np matching of each weak
block A of $G[T]$ for $T \in V_2$ unless $A = D$ and

$$(4.3.35) x^D(E(D)) = \frac{b(V(D)) - 3}{2}.$$

Case 1b. $b_i = 1$ for every node $i \in V(D)$ met by an
edge of B . (see Figure 4.2) Then by our case 1 hypothesis
there must be distinct $v, w \in S$ incident with edges $h,$
 $k \in \delta_B(S)$ respectively and every path in $G[S]$ from v to
 w must contain an edge of D . Since D is a weak block of
 $G[S]$ there is a unique node $p \in V(D)$ which is the first
node of D in any such path. If $p \neq v$ then p is a
strong cutnode of $G[S]$ and hence is not a cutnode of D .

If $p = v$ then $b_p = 1$ by our Case 1b hypothesis so p cannot be a cutnode of D . Thus there is a component \bar{H} of $G[S - \{p\}]$ such that $V(D) - \{p\} \subseteq V(\bar{H})$. Let $H \equiv G[V(\bar{H}) \cup \{p\}]$, let $K \equiv G[S - V(\bar{H})]$. (K may consist of just the single node p .) Then $V(H) \cup V(K) = S$ and $V(H) \cap V(K) = \{p\}$. Clearly the weak blocks of $G[S]$ are the weak blocks of H and K so by (4.3.7), H and K are shrinkable. Moreover, $v \in V(K)$, $w \in V(H)$ and $p \neq w$. Let x^H be a np matching of H deficient at w . Since $p \neq w$, there is some $\ell \in E(D) \cap \delta(p)$ such that $x_\ell^H = 1$. Let x^K be a np matching of K deficient at v . Let $\{t\} \equiv \psi(k) - S$, let u be the node of $V(B) - \{S\}$ met by k . Let \tilde{x} be the np matching of B deficient at u . Since $|\delta_B(S)| = 2$ and since S is a terminal node of B , S must belong to the odd polygon of B . Therefore h is the first edge in a path of length two from a node in the polygon to u . Therefore by (3.3.12) and (3.3.5) $\tilde{x}_h = 1$ and $\tilde{x}_k = 0$. For each $T \in V_2 - \{u\}$ define x^T as in (4.3.32) and if $u \in V_2$ then define x^u as in (4.3.33). Now define x^D by

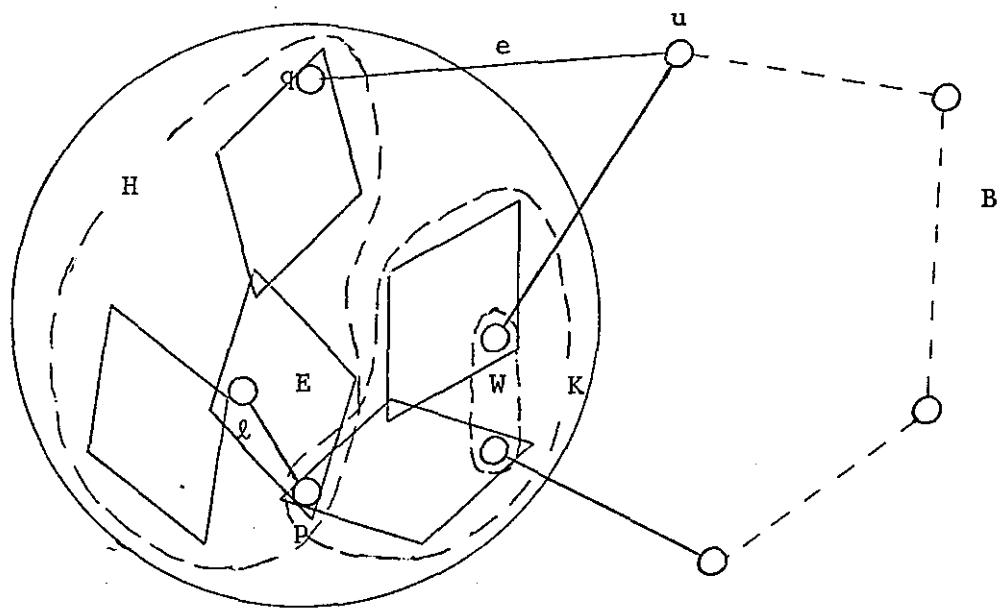
$$\begin{aligned}
 x_j^H & \text{ for } j \in E(H) - \{\ell\} \\
 0 & \text{ for } j \in \{\ell\} \cup C, \\
 x_j^D & \equiv \begin{cases} x_j^K & \text{for } j \in E(K) \\ \tilde{x}_j & \text{for } j \in E(B) - \{k\} \\ 1 & \text{for } j = k \end{cases} \\
 x_j^T & \text{ for } j \in \gamma(T) \text{ for } T \in V_2 - \{S\}.
 \end{aligned}$$

It can now be seen that x^D is a np matching of G satisfying (4.3.34) and (4.3.35).

Case 2. D is a weak block of G' . (See Figure 4.3).

Let W be the set of nodes of S incident with edges of B . There must be a node $p \in V(D)$ which is the first node of D in any path in $G[S]$ from a node in W to a node in D , otherwise D would not be a weak block of G' . p is a strong cutnode of $G[S]$ unless $W = \{p\}$. Since D is not a weak block of G , there is some edge $e \in C \cap \delta(S)$ such that where $\{q\} \equiv \psi(e) \cap S$, there is a path in $G[S]$ from q to a node of B which does not contain p . Let \bar{H} be the component of $G[S - \{p\}]$ which contains q , let $H \equiv G[V(\bar{H}) \cup \{p\}]$, let $K \equiv G[S - V(\bar{H})]$. (If $W = \{p\}$ then K may simply consist of p .) Let u be the node of $V(B) - \{S\}$ met by e and let \tilde{x} be the np matching of B deficient at u . Let x^H be a np matching of H deficient at q . There must be $\ell \in E(D) \cap \delta(p)$ such that $x_\ell^H = 1$. Let x^K be a np matching of K deficient at the node $w \in W$ met by an edge $h \in E(B)$ for which $\tilde{x}_h = 1$. For each $T \in V_2 - \{u\}$ define x^T as in (4.3.32). If $u \in V_2$ then let $\{t\} \equiv \psi(e) - S$ and define x^u as in (4.3.33). Now define x^D as follows.

$$\begin{aligned}
 x_j^D &= \begin{cases} x_j^H & \text{for } j \in E(H) - \{\ell\}, \\ 0 & \text{for } j \in \{\ell\} \cup C - \{e\}, \end{cases} \\
 x_j^D &\equiv \begin{cases} x_j^K & \text{for } j \in E(K), \\ \tilde{x}_j & \text{for } j \in E(B), \\ 1 & \text{for } j = e, \\ x_j^T & \text{for } j \in \gamma(T) \text{ for } T \in V_2 - \{S\}. \end{cases}
 \end{aligned}$$

FIGURE 4.3

It can be seen that x^D is a np matching of G satisfying (4.3.34) and (4.3.35).

Let Z be the set of all weak blocks of $G[S]$ for all $S \in V_2$ and let $Z' = \{D \in Z : D \text{ is not a weak block of } G\}$. Let $X_4 = \{x^D : D \in Z'\}$. Then x^D satisfies (4.3.34) and (4.3.35) for every $x^D \in X_4$. Therefore by lemma (4.3.1), (4.3.18), (4.3.22), (4.3.30) we have that

$X_1 \cup X_2 \cup X_3 \cup X_4$ is linearly independent.

Now we evaluate $|X_1 \cup X_2 \cup X_3 \cup X_4|$.

$$\begin{aligned}
 (4.3.36) \quad & |X_1 \cup X_2 \cup X_3 \cup X_4| \\
 &= |V_1| + \sum(|\gamma(S)| - \beta(G[S]) + 1 : S \in V_2) + \\
 & \quad |C| + |Z'| \\
 &= |V_1| + |V_2| + \sum(|\gamma(S)| : S \in V_2) + \\
 & \quad |C| - (|Z| - |Z'|)
 \end{aligned}$$

Since $V_1 \cup V_2 = V(B)$ and $|V(B)| = |E(B)|$ (by (3.3.7)) and since $\bigcup_{S \in V_2} \gamma(S) \cup E(B) \cup C$ is a partition of E ,

$$(4.3.37) \quad |E| = |V_1| + |V_2| + \sum(|\gamma(S)| : S \in V_2) + |C|.$$

Since the weak blocks of G are the members of $Z - Z'$ together with the weak block containing $E(B)$, we have

$$(4.3.38) \quad |Z| - |Z'| = \beta(G) - 1.$$

Thus (4.3.36)-(4.3.38) combine to give

$$|X_1 \cup X_2 \cup X_3 \cup X_4| = |E| - (\beta(G) - 1)$$

and the theorem now follows by induction. \square

We now are able to give the dimension of all faces of $P(G, b)$ obtained by making one of the inequalities (4.0.3) an equation.

(4.3.39) Theorem. Let $F \equiv \{x \in P(G, b) : x(\gamma(S)) = q_S\}$ for some $S \in Q^0$. Then $\dim(F) = |E| - \beta(G[S])$.

Proof. First we show

$$(4.3.40) \quad \dim(F) \leq |E| - \beta(G[S]).$$

Let Z be the set of weak blocks of $G[S]$. Since $S \in Q^0$, $G[S]$ is shrinkable so by (4.3.7) each $D \in Z$ is shrinkable. Let $W \equiv \{V(D) : D \in Z\}$. Then $W \subseteq Q^0$. By (4.3.10), any $x \in P(G, b)$ that satisfies $x(\gamma(S)) = q_S$ will also satisfy

$$(4.3.41) \quad x(\gamma(T)) = q_T \text{ for all } T \in W.$$

The inequalities (4.0.1)-(4.0.3) can be represented by $Ax \leq d$ where $A \in \mathbb{R}^{(E \cup V \cup Q^0) \times E}$ and $d \in \mathbb{R}^{(E \cup V \cup Q^0)}$ are appropriately defined. By (4.3.41), if I is the equality set of F (see section 2.1) then $W \subseteq I$. By (4.3.5) the rows of A_W are linearly independent, so $\text{rank}(A_I) \geq |W|$ and hence $\dim(F) \leq |E| - |Z|$ which proves (4.3.30).

We now show

$$(4.3.42) \quad \dim(F) \geq |E| - \beta(G[S])$$

by displaying $|E| - \beta(G[S]) + 1$ linearly independent members x of $P(G, b)$ which satisfy

$$(4.3.43) \quad x(\gamma(S)) = q_S.$$

By (4.3.15) $G[S]$ has a set \hat{X}_1 of $|\gamma(S)| - \beta(G[S]) + 1$ linearly independent np matchings. We extend each $\hat{x} \in \hat{X}_1$ to a matching x of G by letting

$$(4.3.44) \quad x_j = \begin{cases} \hat{x}_j & \text{for } j \in \gamma(S) \\ 0 & \text{for } j \in E - \gamma(S). \end{cases}$$

Let X_1 be the set of matchings thereby obtained, each $x \in X_1$ satisfies (4.3.43).

Let $k \in \delta(S)$, let $\{v\} \equiv \psi(k) \cap S$. Let \tilde{x} be a np matching of $G[S]$ deficient at v and let x^k be defined by

$$(4.3.45) \quad x_j^k = \begin{cases} \tilde{x}_j & \text{for } j \in \gamma(S) \\ 1 & \text{for } j = k \\ 0 & \text{for } j \in E - (\gamma(S) \cup \{k\}). \end{cases}$$

For any $k \in \gamma(V - S)$ let \tilde{x} be any np matching of $G[S]$ and let x^k be defined as in (4.3.45). Let $X_2 = \{x^k : k \in E - \gamma(S)\}$. Each $x^k \in X_2$ satisfies (4.3.43) and since by (4.3.44) and (4.3.45) x^k is the unique member x of $X_1 \cup X_2$ for which $x_k \neq 0$,

$X_1 \cup X_2$ is linearly independent

Since $|X_1 \cup X_2| = |E| - \beta(G[S]) + 1$, (4.3.42) now follows.

Combining (4.3.40) and (4.3.42) proves the theorem. \square

Theorem (4.3.39) specializes to the following.

(4.3.46) Theorem. $x(\gamma(S)) \leq q_S$ for $S \in Q^0$ gives a facet of $P(G, b)$ if and only if $G[S]$ contains no strong cutnode.

We call such facets blossom facets of $P(G, b)$. Notice that since nonnegativity facets of $P(G, b)$ all contain the point $0 \in \mathbb{R}^E$ and since neither degree constraint facets nor blossom facets contain 0 , nonnegativity facets are different from blossom facets and degree constraint facets. For any $i \in V$ and any $S \in Q^0$, $\delta(i)$ is the edge set of a tree and $\gamma(S)$ contains the edge set of a polygon. Therefore the equations $x(\delta(i)) = b_2$ and $x(\gamma(S)) = q_S$ are distinct. Thus no degree constraint facet of $P(G, b)$ is a blossom facet of $P(G, b)$. Consequently

(4.3.47) the facets of $P(G, b)$ are partitioned into nonnegativity facets, degree constraint facets and blossom facets.

Finally, by making use of (4.3.46) and (4.2.1) we obtain the following.

(4.3.48) Theorem. For any $S \subseteq V$ such that $b(S)$ is odd, $x(\gamma(S)) \leq q_S$ gives a facet of $P(G, b)$ if and only if

(4.3.49) $|S| \geq 3$, $G[S]$ is shrinkable and contains no strong cutnode,
or

(4.3.50) there is $i \in S$ such that $\delta(i) = \gamma(S)$,
 $b_i = q_S$ and i satisfies (4.2.2) or (4.2.3).

Proof. Let $F \equiv \{x \in P(G, b) : x(\gamma(S)) = q_S\}$. If F is a nonnegativity facet of $P(G, b)$ then $q_S = 0$ and so $b(S) = 1$ and hence $|S| = 1$. Therefore since G has no

loops, $\gamma(S) = \phi$. Thus every member of $P(G, b)$ satisfies $x(\gamma(S)) = q_S$ so F is not a proper face of $P(G, b)$, a contradiction. Therefore F is not a nonnegativity facet of $P(G, b)$.

Consequently F is a facet of $P(G, b)$ if and only if F is a degree constraint facet of $P(G, b)$ or a blossom facet of $P(G, b)$. By (4.2.1) F is a degree constraint facet of $P(G, b)$ if and only if (4.3.50) holds, by (4.3.46) F is a blossom facet of $P(G, b)$ if and only if (4.3.49) holds. The theorem follows. \square

In the case of 1-matchings, (4.3.48) can be specialized as follows.

(4.3.51) Theorem. For any $S \subseteq V$ such that $|S| \geq 3$, $x(\gamma(S)) \leq 1/2(|S| - 1)$ gives a facet of $P(G, 1)$ if and only if

(4.3.52) $G[S]$ is shrinkable and nonseparable.

(4.3.53) $|S| = 3$ and there is $i \in S$ such that $\delta(i) = \gamma(S)$ and i satisfies (4.2.11) or (4.2.12).

4.4. b-critical Graphs

In this section we give two characterizations of shrinkable graphs and in doing so we give two more characterizations of the blossom facets of $P(G, b)$. We also show that the blossom algorithm can be applied to a graph $G = (V, E, \psi)$ for which $b(V)$ is odd so as to determine whether or not G is shrinkable.

We say that $G = (V, E, \psi)$ is b-critical if there is a np matching x^v of G deficient at v for each $v \in V$. This of course implies that $b(V)$ is odd. In the case of 1-matchings we have G is 1-critical if for any $v \in V$, $G[V - \{v\}]$ has a perfect 1-matching.

If G is b-critical, and hence has a np matching, then $D(G, V) = 1$. (See (3.7.9), (3.7.10) for the definition of $D(G, V)$.)

We saw in (3.10.33) (Tutte's Theorem) that G has a perfect b-matching if and only if for every $X \subseteq V$

$$(4.4.1) \quad b(X) \geq |C_1(X)| + b(\cup C_0(X))$$

where $C_0(X)$, $C_1(X)$ are as defined in (3.10.20), (3.10.21).

The inequality (4.4.1) is commonly called Tutte's condition.

If $b(V)$ is odd then clearly if we take $X \equiv \phi$, we will violate Tutte's condition. However our next theorem shows that if G is b-critical if and only if G is connected and $X \equiv \phi$ is the only subset of V which violates Tutte's condition. It also shows that G is b-critical if and only if G is shrinkable.

(4.4.2) Theorem. Let $G = (V, E, \psi)$ be a graph, let $b = (b_i : i \in V)$ be a vector of positive integers. The following conditions are equivalent.

(4.4.3) G is shrinkable;

(4.4.4) G is b-critical;

(4.4.5) G is connected, $b(V)$ is odd and every

nonempty $X \subset V$ satisfies (4.4.1) (Tutte's condition).

Proof. (4.4.3) implies (4.4.4). This is simply (3.3.21).

(4.4.4) implies (4.4.5). If G is b-critical then $b(V)$ is odd. Suppose that H and K are distinct components of G . Then each must be b-critical so $b(V(H))$ and $b(V(K))$ are odd. Let x be a np matching of G deficient at $v \in V(H)$. Then $x|E(K)$ is a perfect matching of K and so $b(V(K))$ is even, a contradiction. Therefore G is connected.

Let X be any nonempty subset of V and let $v \in X$. Let x be a np matching of G deficient at v . For any $i \in C_0(X)$ we have

$$(4.4.6) \quad x(\delta(i)) = b_i.$$

For any $S \in C_1(X)$ we have

$$(4.4.7) \quad x(\delta(S)) \geq 1$$

since $b(S)$ is odd and $x(\delta(i)) = b_i$ for all $i \in S$. Since $x(\delta(i)) \leq b_i$ for all $i \in X$ we have

$$(4.4.8) \quad x(\delta(X)) \leq b(X).$$

Since $\bigcup_{\{i\} \in C_0(X)} \delta(i) \cup \bigcup_{S \in C_1(X)} \delta(S)$ partitions a subset of $\delta(X)$, we have

$$(4.4.9) \quad \sum(x(\delta(i)) : \{i\} \in C_0(X)) + \sum(x(\gamma(S)) : S \in C_1(X)) \leq x(\delta(X)).$$

Combining (4.4.6)-(4.4.9) gives

$$\sum_{i \in \{i\} \subseteq C_0(X)} b_i + |C_1(X)| \leq b(X)$$

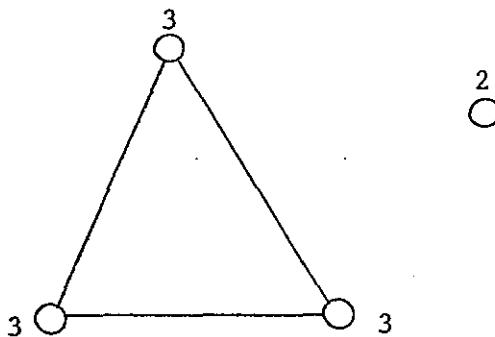
so X satisfies (4.4.1).

(4.4.5) implies (4.4.3). Suppose that $b(V)$ is odd, G is connected and G is not shrinkable. If we apply the matching algorithm to G and attempt to find a perfect matching, the algorithm must terminate with a Hungarian forest F with respect to a matching \bar{x} in a graph \bar{G} obtained from G by shrinking a set of disjoint shrinkable subsets of G . Since G is connected and nonshrinkable, F must have a nonempty set X of odd nodes, by (3.7.5) each of these is a node of G . Let W_0 be the set of even nodes of F which are nodes of G , let W_1 be the set of even pseudonodes of F . By (3.7.3), (3.7.4), $W_0 \subseteq C_0(X)$. By (3.7.3), (3.7.4) and (3.3.19), $W_1 \subseteq C_1(X)$. Since $b_S = 1$ for every $S \in W_1$ we have $b(W_1) \leq |C_1(X)|$. By (3.6.12),

$$\begin{aligned} b(X) &< b(W_0) + b(W_1) \\ &\leq b(\cup(C_0(X))) + |C_1(X)| \end{aligned}$$

so X violates (4.4.1). Since $X \neq \emptyset$, the result follows. \square

If we omit the connectivity condition from (4.4.5) then it no longer implies (4.4.3) or (4.4.4) for general b . If G is the graph represented in Figure 4.4 then $b(V(G)) = 11$ and every nonempty $X \subseteq V$ satisfies (4.4.1) but G does not have a np matching and so is not b -critical.

FIGURE 4.4

The number beside each node is the degree constraint of the node.

However if $b_i = 1$ for all $i \in V$ and if $|V|$ is odd and if every nonempty $X \subseteq V$ satisfies (4.4.1) then G is connected, for suppose G is not connected. Then G must have an odd number k of components H for which $|V(H)|$ is odd. If $k = 1$ then G must also have a component K for which $|V(K)|$ is even. Let $v \in V(K)$ and let $X \equiv \{v\}$. Then $b(X) = |X| = 1$ but $G[V - X]$ has at least two components with an odd number of nodes so (4.4.1) fails for X . If $k \geq 3$ then let $X \equiv \{v\}$ for any node v of G . Then $G[V - X]$ has at least two components with an odd number of nodes but $b(X) = 1$, again contradicting (4.4.1).

Thus when considering 1-matchings we obtain the following specialization of (4.4.2).

(4.4.10) Theorem. Let $G = (V, E, \psi)$ be a graph, let every node of G have a degree constraint of 1. Then the following are equivalent.

(4.4.11) G is shrinkable

(4.4.12) G is 1-critical

(4.4.13) $|V|$ is odd and for every nonempty $X \subseteq V$, the number of components H of $G[V - X]$ for which $|V(H)|$ is odd is no greater than $|X|$.

The blossom algorithm of chapter 3 provides an efficient method for determining whether or not a graph G satisfies the equivalent conditions (4.4.3)-(4.4.5). For if we apply the algorithm to a graph $G = (V, E, \psi)$ for which $b(V)$ is odd then, as noted in the proof of (4.4.2), it will either find a shrinking family of G or else will terminate with a Hungarian forest F and a node $i \in V$ which is not an even node of F or contained in an even pseudonode of F . By (3.7.38) in theorem (3.7.36) G can have no np matching deficient at i so G is not b-critical and violates (4.4.3)-(4.4.5).

Finally notice that if $b(V)$ is odd, then for any $X \subseteq V$, $b(X)$ and $|C_1(X)| + b(\cup(C_0(X)))$ must always have opposite parity so we can never have equality in (4.4.1). Thus if desired we could replace (4.4.1) with

$$(4.4.14) \quad b(X) > |C_1(X)| + b(\cup(C_0(X)))$$

in (4.4.5).

Now we can apply (4.4.2) to (4.3.46) and obtain the following two characterizations of the blossom facets of $P(G, b)$.

(4.4.15) Theorem. For any set S such that $b(S)$

is odd, $x(\gamma(S)) \leq q_S$ gives a blossom facet of $P(G, b)$
if and only if

(4.4.16) $|S| \geq 3$, $G[S]$ is b -critical and has no
cutnode v for which $b_v = 1$.

(4.4.17) Theorem. For any set S such that $b(S)$
is odd, $x(\gamma(S)) \leq q_S$ gives a blossom facet of $P(G, b)$ if
and only if

(4.4.18) $b(S)$ is odd, $|S| \geq 3$, $G[S]$ is connected
and has no cutnode v for which $b_v = 1$ and for every
nonempty $X \subseteq S$,

$$b(X) \geq b(u(C_0(X \cup (V - S)))) + |C_1(X \cup (V - S))| .$$

We are now able to combine theorems (4.1.2), (4.2.1)
and (4.3.46) to obtain the following.

(4.4.19) Theorem. The following is the minimal
subset of the inequalities (3.4.2)-(3.4.4) which is sufficient
to define $P(G, b)$.

$x_j \geq 0$ for all $j \in E$
 $x(\delta(i)) \leq b_i$ for all $i \in V$ satisfying (4.2.2)
or (4.2.3)

$x(\gamma(S)) \leq q_S$ for all $S \in Q^0$ which satisfy the
equivalent conditions (4.3.49), (4.4.16) or (4.4.18).

As we discussed in section 3.10, we can now use linear
programming duality to obtain a "best-possible" min-max
theorem. Let $W \subseteq V$ contain exactly one node of each balanced

edge of G . Let $V^* \equiv \{i \in V : i \text{ satisfies (4.2.3)}\} \cup W$,
 $Q^* \equiv \{S \in Q : S \text{ satisfies the equivalent conditions (4.3.49), (4.4.16) or (4.4.18)}\}$.

(4.4.20) Theorem. Let $G = (V, E, \psi)$ be a graph, let $b = (b_i : i \in V)$ be a vector of positive integers and let $c = (c_j : j \in E)$ be an arbitrary real vector. Then the maximum value of $c \cdot x$ for any matching x of G which satisfies

$$x(\delta(i)) \leq b_i \text{ for all } i \in V$$

is equal to the minimum value of

$$\sum(b_i y_i : i \in V^*) + \sum(q_S y_S : S \in Q^*)$$

where

$$y_i \geq 0 \text{ for all } i \in V^*,$$

$$y_S \geq 0 \text{ for all } S \in Q^*,$$

$$y(\psi^*(j)) + y(Q^*(j)) \geq c_j \text{ for all } j \in E.$$

$$(\psi^*(j) \equiv \psi(j) \cap V^*, Q^*(j) \equiv \{S \in Q^* : j \in \gamma(S)\})$$

for all $j \in E$.)

This theorem is best possible in the sense that if either V^* or Q^* were replaced by a smaller set then the min-max relationship of (4.4.20) would not hold for all $c \in \mathbb{R}^E$.

By combining (3.10.23) and (4.4.2) we can obtain the following strengthenings of Tutte's theorems (3.10.33) and (3.10.35).

(4.4.21) Theorem. $G = (V, E, \psi)$ has a perfect

matching if and only if for every $X \subseteq V$ such that

$$C_2(X) = \emptyset,$$

$G[S]$ is b-critical for every $S \in C_1(X)$

we have

$$b(X) \geq c_1(X) + b(\cup(C_0(X))).$$

Proof. If G has a perfect matching then by (3.10.23), for any $X \subseteq V$ we have

$$\frac{1}{2}b(V) + \frac{1}{2}(b(X) - |c_1(X)| - b(\cup(C_0(X)))) \geq \frac{1}{2}b(V)$$

$$\text{so } b(X) \geq |c_1(X)| + b(\cup(C_0(X))).$$

Suppose G has no perfect matching. Then

$\frac{1}{2}b(V) + \frac{1}{2} \min\{b(X) - |c_1(X)| - b(\cup(C_0(X)))\} < \frac{1}{2}b(V)$ by (3.10.23). By (3.10.23a) we can choose a set X^* which minimizes $b(X) - |c_1(X)| - b(\cup(C_0(X)))$ and which satisfies $c_2(X^*) = \emptyset$ and $c_1(X^*) \subseteq Q^0$. Then $b(X^*) < |c_1(X^*)| + b(\cup(C_0(X^*)))$ and since $c_1(X^*) \subseteq Q^0 = \{S \subseteq V : |S| > 3\}$ and S is a shrinkable subset of $\{V\}$ it follows from (4.4.2) that $G[S]$ is b-critical for all $S \in C_1(X^*)$. \square

If H is a component of G such that $|V(H)| = 1$ and $b_v = 1$ where $\{v\} \equiv V(H)$ then H is 1-critical. Therefore (4.4.21) becomes in the case of 1-matchings.

(4.4.22). Theorem. $G = (V, E, \psi)$ has a perfect 1-matching if and only if for every $X \subseteq V$ such that $G[V - X]$ consists of 1-critical components, the number of components of $G[V - X]$ is no greater than $|X|$.

We close this section by observing the relationship between b-critical graphs and graphs having large numbers of linearly independent "best possible" matchings. For any graph $G = (V, E, \psi)$ and vector $b = (b_i : i \in V)$ of positive integers such that $b(V)$ is odd, the largest number of linearly independent np matchings of G that we could hope to find is $|E|$, since each such matching is a vector in \mathbb{R}^E . If $E = \emptyset$ then G trivially has $|E| = 0$ linearly independent np matchings. We show in theorem (4.4.23) that if $E \neq \emptyset$, then G has $|E|$ linearly independent np matchings if and only if G is one of three sorts of graphs.

Let K be a connected graph for which there is some $v \in V(K)$ such that $\delta_K(v) = E(K)$ and let $b = (b_i : i \in V(K))$ be a vector of positive integers such that $b_v = b(V(K) - \{v\}) - 1$. We call K a b-star.

(4.4.23) Theorem. If $E \neq \emptyset$ then $G = (V, E, \psi)$ has $|E|$ linearly independent near perfect b-matchings if and only if

(4.4.24) G is b-critical and has no strong cutnode or

(4.4.25) G is a b-star

or

(4.4.26) G has two components, one being a balanced edge (as defined in 4.2) the other consisting of a single node v for which $b_v = 1$.

Proof. If $b(V)$ is even then G cannot have a np matching nor can G satisfy any of (4.4.24)-(4.4.26). Hence we assume $b(V)$ is odd. Then G has $|E|$ linearly independent np matchings if and only if $x(E) \leq 1/2(b(V) - 1)$ gives a facet of $P(G, b)$. By (4.3.48) this is true if and only if one of (4.3.49) or (4.3.50) holds. In view of (4.4.2), (4.3.49) is equivalent to (4.4.24). Moreover (4.3.50) is easily seen to be equivalent to one of (4.4.25) or (4.4.26) holding, completing the proof. \square

In the case of 1-matchings, (4.4.21) specializes as follows.

(4.4.27) Theorem. If $E \neq \emptyset$ then G has $|E|$ linearly independent near perfect 1-matchings if and only if

(4.4.28) G is 1-critical and nonseparable
or

$$(4.4.29) |V| = 3.$$

4.5 Vertices of Matching Polyhedra.

In this section we characterize the matchings of G which are vertices of $P(G, b)$. For the case of 1 matchings this problem is rather simply solved; every matching x^0 belonging to $P(G, 1)$ is a vertex of $P(G, 1)$. For if we define $c = (c_j : j \in E)$ by

$$c_j = \begin{cases} 1 & \text{if } x_j^0 = 1 \\ -1 & \text{if } x_j^0 = 0 \end{cases}$$

then x^0 is clearly the unique member of $P(G, b)$ which maximizes $c \cdot x$ for $x \in P(G, b)$. Therefore by (2.4.1) x^0 is a vertex of $P(G, b)$.

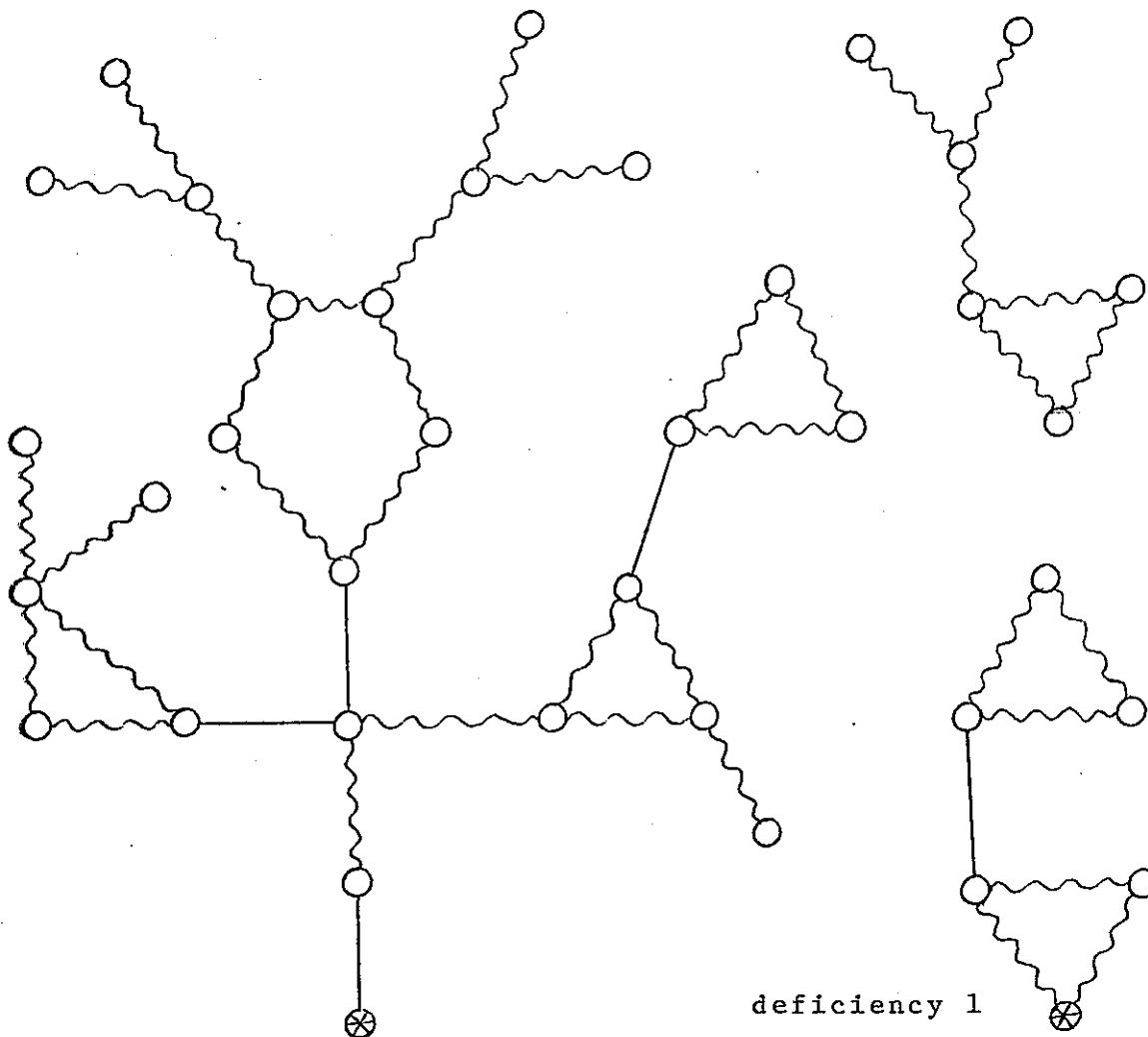
However in the general b -matching case, the problem becomes less trivial. In fact we show that the vertices of $P(G, b)$ are precisely the matchings produced by the blossom algorithm of chapter 3. Thus the set of matchings produced by the blossom algorithm is as small as possible, by (2.4.5) for any $c \in \mathbb{R}^E$, $c \cdot x$ is maximized over $P(G, b)$ by a vertex (and perhaps some other members of $P(G, b)$); by (2.4.1) every vertex of $P(G, b)$ is the unique member x of $P(G, b)$ maximizing $c \cdot x$ for some $c \in \mathbb{R}^E$.

(4.5.1) For any graph $G = (V, E, \psi)$ and any $x \in \mathbb{R}^E$ we let $G^+(x)$ be the spanning subgraph of G whose edges are those edges of G for which $x_j > 0$. Thus

$$G^+(x) \equiv (V, E^+, \psi|_{E^+})$$

where $E^+ \equiv \{j \in E : x_j > 0\}$.

Let H and K be subgraphs of a component of G . Let $v \in V$. We say that π is a path in G from v to H if π is a path in G from v to some $w \in V(H)$ and $V(\pi) \cap V(\pi) = \{v\}$. We say that π is a path in G from H to K if π is a path from some $v \in V(H)$ to K and $V(\pi) \cap V(\pi) = \{v\}$. By the distance in G from H (or v) to K we mean the length of the shortest path in G from H (or v) to K . Clearly no edge of H or K could be in a path from H to K .

FIGURE 4.5 Sample Vertex of $P(G, b)$.

Edge j : such that $x_j > 0$

Edge j such that $x_j = 1$

Node at which matching deficient

(4.5.3) Theorem. (See Figure 4.5) $x^0 \in P(G, b)$

is a vertex of $P(G, b)$ if and only if each component H of $G^+(x^0)$ satisfies the following:

(4.5.4) H contains no even polygon;

(4.5.5) H contains at most one node at which x^0 is deficient;

(4.5.6) if H contains more than one odd polygon then there is an isthmus j of H for which $x_j^0 = 1$ in any path in H joining any two of these polygons;

(4.5.7) if H contains a node v at which x^0 is deficient and some odd polygons then either v has deficiency 1 or else for any odd polygon P contained in H there is an isthmus $j(P)$ of H for which $x_{j(P)}^0 = 1$ in any path in H from v to P .

Proof. We first prove the necessity of (4.5.4)-(4.5.7) by showing that every matching produced by the blossom algorithm satisfies the conditions of the theorem. This will prove the necessity for by (2.4.1) for any vertex x^0 of $P(G, b)$ there is some $c^0 \in \mathbb{R}^E$ such that x^0 is the unique member of $P(G, b)$ which maximizes $c^0 \cdot x$ for $x \in P(G, b)$. If we use the blossom algorithm to maximize $c^0 \cdot x$ for $x \in P(G, b)$, x^0 must be the matching obtained.

Since we are maximizing over $P(G, b)$, $V^{\leq} = V$ and $V^{\geq} = \emptyset$. Therefore the blossom algorithm must terminate in step 11. If x , R and $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ are defined as at the start of step 11, and $\bar{x} \equiv x|_{\bar{E}}$ then by (3.8.13)-(3.8.16), each

component H of $\bar{G}^+(\bar{x})$ satisfies (4.5.4)-(4.5.7). We show that the operations of Step 12 preserve this property.

It is easily seen that

(4.5.8) if B is a blossom and x is a np matching of B then each component H of $B^+(x)$ satisfies (4.5.4)-(4.5.7).

Suppose x , R and $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi})$ are such that each component H of $\bar{G}^+(\bar{x})$ satisfy (4.5.4)-(4.5.7) where $\bar{x} \equiv x|_{\bar{E}}$. Suppose we perform a cycle of Step 12. This will involve executing Step 12c or Step 12d since $V'' = \emptyset$. Let S , $B(S)$ and G' be as defined in Step 12b.

Suppose we perform Step 12c. Then there is a unique $j \in \delta(S)$ for which $\bar{x}_j = 1$ and $\bar{x}_k = 0$ for all $k \in \delta(S) - \{j\}$. We let v be the node of $B(S)$ met by j and \hat{x} is the np matching of $B(S)$ deficient at v . Then j is an isthmus for which $\bar{x}_j = 1$ joining the component H of $B(S)^+(\hat{x})$ containing v to the component K of $\bar{G}^+(\bar{x})$ containing j . If x' is defined as in step 12c, x' will not be deficient at any node $i \in V(B(S))$ so using (4.5.8) we see that each component of $G'^+(\hat{x}'|_{\bar{E}'})$ satisfies (4.5.4)-(4.5.7).

Suppose we perform Step 12d. Then $\bar{x}_j = 0$ for all $j \in \delta(S)$. In Step 12d we defined \hat{x} to be a np matching of $B(S)$ deficient at $r \in B(S)$. Then where x' is as defined in Step 12d, the components of $G'^+(x')$ are precisely the components of $B(S)^+(\hat{x})$ together with the components of $\bar{G}^+(\bar{x})$. Therefore by (4.5.8) it follows that (4.5.4)-(4.5.7)

are satisfied.

Thus step 12 preserves properties (4.5.4)-(4.5.7) for each component H of $\bar{G}(x)$, suppose \bar{G} and x are the last such graph and matching defined in step 12. Then \bar{G} is a spanning subgraph of the original graph G and $x_j = 0$ for all $j \in E(G) - E(\bar{G})$. Therefore each component H of $G^+(x)$ satisfies (3.5.4)-(3.5.7) and the necessity of our conditions is proved.

In (4.5.21) we describe a procedure which expresses any matching $x \in P(G, b)$ for which a component H of $G^+(x)$ violates (4.5.4)-(4.5.7) as a convex combination of matchings $x^1, x^2 \in P(G, b) - \{x\}$. This then provides an alternative, more direct proof of the necessity of (4.5.4)-(4.5.7).

Now we prove the sufficiency. Suppose $\tilde{x} \in P(G, b)$ is a matching such that every component H of $G^+(\tilde{x})$ satisfies (4.5.4)-(4.5.7). We will show that there are $J \subseteq E$, $W \subseteq V$ and $R \subseteq Q^0$ such that $x \in \mathbb{R}^E$ satisfies

$$(4.5.9) \quad x_j = 0 \quad \text{for all } j \in J$$

$$(4.5.10) \quad x(\delta(i)) = b_i \quad \text{for all } i \in W$$

$$(4.5.11) \quad x(\gamma(s)) = q_s \quad \text{for all } s \in R$$

if and only if $x = \tilde{x}$, for then it will follow that $\{\tilde{x}\}$ is a single element face of $P(G, b)$, that is, \tilde{x} is a vertex.

Let $J \equiv \{j \in E: \tilde{x}_j = 0\}$, let $W \equiv \{i \in V: \tilde{x}(\delta(i)) = b_i\}$. We now show that it is possible to define R so that \tilde{x} will be the unique member of \mathbb{R}^E satisfying (4.5.9)-(4.5.11). We prove by induction on the number of polygons in $G^+(\tilde{x})$. If $G^+(\tilde{x})$ is a forest and thus contains no polygons, let

$R \equiv \phi$. By (4.5.5) each tree T in $G^+(\tilde{x})$ has at most one node i not belonging to W . Therefore by (3.1.11) \tilde{x} is the unique member of \mathbb{R}^E satisfying (4.5.9) and (4.5.10).

Suppose the result true for graphs \bar{G} and matchings \bar{x} such that $\bar{G}^+(\bar{x})$ contains fewer polygons than $G^+(\tilde{x})$, and suppose $G^+(\tilde{x})$ is not a forest. Let H be any component of $G^+(\tilde{x})$ which contains a polygon. If H has a node r at which \tilde{x} is deficient then we designate r as the root of H , if H has no such node then designate any polygon C contained in H as the root.

If H is rooted at a polygon C and if C is the only polygon contained in H then by (3.1.16), if x is any member of \mathbb{R}^E satisfying (4.5.9) and (4.5.10) then $x|_{E(H)} = \tilde{x}|_{E(H)}$. If we let $G' \equiv G[V - V(H)]$ and let $\tilde{x}' \equiv \tilde{x}|_{E(G')}$ then by our induction hypothesis there is a set $R \subseteq \{S \in Q^0 : S \subseteq V - V(H)\}$ such that \tilde{x}' is the unique solution to

$$x_j = 0 \text{ for all } j \in J \cap E(G'),$$

$$x(\delta(i)) = b_i \text{ for all } i \in W \cap V(G'),$$

$$x(\gamma(S)) = q_S \text{ for all } S \in R.$$

Therefore \tilde{x} is the unique solution to (4.5.9)-(4.5.11) taking R so defined and the result follows by induction.

Assume that H either contains at least two polygons or else contains a polygon and a node at which \tilde{x} is deficient. Suppose that no path in H from a polygon of H to the root of H contains an isthmus j of H for which $\tilde{x}_j = 1$. If H contains distinct polygons P and P' then there must

be a path in H from P to P' containing no isthmus j of H for which $\tilde{x}_j = 1$, contradictory to (4.5.6). Hence H contains a unique polygon P and by (4.5.7) the root r of H must be a node at which x has a deficiency of 1. Thus

(4.5.12) H is a blossom, $\tilde{x}|_{E(H)}$ is a np matching of H deficient at r .

On the other hand, let P be a polygon of H for which every path in H from P to the root r of H contains an isthmus j of H for which $\tilde{x}_j = 1$, and for which the distance in H from r to P is as great as possible. Let π be a path in H from P to r . π may contain edges of other polygons but by (4.5.6) there is an isthmus j of H for which $\tilde{x}_j = 1$ in π before any edge belonging to a polygon of H . Let k be the first isthmus of H in π for which $\tilde{x}_k = 1$. Let v be the end of k furthest from the root of H . If we delete k from H we obtain two components, one of which, B , contains P and v . It is easily verified that

(4.5.13) B is a blossom, $\tilde{x}|_{E(B)}$ is a np matching of B deficient at v .

If (4.5.12) applies, let $B \equiv H$ and $v \equiv r$. Now whichever case applies, if $x \in \mathbb{R}^E$ satisfies (4.5.9),

(4.5.14) $x(\delta(i)) = b_i$ for $i \in V(B) - \{v\}$,
and

(4.5.15) $x(\gamma(V(B))) = q_{V(B)}$,

Then since $x(\gamma(V(B))) = 1/2 \sum x(\delta(i)) : i \in V(B)$ and $q_{V(B)} = 1/2(b(V(B)) - 1)$ it is easily seen that $x(\delta(v)) = b_v - 1$. Therefore by (3.1.16)

$$(4.5.16) \quad x|_{\gamma(V(B))} = \tilde{x}|_{\gamma(V(B))}.$$

Let $G' = (V', E', \psi')$ be the graph obtained from G by shrinking $V(B)$. Let $x' \equiv \tilde{x}|_{E'}$. Clearly each component H of $G'^+(x')$ satisfies (4.5.4)-(4.5.7) and $G'^+(x')$ contains one fewer polygon than $G^+(\tilde{x})$. Therefore by our induction hypothesis there is a set R' of shrinkable subsets of G' such that $x \in \mathbb{R}^{E'}$ satisfies

$$(4.5.17) \quad x_j = 0 \quad \text{for } j \in J \cap E',$$

$$(4.5.18) \quad x(\delta_{G'}(i)) = b_i \quad \text{for } i \in W'$$

$$(4.5.19) \quad x(\gamma_{G'}(S)) = q_S \quad \text{for all } S \in R'$$

if and only if $x = x'$, where if $B = H$ then $W' \equiv W \cap V'$, if $B \neq H$ then $W' \equiv W \cap V' \cup \{V(B)\}$ (and $b_{V(B)} \equiv 1$).

Now notice that if $x \in \mathbb{R}^E$ satisfies $x(\delta(i)) = b_i$ for all $i \in V(B)$ and (4.5.15) then

$$x(\delta(V(B))) = 1 = b_{V(B)}.$$

Thus for any $x \in \mathbb{R}^E$ which satisfies (4.5.9), (4.5.10) and (4.5.15), $x|_{E'}$ satisfies (4.5.17) and (4.5.18).

For any $S \in R'$ such that $V(B) \in S$, let $\zeta(S) \equiv (S - \{V(B)\}) \cup V(B)$. Then $\zeta(S) \in Q^0$ (since S was a shrinkable subset of G'). Moreover (4.5.15) and $x(\gamma(\zeta(S))) = q_{\zeta(S)}$ imply $x(\gamma_{G'}(S)) = q_S$.

Let $\bar{R} = \{S \in R' : V(B) \in S\}$. Let $R \equiv \{V(B)\} \cup (R - R') \cup \{\zeta(S) : S \in R'\}$. Then if $x \in \mathbb{R}^E$ satisfies (4.5.9)-(4.5.11) then $x|_{E'}$ satisfies (4.5.17)-(4.5.19) so $x|_{E'} = \tilde{x}|_{E'}$. Moreover x satisfies (4.5.14) and (4.5.15) (and (4.5.9)) so (4.5.16) holds. Therefore $x \in \mathbb{R}^E$ satisfies (4.5.9)-(4.5.11) taking R as defined above if and only if $x = \tilde{x}$ and the theorem now follows by induction. \square

As a result of this theorem and theorem (2.4.14) we have the following result.

(4.5.20) Theorem. Let $\hat{x} \in P(G, b)$. There is a set $X \subseteq P(G, b)$ such that for each $x \in X$ every component H of $G^+(x)$ satisfies (4.5.4)-(4.5.7), $|X| \leq |E|$ and \hat{x} is a convex combination of the members of X .

We next describe a procedure which will express any matching $x^0 \in P(G, b)$ which is not a vertex of $P(G, b)$ as a convex combination of two different matchings which are simpler in a certain sense.

(4.5.21) Matching Simplification Algorithm.

Step 1. If $G^+(x^0)$ contains no even polygon then go to Step 2. Otherwise let P be an even polygon, let $v \in V(P)$ and let τ be a track from v to v induced by P . Let J be the set of even edges of τ . Let

$$\lambda \equiv \min\{x_j^0 : j \in J\}$$

$$\sigma \equiv \min\{x_j^0 : j \in E(P) - J\}.$$

Then λ, σ are positive integers. Define $z \in \mathbb{R}^E$ by

$$\begin{aligned} & 1 \quad \text{if } j \in J \\ z_j & \equiv -1 \quad \text{if } j \in E(P) - J \\ & 0 \quad \text{if } j \in E - E(P). \end{aligned}$$

Then

$$(4.5.22) \quad x^0 = \frac{\lambda}{\lambda+\sigma} (x^0 + \sigma z) + \frac{\sigma}{\lambda+\sigma} (x^0 - \lambda z).$$

Therefore x^0 is a convex combination of different matchings $x^0 + \sigma z$ and $x^0 - \lambda z$, both of which are members of $P(G, b)$.

Moreover,

$$(4.5.23) \quad |E(G^+(x^0 + \sigma z))| < |E(G^+(x^0))|,$$

$$(4.5.24) \quad |E(G^+(x^0 - \lambda z))| < |E(G^+(x^0))|.$$

Exit from the algorithm.

Step 2. If no component of $G^+(x^0)$ has more than one node at which x^0 is deficient, then go to Step 3. Otherwise let H be such a component and let $v, w \in V(H)$ be nodes at which x^0 is deficient. Let π be a path in H from v to w , let J be the set of odd edges of π . Let

$$\lambda' \equiv \min\{x_j^0 : j \in J\}$$

$$\sigma' \equiv \min\{b_v - x^0(\delta(v))\} \cup \{x_j^0 : j \in E(\pi) - J\}.$$

If π is of even length let $\sigma \equiv \sigma'$ and let

$$\lambda \equiv \min\{\lambda', b_w - x^0(\delta(w))\},$$

if π is of odd length, let $\lambda \equiv \lambda'$ and let

$$\sigma \equiv \min\{\sigma', b_w - x^0(\delta(w))\}.$$

Then both λ and σ are positive integers. Define $z \in \mathbb{R}^E$ by

$$\begin{aligned} z_j &= 1 \text{ if } j \in J \\ z_j &= -1 \text{ if } j \in E(\pi) - J \\ z_j &= 0 \text{ if } j \in E - E(\pi). \end{aligned}$$

Then (4.5.22) holds and x^0 is a convex combination of $x^0 + \sigma z$ and $x^0 - \lambda z$ which by our choice of σ and λ are matchings belonging to $P(G, b) - \{x^0\}$. Moreover, either (4.5.23) holds or

(4.5.25) $x^0 + \sigma z$ is deficient at fewer nodes of G than x^0 .

Similarly, either (4.5.24) holds or

(4.5.26) $x^0 - \lambda z$ is deficient at fewer nodes of G than x^0 .

Exit from the algorithm.

Step 3. If every path in $G^+(x^0)$ which joins two odd polygons in $G^+(x^0)$ contains an isthmus j for which $x_j^0 = 1$ then go to Step 4. Otherwise, notice that since we bypassed Step 1, every edge of $G^+(x^0)$ which is not an isthmus of $G^+(x^0)$ belongs to an odd polygon of $G^+(x^0)$. Therefore we can choose odd polygons P_1 and P_2 contained in $G^+(x^0)$ and a path π from $v \in V(P_1)$ to $w \in V(P_2)$ such that each edge j of π is an isthmus of $G^+(x^0)$ for which $x_j^0 \geq 2$. Let τ_1 be a track from v to v induced by P_1 , let τ_2 be a track from w to w induced by P_2 . If π

is of odd length then let J be the set of odd edges of τ_1 and τ_2 together with the set of even edges of π . If π is of even length, let J be the set of odd edges of τ_1 together with the set of even edges of π and τ_2 . Let

$$\lambda \equiv \min(\{x_j^0 : j \in J - E(\pi)\} \cup \{\frac{1}{2}x_j^0 : j \in J \cap E(\pi)\})$$

$$\sigma \equiv \min(\{x_j^0 : j \in E(\tau_1) \cup E(\tau_2) - J\} \cup \{\frac{1}{2}x_j^0 : j \in E(\pi) - J\}).$$

Then λ and σ are positive integers. Define $z \in \mathbb{R}^E$ by

$$\begin{aligned} z_j &\equiv \begin{cases} 1 & \text{if } j \in J - E(\pi) \\ -1 & \text{if } j \in E(\tau_1) \cup E(\tau_2) - J \\ 2 & \text{if } j \in J \cap E(\pi) \\ -2 & \text{if } j \in E(\pi) - J \\ 0 & \text{if } j \in E - E(\pi) \cup E(\tau_1) \cup E(\tau_2). \end{cases} \end{aligned}$$

Then (4.5.21) holds; x^0 is a convex combination of matchings $x^0 + \sigma z$, $x^0 - \lambda z \in P(G, b) - \{x^0\}$. For any $x \in P(G, b)$ let

$$I(x) \equiv \{j \in E : j \text{ is an isthmus of } G^+(x) \text{ and } x_j = 1\}.$$

Then we have either (4.5.23), (4.5.25) or

$$(4.5.27) \quad I(x^0 + \sigma z) \supset I(x^0)$$

and either (4.5.24), (4.5.26) or

$$(4.5.28) \quad I(x^0 - \lambda z) \supset I(x^0).$$

Exit from the algorithm.

Step 4. If every component H of $G^+(x^0)$ containing both an odd polygon and a node v at which x^0 has a deficiency of at least two has a member of $I(x^0)$ in every path from v to an odd polygon of H , then stop, x^0 is a vertex of $P(G, b)$. Otherwise let v be a node of a component H of $G^+(x^0)$ at which x^0 has a deficiency of at least two, let π be a path in H from v to a node w of an odd polygon P contained in H such that every $j \in E(\pi)$ is an isthmus for which $x_j^0 \geq 2$. Let τ be a track from w to w induced by P . If π is of odd length then let J be the set of odd edges of π together with the even edges of τ , if π is of even length, let J be the set of odd edges of π and τ . Let

$$\lambda \equiv \min(\{x_j^0 : j \in J \cap E(\tau)\} \cup \{\frac{1}{2}x_j^0\} : j \in J \cap E(\pi)\},$$

$$\sigma \equiv \min(\{[1/2(b_v - x^0(\delta(v)))]\} \cup \{\frac{1}{2}x_j^0\} : j \in E(\pi) - J \cup \{x_j^0 : j \in E(\tau) - J\}).$$

Then σ, λ are positive integers. Define $z \in \mathbb{R}^E$ by

$$\begin{aligned} z_j &\equiv \begin{cases} 1 & \text{if } j \in J \cap E(\tau) \\ -1 & \text{if } j \in E(\tau) - J \\ 2 & \text{if } j \in J \cap E(\pi) \\ -2 & \text{if } j \in E(\pi) - J \\ 0 & \text{if } j \in E - (E(\pi) \cup E(\tau)). \end{cases} \end{aligned}$$

Then (4.5.21) holds; x^0 is a convex combination of matchings $x^0 + \sigma z, x^0 - \lambda z \in P(G, b) - \{x^0\}$. We have either (4.5.23),

(4.5.25), (4.5.27) or

(4.5.29) there is $i \in V$ such that the deficiency of x^0 at i is at least two and the deficiency of $x^0 + \sigma z$ at i is one.

Similarly, we have (4.5.24), (4.5.26), (4.5.28) or

(4.5.30) there is $i \in V$ at which x^0 has deficiency at least two and at which $x^0 - \lambda z$ has a deficiency of one.

This ends the algorithm. \square

This algorithm has several uses. First it reproves the necessity of (4.5.4)-(4.5.7) in theorem (4.5.3), for it shows that if x^0 violates anyone of (4.5.4)-(4.5.7) then it is a convex combination of two different matchings belonging to $P(G, b)$. If $x = \lambda x^1 + (1 - \lambda)x^2$ for $\lambda \in \mathbb{R}$ satisfying $0 \leq \lambda \leq 1$ then for any $c \in \mathbb{R}^E$, $c \cdot x = \lambda c \cdot x^1 + (1-\lambda)c \cdot x^2$ so either $c \cdot x^1$ or $c \cdot x^2$ must be at least as large as $c \cdot x^0$. Therefore by (2.4.1) x^0 cannot be a vertex of $P(G, b)$.

Second, we can use this algorithm for the following problem. Let $c \in \mathbb{R}^E$ and a matching $x^0 \in P(G, b)$ be given. We wish to find a vertex x^* of $P(G, b)$ such that $c \cdot x^* \geq c \cdot x^0$. Apply the following procedure. Let $x \equiv x^0$.

Step A. Apply (4.5.21) to x . If it terminates with the information that x is a vertex of $P(G, b)$ then let $x^* \equiv x$ and stop. Otherwise it provides matchings

$x^1, x^2 \in P(G, b) - \{x\}$ such that x is a convex combination of x^1 and x^2 . At least one of $c \cdot x^1$ and $c \cdot x^2$ must be no less than $c \cdot x$; replace x with that x^1 or x^2 and return to step A.

This describes the procedure, we now show why it is finite. Notice that in the course of this procedure if at some point we perform step i of (4.5.21) then at no later point do we perform step k of (4.5.21) for $k < i$. By (4.5.23) and (4.5.24), each application of step 1 decreases $|E(G^+(x))|$. By (4.5.23)-(4.5.26) each application of step 2 decreases $|E(G^+(x))|$ or $|\{i \in V: x(\delta(i)) < b_i\}|$. By (4.5.23)-(4.5.28) each application of step 3 decreases $|E(G^+(x)) - I(x)|$. Finally, by (4.5.23)-(4.5.30) each application of step 4 decreases $|E(G^+(x)) - I(x)|$ or $|\{i \in V: b_i - x(\delta(i)) \geq 2\}|$. Therefore steps 1 through 4 can be applied at most $|E(G^+(x^0))| + |\{i \in V: x(\delta(i)) < b_i\}|$ times. Thus we will find x^* after at most $|E(G^+(x^0))| + |\{i \in V: x^0(\delta(i)) < b_i\}|$ applications of (4.5.21).

A third problem to which (4.5.21) applies is that of representing any matching $x^0 \in P(G, b)$ as a convex combination of the members of a set X of vertices of $P(G, b)$. Let $X \equiv \{x^0\}$, let $\alpha_{x^0} \equiv 1$.

Step A. Suppose we have a finite set X of matchings contained in $P(G, b)$ and $(\alpha_x: x \in X) \in \mathbb{R}$ such that

$$(4.5.22) \quad 0 \leq \alpha_x \text{ for all } x \in X,$$

$$(4.5.23) \quad \sum (\alpha_x: x \in X) = 1,$$

$$(4.5.24) \quad x^0 = \sum_{x \in X} (\alpha_x x : x \in X).$$

If every member of X is a vertex of $P(G, b)$ then stop, X is the set we require. Otherwise, suppose $\bar{x} \in X$ is not a vertex of $P(G, b)$. Apply (4.5.21) to \bar{x} , thereby obtaining matchings $x^1, x^2 \in P(G, b) - \{\bar{x}\}$ and positive $\mu_1, \mu_2 \in \mathbb{R}$ for which $\mu_1 + \mu_2 = 1$ such that $\bar{x} = \mu_1 x^1 + \mu_2 x^2$. For each $i \in \{1, 2\}$, if $x^i \in X$ then let

$$\alpha'_{x^i} \equiv \alpha_{x^i} + \mu_i \alpha_{\bar{x}};$$

if $x^i \notin X$ then let

$$\alpha'_{x^i} \equiv \mu_i \alpha_{\bar{x}}$$

For every $x \in X - \{\bar{x}, x^1, x^2\}$ let

$$\alpha'_{x^*} \equiv \alpha_x.$$

Let $X' \equiv X \cup \{x^1, x^2\} - \{\bar{x}\}$. Then if we replace X by X' and α_x by α'_x , (4.5.22)-(4.5.24) still hold; return to step A.

This describes the procedure; an argument similar to that given by the preceding procedure proves that it is finite. Unfortunately however the size of X tends to increase exponentially with the size of $|E(G^+(x^0))| + |\{i \in V : x^0(\delta(i)) < b_i\}|$. By (4.5.20) there is a set X^* of vertices of $P(G, b)$ such that x^0 is a convex combination of the members of X^* and $|X^*| \leq |E|$; it seems unlikely that the procedure described here will find such an X^* .

Chapter 5

Optimizing over Faces of $P(G, b)$

Throughout this chapter $G = (V, E, \psi)$ is a graph and $b = (b_i : i \in V)$ is a vector of positive integers. We let $c = (c_j : j \in E)$ be an arbitrary real vector. In Chapter 3 we described the blossom algorithm which solved the problem of maximizing $c \cdot x$ for $x \in P(G, b)$. In this chapter we present an algorithm called the face optimization algorithm which solves the problem of maximizing $c \cdot x$ for x belonging to any face of $P(G, b)$. This algorithm actually has two parts. The first part is a preconditioning process which is used to obtain an equivalent problem with a simpler structure. The second part, which uses a modification of the blossom algorithm as a subroutine, solves this simpler problem.

We also describe how in principle the problem of optimizing over a face can be reduced to an ordinary matching problem. Finally we show how a certain type of so called "multi-optimization" problems can be solved by solving a sequence of face optimization problems.

In Chapter 3 (Theorem(3.4.5))we proved the theorem of Edmonds, that $P(G, b)$ is the solution set of the linear inequalities (3.4.6)-(3.4.8). In view of this and (3.1.7),

$$P(G, b) = \{x \in \mathbb{R}^E :$$

$$(5.0.1) \quad x_j \geq 0 \quad \text{for all } j \in E,$$

$$(5.0.2) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V,$$

$$(5.0.3) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q'$$

where $Q' \equiv \{S \subseteq V : b(S) \text{ is odd}\}$ and $q_S \equiv 1/2(b(S) - 1)$ for all $S \in Q'$.

The difference between this set of inequalities and that prescribed in theorem (3.4.5) is that in (5.0.3) we have a "blossom inequality" for every $S \subseteq V$ such that $b(S)$ is odd and in (3.4.8) we only had such inequalities for shrinkable sets. In general then the set of inequalities (5.0.3) is far from minimal (see (4.4.19)). However by using these redundant inequalities we are able to obtain a relatively simple description of the faces of $P(G, b)$ by means of a preconditioning process.

5.1. The Faces of $P(G, b)$.

Let $W \subseteq V$, $J \subseteq E$ and $N \subseteq Q'$. Then we define the face $F(J, W, N)$ of $P(G, b)$ to be the set of all $x \in P(G, b)$ satisfying

$$(5.1.1) \quad x_j = 0 \quad \text{for all } j \in J$$

$$(5.1.2) \quad x(\delta(i)) = b_i \quad \text{for all } i \in W,$$

$$(5.1.3) \quad x(\gamma(S)) = q_S \quad \text{for all } S \in N.$$

In general there are many different choices of J , W and N which give the same face of $P(G, b)$. It is useful here to find J , W and N such that N is a nested family of sets (see Section 3.2). The following propositions form the basis of an efficient preconditioning algorithm which when presented with sets $J \subseteq E$, $W \subseteq V$ and $N \subseteq Q'$ find sets $J' \subseteq E$, $W' \subseteq V$ and $N' \subseteq Q'$ such that N' is a nested family

and $F(J, W, N) = F(J', W', N')$.

Throughout the remainder of this chapter we assume
 $J \subseteq E, W \subseteq V, N \subseteq Q'$.

(5.1.4) Proposition. Let $S, T \in N$ be such that
 $b(S \cap T)$ is odd. Let $K \equiv \gamma(S \cup T) - (\gamma(S) \cup \gamma(T))$. Then
 $F(J, W, N) = F(J \cup K, W, N - \{S, T\} \cup \{S \cap T, S \cup T\})$.

Proof. First observe that for any $x \in \mathbb{R}^E$,

$$(5.1.5) \quad x(\gamma(S \cap T)) + x(\gamma(S \cup T)) = x(\gamma(S)) + x(\gamma(T)) + x(K).$$

If $x \in F(J, W, N)$ then $x(\gamma(S)) = q_S$ and $x(\gamma(T)) = q_T$. Since $b(S \cap T)$ is odd, $b(S \cup T)$ is also odd and so since $x \in P(G, b)$, $x(\gamma(S \cup T)) \leq q_{S \cup T}$ and $x(\gamma(S \cap T)) \leq q_{S \cap T}$. Thus by (5.1.5),

$$\begin{aligned} q_S + q_T &= x(\gamma(S \cap T)) + x(\gamma(S \cup T)) - x(K) \\ (5.1.6) \quad &\leq q_{S \cap T} + q_{S \cup T} - 0 \\ &= 1/2(b(S \cap T) - 1) + b(S \cup T) - 1 \\ &= 1/2(b(S) - 1) + 1/2(b(T) - 1) \\ &= q_S + q_T. \end{aligned}$$

Therefore equality must hold in (5.1.6) and so

$$x(\gamma(S \cap T)) = q_{S \cap T},$$

$$x(\gamma(S \cup T)) = q_{S \cup T},$$

$$x(K) = 0$$

and $x \in F(J \cup K, W, N - \{S, T\} \cup \{S \cup T, S \cap T\})$.

Conversely, if $x \in F(J \cup K, W, N - \{S, T\} \cup \{S \cup T, S \cap T\})$
then by (5.1.5)

$$\begin{aligned} x(\gamma(S)) + x(\gamma(T)) &= q_{S \cup T} + q_{S \cap T} = 0 \\ &= q_S + q_T \end{aligned}$$

so since $x \in P(G, b)$ implies $x(\gamma(S)) \leq q_S$ and $x(\gamma(T)) \leq q_T$,
we have $x \in F(J, W, N)$. \square

(5.1.7) Proposition. Let $S, T \in N$ be such that $b(S \cap T)$ is even. Let $L \equiv \delta(S \cap T) \cap \delta(S \cup T)$. Then $F(J, W, N) = F(J \cup L, W \cup (S \cap T), N - \{S, T\} \cup \{S - T, T - S\})$.

Proof. First observe that for any $x \in \mathbb{R}^E$,

$$(5.1.8) \quad x(\gamma(S)) + x(\gamma(T)) = x(\gamma(S - T)) + x(\gamma(T - S)) + \sum(x(\delta(i)) : i \in S \cap T) - x(L).$$

Since $b(S \cap T)$ is even, both $b(S - T)$ and $b(T - S)$ must be odd. Suppose $x \in F(J, W, N)$. Then $x(\gamma(S)) = q_S$ and $x(\gamma(T)) = q_T$. Since $x \in P(G, b)$ it follows that $x(\gamma(S - T)) \leq q_{S-T}$, $x(\gamma(T - S)) \leq q_{T-S}$, $\sum(x(\delta(i)) : i \in S \cap T) \leq b(S \cap T)$ and $x(L) \geq 0$. These facts together with (5.1.8) imply $q_S + q_T = x(\gamma(S - T)) + x(\gamma(T - S)) + \sum(x(\delta(i)) : i \in S \cap T) - x(L)$

$$\begin{aligned} (5.1.9) \quad &\leq q_{S-T} + q_{T-S} + b(S \cap T) - 0 \\ &= q_S + q_T . \end{aligned}$$

Therefore equality must hold in (5.1.9), that is

$$x(\gamma(S - T)) = q_{S-T}, \quad x(\gamma(T - S)) = q_{T-S}, \quad x(\delta(i)) = b_i \quad \text{for all } i$$

$i \in S \cap T$ and $x(L) = 0$. Therefore

(5.1.10) $x \in F(J \cup L, W \cup (S \cap T), N - \{S, T\} \cup \{S - T, T - S\})$.

Conversely, if x satisfies (5.1.10) then (5.1.8) gives

$$\begin{aligned} x(\gamma(S)) + x(\gamma(T)) &= q_{S-T} + q_{T-S} + b(S \cap T) \\ &= q_S + q_T \end{aligned}$$

so since $x \in P(G, b)$ implies $x(\gamma(S)) \leq q_S$ and $x(\gamma(T)) \leq q_T$ we have $x(\gamma(S)) = q_S$ and $x(\gamma(T)) = q_T$. Therefore $x \in F(J, W, N)$. \square

The operations indicated by these two propositions will provide the core of our preconditioning algorithm. Now we show that by repeatedly applying these operations to an arbitrary family of sets we will eventually obtain a nested family of sets.

The following results apply to any set N of subsets of a set V , that is the value of $b(S)$ for $S \in N$ is of no significance. However our use of them will be restricted to sets $N \subseteq Q'$.

Let S and T be sets. We say that S cuts T or S and T cut each other if

$$\begin{aligned} S \cap T &\neq \emptyset, \\ S \not\subseteq T \text{ and } T \not\subseteq S. \end{aligned}$$

(5.1.11) Proposition. Let H, S and T be subsets of V and suppose S cuts T .

(5.1.12) If H cuts $S \cap T$ or $S \cup T$ then H

cuts S or H cuts T;

(5.1.13) If H cuts S ∩ T and S ∪ T then H cuts S and T;

(5.1.14) If H cuts S - T or T - S then H cuts S or T;

(5.1.15) If H cuts S - T and T - S then H cuts S and T.

Proof. If H cuts S ∩ T then $H \cap (S \cap T) \neq \emptyset$ so

(5.1.16) $H \cap S \neq \emptyset$ and $H \cap T \neq \emptyset$.

Moreover $S \cap T \not\subseteq H$ so

(5.1.17) $S \not\subseteq H$ and $T \not\subseteq H$.

Moreover $H \not\subseteq S \cap T$ so

(5.1.18) $H \not\subseteq S$ or $H \not\subseteq T$.

Combining (5.1.16)-(5.1.18) proves that if H cuts S ∩ T then H cuts S or T.

If H cuts S ∪ T then $H \not\subseteq S \cup T$ so

(5.1.19) $H \not\subseteq S$ and $H \not\subseteq T$.

Thus (5.1.16), (5.1.17) and (5.1.19) prove (5.1.13). If H cuts S ∪ T then $H \cap (S \cup T) \neq \emptyset$ so

(5.1.20) $H \cap S \neq \emptyset$ or $H \cap T \neq \emptyset$.

Moreover, suppose $H \cap T = \emptyset$. Then $H \cap (S \cap T) = \emptyset$ and

since $S \cap T \neq \emptyset$ (because S cuts T),

$$(5.1.21) \quad S \not\subseteq H.$$

Thus if $H \cap T = \emptyset$, combining (5.1.19)-(5.1.21) proves that H cuts S and similarly if $H \cap S = \emptyset$ we can see that H cuts T . Thus (5.1.12) is proved.

Suppose that H cuts $S - T$. Then

$$(5.1.22) \quad H \cap S \neq \emptyset, \quad S \not\subseteq H, \quad H \not\subseteq T.$$

If $H \not\subseteq S$ then we have immediately that H cuts S . If $H \subseteq S$ then since $H \not\subseteq S - T$ we must have $H \cap T \neq \emptyset$. If $T \subseteq H$ then we would have $T \subseteq S$, contradictory to the fact that S cuts T . Therefore $T \not\subseteq H$ and so H cuts T .

If H cuts $T - S$ then

$$(5.1.23) \quad H \cap T \neq \emptyset, \quad T \not\subseteq H \quad \text{and} \quad H \not\subseteq S$$

and a similar argument shows that H cuts S or T and (5.1.14) follows.

Finally, if H cuts both $S - T$ and $T - S$ then by combining (5.1.22) and (5.1.23) we see that H cuts both S and T , proving (5.1.15). \square

Let N be an arbitrary set of subsets of V . Let $K(N) \equiv \{\{S, T\} \in N : S \text{ cuts } T\}$. Let $k(N) \equiv |K(N)|$. Observe that $k(N) = 0$ if and only if N is a nested family of sets.

(5.1.24) Proposition. Let N be a set of subsets of V for which $k(N) > 0$. Let $\{S, T\} \in K(N)$. Let

$$\underline{N' \equiv N - \{S, T\} \cup \{S \cap T, S \cup T\}}.$$

$$N'' \equiv N - \{S, T\} \cup \{S - T, T - S\}.$$

Then $k(N') \leq k(N) - 1$ and $k(N'') \leq k(N) - 1$.

Proof. By using the correspondences suggested by (5.1.12)-(5.1.15) it is easy to exhibit one to one functions from $K(N')$ into $K(N) - \{S, T\}$ and from $K(N'')$ into $K(N) - \{S, T\}$, since $\{S \cup T, S \cap T\} \notin K(N')$ and $\{S - T, T - S\} \notin K(N'')$. The result now follows. \square

It should be observed that $k(N')$ and $k(N'')$ need not equal $k(N) - 1$, generally they will be much smaller.

The following theorem now follows directly.

(5.1.25) Theorem. Let $F(J, W, N)$ be a face of $P(G, b)$. There are J' , W' and N' such that $J \subseteq J' \subseteq E$, $W \subseteq W' \subseteq V$ and N' is a nested family of members of Q' such that $F(J, W, N) = F(J', W', N')$.

Proof. Let J^0, W^0, N^0 be such that $J \subseteq J^0 \subseteq E$, $W \subseteq W^0 \subseteq V$, $F(J^0, W^0, N^0) = F(J, W, N)$ and $k(N^0)$ is as small as possible. If $k(N^0) = 0$ then N^0 is a nested family and we are finished. Otherwise, let $\{S, T\} \in K(N^0)$. If $b(S \cap T)$ is odd then let $N^1 \equiv N^0 - \{S, T\} \cup \{S \cup T, S \cap T\}$, let $J^1 \equiv J^0 \cup (\gamma(S \cup T) - (\gamma(S) \cup \gamma(T)))$. Then by (5.1.4), $F(J^1, W, N^1) = F(J, W, N)$ and by (5.1.24) $k(N^1) < k(N^0)$, a contradiction. If $b(S \cap T)$ is even then let $N^1 \equiv N^0 - \{S, T\} \cup \{S - T, T - S\}$, let $J^1 \equiv J^0 \cup (\delta(S \cap T) \cap \delta(S \cup T))$, let $W^1 \equiv W^0 \cup (S \cap T)$. Then by (5.1.7), $F(J^1, W^1, N^1) = F(J, W, N)$ and by (5.1.24) $k(N^1) < k(N^0)$, a contradiction. \square

5.2 A Preconditioning Algorithm.

In this section we present an algorithm which when presented with sets $J \subseteq E$, $W \subseteq V$ and $N \subseteq Q'$ will find $J^* \subseteq E$, $W^* \subseteq V$ and $N^* \subseteq Q'$ such that N^* is a nested family and $F(J, W, N) = F(J^*, W^*, N^*)$. It is based upon the proof of (5.1.23) but manipulates the data in such a way that in a sense the algorithm is as efficient as could be hoped for. It relies on the following proposition.

(5.2.1) Proposition. Let N be a set of subsets of V , let S be a minimal member of N and let T be a member of N which cuts S . Then for any $H \in N$, if H does not cut S then H does not cut $S \cap T$ or $S - T$.

Proof. Since H does not cut S and S is a minimal member of N either $H \cap S = \emptyset$ or $H \supseteq S$. If $H \cap S = \emptyset$ then $H \cap (S \cap T) = H \cap (S - T) = \emptyset$ so H does not cut $S \cap T$ or $S - T$. If $H \supseteq S$ then $H \supseteq (S \cap T)$ and $H \supseteq (S - T)$ so again H does not cut $S \cap T$ or $S - T$. \square

Suppose there is $S \in N$ such that $|S| = 1$. Let $\{v\} \equiv S$. If $b_v = 1$ then every $x \in \mathbb{R}^E$ satisfies

$$(5.2.2) \quad x(\gamma\{v\}) = 1/2(b_v - 1),$$

if $b_v > 1$ then no $x \in \mathbb{R}^E$ satisfies (5.2.2) and $F(J, W, N) = \emptyset$. Thus when we detect a singleton $\{v\}$ during the preconditioning algorithm we will either ignore it if $b_v = 1$ or else stop with the information that $F(J, W, N) = \emptyset$ if $b_v > 1$.

(5.2.3) Preconditioning Algorithm.

Initially we have sets $J \subseteq E$, $W \subseteq V$, $N \subseteq Q'$, we will terminate with sets $J^* \subseteq E$, $W^* \subseteq V$ and $N^* \subseteq Q'$ such that N^* is a nested family and $F(J, W, N) = F(J^*, W^*, N^*)$ unless $F(J, W, N) = \emptyset$ in which case we terminate with that information.

Step 0. Let $\bar{J} \equiv J$, $\bar{W} \equiv W$, $R \equiv \emptyset$ and $\bar{R} \equiv N$.

Step 1. If $\bar{R} = \emptyset$ then go to Step 5. Otherwise scan \bar{R} to find a minimal member S . If $|S| = 1$ then go to Step 4, otherwise go to Step 2.

Step 2. Test each $T \in \bar{R} - \{S\}$ in turn, if T does not cut S then do nothing. If T cuts S then go to Step 2a or 2b according as $b(S \cap T)$ is odd or even. When all members of $\bar{R} - \{S\}$ have been tested, go to Step 3.

Step 2a. Replace \bar{J} with $\bar{J} \cup (\gamma(S \cup T) - (\gamma(S) \cup \gamma(T)))$ and replace \bar{R} with $\bar{R} - \{S, T\} \cup \{S \cup T, S \cap T\}$. Replace S and T with $S \cap T$ and $S \cup T$ respectively. If $|S| = 1$ then go to Step 4, otherwise return to Step 2 and resume testing members of $\bar{R} - \{S, T\}$ which have not been previously tested in this execution of Step 2.

Step 2b. Replace \bar{W} with $\bar{W} \cup (S \cap T)$, \bar{J} with $\bar{J} \cup (\delta(S \cap T) \cap \delta(S \cup T))$ and \bar{R} with $\bar{R} - \{S, T\} \cup \{S-T, T-S\}$. Replace S and T with $S - T$ and $T - S$ respectively. If $|S| = 1$ then go to Step 4, otherwise return to Step 2 and resume testing untested members of $\bar{R} - \{S, T\}$.

Step 3. Now the current S cuts no member of $\bar{R} - \{S\}$. Replace \bar{R} with $\bar{R} - \{S\}$ and if $S \notin R$ then replace R with

$R \cup \{S\}$. Go to Step 1.

Step 4. S is a singleton, let $\{v\} \equiv S$. If $b_v > 1$ then stop, $F(\bar{J}, \bar{W}, \bar{R} \cup R) = \phi$. If $b_v = 1$ then replace \bar{R} with $\bar{R} - \{S\}$ and go to step 1.

Step 5. Let $J^* \equiv \bar{J}$, $W^* \equiv \bar{W}$ and $N^* \equiv R$ and terminate the algorithm.

In view of (5.1.4) and (5.1.7), at every point in the algorithm $F(J, W, N) = F(\bar{J}, \bar{W}, \bar{R} \cup R)$. Since the size of \bar{R} is reduced by one each time we perform Step 3 and since we either terminate or reduce the size of \bar{R} by one in Step 4, the algorithm terminates after a finite number of steps. N^* is a nested family of sets for the following reason: at each stage of the algorithm R is a nested family and no member of R cuts a member of \bar{R} . This can be seen as follows. Initially $R = \phi$ and it is trivially true. It follows from (5.1.11) that each application of Step 2a or Step 2b maintains this property. Step 3 simply involves transferring a member from \bar{R} to R so this property is preserved. Step 4 either terminates or else deletes a member from \bar{R} so this property is maintained.

The importance of Proposition (5.2.1) is that after completing Step 2a or 2b we can resume our scan of $\bar{R} - \{S\}$ from where we were, we do not need to retest the members of $\bar{R} - \{S\}$ which have already been tested.

We now determine an upper bound on the amount of work done by the algorithm in solving a problem. We perform Steps 1 and 3 or 4 $|N|$ times, once for each member of N . We perform Step 2 $|\bar{R}| - 1$ times, when scanning in Step 1 we

consider $|\bar{R}|$ sets. Since $|\bar{R}| \leq |N|$ an upper bound on the amount of work done in solving a problem is of the order $|N|^2 \cdot f(V, E)$ where $f(V, E)$ is a measure of the efficiency of the set handling routines which perform the manipulations of Steps 1, 2a and 2b and so will generally depend upon $|V|$ and $|E|$ but not $|N|$.

The order of this bound seems as good as can be expected for the following reason. There are $\binom{|N|}{2}$ pairs of sets in N and the members of each such pair have to be tested to see whether or not they cut each other, since the relation "cut" is nontransitive. Thus we would expect that our bound on an algorithm to replace N with a nested family N^* would be of the order $|N|^2 \cdot f'(V, E)$ where $f'(V, E)$ is some measure of our set handling efficiency.

We now have sets J^*, W^* and N^* such that $F(J, W, N) = F(J^*, W^*, N^*)$ and N^* is a nested family, or else know that $F(J, W, N) = \emptyset$. The original set N may have been very large, if b_i is odd for all $i \in V$ then $|Q'| = 2^{|V|-1}$. However N^* is relatively small; by (3.2.3), $|N^*| \leq |V| - 1$.

In the following sections we show how to maximize $c \cdot x$ for $x \in F(J, W, N)$ where N is a nested subset of Q' which contains no singletons. This then can be combined with the preconditioning algorithm of this section to provide an efficient algorithm for solving the problem of optimizing over an arbitrary face of $P(G, b)$.

5.3 Pseudo Hungarian Forests

Let $G = (V, E, \psi)$ be a graph, let $V' \subseteq V$ and let x

be a matching of G which satisfies

$$(5.3.1) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V.$$

In (3.7.9) we defined $d(G, V^=; x)$, a measure of the amount by which x fails to be a feasible matching. Let $N \subseteq Q \equiv \{S \subseteq V: b(S) \text{ is odd and } |S| \geq 3\}$ and let x be a matching of G which satisfies (5.3.1) and

$$(5.3.2) \quad x(\gamma(S)) = q_S \quad \text{for all } S \in N.$$

We define

$$(5.3.3) \quad d(G, V^=, N; x) \equiv d(G, V^=; x) = \sum (b_i - x(\delta(i)) : i \in V^=).$$

If x satisfies (5.3.1) but violates (5.3.2) then we define

$$d(G, V^=, N; x) \equiv \infty.$$

Let X be the set of all matchings of G which satisfy (5.3.1). We define

$$(5.3.4) \quad D(G, V^=, N) \equiv \min\{d(G, V^=, N; x) : x \in X\}.$$

Clearly $D(G, V^=, N) < \infty$ if and only if G has a matching x satisfying (5.3.1) and (5.3.2) and $D(G, V^=, N) = 0$ if and only if G has a matching x satisfying (5.3.1), (5.3.2) and

$$(5.3.5) \quad x(\delta(i)) = b_i \quad \text{for all } i \in V^=.$$

Finally, observe that

(5.3.6) $d(G, V, N; x) = 1$ if and only if x is a np matching of G which satisfies (5.3.2) and consequently

(5.3.7) $D(G, V, N) = 1$ if and only if G has a np matching x which satisfies (5.3.2).

We say that a nested family N of members of Q is a shrinkable family if $G[S] \times N[S]$ is shrinkable for all $S \in Q$. (Recall $N[S] \equiv \{T \in N: T \subset S\}$).

Throughout much of the remainder of this section we will be assuming that N is a shrinkable family of members of Q . This is because the algorithm presented in the following section replaces the sets J, W, N where N is a nested family of members of Q' with sets J', W', N' where N' is a shrinkable family of subsets of V and such that

$$F(J, W, N) = F(J', W', N').$$

Let N be a shrinkable family of subsets of V . We saw in (3.7.12) that any matching \bar{x} of $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G \times N$ which satisfied $\bar{x}(\delta_{\bar{G}}(i)) \leq b_i$ for all $i \in \bar{V}$ could be extended to a matching x of G which satisfied (5.3.1) and for which $d(G, V^=; x) = d(\bar{G}, \bar{V}^=; \bar{x})$ where

$$(5.3.8) \quad \bar{V}^= \equiv (V^= \cap \bar{V}) \cup \{\text{maximal } S \in N: S \subseteq V^=\}.$$

It is easy to see that x can be constructed so as to satisfy (5.3.2). Thus we have

(5.3.9) Proposition. Let $G = (V, E, \psi)$ be a graph and let N be a shrinkable family of subsets of V .

Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G \times N$, let $V^= \subseteq V$ and let $\bar{V}^=$ be defined as in (5.3.8). Then any matching \bar{x} of \bar{G} which satisfies

$$\frac{\bar{x}(\delta_G(i)) \leq b_i}{\bar{G}} \text{ for all } i \in \bar{V}$$

can be extended to a matching x of G which satisfies (5.3.1) and (5.3.2). Moreover

$$d(G, V^=, N; x) = d(\bar{G}, \bar{V}^=; \bar{x}).$$

What is of special interest to us here however, is that when we have constraints (5.3.2), we have the following complementary result.

(5.3.10) Proposition. Let $G = (V, E, \psi)$ be a graph and let N be a shrinkable family of subsets of V . Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G \times N$, let $V^= \subseteq V$ and let $\bar{V}^=$ be defined as in (5.3.8). Then for any matching x of G which satisfies (5.3.1) and (5.3.2), $\bar{x} \equiv x|_{\bar{E}}$ is a matching of \bar{G} satisfying

$$(5.3.11) \quad \frac{\bar{x}(\delta_G(i)) \leq b_i}{\bar{G}} \text{ for all } i \in \bar{V},$$

$$(5.3.12) \quad d(G, V^=, N; x) \geq d(\bar{G}, \bar{V}^=; \bar{x})$$

Proof. Suppose x is a matching of G which satisfies (5.3.1) and (5.3.2). Then since $x(\gamma(S)) = q_S$ for all $S \in N$, it follows from (5.3.1) that

$$x(\delta(S)) \leq 1 = b_S \text{ for all } S \in N.$$

This combined with (5.3.1) proves (5.3.11).

By (5.3.1) and (5.3.2), for any pseudonode $S \in \bar{V} \cap N$,

$x|\gamma(S)$ is a np matching of $G[S]$ deficient at some node

$v(S) \in S$. Therefore

$$(5.3.13) \quad \sum(b_i - x(\delta(i)) : i \in S) = b_{v(S)} - x(\delta(v(S)))$$

$$(5.3.14) \quad = b_S - \bar{x}(\delta_{\bar{G}}(S)).$$

Therefore

$$d(G, V^=, N; x) = \sum(b_i - x(\delta(i)) : i \in V^=)$$

$$= \sum(b_i - x(\delta(i)) : i \in V^= - \cup(\bar{N}))$$

$$+ \sum(b_i - x(\delta(i)) : i \in S \cap V^=, S \in \bar{N})$$

where \bar{N} is the set of maximal members of N .

Therefore by (5.3.13)

$$(5.3.15) \quad d(G, V^=, N; x) = \sum(b_i - x(\delta(i)) : i \in V^= - \cup(\bar{N}))$$

$$+ \sum(b_{v(S)} - x(\delta(v(S)))) : S \in \bar{N}, v(S) \in V^=.$$

For any $S \in \bar{N}$, $S \in \bar{V}^=$ only if $S \subseteq V^=$ and hence only if

$v(S) \in V^=$. Therefore by (5.3.14) and (5.3.15)

$$d(G, V^=, N; x) \geq \sum(b_i - x(\delta(i)) : i \in V^= - \cup(\bar{N}))$$

$$+ \sum(b_S - \bar{x}(\delta_{\bar{G}}(S)) : S \in \bar{N} \cap \bar{V}^=)$$

$$= d(\bar{G}, \bar{V}^=; x)$$

and (5.3.12) is proved. Notice that we have strict inequality in (5.3.12) if and only if for some $S \in \bar{N}$, we have $S \notin \bar{V}^=$, and $x(\delta(S)) = 0$ and $x(\delta(i)) = b_i - 1$ for some $i \in S \cap V^=$. \square

Combining (5.3.9) and (5.3.10) we have

(5.3.16) Theorem. If G , \bar{G} , $V^=$, $\bar{V}^=$ and N are as in (5.3.10) then

$$D(G, V^=, N) = D(\bar{G}, \bar{V}^=).$$

Proposition (5.3.10) states a major difference between finding matchings satisfying (5.3.1) and (5.3.2) and simply finding matchings satisfying (5.3.1). In this latter case it is not true that every such matching of G is a matching of \bar{G} satisfying (5.3.11). Thus in the simpler problem, shrinking was never permanent, in Step 9e of the blossom algorithm we allowed for the possibility of expanding odd pseudonodes of Hungarian forests. However we shall see that when treating the problem of this chapter, shrinking can be permanent whenever we have a constraint $x(\gamma(S)) = q_S$ for a set S which we shrink.

Let $G = (V, E, \psi)$ be a graph, let R be a shrinkable family of subsets of V and let $N \subseteq R$. Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G \times R$ and let F be an alternating forest contained in \bar{G} with respect to a matching \bar{x} of \bar{G} which satisfies

$$\bar{x}(\delta_{\bar{G}}(i)) \leq b_i \text{ for all } i \in \bar{V}.$$

Let $V^=$ be a subset of V and let

$$(5.3.17) \quad \bar{V}^= \equiv (V^= \cap \bar{V}) \cup \{S \in R \cap \bar{V}: S \subseteq V^=\}.$$

We say that F is a pseudo Hungarian forest over N with respect to \bar{x} if F satisfies (3.7.2)-(3.7.4), (3.7.6), (3.7.7) and

$$(5.3.18) \quad \text{every odd node of } F \text{ is either a node}$$

of G or a member of N .

Thus the difference between a Hungarian forest and a pseudo Hungarian forest is that we allow odd nodes in pseudo Hungarian forests to be pseudonodes, a situation which was not permitted for Hungarian forests. We will see that when we require $x(\gamma(S)) = q_S$ for all $S \in N$, then pseudo Hungarian forests play a role analogous to that played by Hungarian forests when we make no such requirement.

(5.3.19) Theorem. Let $G = (V, E, \psi)$ be a graph, let $V^{\equiv} \subseteq V$ and let \bar{V}^{\equiv} be as defined in (5.3.17). Let R be a shrinkable family of subsets of V and let $N \subseteq R$. Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G \times R$ and let F be a pseudo Hungarian forest over N contained in \bar{G} with respect to a matching \bar{x} of \bar{G} . Let $K \subseteq \bar{V}$ be the set of roots of trees of F . Then $D(G, V^{\equiv}, N) = \sum (b_i - \bar{x}(\delta(i))) : i \in K$.

Proof. Let $G' = (V', E', \psi') \equiv G \times N$, let $V' = (V^{\equiv} \cap V') \cup \{S \in N \cap V' : S \subseteq V^{\equiv}\}$. By (5.3.16),

$$(5.3.19a) D(G, V^{\equiv}, N) = D(G', V') =$$

Let $\bar{R} = \{S \in R : S \not\subseteq T \text{ for any } T \in N\}$. That is, \bar{R} is the set of members of R which are not contained in pseudonodes of G' . For each $S \in \bar{R}$ we define

$$(5.3.20) V'(S) = (S \cap V') \cup \{T \in N \cap V' : T \subseteq S\}.$$

Thus $V'(S)$ is at the set of nodes of V' which correspond in the natural way to S . Let

$$(5.3.21) R' = \{V'(S) : S \in \bar{R}\}.$$

It is easily seen that R' is a shrinkable family of subsets of V' and that $G' \times R'$ is isomorphic with \bar{G} . Moreover $E(F)$ is the edge set of a Hungarian forest F' in $G' \times R'$ with respect to x' . Therefore it follows from (3.7.17) that

$$(5.3.22) \quad D(G', V') = \sum (b_i - \bar{x}(\delta(i)) : i \in K')$$

where K' is the set of roots of trees of F' . But $\sum (b_i - \bar{x}(\delta(i)) : i \in K') = \sum (b_i - \bar{x}(\delta(i)) : i \in K)$ so (5.3.19a) and (5.3.22) combine to prove the theorem. \square

This theorem is used in the algorithm to justify terminating when no feasible solution exists. We make use of the following analogue of Theorem (3.7.36) to justify replacing a constraint $x(\gamma(S)) = q_S$ for some $S \in Q'$ with a set of constraints

$$x(\delta(i)) = b_i \quad \text{for } i \in W(S) \subseteq V$$

$$x_j = 0 \quad \text{for } j \in J(S) \subseteq E$$

$$x(\gamma(T)) = q_T \quad \text{for } T \in N(S) \subseteq Q^0.$$

(5.3.23) Theorem. Let $G = (V, E, \psi)$ be a graph, let P be a subset of V , let R be a shrinkable family of subsets of P and let $N \subseteq R$. Suppose $D(G[P], P, N) = 1$. Let $\bar{G} = (\bar{V}, \bar{E}, \bar{\psi}) \equiv G[P] \times R$, let F be a pseudo Hungarian forest over N contained in \bar{G} , let I and Z be the sets of odd and even nodes of F respectively. Then a matching x of G satisfying (5.3.1) and (5.3.2) satisfies

$$(5.3.24) \quad x(\gamma(P)) = q_P$$

if and only if

$$(5.3.25) \quad \underline{x_j = 0} \quad \text{for all } j \in (\cup_{i \in I} \delta(i) \cup \delta(P)) - \cup_{i \in Z} \delta(i),$$

$$(5.3.26) \quad \underline{x(\delta(i)) = b_i} \quad \text{for all } i \in P - (Z \cup \cup(Z \cap R))$$

$$(5.3.27) \quad \underline{x(\gamma(S)) = q_S} \quad \text{for all } S \in Z \cap R.$$

Proof. Let $G' = (V', E', \psi') \equiv G[P] \times N$. By (5.3.16),

$$(5.3.28) \quad D(G', V') = D(G[P], P, N) = 1.$$

Now suppose x satisfies (5.3.1) and (5.3.2). We show

$$(5.3.29) \quad d(G[P], P, N; x) = 1 \quad \text{if and only if} \\ d(G', V'; x|E') = 1.$$

Suppose $d(G[P], P, N; x) = 1$. Then by (5.3.10)
 $d(G', V', x|E') \leq 1$, by (5.3.28) therefore we have
 $d(G', V', x|E') = 1$.

Conversely, suppose $d(G', V', x|E') = 1$. Let N' be the set of maximal members of N . Then

$$x(\gamma(P)) = x(E') + \sum(x(S): S \in N').$$

Since $d(G', V', x|E') = 1$, $x(E') = 1/2(b(V') - 1)$. Therefore, using this and (5.3.2)

$$\begin{aligned} x(\gamma(P)) &= 1/2(b(V') - 1) + 1/2\sum(b(S) - 1: S \in N') \\ &= 1/2(b(V' - N') + b(V' \cap N') - 1 \\ &\quad + \sum(b(S) - 1: S \in N')). \end{aligned}$$

But $b_v = 1$ for all $v \in V' \cap N'$ so we have

$$\begin{aligned}x(\gamma(P)) &= 1/2(b(V' - N') + |N'| - 1 + \sum(b(S) - 1 : S \in N')) \\&= 1/2(b(P) - 1)\end{aligned}$$

and so $d(G[P], P, N; x) = 1$. Thus (5.3.29) is established.

Now let $\bar{R} \equiv \{S \in R : S \not\subseteq T \text{ for any } T \in N\}$. For each $S \in \bar{R}$ let $V'(S)$ be defined as in (5.3.20) and let $R' \equiv \{V'(S) : S \in \bar{R}\}$. Then R' is easily seen to be a shrinkable family of subsets of V' and $G' \times R'$ is isomorphic with \bar{G} . Moreover, $E(F)$ is the edge set of a Hungarian forest F' in $G' \times R'$. Therefore by (3.7.36)

$$d(G', V', x|E') = 1 \text{ if and only if}$$

$$(5.3.30) \quad x(\gamma_{G'}(S)) = q_S \text{ for every } S \in R' \cap V(F'),$$

$$(5.3.31) \quad x(\delta_{G'}(i)) = b_i \text{ for every odd node } i \text{ of } F' \text{ and for every } i \in V' - V(F') - \cup(R' \cap V(F')),$$

$$(5.3.32) \quad x_j = 0 \text{ for all } j \in \cup_{i \in I} \delta_{G'}(i) - \cup_{i \in Z} \delta_{G'}(i).$$

In view of (5.3.1) and (5.3.2) it is easily seen that (5.3.30) and (5.3.27) are equivalent and that (5.3.31) is equivalent to (5.3.26) and

$$x_j = 0 \text{ for all } j \in \delta(S) - \cup_{i \in Z} \delta(i).$$

It is easily seen that (5.3.32) is equivalent to

$$x_j = 0 \text{ for all } j \in \cup_{i \in I} \delta(i) - \cup_{i \in Z} \delta(i).$$

The theorem now follows from these facts and (5.3.29). \square

5.4 The Face Optimization Algorithm (Phase II)

We are given a graph $G = (V, E, \psi)$ and a vector $b = (b_i : i \in V)$ of positive integers. Let c be an arbitrary real vector, let J be a subset of E , let W be a subset of V and let N be a nested family of members of Q . We wish to solve the problem: maximize $c \cdot x$ over x belonging to the face $F(J, W, N)$ of $P(G, b)$.

By (3.4.5) and (3.1.7) the linear program we wish to solve is

$$\text{maximize } c \cdot x$$

over $x \in \mathbb{R}^E$ which satisfy

$$(5.4.1) \quad x_j = 0 \quad \text{for all } j \in J$$

$$(5.4.1a) \quad x_j \geq 0 \quad \text{for all } j \in E - J$$

$$(5.4.2) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V - W$$

$$(5.4.3) \quad x(\delta(i)) = b_i \quad \text{for all } i \in W,$$

$$(5.4.4) \quad x(\gamma(S)) \leq q_S \quad \text{for all } S \in Q - N,$$

$$(5.4.5) \quad x(\gamma(S)) = q_S \quad \text{for all } S \in N.$$

The dual linear program is

$$\text{minimize } \sum(b_i y_i : i \in V) + \sum(q_S y_S : S \in Q)$$

for $y \in \mathbb{R}^{V \cup Q}$ which satisfy

$$(5.4.6) \quad y_S \geq 0 \quad \text{for all } S \in Q - N,$$

$$(5.4.7) \quad y_S \text{ unrestricted in sign for } S \in N,$$

$$(5.4.8) \quad y_i \geq 0 \quad \text{for all } i \in V - W,$$

$$(5.4.9) \quad y_i \text{ unrestricted in sign for } i \in W$$

$$(5.4.10) \quad y(\psi(j)) + y(Q(j)) \geq c_j \quad \text{for all } j \in E - J$$

where $Q(j) \equiv \{S \in Q: j \in \gamma(S)\}$ for any $j \in E$.

The important difference between this dual linear program and the linear program (3.5.6)-(3.5.9) is (5.4.7), for this will enable us to let the dual variables of some pseudonodes take on negative values, and consequently they can be kept shrunk throughout the course of the algorithm.

The complementary slackness conditions for optimality of a solution x to (5.4.1)-(5.4.5) and a solution y to (5.4.6)-(5.4.10) are

$$(5.4.11) \quad x_j > 0 \quad \text{only if} \quad y(\psi(j)) + y(Q(j)) = c_j$$

for all $j \in E - J$,

$$(5.4.12) \quad y_i > 0 \quad \text{only if} \quad x(\delta(i)) = b_i \quad \text{for} \\ i \in V - W,$$

$$(5.4.13) \quad y_S > 0 \quad \text{only if} \quad x(\gamma(S)) = q_S \quad \text{for} \\ S \in Q - N.$$

The general approach of our algorithm is to process each member S of N in turn, finding sets $J(S) \subseteq E$, $W(S) \subseteq V$ and a shrinkable family $N(S) \subseteq Q^0$ such that we can replace the constraint

$$x(\delta(S)) = q_S$$

with the constraints

$$x(\delta(i)) = b_i \text{ for all } i \in W(S),$$

$$x_j = 0 \text{ for all } j \in J(S),$$

$$x(\gamma(T)) = q_T \text{ for all } T \in N(S).$$

Then we carry on, applying the algorithm to this modified problem.

Eventually we find sets $\bar{J} \subseteq E$, $\bar{W} \subseteq V$ and a shrinkable family \bar{N} such that

$$F(\bar{J} \cup J, \bar{W} \cup W, \bar{N}) = F(J, W, N)$$

and we find an optimal solution x^* to the problem of maximizing $c \cdot x$ over $F(J \cup \bar{J}, W \cup \bar{W}, N)$. We also obtain an optimal dual solution y^* to this problem. Thus x^* satisfies (5.4.1)-(5.4.5), y^* satisfies (5.4.6)-(5.4.10) and x^* and y^* satisfy (5.4.11)-(5.4.13) where we replace J , W and N with $J \cup \bar{J}$, $W \cup \bar{W}$ and \bar{N} respectively.

At each stage of the algorithm we have a set $M \subseteq N$ of processed members of N . This set has the property

$$(5.4.14) \text{ if } S \in M, T \in N \text{ and } T \subseteq S \text{ then } T \in M.$$

(In other words, we always chose a minimal member of $N - M$ for processing).

Initially, we let $M \equiv \emptyset$.

For each $S \in M$ we have sets $J(S) \subseteq E$, $W(S) \subseteq S$ and $N(S) \subseteq Q^0[S] \cup \{S\}$ which have the properties described in (5.4.24)-(5.4.26). We let $\bar{J} \equiv \bigcup_{S \in M} J(S)$, $\bar{W} \equiv \bigcup_{S \in M} W(S)$ and $\bar{N} \equiv \bigcup_{S \in M} N(S)$. Then

$$(5.4.15) \quad F(J, W, M) = F(\bar{J} \cup J, \bar{W} \cup W, \bar{N}).$$

Initially, of course, $\bar{J} = \bar{W} = \bar{N} = \emptyset$.

We have a dual variable y_i defined for every $i \in U(M)$.

These are the only nodes for which a dual variable is defined at present. For every $S \in Q^0$ we have defined a dual variable y_S . This dual solution satisfies

$$(5.4.16) \quad y(\psi(j)) - y(Q^0(j)) \geq c_j \quad \text{for all } j \in \bigcup_{S \in M} \gamma(S) - (\bar{J} \cup J).$$

Let $E^{\equiv} \equiv \{j \in \bigcup_{S \in M} \gamma(S) - (\bar{J} \cup J) : y(\psi(j)) + y(Q^0(j)) = c_j\}$.

Let G^{\equiv} be the graph $(V, E^{\equiv}, \psi|_{E^{\equiv}})$.

We have defined a shrinkable family R of subsets of V such that

$$(5.4.17) \quad \bar{N} \subseteq R,$$

(5.4.18) for any $S \in R$ there is a set $T \in M$ such that $S \subseteq T$.

The sets $S \in R$ have been constructed in applications of the blossom algorithm in earlier executions of Step 2 of the algorithm to be described. The members of R satisfy

(5.4.19) for each $S \in R$, $H(S) \equiv G^{\equiv}[S] \times R[S]$ is spanned by a blossom $B(S)$.

The dual solution y has the properties

$$(5.4.20) \quad y_S = 0 \quad \text{for all } S \in Q^0 - R,$$

$$(5.4.21) \quad y_S \geq 0 \quad \text{for all } S \in R - \bar{N}.$$

We have a matching x of G defined such that

$$(5.4.22) \quad x_j = 0 \quad \text{for all } j \in E - E^+,$$

(5.4.23) $x|_{E(B(S))}$ is a np matching of $B(S)$
and $x_j = 0$ for all $j \in E(H(S)) - E(B(S))$ for all $S \in R$.

Finally, for each $S \in M$ we let $R \langle S \rangle \equiv \{T \in R: T \subseteq S\}$. With each $S \in M$ we have associated a pseudo Hungarian tree $F(S)$ over $\bar{N}[S] \equiv \{T \in \bar{N}: T \subset S\}$ contained in the subgraph G_S of $G[S] \times R \langle S \rangle$ obtained by deleting the members of $J \cup \bigcup_{T \in \bar{N}[S]} J(T)$. Moreover where $I(S)$ and $Z(S)$ are the sets of odd and even nodes of $F(S)$ respectively, we have

$$(5.4.24) \quad J(S) = \{j \in E: (j \in \delta(S) \cup j \in \bigcup_{i \in I(S)} \delta_{G_S}(i)) - \bigcup_{i \in Z(S)} \delta_{G_S}(i)\},$$

$$(5.4.25) \quad W(S) = S - Z(S) - \cup(Z(S) \cap R \langle S \rangle),$$

$$(5.4.26) \quad N(S) = \{T \in Q^0: T \in Z(S) \cap R \langle S \rangle\}.$$

Now we describe the algorithm.

Step 1. If $M = N$ then go to Step 5. Otherwise choose a minimal set $S \in N - M$ which we will now process. First we define dual variables y'_i for $i \in S$ and y'_T for $T \in Q^0 \langle S \rangle \equiv \{T \in Q^0: T \subseteq S\}$ so that

$$(5.4.27) \quad y'(\psi(j)) + y'(Q^0(j)) \geq c_j \quad \text{for all } j \in \gamma(S) - (\bar{J} \cup J)$$

$$(5.4.28) \quad y'(\psi(j)) + y'(Q^0(j)) = y(\psi(j)) + y(Q^0(j))$$

for all $j \in \bigcup_{T \in R[S]} \gamma(T) - (\bar{J} \cup J)$,

This is easy to do unless there are edges $j \in \gamma(S) - (\bar{J} \cup J)$ incident with nodes belonging to two distinct maximal members of $R[S]$ and such that $y(\psi(j)) < c_j$. In this case let Y be the set of all such edges and let

$$\sigma \equiv 1/2 \max\{c_j - y(\psi(j)) : j \in Y\}.$$

(Note that by (5.4.14) $S \notin T$ for any $T \in M$ so by (5.4.18) and since N is a nested family, a maximal member of $R[S]$ is a maximal member of R and hence $y(\psi(j)) + y(Q^0(j)) = y(\psi(j)) + y(R(j))$ by (5.4.20). But $R(j) = \emptyset$ for all $j \in Y$ so we have $y(\psi(j)) = y(\psi(j)) + y(Q^0(j))$.)

Let T be any maximal member of $R[S]$ such that $Y \cap \delta(T) \neq \emptyset$. By (5.4.14) and (5.4.18) T is a maximal member of R . By (5.4.16) $j \notin \gamma(P)$ for any $P \in M$. Thus $j \in \delta(D)$ for some $D \in M$ such that $T \in N(D)$. There is a Hungarian tree $F(D)$ defined, by (5.4.24) and the definition of \bar{J} , T must be an even pseudonode of $F(D)$.

Define y' as follows.

$$(5.4.29) \quad y'_i \equiv \begin{cases} y_i + \sigma & \text{for all } i \in D \cap (Z(D) \cup \cup(Z(D) \cap R)) \\ y_i - \sigma & \text{for all } i \in D \cap (I(D) \cup \cup(I(D) \cap R)) \\ y_i & \text{for all } i \in D - V(F(D)) - \cup(V(F(D)) \cap R). \end{cases}$$

$$(5.4.30) \quad y'_P \equiv \begin{cases} y_P + 2\sigma & \text{for every } P \in I(D) \cap R \\ y_P - 2\sigma & \text{for every } P \in Z(D) \cap R \\ y_P & \text{for all } P \in Q^0 \cap D - V(F(D)). \end{cases}$$

5.40

Notice that the only nodes for which the dual variables are decreased are odd nodes of $F(D)$ and nodes contained in odd pseudonodes of $F(D)$. By (5.4.24) any edge j meeting such a node and which is not a member of $\bar{J} \cup J$ must also meet a node whose dual variable increased by σ . Thus y' will satisfy our feasibility criteria. Moreover for any $j \in \gamma(D) - (\bar{J} \cup J)$ we have

$$y'(\psi(j)) + y'(Q^0(j)) = y(\psi(j)) + y(Q^0(j))$$

so (5.4.22) will still be satisfied when $E^=$ is defined relative to y' .

We define y' in this manner for all $D \in M$ which contain a maximal member T of $R[S]$ such that $Y \cap \delta(T) \neq \emptyset$. Thus we have

$$y'(\psi(j)) = y(\psi(j)) + 2\sigma \quad \text{for all } j \in Y$$

since each $j \in Y$ joins even nodes or nodes contained in even pseudonodes of pseudo Hungarian forests. We let $y'_i \equiv y_i$ for all $i \in \cup(R[S])$ which have not yet had y'_i defined and we let y'_i be defined for $i \in S - \cup(R[S])$ sufficiently large that (5.4.27) will hold. We let $y'_T \equiv y_T$ for all $T \in Q^0 < S >$ which have not yet had a dual variable y'_T defined.

Notice that (5.4.29) and (5.4.30) may have caused y'_i to become negative for some $i \in S$ and caused y'_p to become negative for some $P \in Q^0 < S >$. However any such i and P belong to \bar{W} and \bar{N} respectively and are not required to have nonnegative dual variables.

Step 2. Apply the blossom algorithm with the restrictions (5.4.31) and (5.4.32) to the graph $G(S) = (S, \gamma(S) - (\bar{J} \cup J), \psi|\gamma(S) - (\bar{J} \cup J))$ to attempt to find a solution to the problem

$$\text{maximize } \sum_j c_j x_j : j \in \gamma(S) - (\bar{J} \cup J)$$

where

x_j is a nonnegative integer for all $j \in \gamma(S) - (\bar{J} \cup J)$,

$$x(\delta(i)) = b_i \text{ for all } i \in S.$$

We start with the initial solution $x|\gamma(S) - (\bar{J} \cup J), y'$ and the nested family of sets $R[S]$. These are easily seen to satisfy our requirements for a starting set of values for the blossom algorithm except for the fact that there may be members T of $\bar{N}[S] \subseteq R[S]$ for which $y_T < 0$. This problem is handled by the restrictions

(5.4.31) we do not consider members of \bar{N} when computing the value of ϵ_3 in Step 9a of the blossom algorithm;

(5.4.32) we do not allow the blossom algorithm to expand members of \bar{N} in Step 9e.

Since $S \in N \subseteq Q$, $b(S)$ is odd and the algorithm must terminate in Step 10 with a new matching \bar{x} of $G(S)$, a new dual solution \bar{y} , a new nested family \bar{R} and a pseudo Hungarian forest $F(S)$ over $\bar{N}[S]$ contained in the subgraph G_S of $G[S] \times \bar{R} \langle S \rangle$ obtained by deleting all members of $\bar{J} \cup J$.

Step 3. Let K be the set of roots of trees of $F(S)$,

let

$$d \equiv \sum_{S} (b_i - \bar{x}(\delta_{G_S}(i))) : i \in K.$$

If $d = 1$ then go to Step 4. Otherwise $d \geq 2$ so by

(5.3.19), $D(G(S), S, \bar{N}[S]) \geq 2$. Therefore

$F(\bar{J} \cup J, \bar{W}, \bar{N} \cup \{S\}) = \emptyset$ so by (5.4.15) we have $F(J, W, N) = \emptyset$ and we terminate the algorithm with this information.

Step 4. ($d = 1$) $F(S)$ consists of a single tree rooted at a node $r(S) \in V(G_S)$ and $\bar{x}(\delta_{G_S}(r(S))) = b_{r(S)} - 1$.

Let

$J(S) \equiv \{j \in E : (j \in \delta(S) \text{ or } j \text{ is incident with an odd node of } F(S)) \text{ and } (j \text{ is not incident with an even node of } F(S)\}$,

$W(S) \equiv \{i \in S : i \text{ is not an even node of } F(S) \text{ or contained in an even pseudonode of } F(S)\}$,

$N(S) \equiv \{T \in Q^0 : T \text{ is an even pseudonode of } F(S)\}$.

By Theorem (5.3.23)

$F(\bar{J} \cup J, \bar{W} \cup (W \cap S), \bar{N} \cup \{S\}) = F(\bar{J} \cup J \cup J(S), \bar{W} \cup W(S), \bar{N} \cup N(S))$.

Then we replace M with $M \cup \{S\}$ and \bar{J}, \bar{W} and \bar{N} with $\bar{J} \cup J(S), \bar{W} \cup W(S)$ and $\bar{N} \cup N(S)$ respectively and (5.4.15) is still satisfied.

We define \hat{x} by

$$\begin{aligned} \hat{x}_j &\equiv \bar{x}_j & \text{if } j \in \gamma(S) - (\bar{J} \cup J) \\ &\equiv x_j & \text{if } j \in E - (\gamma(S) - (J \cup \bar{J})) \end{aligned}$$

We define \hat{y} by

$$\begin{aligned}\hat{y}_i &\equiv \bar{y}_i \quad \text{if } i \in S, \\ &y_i \quad \text{if } i \in U(M) - S\end{aligned}$$

$$\begin{aligned}\hat{y}_T &\equiv \bar{y}_T \quad \text{if } T \in Q^0[M] \\ &y_T \quad \text{if } T \in Q^0 - Q^0[M].\end{aligned}$$

Replace R with $R - R[S] \cup \bar{R}$ and x and y with \hat{x} and \hat{y} and return to Step 1.

Step 5. We have now processed all the members of N and we are going to apply the blossom algorithm to the graph G' obtained from G by deleting the edges in $\bar{J} \cup J$. First we define dual variables y'_i for the nodes $i \in V$ and y'_T for all $T \in Q^0$ so that

$$(5.4.33) \quad y'(\psi(j)) + y'(Q^0(j)) \geq c_j \quad \text{for all } j \in E - (\bar{J} \cup J)$$

$$(5.4.34) \quad y'(\psi(j)) + y'(Q^0(j)) = y(\psi(j)) + y(Q^0(j)) \quad \text{for all } j \in \cup_{T \in R} \gamma(T) - (\bar{J} \cup J)$$

$$(5.4.35) \quad y'_i \geq 0 \quad \text{for all } i \in V - \bar{W}.$$

Let Y be the set of all edges $j \in \gamma(S) - (\bar{J} \cup J)$ such that $y(\psi(j)) < c_j$ and the ends of j are two maximal members of R . Let

$$\sigma_1 \equiv 1/2 \max\{c_j - y(\psi(j)): j \in Y\}.$$

Let

$$\sigma_2 \equiv \max\{-y_i: i \in V - \bar{W} \text{ and } y_i < 0\}.$$

Let

$$\sigma \equiv \max\{\sigma_1, \sigma_2\}.$$

Now for any $D \in N$ which contains a maximal member T of R such that $Y \cap \delta(T) \neq \emptyset$ or contain a node $i \in T - \bar{W}$ such that $y_i < 0$ we define y'_i and y'_p as in (5.4.29) and (5.4.30). By (5.4.25) any $i \in T - \bar{W}$ is an even node of $F(D)$ or is contained in an even pseudonode of $F(D)$, so we add σ to the dual variable of such a node. In Step 1 we discussed the effect that this dual change had on the feasibility of the constraints $y(\psi(j)) + y(Q^0(j)) \geq c_j$, the same remarks apply here.

We let $y'_i = y_i$ for all $i \in u(R)$ which have not yet had y'_i defined and we let y'_i be defined for $i \in V-u(R)$ sufficiently large that (5.4.33) and (5.4.35) will hold. We let $y'_T = y_T$ for any $T \in Q^0$ which have not yet had a dual variable y'_T defined.

Now we apply the blossom algorithm to G' , the graph obtained from G by deleting the edges in $\bar{J} \cup J$, taking $V' = \bar{W} \cup W$ and again applying the restrictions (5.4.31) and (5.4.32). We take y' , x and R as starting solutions.

The blossom algorithm may terminate in Step 10 with a pseudo Hungarian forest F over \bar{N} . If this is the case, then by (5.3.19) $D(G', W \cup \bar{W}, \bar{N}) \geq 1$ and consequently $F(\bar{J} \cup J, W \cup \bar{W}, \bar{N}) = \emptyset$. Thus by (5.4.15) $F(J, W, M) = \emptyset$ and consequently there exists no solution to our problem; we terminate the algorithm with this information.

Otherwise the blossom algorithm terminates in Step 11 with a matching \hat{x} and a dual solution y^* . We define a matching x^* of G by

$$x_j^* = \begin{cases} \hat{x}_j & \text{for } j \in E - (\bar{J} \cup J) \\ 0 & \text{for } j \in \bar{J} \cup J. \end{cases}$$

The matching x^* is the solution we seek. Because of our restriction (5.4.32) we must have $x^*(\gamma(S)) = q_S$ for all $S \in \bar{N}$. By the operation of the blossom algorithm $x^*(\delta(i)) = b_i$ for all $i \in \bar{W} \cup W$. By our definition of x^* , $x_j^* = 0$ for all $j \in J \cup \bar{J}$. Thus $x^* \in F(J \cup \bar{J}, W \cup \bar{W}, \bar{N})$; x^* is optimal for the following reason. The matching x^* and dual solution y^* can be seen to satisfy the conditions (5.4.1)-(5.4.13), substituting $\bar{J} \cup J$ for J , $\bar{W} \cup W$ for W and \bar{N} for N . Therefore x^* maximizes $c \cdot x$ over $F(\bar{J} \cup J, \bar{W} \cup W, \bar{N})$. By (5.4.15), $F(J, W, N) = F(\bar{J} \cup J, \bar{W} \cup W, \bar{N})$ so x^* maximizes $c \cdot x$ over $F(J, W, N)$. If an optimal dual solution \bar{y} to the original problem is required, then perform the following step, Step 6. Otherwise terminate the algorithm.

Step 6. Our optimal solutions x^* and y^* satisfy the complementary slackness conditions, however in general y^* will not satisfy the conditions (5.4.6)-(5.4.8) and (5.4.10). That is there may be edges $j \in \bar{J} - J$ such that $y^*(\psi(j)) + y^*(Q^0(j)) < c_j$, there may be nodes $i \in \bar{W} - W$ such that $y_i < 0$ and there may be sets $S \in \bar{N} - N$ such that $y_S < 0$. We now describe how to obtain a vector \bar{y} which will satisfy the complementary slackness conditions (5.4.11)-(5.4.13) relative to x^* and which will satisfy (5.4.6)-(5.4.10).

Initially, let $M \equiv N$. M is the set of unprocessed numbers of N . We define a vector $\bar{y} \in \mathbb{R}^{V \cup Q}$ by

$$\begin{aligned}\bar{y}_i &\equiv y_i^* \text{ for all } i \in V, \\ \bar{y}_S &\equiv y_S^* \text{ for all } S \in Q^0, \\ \bar{y}_S &\equiv 0 \text{ for all } S \in Q - Q^0.\end{aligned}$$

At each stage we have

$$(5.4.36) \quad \bar{y}(\psi(j)) + \bar{y}(Q(j)) \geq c_j \quad \text{for all } j \in E - J - \bigcup_{S \in M} J(S)$$

$$(5.4.37) \quad \bar{y}_i \geq 0 \quad \text{for all } i \in V - W - \bigcup_{S \in M} W(S)$$

$$(5.4.38) \quad \bar{y}_T \geq 0 \quad \text{for all } T \in Q - N - \bigcup_{S \in M} N(S).$$

Step 6a: If $M = \emptyset$ then stop, by (5.4.36)-(5.4.38)

\bar{y} must satisfy (5.4.6)-(5.4.10). Otherwise choose a maximal member S of M . Let

$$\sigma_1 \equiv \max\{0\} \cup \{c_j - \bar{y}(\psi(j)) - \bar{y}(Q(j)): j \in J(S) - J\},$$

$$\sigma_2 \equiv \max\{0\} \cup \{-\bar{y}_i: i \in W(S) - W\},$$

$$\sigma_3 \equiv 1/2 \max\{0\} \cup \{-\bar{y}_T: T \in N(S) - N\}.$$

Let $\sigma \equiv \max\{\sigma_1, \sigma_2, \sigma_3\}$. If $\sigma \equiv 0$ then replace M with $M - \{S\}$, we still have (5.4.36)-(5.4.38) satisfied, return to Step 6a.

Otherwise, let $F(S)$ be the pseudo Hungarian forest as defined in the algorithm, let $J(S)$ and $Z(S)$ be the sets of odd and even nodes of $F(S)$ respectively. We define a dual solution y' by

\bar{y}_i if $i \in (V - S)$ or if i belongs to $Z(S)$

or is contained in a pseudonode of $Z(S)$,

$\bar{y}_i + 2\sigma$ if i belongs to $I(S)$ or is contained

$y'_i =$ in a pseudonode belonging to $I(S)$,

$\bar{y}_i + \sigma$ if $i \in S$ and i is not a node of $F(S)$

or contained in a pseudonode of $F(S)$.

\bar{y}_T if $T \in Q - V(F(S)) - \{S\}$,

$\bar{y}_T - 2\sigma$ if $T = S$ or if T is an odd pseudonode

$y'_T =$ of $F(S)$,

$\bar{y}_T + 2\sigma$ if T is an even pseudonode of $F(S)$.

Now if there is any edge $j \in \delta(S)$ such that $x_j^* > 0$ then $j \in E - (J \cup \bar{J})$ so j must meet an even node of $F(S)$ or a node contained in an even pseudonode of $F(S)$. If there is any node $i \in \delta(S)$ such that $x^*(\delta(i)) < b_i$ then $i \in V - (W \cup \bar{W})$ so i must be an even node of $F(S)$ or be contained in an even pseudonode of $F(S)$. For any edge $j \in \gamma(S)$ such that $y'(\psi(j)) + y'(\eta(j)) > \bar{y}(\psi(j)) + \bar{y}(\eta(j))$, one end of j must be an odd node of $F(S)$ or a node contained in an odd pseudonode of $F(S)$ and the other end of j must not be an even node of $F(S)$ or contained in an even pseudonode of $F(S)$. Therefore $j \in J(S)$ and consequently $x_j^* = 0$. Thus it can be seen that x^* and \bar{y} satisfy the complementary slackness conditions (5.4.11)-(5.4.13).

Finally, note that $y'_i \geq \bar{y}_i$ for all $i \in V$ and

$y'_T \geq \bar{y}_T$ for all $T \in Q - \{S\} - I(S)$. S belongs to N and

if $T \in Q \cap I(S)$ then T must be a member of $N(P)$ for some $P \subseteq S$. Thus $T \notin \cup_{S \in N-M} N(S)$. Therefore we replace \bar{y} with y' and M with $M - \{S\}$ and (5.4.36)-(5.4.38) are seen to be satisfied. Go to Step 6a.

This completes the description of the algorithm. We now show that the amount of work performed by the algorithm has an upper bound of the order $b(V) \cdot |V| \cdot |E|$, the same as the blossom algorithm.

For any set $S \in N$ we let $N^*(S)$ be the set of maximal members of $N[S]$. It is easily seen that an upper bound on the amount of work done in each execution of step 1 is of the order $|V| \cdot |E|$. In Step 2 we apply the blossom algorithm to $G(S)$. From (5.4.23) it can be deduced that $x|\gamma(T)$ is a np matching of $G[T]$ for each $T \in N^*(S)$. Therefore $\Delta(G(S), x|\gamma(S) - (\bar{J} \cup J), \bar{y}) \leq b(S) - \sum(b(T) - 1: T \in N^*(S))$, where $\Delta(G(S), x|\gamma(S) - (\bar{J} \cup J), \bar{y})$ is as defined in (3.8.22). Therefore by (3.9.1) an upper bound on the amount of work performed by this execution of the blossom algorithm is of the order $(b(S) - \sum(b(T) - 1: T \in N^*(S))) \cdot |S| \cdot |\gamma(S)| \leq (b(S) - \sum(b(T): T \in N^*(S))) + |N^*(S)| \cdot |V| \cdot |E|$. The amount of work performed in each execution of Steps 3 and 4 has an upper bound of the order $|V| + |E|$. Thus

(5.4.39) for each $S \in N$, the amount of work performed in processing S has an upper bound of the order $b(S) - \sum(b(T): T \in N^*(S)) + |N^*(S)| \cdot |V| \cdot |E|$.

In Step 5 we apply the blossom algorithm to the graph G' . Where N^* is the set of maximal members of N , it can

be seen that an upper bound on the amount of work performed in Step 5 is of the order $(b(V) - \sum(b(T) : T \in N^*) + |N^*|) \cdot |V| \cdot |E|$. If we add this to the sum of the bounds (5.4.39) for all $S \in N$, we see that an upper bound on the amount of work performed in Steps 1 to 5 of this algorithm is of the order $(b(V) + |N|) \cdot |V| \cdot |E|$.

It is easily seen that an upper bound on the amount of work performed in Step 6 (if this step is performed) is of the order $|N| \cdot (|V| + |E|)$. Thus

(5.4.40) An upper bound on the amount of work performed by the face optimization algorithm is of the order $(b(V) + |N|) \cdot |V| \cdot |E|$.

However N is a nested family of sets which contains no singletons so by (3.2.4), $|N| \leq |V| - 1$. Thus since $b_v \geq 1$ for all $v \in V$, we can obtain the following from (5.4.40).

(5.4.41) Theorem. An upper bound on the amount of work performed by the face optimization algorithm (phase II) in solving a problem is of the order $b(V) \cdot |V| \cdot |E|$.

We saw in Section 5.2 that an upper bound on the amount of work performed in the first phase of the face optimization algorithm, the preconditioning algorithm, was of the order $|N|^2 f(V, E)$ where N was the original, not necessarily nested, family of members of Q and $f(V, E)$ was a bound on how efficiently we could perform the set manipulations of the algorithm. In practice, if $|N|$ is reasonably small, the amount of work performed in the preconditioning phase

will be small compared to the amount of work performed in the second phase. However since $|N|$ could be as large as $2^{|V|-1}$, there could arise situations in which the preconditioning phase was the more time consuming phase of the algorithm.

5.5. The "Big-M" Method.

In this section we describe how the problem of maximizing $c \cdot x$ over a face $F(J, W, N)$ of $P(G, b)$ can be solved by a straightforward application of the blossom algorithm. Recall that the blossom algorithm described in Chapter 3 solved the problem of maximizing $c \cdot x$ over a face $F(\phi, W, \phi)$ of $P(G, b)$. We could use it to maximize over a face $F(J, W, \phi)$ of $P(G, b)$ by applying it to the graph $G' \equiv (V, E - J, \psi|E - J)$ and if an optimal matching x' was found, we obtain our solution x^* by defining

$$(5.5.1) \quad x_j^* = \begin{cases} x'_j & \text{for } j \in E - J, \\ 0 & \text{for } j \in J. \end{cases}$$

We construct a new objective function $c' = (c'_j : j \in E - J)$ with the property that if x' maximizes $c' \cdot x'$ over the face $F(\phi, W, \phi)$ of $P(G', b)$ and if $F(J, W, N) \neq \phi$ then x^* defined in (5.5.1) maximizes $c \cdot x$ over the face $F(J, W, N)$ of $P(G, b)$.

For every $x \in F(\phi, \phi, N) \subseteq P(G, b)$

$$\sum(x(\gamma(s)) : s \in N) = \sum(q_s : s \in N) \equiv q(N).$$

For any matching $x \in P(G, b) = F(\phi, \phi, N)$

$$\sum(x(\gamma(s)) : s \in N) \leq q(N) - 1.$$

Therefore if we define

$$f_j \equiv |\{s \in N : j \in \gamma(s)\}| \text{ for all } j \in E$$

then

$$(5.5.2) \quad c \cdot x = q(N) \text{ if } x \in F(\phi, \phi, N),$$

(5.5.3) $c \cdot x \leq q(N) - 1$ if x is a matching belonging to $P(G, b) = F(\phi, \phi, N)$.

Let $\lambda^* \equiv \max(\{0\} \cup \{c_j : j \in E\})$ and let $\lambda_* \equiv \min(\{0\} \cup \{c_j : j \in E\})$. Then for any $x \in \mathbb{R}^E$

$$\lambda_* x(E) \leq c \cdot x \leq \lambda^* x(E).$$

If $x \in P(G, b)$ then $x(\delta(i)) \leq b_i$ for all $i \in V$ so

$$x(E) \leq 1/2b(V).$$

Since $\lambda^* \geq 0$, $\lambda_* \leq 0$ we have therefore

$$(5.5.4) \quad 1/2b(V)\lambda_* \leq c \cdot x \leq 1/2b(V)\lambda^*.$$

Let

$$(5.5.5) \quad M \equiv \frac{b(V)}{2}(\lambda^* - \lambda_*) + 1.$$

For each $j \in E$ define

$$(5.5.6) \quad c'_j \equiv M \cdot f_j + c_j.$$

Then for any $x \in F(\phi, \phi, N)$ by (5.5.2) and (5.5.6)

$$(5.5.7) \quad \begin{aligned} c' \cdot x &= M \cdot q(N) + c \cdot x \\ &\geq M \cdot q(N) + 1/2b(V)\lambda_* \text{ by (5.5.4).} \end{aligned}$$

For any matching $x \in P(G, b) - F(\phi, \phi, N)$ by (5.5.3) and (5.5.6)

$$(5.5.8) \quad \begin{aligned} c' \cdot x &\leq M \cdot (q(N) - 1) + c \cdot x \\ &\leq M \cdot q(N) - (\frac{b(V)}{2}(\lambda^* - \lambda_*) + 1) + \frac{b(V)}{2}\lambda^* \\ &\quad \text{by (5.5.4) and (5.5.5)} \\ &= M \cdot q(N) + 1/2b(V)\lambda_* - 1. \end{aligned}$$

Thus (5.5.7) and (5.5.8) show that any member x of $F(\phi, \phi, N)$ makes $c' \cdot x$ take on a value at least one larger than does any matching x belonging to $P(G, b) - F(\phi, \phi, N)$. Moreover, (5.5.7) shows that for any members x^1 and x^2 of $F(\phi, \phi, N)$, $c' \cdot x^1 - c' \cdot x^2 = c \cdot x^1 - c \cdot x^2$ so that the relative values of the matchings $x \in F(\phi, \phi, N)$ with respect to c' are the same as their relative values with respect to c .

Now we use the blossom algorithm to solve the problem of maximizing $(c'|E - J) \cdot x$ over x belonging to the face $F(\phi, W, \phi)$ of $P(G', b)$ where $G' \equiv (V, E - J, \psi|E - J)$. If the algorithm terminates with a Hungarian forest then the face $F(\phi, W, \phi)$ of $P(G', b)$ is empty, and so the face $F(J, W, N)$ of $P(G, b)$ is empty.

If the algorithm terminates with an optimum matching x' then let x^* be defined as in (5.5.1). If $c' \cdot x^* \leq M \cdot q(N) + 1/2b(V)\lambda_* - 1$ then $x^* \notin F(\phi, \phi, N)$;

since any $x \in F(\phi, \phi, N)$ would make $c' \cdot x$ take on a larger value. $F(\phi, \phi, N) = \phi$ and hence $F(J, W, N) = \phi$.

However if $c' \cdot x^* \geq M \cdot q(N) + 1/2b(V)\lambda^*$ then $x^* \in F(J, W, N)$ and so x^* must maximize $c' \cdot x^*$ over $F(J, W, N)$. Moreover the value of the solution is easily computed, by (5.5.7)

$$c \cdot x^* = c' \cdot x^* - M \cdot q(N).$$

Thus this procedure reduces the face optimization problem to a matching problem which can be handled by the blossom algorithm. Since (Theorem (5.9.2)) the bound on the amount of work performed by the blossom algorithm is independent of the edge costs, the bound on this procedure is the same as that of applying the blossom algorithm to simply maximize $c \cdot x$ over $P(G, b)$. The only drawback with this approach is that if $b(V)$, λ^* and $q(N)$ are large then a computer implementation might experience some difficulty in storing all the significant digits in the numbers c'_j and in the dual variables. The algorithm of Sections 5.2 and 5.4 does not have this difficulty.

5.6. Multi-Optimization in Matching Problems

In this section we describe how the principle of complementary slackness can be used with the algorithms of this chapter to solve matching problems in which we have specified not just one, but several objective functions to be maximized according to some levels of priorities. For example, we may be given a subset J of the edges of

$G = (V, E, \psi)$ and a vector $c \in \mathbb{R}^E$ and wish to find a matching $\hat{x} \in P(G, b)$ for which $\hat{x}(J)$ is maximal over $P(G, b)$ and for which $c \cdot \hat{x}$ is maximal over the members of $P(G, b)$ which maximize $x(J)$.

The method described is based on Theorem (2.1.8) which shows that if $c \cdot x$ has a maximum value z over a polyhedron P then $\{x \in P: c \cdot x = z\}$ is a face of P . A matching polyhedron $P(G, b)$ is a bounded polyhedron (for every $j \in E$, $0 \leq x_j \leq \min\{b_i: i \in \psi(j)\}$, for all $x \in P(G, b)$) and consequently for any $c \in \mathbb{R}^E$, cx has an upper bound over $P(G, b)$. Therefore for any $c \in \mathbb{R}^E$, cx is maximized over $P(G, b)$ by precisely the members of some face of $P(G, b)$.

Now we describe the first sort of multi-optimization problem considered. We are given a graph $G = (V, E, \psi)$, a vector $b = (b_i: i \in V)$ of positive integers and a sequence c^1, c^2, \dots, c^k of members of \mathbb{R}^E . Let $X_0 \equiv P(G, b)$ and for each $i \in \{1, 2, \dots, k\}$ we let

$$X_i \equiv \{x \in X_{i-1}: c^i \cdot x \text{ is maximized over } X_{i-1}\}.$$

The multi-optimization problem is to find a matching $x^* \in X_k$.

(5.6.1) Multi-Optimization Algorithm.

Step 0: Let $i \equiv 0$, let $J_0 \equiv W_0 \equiv N_0 \equiv \emptyset$. Then trivially we have $X_0 = P(G, b) = F(J_0, W_0, N_0)$.

Step 1: We assume we know sets $J_i \subseteq E$, $W_i \subseteq V$ and $N_i \subseteq Q$ such that $X_i = F(J_i, W_i, N_i)$. We now use the face optimization algorithm to find a solution x^{i+1} to the problem

maximize $c^{i+1} \cdot x$

over $x_i = F(J_i, W_i, N_i)$.

If $i = k - 1$ then x^k is the required solution to the multi-optimization problem, stop the algorithm. Otherwise go to Step 2.

Step 2. Let y^{i+1} be the dual solution supplied by the face optimization algorithm. By the complementary slackness conditions (5.4.11)-(5.4.13) $\hat{x} \in F(J_i, W_i, N_i)$ maximizes $c \cdot x$ over $F(J_i, W_i, N_i)$ if and only if

$$(5.6.2) \quad \hat{x}_j = 0 \quad \text{for all } j \in E \quad \text{such that} \\ y^{i+1}(\psi(j)) + y^{i+1}(Q(j)) > c_j;$$

$$(5.6.3) \quad \hat{x}(\delta(v)) = b_v \quad \text{for all } v \in V \quad \text{such that} \\ y^{i+1}_v > 0;$$

$$(5.6.4) \quad \hat{x}(\gamma(S)) = q_S \quad \text{for all } S \in Q \quad \text{such that} \\ y^{i+1}_S > 0.$$

Thus $X_{i+1} = \{\hat{x} \in X_i : \hat{x} \text{ satisfies (5.6.2)-(5.6.4)}\}$ and so we define

$$J_{i+1} \equiv J_i \cup \{j \in E : y^{i+1}(\psi(j)) + y^{i+1}(Q(j)) > c_j\},$$

$$W_{i+1} \equiv W_i \cup \{v \in V : y^{i+1}_v > 0\},$$

$$N_{i+1} \equiv N_i \cup \{S \in Q : y^{i+1}_S > 0\}.$$

Now

$$X_{i+1} = F(J_{i+1}, W_{i+1}, N_{i+1}).$$

Replace i with $i + 1$ and return to Step 1.

This completes the description of the algorithm. It is clear that solving this multi-optimization algorithm will involve k applications of the face optimization algorithm. Generally the graph considered in each successive application of the face optimization algorithm in Step 1 will have fewer edges than the one handled in the previous cycle, since growth of the sets J_i is equivalent to deleting edges of the graph. Thus we would expect the multi-optimization algorithm to perform somewhat better, certainly no worse, than solving k face optimization problems for the original graph.

The following problem is a variant of this multi-optimization problem. We wish to find, if one exists, an element x^* of $P(G, b)$ such that $c^i \cdot x$ is maximized over $P(G, b)$ by x^* for all $i \in \{1, 2, \dots, k\}$. This we do by finding for each $i \in \{1, 2, \dots, k\}$ the face F_i of $P(G, b)$ containing all those x which maximize $c^i \cdot x$ over $P(G, b)$ and then we find a matching $x^* \in \bigcap_{i=1}^k F_i$ if this set is not empty.

For each $i \in \{1, 2, \dots, k\}$ we use the blossom algorithm to find an optimal primal solution and an optimal dual solution y^0 to the problem of maximizing $c^i \cdot x$ over $P(G, b)$. We let

$$J_i \equiv \{j \in E: y^0(\psi(j)) + y^0(Q(j)) > c_j^i\},$$

$$W_i \equiv \{v \in V: y_v^0 > 0\},$$

$$N_i \equiv \{s \in Q: y_s^0 > 0\}.$$

By complementary slackness (Theorem(1.5.16)) applied to the linear programs (3.5.1)-(3.5.5) and (3.5.6)-(3.5.9) (taking $V^* \equiv \phi$ and $c \equiv c^i$) $x \in P(G, b)$ maximizes $c^i \cdot x$ over $P(G, b)$ if and only if $x \in F(J_i, W_i, N_i)$. By (2.1.4) we have

$$\bigcap_{i=1}^k F(J_i, W_i, N_i) = F(J, W, N)$$

where

$$J \equiv \bigcup_{i=1}^k J_i,$$

$$W \equiv \bigcup_{i=1}^k W_i,$$

$$N \equiv \bigcup_{i=1}^k N_i.$$

Thus we can now use the face optimizing algorithm to find a matching $x^* \in F(J, W, N)$ if such a matching exists. (Take $c_j = 0$ for all $j \in E$ as an objective function to be maximized). The algorithm will either terminate with the information that $F(J, W, N) = \phi$ or with the matching x^* which we require.

This process involves k applications of the blossom algorithm and one application of the face optimizing algorithm. The face optimizing algorithm is applied to a problem of a particularly simple type, one in which $c_j = 0$ for all $j \in E$. This means that if the dual variables are defined initially to be zero, then they will never be changed in the course of the algorithm.

If we find a solution x^* then clearly we could have found it by using the first algorithm described in this section.

However if we simply apply the first algorithm when presented with a problem of this sort we will not know if the matching x^k produced by this algorithm maximizes $c^i \cdot x$ over $P(G, b)$ for $i \in \{2, 3, \dots, k\}$ unless we check it for all such i . This checking procedure involves $k - 1$ applications of the blossom algorithm. Thus altogether we would have to solve k face optimizing problems and $k - 1$ ordinary matching problems so the advantage of the second method of solution is apparent.

We can combine these two methods in an obvious fashion to solve multi-optimization problems of the following sort. Let C_1, C_2, \dots, C_k be a sequence of finite nonempty subsets of \mathcal{R}^E , let $X_0 \equiv P(G, b)$ and for each $i \in \{1, 2, \dots, k\}$ let

$$X_i \equiv \{x \in X_{i-1} : c \cdot x \text{ is maximized over } X_{i-1} \text{ for all } c \in C_i\}.$$

We wish to find a matching $x^* \in X_k$ if such a matching exists. If such a matching exists, we have to solve $l + \sum_{i=1}^k |C_i|$ face optimization problems to find it. However if $X_k = \emptyset$ then it may well happen that $X_i = \emptyset$ for some $i < k$ and so we would discover this without solving so many problems.

A first approach which might be considered for solving multi-optimization problems is a generalization of the "Big-M" method of the previous section. This would involve selecting positive constants $M_1 \gg M_2 \gg \dots \gg M_k$ and letting $c'_j \equiv \sum_{i=1}^k M_i c_j^i$ for all $j \in E$. Then it is easily

seen that a solution x^* to the problem of maximizing $c' \cdot x$ over $P(G, b)$ is a solution to the first multi-optimization problem discussed. However, although this method is fine in theory, in practice if k is reasonably large then the huge number of significant digits which would have to be handled for the c'_j makes this method infeasible.

The methods described in this section could be applied to other classes of multi-optimization problems besides matching problems provided a face optimizing algorithm were known which provided an optimal dual solution. However, for linear programs in which the number of constraints is of a manageable size there are more direct methods (for example, a generalization of the two phase method of obtaining a starting basic feasible solution; see Dantzig [D1] Chapter 5 Section 2 for a discussion of the two phase method.)

Chapter 6

A Post-Optimality Problem

In this chapter we discuss one aspect of post-optimality for matching problems, modification of the degree constraints. Assume that we are given a graph $G = (V, E, \psi)$, a vector b of positive integral degree constraints and an arbitrary vector $c \in \mathbb{R}^E$. We suppose that we have applied the blossom algorithm of Chapter 3 to the matching problem

$$(6.0.1) \quad \text{maximize } c \cdot x$$

over $x \in \mathbb{R}^E$ which satisfy

$$(6.0.2) \quad x_j \text{ is a nonnegative integer for all } j \in E,$$

$$(6.0.3) \quad x(\delta(i)) = b_i \text{ for all } i \in V^= \subseteq V,$$

$$(6.0.4) \quad x(\delta(i)) \leq b_i \text{ for all } i \in V^{\leq} \equiv V - V^=,$$

and x^0 and y^0 are the optimal matching and dual solution thereby obtained. Now we wish to solve (6.0.1)-(6.0.4) again, replacing b with a new vector b' of degree constraints. We could simply reapply the blossom algorithm taking $x_j \equiv 0$ for all $j \in E$ as an initial matching. If we do this, the upper bound on the amount of work required (see (3.9.2)) is of the order

$$b'(V) \cdot |V| \cdot |E|.$$

In this chapter we describe a method of solving this problem which utilizes the solutions x^0 and y^0 which we already know. We show that an upper bound on the amount of work required by this method is of the order

$$(\sum(|b_i - b'_i| : i \in V) + |V|) \cdot |V| \cdot |E|.$$

Thus it is clear that if the values of $|b_i - b'_i|$ are small relative to the values of b'_i for $i \in V$ then our new method has a somewhat better bound than a direct application of the blossom algorithm.

6.1 Obtaining a Starting Solution

Throughout this chapter we let $G = (V, E, \psi)$ be a graph, we let $b = (b_i : i \in V)$ be a vector of positive integers, we let $c = (c_j : j \in E)$ be an arbitrary real vector and let y be a feasible dual solution of (6.0.1)-(6.0.4) (see Section 3.7). Let R be a shrinkable family of subsets of G and let $\bar{G} \equiv G \times R$. In (3.8.22) we defined $\Delta(\bar{G}; x, y)$ for any matching x of G which satisfied $x(\delta(v)) \leq b_v$ for all $v \in V(\bar{G})$. In the case $R = \emptyset$ we have

$$\Delta(G; x, y) = \sum(b_i - x(\delta(i)) : i \in V^= \text{ or } (i \in V^{\leq} \text{ and } y_i > 0))$$

where x is a matching satisfying

$$(6.1.1) \quad x(\delta(i)) \leq b_i \quad \text{for all } i \in V$$

and y is any feasible dual solution. The value of $\Delta(G; x, y)$ measures, in a sense, how close x and y are to being optimal feasible solutions to the matching problem.

Now suppose that y is any feasible dual solution and x is any matching of G , that is x need not satisfy (6.1.1). We define

$$\begin{aligned}\Delta(G, b; x, y) \equiv & \sum(|b_i - x(\delta(i))| : i \in V^= \text{ or} \\ & (i \in V^{\leq} \text{ and } y_i > 0)) \\ & + \sum(\max\{0, x(\delta(i)) - b_i\} : i \in V^{\leq} \text{ and } y_i = 0).\end{aligned}$$

(In Chapter 3 the vector b was constant so we did not introduce it as a parameter of $\Delta(G; x, y)$. Here however we consider more than one vector of degree constraints so we include b as a parameter of $\Delta(G, b; x, y)$. Throughout this chapter the set $V^=$ is constant, so we do not include it as a parameter of $\Delta(G, b; x, y)$ although of course it does affect this value.)

Notice that

$$(6.1.1a) \quad \text{if } x \text{ satisfies (6.1.1) then} \\ \Delta(G, b; x, y) = \Delta(G; x, y).$$

(6.1.2) Proposition. For any matching x of G
there exists a matching x' of G such that

$$(6.1.3) \quad x'_j \neq 0 \quad \underline{\text{only if}} \quad x_j \neq 0 \quad \underline{\text{for all }} j \in E,$$

$$(6.1.4) \quad \underline{\Delta(G, b; x', y)} \leq \underline{\Delta(G, b; x, y)}$$

$$(6.1.5) \quad \underline{x'(\delta(i)) \leq b_i} \quad \text{for all } i \in V.$$

Proof. Our proof consists of an algorithm for constructing the matching x' .

Step 0. Let $d_i \equiv 0$ for all $i \in V$. Let $J \equiv \emptyset$. At each stage of the algorithm we have a nonnegative integer x'_j defined for all $j \in J \subset E$ such that

$$(6.1.6) \quad x'(\delta(i) \cap J) = d_i \leq b_i \quad \text{for all } i \in V.$$

Step 1. If $J = E$ then x' is the matching we require, stop the algorithm. Otherwise choose $j \in E - J$, let $\{u, v\} \equiv \psi(j)$. Let

$$x'_j \equiv \min\{b_u - d_u, b_v - d_v, x_j\}.$$

Replace d_u and d_v with $d_u + x'_j$ and $d_v + x'_j$ respectively, add j to J and return to Step 1.

This describes the algorithm, we now discuss why it works. Clearly our actions in Step 1 preserve (6.1.6). Properties (6.1.3) and (6.1.5) are immediate consequences of our definition of x'_j and (6.1.6). For any $J \subseteq E$ as constructed in the algorithm we define a matching x^J by

$$x_j^J \equiv \begin{cases} x_j & \text{if } j \in E - J \\ x'_j & \text{if } j \in J. \end{cases}$$

We show that $\Delta(G, b; x^J, y) \leq \Delta(G, b; x, y)$ for all such J . Since $x^E = x'$ this will prove (6.1.4).

Initially $J = \emptyset$ and the result is trivial. Suppose it holds for some $J \subset E$ and let j be as chosen in Step 1. If $x'_j = x_j$ then $x^J = x^{J \cup \{j\}}$ and the result is trivial. Otherwise there is $u \in \psi(j)$ such that $x'_j = b_u - d_u$ and $x'_j \leq b_v - d_v$ where $\{v\} \equiv \psi(j) - \{u\}$. Therefore, by (6.1.6) $x'_j + x'(\delta(u) \cap J) = b_u$ and $x'_j + x'(\delta(u) \cap J) \leq b_v$. The term in $\Delta(G, b; x^J, y)$ corresponding to u contributes $x_j - x'_j$ more to this sum than the corresponding term contributes to $\Delta(G, b; x^{J \cup \{j\}}, y)$. However the term in $\Delta(G, b; x^J, y)$

corresponding to v contributes at most $x_j - x'_j$ more to this sum than the corresponding term contributes to $\Delta(G, b; x^{J \cup \{j\}}, y)$. Since all other terms contribute the same amount to both sums, $\Delta(G, b; x^{J \cup \{j\}}, y) \leq \Delta(G, b; x^J, y) \leq \Delta(G, b; x, y)$ by our hypothesis and the result follows. \square

Observe that the amount of work performed by this algorithm is of the order $|E|$.

As in Chapter 4, for any graph $G = (V, E, \psi)$ and any matching x we define the graph

$$G^+(x) = (V, E^+(x), \psi|_{E^+(x)})$$

where

$$E^+(x) \equiv \{j \in E : x_j > 0\}.$$

In Section 4.5 we described the structure possessed by the vertices of $P(G, b)$ and hence of the matchings produced by the blossom algorithm. We next show how from any matching $x \in P(G, b)$ we can obtain a matching x' of $P(G, b)$ which will have several of the same characteristics as vertices of $P(G, b)$ and such that $\Delta(G, b; x', y) \leq \Delta(G, b; x, y)$. In the uses that we make of this procedure, the matchings x with which we start will be such that x' will be a special type of vertex of $P(G, b)$, a vertex x' for which any component of $G^+(x')$ contains at most one polygon.

(6.1.7) Theorem. For any matching x of G satisfying (6.1.1) there is a matching x' of G satisfying (6.1.3)-(6.1.5) and for which each component H of $G^+(x')$ satisfies

(6.1.8) H contains at most one node at which x' is deficient,

(6.1.9) if H contains an odd polygon and a node v at which x' is deficient then either x' has a deficiency of 1 at v or else there is a set $J \subseteq E$ such that $x'_j = 1$ for all $j \in J$ and any path in H from v to an odd polygon of H contains a member of J.

Proof. Again, our proof describes an algorithm for actually constructing x' . The operations of the algorithm are similar to Steps 2 and 4 of the Matching Simplification Algorithm (4.5.21).

Initially, let $x' \equiv x$.

Step 1. If each node of G at which x' is deficient belongs to a distinct component of $G^+(x')$ then (6.1.8) holds for all components H of $G^+(x')$ and we go to Step 2. Otherwise let v and w be nodes belonging to a component H of $G^+(x')$ such that x' is deficient at both v and w and

(6.1.10) if $y_v = 0$ and $v \in V^{\leq}$ then $y_w = 0$ and $w \in V^{\leq}$.

(If our original choice of v and w violated (6.1.10) we simply interchange v and w). Let π be a path in H from v to w. Let

$$\sigma_1 \equiv \min\{x'_j : j \text{ is an even edge of } \pi\}.$$

Since $j \in E^+(x')$ for all $j \in E(\pi)$, $\sigma_1 \geq 1$. Let

$\sigma \equiv$

$$\begin{aligned} & \min\{\sigma_1, b_v - x'(\delta(v))\} \text{ if } \pi \text{ is of even length,} \\ & \min\{\sigma_1, b_v - x'(\delta(v)), b_w - x'(\delta(w))\} \text{ if } \pi \text{ is} \\ & \quad \text{of odd length.} \end{aligned}$$

Define x'' by

$$\begin{aligned} x'_j + \sigma & \text{ if } j \text{ is an odd edge of } \pi, \\ x''_j & \equiv x'_j - \sigma \text{ if } j \text{ is an even edge of } \pi, \\ x'_j & \text{ if } j \in E - E(\pi). \end{aligned}$$

Replace x' with x'' and return to Step 1.

Step 2. If (6.1.9) is satisfied then stop, x' is the desired solution. Otherwise let v be a node in a component H of $G^+(x')$ such that x' has a deficiency of at least 2 at v and let π be a path from v to an odd polygon P in H such that $x'_j \geq 2$ for all $j \in E(\pi)$. Let $\{w\} \equiv V(P) \cap V(\pi)$, let τ be a track from w to w induced by P . Let

$$\begin{aligned} \sigma_1 & \equiv \min\{x'_j : j \text{ is an even edge of } \pi\}, \\ \sigma_2 & \equiv \min\{x'_j : j \text{ is an even edge of } \tau\} \text{ if } \pi \text{ has even length,} \\ & \min\{x'_j : j \text{ is an odd edge of } \tau\} \text{ if } \pi \text{ has odd length,} \end{aligned}$$

Let

$$\sigma \equiv \min\{\frac{1}{2}(b_v - x'(\delta(v))), [\frac{1}{2}\sigma_1], \sigma_2\}.$$

Then $\sigma \geq 1$. Now define x'' as follows.

$x'_j = 2\sigma$ if j is an even edge of π ,

$x'_j + 2\sigma$ if j is an odd edge of π ,

$x'_j + \sigma$ if j is an even edge of τ and
 $|E(\pi)|$ is odd
or if j is an odd edge of τ and
 $|E(\pi)|$ is even,

$x''_j \equiv x'_j - \sigma$ if j is an even edge of τ and
 $|E(\pi)|$ is even,
or if j is an odd edge of τ and
 $|E(\pi)|$ is odd,

x'_j if $j \in E - (E(\pi) \cup E(\tau))$.

Replace x' by x'' and return to step 2.

This describes the algorithm, we now show why it works.

Each time we perform Step 1 we either decrease the number of deficient nodes or introduce a new edge j for which $x''_j = 0$ (or both). Each time we perform Step 2 we decrease the deficiency of a node (by at least 2). Since neither Step 1 nor Step 2 introduce new deficient nodes and since in both Step 1 and Step 2, $x''_j \neq 0$ only if $x'_j \neq 0$, the algorithm will terminate and (6.1.3) and (6.1.5) are easily seen to be satisfied. By virtue of the fact that we terminated; every component H of $G^+(x')$ must satisfy (6.1.8) and (6.1.9) for the final x' .

In order to see that (6.1.4) is satisfied by our final x' , observe that $|b_i - x'(\delta(i))|$ is increased by an application of Step 1 or 2 in exactly one case, namely in Step 1 when $i = w$ and $|E(\pi)|$ is odd. However in this case we decrease $|b_v - x'(\delta(v))|$ by an identical amount. The

only time the term in $\Delta(G, b; x', y)$ corresponding to v contributes nothing to $\Delta(G, b; x', y)$ is when $v \in V^{\leq}$ and $y_v = 0$. But in this case, by (6.1.10), $w \in V^{\leq}$ and $y_w = 0$ so the term corresponding to w contributes nothing to $\Delta(G, b; x', y)$. Hence in every case $\Delta(G, b; x'', y) \leq \Delta(G, b; x', y)$ following an application of Step 1 or Step 2 and the proof is complete. \square

6.2 The Post-optimality Algorithm.

We describe in this section what could be considered a two phase approach to the post optimality problem. The first phase will involve modifying an existing matching x^0 and corresponding dual solution y^0 to obtain a matching x and dual solution y which are acceptable as input to the blossom algorithm with the new degree constraints b' . Then the second phase will consist of a straightforward application of the blossom algorithm.

The two phase structure of this algorithm makes it particularly attractive for computer implementation, for given that we have a computer code of the blossom algorithm, we need only write new code for the first phase; no reprogramming of the blossom algorithm is required.

Let $b = (b_i : i \in V)$ and $b' = (b'_i : i \in V)$ be two real vectors indexed by V . We measure the difference of b' and b by

$$\|b - b'\| = \sum_i (|b_i - b'_i| : i \in V).$$

This is commonly called the 1-norm of the vector $(b - b')$ (Isaacson and Kellar [11], p. 4).

Suppose that we have used the blossom algorithm to solve the matching problem (6.0.1)-(6.0.4) and that x^0 is the optimal feasible solution found, y^0 is the optimal dual solution (see Section 3.7) and R is the nested family of sets which we had when we reached Step 11 of the blossom algorithm. (The knowledge of R is not essential, given an x^0 and y^0 as above, we can construct a nested family R' of members of Q^0 which will suffice. This we discuss in Section 6.3).

Phase 1. Initialization.

Step 1: Define vectors x^1 and y^1 as follows. Let $J \equiv \cup_{S \in R} \delta(S)$. Let

$$(6.2.1) \quad x_j^1 \equiv \begin{cases} 0 & \text{if } j \in J, \\ x_j^0 & \text{if } j \in E - J. \end{cases}$$

For any $i \in V$ let $R(i) = \{S \in R : i \in S\}$. Let

$$(6.2.2) \quad y_i^1 \equiv y_i^0 + 1/2 \sum (y_S^0 : S \in R(i)) \text{ for all } i \in V,$$

$$(6.2.3) \quad y_S^1 \equiv 0 \text{ for all } S \in Q^0.$$

Go to Step 2.

Notice that

(6.2.3a) each component H of $G^+(x^1)$ can contain at most one polygon, for by (3.8.14) each component of $G \times R^+(x^1)$ contained at most one odd polygon and for every $S \in R$, each component of $(G[S] \times R[S])^+(x^1)$ is a subgraph of a blossom

and so contains at most one polygon and $x_j^1 = 0$ for all $j \in \delta(S)$ for all $S \in R$.

We now show that y^1 has the following two properties.

$$(6.2.4) \quad y^1(\psi(j)) + y^1(Q^0(j)) \geq y^0(\psi(j)) + y^0(Q^0(j))$$

for all $j \in J$,

$$(6.2.5) \quad y^1(\psi(j)) + y^1(Q^0(j)) = y^0(\psi(j)) + y^0(Q^0(j))$$

for all $j \in E - J$.

Let $j \in E$ and let $R(j) \equiv \{S \in R: j \in \gamma(S)\}$. By (3.8.9), $y_S^0 = 0$ for all $S \in Q - R$, so

$$(6.2.6) \quad y^0(Q^0(j)) = y^0(R(j)).$$

If we let $\{u, v\} \equiv \psi(j)$ then we have

$$(6.2.7) \quad R(j) = R(u) \cap R(v) \text{ for all } j \in E$$

and hence, since $y_S^0 \geq 0$ for all $S \in Q^0$,

$$(6.2.8) \quad y^0(R(j)) \leq 1/2(y^0(R(u)) + y^0(R(v)))$$

for all $j \in E$.

If $j \in E - J$, then $R(u) = R(v)$ and (6.2.7) implies

$$(6.2.9) \quad y^0(R(j)) = 1/2(y^0(R(u)) + y^0(R(v)))$$

for all $j \in E - J$.

Combining (6.2.6) with (6.2.8) and (6.2.9) we obtain

$$(6.2.10) \quad y^0(\psi(j)) + y^0(Q^0(j)) \leq y_u^0 + 1/2y^0(R(u)) + y_v^0 + 1/2y^0(R(v)) \text{ for } j \in J,$$

$$(6.2.11) \quad y^0(\psi(j)) + y^0(Q^0(j)) = y_u^0 + 1/2y^0(R(u)) + y_v^0 + 1/2y^0(R(v)) \quad \text{for } j \in E - J$$

which combined with (6.2.2) and (6.2.3) prove (6.2.4) and (6.2.5) as required.

Since y^0 was a feasible dual solution, (6.2.4) and (6.2.5) immediately imply that

$$(6.2.12) \quad y' \text{ is dual feasible.}$$

Since x^0 and y^0 satisfied the complementary slackness condition (3.5.10) we have by (6.2.1) and (6.2.5) that x^1 and y^1 also satisfy (3.5.10). We have (3.5.12) trivially satisfied because of (6.2.3).

Step 2. Now apply the algorithm described in the proof of Theorem (6.1.2) replacing b with b' so that the new matching x^2 thereby obtained will satisfy

$$(6.2.13) \quad x^2(\delta(i)) \leq b'_i \quad \text{for all } i \in V.$$

Go to Step 3.

By (6.1.3), x^2 will satisfy (3.5.10) and (3.5.12) with respect to y^1 . Moreover by (6.1.4)

$$(6.2.14) \quad \Delta(G, b'; x^2, y^1) \leq \Delta(G, b'; x^1, y^1).$$

Since x^0 and y^0 are optimal solutions to (6.0.1)-(6.0.4), we must have $\Delta(G, b; x^0, y^0) = 0$. We partition R into $R^0 \cup R^1$ where

$$R^0 \equiv \{S \in R: x^0(\delta(S)) = 0\}$$

$$R^1 \equiv \{S \in R: x^0(\delta(S)) = 1\}.$$

For each $S \in R^0$ there is a node $i(S) \in S$ such that

$x^0(\delta(i(S))) = b_{i(S)} - 1$, but for all $i \in S - \{i(S)\}$ we must have $x^0(\delta(i)) = b_i$. Since $\Delta(G, b; x^0, y^0) = 0$, we must have $i(S) \in V^S$ and $y_{i(S)}^0 = 0$. Since by (6.2.2) $y_i^1 > y_i^0$ only if $i \in S \in R$, it follows that

$$(6.2.15) \quad \Delta(G, b; x^0, y^1) \leq |R^0|.$$

For each $S \in R^1$ there is a unique edge $j(S) \in \delta(S)$ such that $x_{j(S)}^0 = 1$. Therefore by (6.2.1),

$$\Delta(G, b; x^1, y^1) \leq \Delta(G, b; x^0, y^1) + 2|R^1|$$

which together with (6.2.15) implies

$$(6.2.16) \quad \Delta(G, b; x^1, y^1) \leq 2|R|.$$

It is easily seen that

$$(6.2.17) \quad \Delta(G, b'; x^1, y^1) \leq \Delta(G, b; x^1, y^1) + ||b - b'||$$

so by (6.2.14), (6.2.16) and (6.2.17),

$$(6.2.18) \quad \Delta(G, b'; x^2, y^1) \leq ||b - b'|| + 2|R|.$$

Step 3. Apply the algorithm we described in the proof of (6.1.7) to the matching x^2 with respect to the vector b' , let x^3 be the matching thereby obtained. Go to Step 4.

Since $x_j^3 \neq 0$ only if $x_j^2 \neq 0$ for all $j \in E$ and by (6.2.5), x^3 and y^1 satisfy (3.5.10). Moreover, by (6.1.7) $\Delta(G, b'; x^3, y^1) \leq \Delta(G, b'; x^2, y^1)$ which with (6.2.18) implies

$$(6.2.19) \quad \Delta(G, b'; x^3, y^1) \leq ||b - b'|| + 2|R|.$$

Let H be any component of $G^+(x^3)$ which contains a node belonging to some $S \in R$. As before, we let

$R[S] \equiv \{T \in R : T \subset S\}$. By (6.2.1), $x_j^1 = 0$ for all $j \in \bigcup_{T \in R[S]} \delta(T) \cup \delta(S)$. Therefore by (6.1.2) and (6.1.7),

$$x_j^2 = x_j^3 = 0 \text{ for all } j \in \bigcup_{T \in R[S]} \delta(T) \cup \delta(S). \text{ Therefore } H$$

is a subgraph of $G[S] \times R[S]$. Since the edges $j \in E(G[S]) \times R[S]$ for which $x_j^0 \neq 0$ are a subset of the edges of a blossom, by (6.1.2) and (6.1.7) H is a subgraph of a blossom and so

(6.2.20) H contains no even polygons and

(6.2.21) H contains at most one odd polygon.

If H contains no node of any $S \in R$, then H is a subgraph of $G \times R$ so (6.2.20) and (6.2.21) follow from (3.8.13) and (3.8.14).

Any component H of $G^+(x^3)$ satisfies (6.1.8) so x^3 and y^1 are almost in a form appropriate for using as starting solutions to the blossom algorithm. However we may still have some components of $G^+(x^3)$ containing both a deficient node and an odd polygon. These are dealt with as follows.

Step 4. Let $R' = \emptyset$. For each component H of $G^+(x^3)$ containing both an odd polygon P and a deficient node v we do the following. Let π be the path in H from v to P . If there is no $j \in E(\pi)$ for which $x_j^3 = 1$ then we must

have $b'_v - x^3(\delta(v)) = 1$ by (6.1.9) and we let $B \in H$ and $r \in v$. Otherwise let k be the last such edge of π . If we delete k from H we are left with two subgraphs, one of which, B , does not contain v . Let $\{r\} \in \psi(k) \cap V(B)$.

In either case it is now easily seen using (6.2.3a) that B is a blossom and $x^3|E(B)$ is a np matching of B deficient at r . Moreover

(6.2.22) $H \times V(B)$ will contain a unique deficient node and no odd polygons.

Add $V(B)$ to R' .

When this process has been completed for all components containing a deficient node and an odd polygon, go to Step 5 where we perform the second phase.

It can now be easily checked that the matching x^3 , the dual solution y^1 and the nested family of (pairwise disjoint) sets R' are suitable input for the blossom algorithm.

Moreover by (6.2.19) and (6.2.1a) we have

$$(6.2.23) \quad \Delta(G; x^3, y^1) \leq ||b - b'|| + 2|R|$$

where $\Delta(G; x^3, y^1)$ is evaluated with respect to b' .

Phase 2. Execution

Step 5. Apply the blossom algorithm to G , with respect to the new vector b' of degree constraints, starting with the matching x^3 , the dual solution y^1 and the nested family R' of shrinkable subsets of V . The optimum solution x^* thereby obtained is an answer to the problem, the dual solution y^* is an optimum dual solution. This completes the

description of the algorithm.

An upper bound on the amount of work performed in step 1 is of the order $|E| + |V|$, an upper bound on the amount of work done in Step 2 is, as we have already seen, of the order $|E|$. In Step 3 we applied the algorithm described in the proof of (6.1.7). Each performance of Step 1 of this algorithm either decreased the number of deficient nodes or introduced a new edge j such that $x'_j = 0$. Thus this step can be performed at most $|V| + |E|$ times and an upper bound on the amount of work done in this step is of the order $|E| \cdot |V|$. Since each component of $G^+(x^2)$ contains at most one odd polygon, it is easily seen that the second step of this algorithm can be performed at most once for each component and so an upper bound on the amount of work performed in this step is of the order $|V|^2$. Step 4 of the post optimality algorithm has an upper bound of the order $|V|^2$.

The bound on the amount of work performed in Phase 2 is a straightforward consequence of (3.9.1) where we saw that this bound was of the order

$$\Delta(G; x, y) \cdot |E| \cdot |V|$$

where x and y are the starting solutions. By (6.2.23) therefore, an upper bound on the amount work performed in step 5 is of the order

$$(6.2.24) \quad (||b - b'|| + 2|R|) \cdot |V| \cdot |E|$$

and since the order of the bound of all previous steps of the algorithm is less than (6.2.24), it follows that a bound on the amount of work performed by the algorithm is of the

order (6.2.24). By (3.2.8) $|R| \leq 1/2(|V| - 1)$ so the total amount of work performed by this algorithm has a bound of the order $(||b - b'|| + |V|) \cdot |V| \cdot |E|$.

In Step 1 of this algorithm we eliminated having to consider R by letting $x_j^1 \equiv 0$ for all $j \in \cup_{S \in R} \delta(S)$ and defining the dual solution y^1 so that $y_S^1 = 0$ for all $S \in R$. It was this operation that introduced the term $2|R|$ in the factor $(||b - b'|| + 2|R|)$ of (6.2.24). An algorithm was developed which was basically a synthesis of the two phases of the algorithm here proposed and which attempted to make as much use of R as possible. However it was abandoned in favour of the algorithm here described for two reasons. In the first place, although the second algorithm was more efficient in certain cases, the theoretical bounds on the amount of work performed by the two algorithms were identical, namely (6.2.24). In the second place the advantage of the second algorithm was that in certain cases it was able to avoid setting $x_j^1 = 0$ for some edges $j \in \cup_{S \in R} \delta(S)$. However practical experience (see Chapter 7) indicated that the size of $|R|$ is normally very small compared to $|V|$ or $|E|$ (in graphs of 100 nodes and 500 edges, we generally had $|R| < 10$). Thus since $x^0(\cup_{S \in R} \delta(S)) \leq |R|$ where x^0 is the initial solution used by the algorithm, the gain is small when compared with the high degree of complexity of the second algorithm.

6.3 Obtaining a Nested Family.

Assume we know an optimal solution x^0 to (6.0.1)-(6.0.4)

and an optimal dual solution y^0 as produced by the blossom algorithm. Thus

$$(6.3.1) \quad x^0 \text{ is a vertex of } P(G, b)$$

(6.3.2) $R' \equiv \{S \in Q^0 : y_S^0 > 0\}$ is a nested family of subsets of V such that for each $S \in R'$, $G^=[S] \times R'[S]$ is shrinkable. ($G^=$ is the equality subgraph relative to y^0 as discussed in Section 3.8, thus $G^= \equiv (V, E^=, \psi|E^=)$

where

$$E^= \equiv \{j \in E : y^0(\psi(j)) + y^0(Q^0(j)) = c_j\}.$$

The nested family R' will in general not be suitable as input to the post optimality algorithms of this chapter, for there may be sets $S \in R'$ such that, where

$\bar{G} \equiv G[S] \times R'[S]$, there is a component H of $\bar{G}^+(x^0)$ containing more than one odd polygon or containing both an odd polygon and a node at which x^0 is deficient. Similarly if we let $\bar{G} \equiv G \times R'$, there may be components H of $\bar{G}^+(x^0)$ having these properties.

In this section, we describe a method for finding a nested family R of members of Q^0 having the properties

$$(6.3.3) \quad R' \subseteq R,$$

(6.3.4) $x^0|_{\gamma(S)}$ is a np matching of $G[S]$ for all $S \in R$,

(6.3.5) where $\bar{G}_S \equiv G[S] \times R[S]$ for all $S \in R$, every component H of $\bar{G}_S^+(x^0)$ satisfies (3.8.13)-(3.8.6)

(6.3.6) where $\bar{G} \equiv G \times R$, every component H of $\bar{G}^+(x^0)$ satisfies (3.8.13)-(3.8.16).

This family R together with x^0 and y^0 will be suitable as input for the post-optimality algorithm of this chapter.

Step 0. Initialization. Let $D \equiv \emptyset$, let $R \equiv \emptyset$. At every stage of this algorithm $R \cup R'$ will be a nested family of members of Q^0 and $D \subseteq R$ is the set of "processed" members of R' .

Step 1. If $D = R'$ then go to step 6. Otherwise, choose a minimal $S \in R' - D$. Let $\bar{G} \equiv G^-[S] \times R[S]$.

Step 2. If there is a component H of $\bar{G}^+(x^0)$ which contains an odd polygon P and a node v at which x^0 is deficient and is such that any path π in H from P to v contains an isthmus j of H for which $x_j^0 = 1$ then go to step 2a, otherwise go to step 3.

Step 2a. Let P be chosen so that a path π in H from P to v is maximal over all paths in H from odd polygons to v . Let j be the first isthmus of H in π such that $x_j^0 = 1$ and let w be the end of j furthest in H from v . Let B be the subgraph of H disconnected from v if j is removed from H . B is easily seen to be a blossom, $x^0|E(B)$ is a np matching of B deficient at w .

(6.3.7) Let $W \equiv \{i \in V: i \in V(B) \text{ or } i \in T \in V(B)$

for some $T \in R\}$. Let $\bar{R} = R \cup \{W\}$, replace R with \bar{R} , replace \bar{G} with $\bar{G}[S] \times \bar{R}[S]$.

Go to Step 2.

Step 3. If no component of $\bar{G}^+(x^0)$ contains both an odd polygon and a node at which x^0 is deficient, then go to Step 4. Otherwise let B be such a component containing a node v at which x^0 is deficient. By (6.3.1), conditions (4.5.6) and (4.5.7) of Theorem (4.5.3) and the condition of Step 2 of this algorithm, x^0 has a deficiency of 1 at v and B contains a unique odd polygon P . It is easily seen that B is a blossom and $x^0|_{E(B)}$ is a np matching of B deficient at v . Perform the operations (6.3.7) and return to Step 3.

Step 4. Now no component of $\bar{G}^+(x^0)$ contains both an odd polygon and a node at which x^0 is deficient. If no component contains more than one odd polygon then go to Step 5. Otherwise let P_1 and P_2 be odd polygons belonging to a component H of $\bar{G}^+(x^0)$ such that a path π in H from P_1 to P_2 is maximal over all paths joining odd polygons in H . By (4.5.6) there is an isthmus j of H for which $x_j^0 = 1$ occurring in π before any edge which belongs to a polygon of H , let k be the first such isthmus in π . Let B be the subgraph of H disconnected from P_2 by removing k . It is easily seen that B is a blossom in H and that $x^0|_{E(B)}$ is a np matching of B . Perform the operations (6.3.7) and return to Step 4.

Step 5. Now every component H of $\bar{G}^+(x^0)$ satisfies (3.8.13)~(3.8.16). Replace R with $R \cup \{S\}$ and replace D by $D \cup \{S\}$. Go to Step 1.

Step 6. Now we have handled every $S \in R'$. Let $\bar{G} \equiv G \times R$. All we need do is ensure that every component H of $\bar{G}^+(x^0)$ satisfies (3.8.13)~(3.8.16). Thus we apply Steps 2, 3, 4 of this algorithm to \bar{G} (replacing S with V). The resulting nested family R is the nested family required for the post optimality algorithms.

Chapter 7

A Computer Implementation of the Blossom Algorithm

In this chapter we discuss a computer implementation of the blossom algorithm we described in Chapter 3. The program was written in PL/1; the reader is assumed to have some knowledge of this programming language. (See [I2] for the language specifications). The design of the program was influenced somewhat by BLOSSOM I (Edmonds, Johnson, Lockhart [E7]), a FORTRAN implementation of a generalization of the blossom algorithm. A special acknowledgement is due to Professor Ellis L. Johnson, who has contributed to both the design and the details of this computer code.

In the next three sections we describe the data structure used and discuss the way the program handles such problems as manipulating trees and blossoms and shrinking subgraphs. Following this we discuss the code itself and in the last section of the chapter we discuss storage requirements and experimental results obtained concerning the algorithm. The program itself is listed in the Appendix.

Throughout the chapter, we refer to statements in the program by means of the PL/1 statement numbers. T and F are bit strings of length one having the values '1'B and '0'B respectively and are used as logical constants having the values "true" or "false".

7.1. Storage of the Graph.

NEDGE and NNODE are binary full words that hold the number of edges and nodes respectively of the graph G. They

do not change throughout the execution of the program, in particular they do not reflect the shrinking of subsets of nodes or the creation of pseudonodes. The edge set of the graph is the set of integers 1, 2, ..., NEDGE; the node set of the graph is a set of NNODE pointer variables which point to the structures holding the information about the nodes.

The graph is represented by an array of edges. EDGES (Statement 4) is an array of NEDGE structures which contain the following information for each edge J.

C(J) is a single precision floating point variable which holds the current "reduced cost". That is, it holds the value $c_J - y(\psi(J)) - y(Q^0(J))$ where c_J is the cost assigned to edge J and y is the current dual solution. Determining the equality subgraph and computing the bound for a dual variable change are facilitated by having this value stored. Initially C(J) should simply be the cost of the edge J; the program (Statements 333 to 346) subtracts the value of the initial dual solution while initializing.

X(J) is a binary halfword that holds the current value of the matching for the edge J.

STATUS(J) is a set of 16 one bit switches available for recording the status of edge J. Only four are used by the algorithm, they are:

EQ(J) = T or F according as J does or does not belong to the current equality subgraph;

SHRNK (J) = T or F according as J has or has not been shrunk in forming a pseudonode;

`FRST(J) = T or F according as J does or does not belong to the alternating forest or to some component of $G^+(X)$;`

`ZER(J) allows the edge J to be omitted from consideration during execution of the program. Any edge J for which $ZER(J) = T$ will be completely ignored, any edge J for which $ZER(J) = F$ will be processed normally. This feature is intended to facilitate processing of subgraphs of the graph G.`

`ENDS(J, *) and ORIGENDS(J, *) are arrays consisting of two pointers. ORIGENDS holds pointers representing the nodes of G with which J is incident and does not change throughout the execution of the algorithm. ENDS reflects any pseudonodes that have been formed. Thus, where R is the nested family of sets described in Chapter 3, if $J \in E(G \times R)$ then ENDS holds pointers to the nodes of $G \times R$ with which J is incident. If $J \notin E(G \times R)$ then the pointers in ENDS point to the pseudonode corresponding to the minimal member of R which contains $\psi(J)$.`

The variables for the real nodes of the graph are stored in an array NODELST (Statement 3). However they are referred to by means of the based structure NODE (Statement 6). Handling the nodes in this way simplifies the treatment of pseudonodes while at the same time allows the algorithm to be as economical with storage as possible.

For each node P, real or pseudo, we have the following values.

$P \rightarrow DEF$ is a binary halfword holding the deficiency of the current matching at the node P , that is, it holds the value $b_p - x(\delta(P))$ where b_p is the degree constraint of P and x is the current matching. If P is contained in a pseudonode, this value may be too large or too small by 1 however this situation is corrected when we expand the pseudonode or correct the matching within it.

$P \rightarrow STATUS$ is a set of 16 one bit switches which reflect the status of node P . Nine of these are actually used.

$P \rightarrow REAL = T$ or F according as P is a real node or a pseudonode.

$P \rightarrow CONSTEQ = T$ if the degree constraint for node P is an equation, $P \rightarrow CONSTEQ = F$ if the degree constraint for node P is an inequality. Thus $P \rightarrow CONSTEQ = T$ or F according as $P \in V^=$ or $V^<$.

$P \rightarrow DEFIC = T$ or F according as P does or does not belong to the alternating forest.

$P \rightarrow ODD = T$ if P is an odd node of the alternating forest, otherwise $P \rightarrow ODD = F$.

$P \rightarrow YRTO = T$ if P belongs to the alternating forest and the tree containing P is rooted at a real node $i \in V^<$ for which $y_i = 0$ or at a pseudonode containing a node $i \in V^<$ for which $y_i = 0$.

$P \rightarrow BLOS = T$ if the components of $G^+(x)$ containing P contains an odd polygon, otherwise it is false.

$P \rightarrow DCHNG$, $P \rightarrow INPATH$ and $P \rightarrow EXPANDED$ are all used by the algorithm and will be discussed later.

$P \rightarrow Y$ is a single precision floating point variable used to hold the current dual variable of the node P . If P is a pseudonode, then it holds the dual variable of the subset of the nodes of G which form the pseudonode.

$P \rightarrow \text{TREE}$, $P \rightarrow \text{EDGEDN}$ and $P \rightarrow \text{STACKUP}$ are used for representation of the trees and blossoms of the algorithm and their use is described in the next two sections.

7.2. Tree Handling.

The manipulation of trees and forests is an important part of the blossom algorithm. There are three properties which we wish our data structure which represents trees to satisfy. First it should provide an easy means of finding the path in the tree from any node of the tree to the root, second it should provide a reasonable means of examining all the nodes and edges of a tree and third it should make convenient such operations as rerooting trees, growing trees and removing portions of trees. The structure used is the "triply linked tree" developed by Johnson [J2]. A description of this structure also appears in Knuth [K3], p. 352.

We are actually storing a planar representation of the tree. We think of a tree being rooted "at the bottom" and consisting of various "levels" of nodes according to their distance from the root (see Figure 7.1).

For any node P of the tree other than the root, $P \rightarrow \text{DN}$ is the node adjacent with P in the level immediately below P , $P \rightarrow \text{EDGEDN}$ is the edge of the tree joining P and $P \rightarrow \text{EDGEDN}$. If P is the root of a tree then $P \rightarrow \text{DN} = \text{NULL}$ and $P \rightarrow \text{EDGEDN} = 0$.

$P \rightarrow UP$ is the leftmost node adjacent with P in the tree belonging to the level of the tree immediately above the level containing P , if such a node exists.
 Otherwise $P \rightarrow UP = \text{NULL}$.

$P \rightarrow RT$ is the first node Q to the right of P in the level of the tree containing P which satisfies $P \rightarrow DN = Q \rightarrow DN$. If no such node Q exists, then $P \rightarrow RT = \text{NULL}$. Observe that if P is the root of a tree then $P \rightarrow RT = \text{NULL}$.

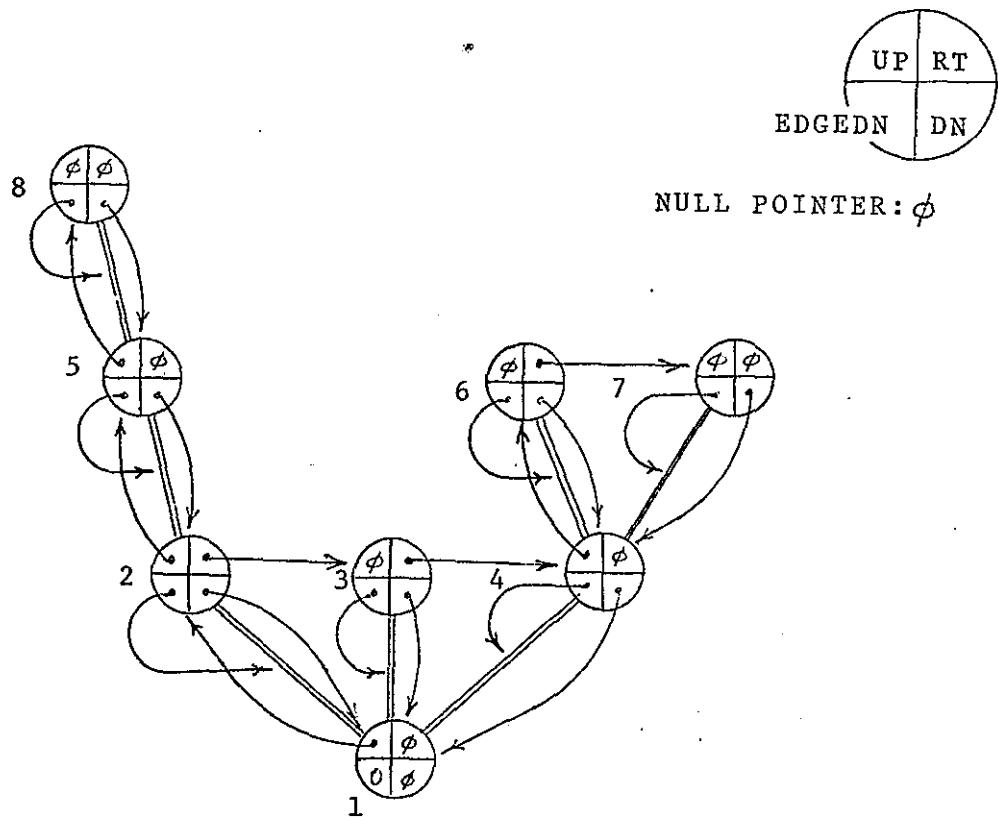


Figure 7.1 Triply Linked Tree

We now describe some of the procedures used by the program in manipulating trees.

ADDON(Q1, Q2, J) (Statements 195-202) uses the edge J which joins nodes Q1 and Q2 to attach a tree rooted at Q2 to the tree containing Q1. It also sets FRST(J) = T to indicate that J is now an edge of the forest.

REMOVE(P1) (Statements 158-175) does the following. P1 is a node belonging to some tree, REMOVE removes P1 and the portion of the subtree above P1 from the tree, thereby creating a new tree rooted at P1. If P1 is already a root, it simply returns having done nothing. Otherwise it finds the other pointers equal to P1 and modifies them appropriately. It sets P1 -> DN = NULL and sets FRST(P1 -> EDGEDN) = F, indicating this edge is no longer part of the forest.

REROOT(P1) (Statements 176-194) reroots the tree containing P1 at P1. This it does by travelling down the path in the tree from P1 to the root, successively removing the portion of the tree above each node in the path and adding that portion to the portion previously removed.

UPSCAN(P1, UPSCALL, SUBRUB, DNCALL, SUBRDN) (Statements 203-234) is a routine which scans through all the nodes of the tree containing P1 which are above P1. These nodes are scanned according to the following rule: UPSCAN always tries to move up the tree; if it cannot do this, it tries to go to the right and then continue moving up; if it cannot

do this it goes down and then tries to go to the right. For example, it would encounter the nodes of the tree of Figure 7.1 in the following order: 1, 2, 5, 8, 5, 2, 3, 4, 6, 7, 4, 1. UPSCALL and DNCALL are one bit strings, if UPSCALL = T then the first time each node is encountered, UPSCALL calls the procedure SUBRUP passing it a pointer to the node. If DNCALL = T then the last time each node is encountered the procedure SUBRDN is called and passed as a parameter a pointer pointing to the node.

Thus depending on the procedures SUBRUP and SUBRDN, UPSCAN can perform a great many functions. The procedures described in Statements 235-324 are all used by means of UPSCAN. We describe the purpose of these procedures in Section 7.5 when describing the main procedure.

The final procedure we discuss in this section is more than just a tree manipulating subroutine. It performs augmentations and at the same time (optionally) helps construct the new alternating forest.

AUGMENT(P1,R1,DELTAX,DESTROY,ODDB) (Statements 40-59)
 alternately subtracts and add DELTAX to the value of X(Q1 → EDGEDN) for each node Q1 ≠ R1 in the path from P1 to R1 in the tree containing these nodes. The pointers Q1 → DN are used to trace down the path. If ODDB (BIT(1)) equals F then the procedure starts with a subtraction, if T then it starts with an addition.

If DESTROY (BIT(1)) = T then everytime an edge becomes

zero, the portion of the tree above that edge is removed and broken into nonzero components. This is done by using UPSCAN, passing it the procedure NONDEFIX which is called the last time each node is encountered. If DESTROY = F then none of this is done.

NONDEFIX(P1) (Statements 250-258) updates the indicators for the node P1. If there is an edge J down from P1 in the tree such that $X(J) = 0$ then P1 is removed from the tree.

7.3. Blossoms, Shrinking and Pseudonodes.

One of the central problems encountered in implementing the blossom algorithm is the problem of shrinking. It has even been suggested (Balinski B[1] p. 232) that the amount of storage required to handle this process would make computer implementation of the blossom algorithm impractical. As was shown by BLOSSOM I and as is shown again by the program of this chapter, such is not the case. An upper bound on the amount of storage required to hold all the information necessary for whatever amount of shrinking is done by the algorithm is only slightly greater than half the amount of storage used to store the information required for the real nodes; in practise we generally require considerably less.

A blossom consists of a special type of alternating tree together with an edge J which forms an odd polygon; this is how it is stored. There is one node R in a

blossom at which the current matching restricted to the edges of the blossom is deficient, the tree is rooted at this node. Since R is the root of a tree, we normally have R \rightarrow EDGEDN = 0. When representing a blossom we let R \rightarrow EDGEDN = J. Thus storing a blossom is no more difficult than storing a tree.

(Components of $G^+(X)$ containing an odd polygon are also stored in this fashion, the only difference being that the root of these components is not deficient.

P \rightarrow BLOS=T for every node P in such a component.)

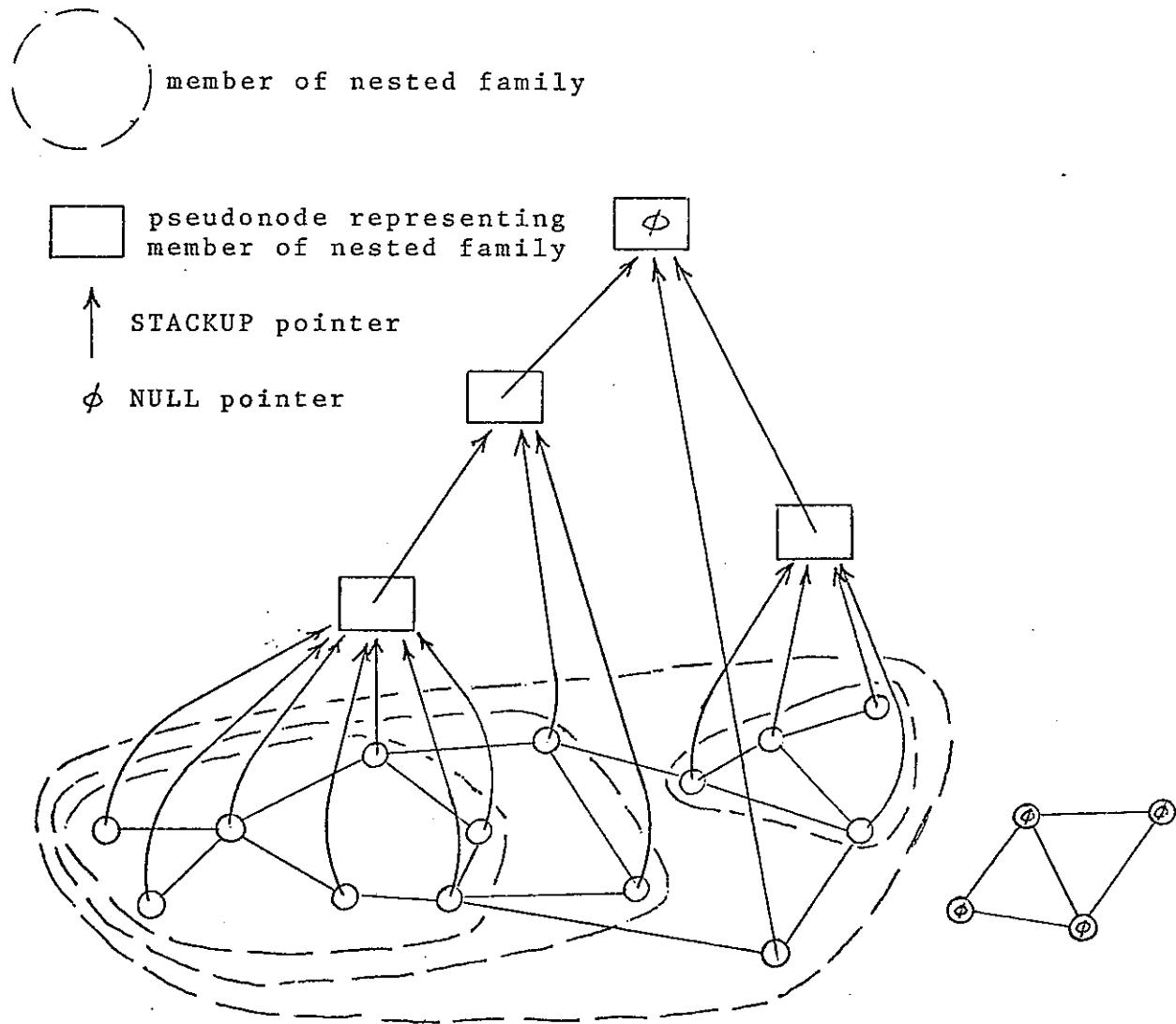
Now we describe the way in which the nested family of shrinkable sets is represented. For each member P of the nested family we have allocated (in Statements 508-516) a structure P \rightarrow PSEUDO. The first seven words of P \rightarrow PSEUDO are used in the same way as the seven words of NODE are used. (The maximal members of the noted family are the current pseudo nodes.) However there is in addition an eighth word \rightarrow ROOT which is the root of the blossom associated with P, that is, the node at which the matching restricted to the edges of the blossom is deficient.

For any real node P1, P1 \rightarrow STACKUP is a pointer to the structure associated with the minimal member of the nested family containing P1 if such a set exists, otherwise P1 \rightarrow STACKUP = NULL. For any member P of the nested family, P \rightarrow STACKUP is a pointer to the structure associated with the minimal member of the family properly containing P,

if such a set exists, otherwise $P \rightarrow \text{STACKUP} = \text{NULL}$.

(See Figure 7.2).

Figure 7.2 Nested Family Representation



Frequently we wish to know the maximal member of the nested family containing a node P , that is, the current pseudonode containing P .

SURF(P) (Statements 13-25) returns the value of the pointer corresponding to the maximal member of the nested family containing P , if such a set exists. If no such set

exists it simply returns P. In Statements 17-22 it searches up the chain of STACKUP pointers starting with $P \rightarrow \text{STACKUP}$ until a null pointer is found. It uses PNEST to count the number of members of the nested family which contain P. This value is not used by the algorithm itself, but the maximum "depth of nesting" is stored in RUNSTAT(3) to provide one indication of the amount of work done by the algorithm.

We now describe the operations performed by the program in shrinking a blossom (Statements 506-558). Figure 3.7 may help to clarify this process. J is an edge joining nodes P1 and P2 which are both even nodes (or in some cases odd nodes) of the same alternating tree. R3 is the first common node of the paths in the tree from P1 and P2 to the root of the tree. R2 is the last node belonging to the blossom in the path in the tree from R3 to the root of the tree. (These nodes have been determined earlier in the program.) We call the path from R3 to R2 the stem of the blossom. The blossom consists of the polygon plus the stem plus any components of $G^+(X)$ containing a node of the polygon or stem.

In Statements 508-516 we allocate the pseudonode P for the blossom and initialize most of its variables. In Statements 519-534 we "mark" the polygon and the stem of the blossom by letting $P3 \rightarrow \text{INPATH} = T$. This is to make it possible to identify all the nodes of the blossom. Then

in Statement 537 we call UPSCAN, passing it the routines UPBLOSS and DNBLOSS. These routines (Statements 296-329) do two things. UPBLOSS set $P_1 \rightarrow \text{STACKUP} = P$ for every node of the blossom. DNBLOSS removes any portions of the tree above the blossom and adjoins them to P . These routines rely on the order in which UPSCAN scans the nodes. SHRNKNG is a one bit switch which is T or F according as the next node to be scanned can or cannot be expected to be part of the blossom. Thus whenever UPBLOSS detects a node P_1 not in the polygon or stem for which $X(P_1 \rightarrow \text{EDGEDN})=0$, PX is set equal to P_1 and SHRNKNG is set equal to F. From then on nothing is done to successive nodes until DNBLOSS is passed the node PX . Then the subtree rooted at PX is removed from the blossom and adjoined to P , SHRNKNG is set T and the process continues.

If SHRNKNG=T then when UPBLOSS is passed a node of the polygon or stem or a node P_1 for which $X(P_1 \rightarrow \text{EDGEDN}) \neq 0$ it sets $P_1 \rightarrow \text{STACKUP} = P$, indicating that this node is part of the blossom.

When UPSCAN has completed its scan the program removes the blossom from the tree and replaces it with the pseudonode P , together with any portions of the tree that DNBLOSS may have adjoined to P (Statements 538-543). Finally, $R2 \rightarrow \text{EDGEDN}$ is set equal to J and the blossom is represented completely.

Now all that remains to be done is to update ENDS to reflect the new pseudonode. This is done in Statements 545-556. At the same time SHRNK(J1) is set equal to T

for any edge such that ENDS(J1,1) and ENDS(J1,2) are nodes of the blossom.

7.4. Parameters Passed and Returned.

In this section we describe the parameters passed and returned when using this code. It should be pointed out that this program is designed to be used by other programs as a subroutine and consequently there is no provision for card input or printer output (except for the trace feature). Thus unlike BLOSSOM 1 this program requires a suitable driver program to prepare its data and output its results if we simply want to solve matching problems.

Parameters Passed.

*NEDGE (BINARY FIXED (16)) holds the number of edges of the graph.

*NNODE (BINARY FIXED (16)) holds the number of nodes of the graph.

*NODELST is an array of NNODE structures having the format described in NODE (Statement 6). Each structure is seven words long and holds the following information. Let $I \in \{1, 2, \dots, NNODE\}$ and let $P = ADDR(NODELST(I))$.

- $P \rightarrow$ DEF(BINARY FIXED (15)) holds the degree constraint of the node P .

- $P \rightarrow$ CONSTEQ(BIT(1)) should be T or F according as the constraint at node P is an equation or an inequality.

- $P \rightarrow$ REAL, $P \rightarrow$ DEFIC (BIT(1)) should be set equal to T initially.

- P -> DCHNG, YRTO, INPATH, EXPANDED, ODD and BLOS (BIT(1)) should initially all be set equal to F.

- P -> Y (DECIMAL FLOAT (SHORT)) holds the initial dual variable of the node. This dual solution must be feasible. If P -> Y is set equal to half the absolute value of the largest edge cost for every node P then this starting dual solution is feasible.

- P -> TREE.UP, RT, DN and STACKUP(POINTER) should all be set equal to NULL.

- P -> EDGEDN (BIN FIXED(16) should be initially zero.

*EDGES is an array of NEDGE structures, each six full words long holding the following information. Let $J \in \{1, 2, \dots, NEDGE\}$.

- C(J) (DECIMAL FLOAT (SHORT)) is the cost of edge J (not the "reduced cost; this is computed by the algorithm").

- X(J) (BINARY FIXED(15)) should be set to zero.

- ZER(J), EQ(J), SHRNK(J) and FRST(J) should all be initially set to F.

- ENDS(J,*), ORIGENDS(J,*) (POINTER) should be the nodes of the graph with which J is incident.

* RUNSTAT (Statement 5) is an array of 10 binary halfwords. The only entry used for input is RUNSTAT(10). A trace of the execution of the program is or is not printed out according as RUNSTAT(10) = 1 or 0. This trace, if obtained, prints a message concerning each edge used by the algorithm together with the values of the matching and the dual solution any time they are changed.

The main use of RUNSTAT is to return statistics concerning the problem solved to the program which invoked BLOSSOM.

These input specifications were based upon the assumption that we were using the zero matching as a starting solution.

Parameters Returned.

The parameters are returned in the following state.

Let P be any node, real or pseudo.

- $P \rightarrow \text{DEF}$ is the deficiency of the current matching X at the node P .

- $P \rightarrow \text{STATUS}$ is set to reflect the status of P at termination.

- $P \rightarrow \text{TREE}$, EDGEDN holds the tree and blossom structure of the final solution.

- $P \rightarrow \text{STACKUP}$ points to the pseudonode representing the minimal member of the nested family containing P , if such a set exists, otherwise it is null.

- $P \rightarrow Y$ is the final dual variable of the node P if P is real, or the final dual variable of the corresponding member of the nested family if P is a pseudo node.

Notice that the invoking program is returned both the optimal dual solution and the final nested family. (This was desired in Chapter 6.)

Let J be any edge of the graph.

- $C(J)$ is the final reduced cost of the edge J .

- $X(J)$ is the maximum matching, that is, the answer to the problem.

- STATUS(J) reflects the status of edge J at termination.

- ENDS(J,*), ORIGENDS(J,*) are both as they were originally, the nodes of the graph met by J.

*RUNSTAT (BINARY FIXED (15)) (Statement 5) is an array of ten indicators showing the amount of work done in solving the problem. The values returned are as follows:

- RUNSTAT(1) is the number of dual variable changes;
- RUNSTAT(2) is the number of times a blossom was shrunk;

- RUNSTAT(3) is the deepest nest of pseudonodes formed (or equivalently, the longest chain of STACKUP pointers);

- RUNSTAT(4) is the number of times pseudonodes were expanded (in Step 9e of the blossom algorithm);

- RUNSTAT(5) is the number of times the forest was grown without making an augmentation (Steps 3a and 7 of the blossom algorithm);

- RUNSTAT(6) is the number of two tree augmentations (Step 4 of the blossom algorithm);

- RUNSTAT(7) is the number of one tree augmentations (Step 5b of the blossom algorithm);

- RUNSTAT(8) is the number of times a component of $G^+(X)$ containing an odd polygon was added to the forest (Step 3c of the blossom algorithm);

- RUNSTAT(9) is the number of so called "pseudo augmentations", augmentations which move a deficiency to a node i of V^S for which $y_i = 0$ (Step 7a of the blossom algorithm);

- RUNSTAT(10) is returned with the value zero or one according as the matching returned is or is not feasible, if RUNSTAT(10) = 1 when returned then the algorithm terminated with a Hungarian forest.

7.5. The Main Procedure.

Now we describe the main procedure itself. The code follows fairly closely the description of the blossom algorithm given in Section 3.8.

Statements 325-349 are for initialization, reduced costs are computed and EQ(J) is set for each edge J reflecting whether or not the edge belongs to the equality subgraph. A procedure FN(J) (Statements 26-39) is used in computing reduced costs. It calculates the sum of the dual variables of the ends of J and of all members of the nested family which contain both ends of J.

Statements 350-634 constitute the "edge processing" loop of the program. JCNT is used to cycle through the edges. Anytime we finish considering an edge, whether or not we have been able to make use of it, we go to ENDA (Statement 633) where JCNT is set equal to $1 + \text{MOD}(\text{JCNT}, \text{NEDGE})$.

Whenever we are able to use the edge JCNT (to augment, grow the forest or shrink), LASTJ is set equal to JCNT. If JCNT ever "catches up" with LASTJ then we have made a complete cycle through the edges without having been able to do anything so we go to Statement 636 and attempt to change the dual variables.

Statements 350-372 test each edge JCNT to see if it belongs to the equality subgraph, has not been shrunk, is not in the forest and meets an even node P1 of the alternating forest F¹. If JCNT violates any of these criteria we go to ENDA. If the other end P2 of JCNT is an odd node of F¹ then it is of no use to us and we go to ENDA, if P2 is an odd node of F⁰ then we go to ODDGROW (Statement 581). Otherwise we set J=JCNT and go to POLYSTEP(Statement 566), GROWSTEP(Statement 559) or DXCALC(Statement 382) depending on the status of P2.

DXCALC:(Statement 382) J joins even nodes P1 and P2 of the forest F, in Statements 382-400 we determine whether they belong to the same or to different trees. At the same time we compute D1 and D2, bounds on the amount of augmentation that can be made. INPATH is used to mark the nodes in the paths from P1 and P2 to the roots of their respective trees. If P1 and P2 belong to different trees then R1 and R2 are the roots of the two trees. If P1 and P2 belong to the same tree then R1 is the root of the tree and R2 is the first common node of the two paths.

If P1 and P2 belong to different trees, then we go to TWOTREE (Statement 401) where we perform the augmentation described in Step 4b of the blossom algorithm compute the new forest and go to ENDA.

If P1 and P2 belong to the same tree then we go to ONETREE (Statement 427). There we determine whether or not an augmentation is possible. If not we go to DEFBLOSS

(Statement 506) where we shrink. If we can make an augmentation we do so, update the tree and then update the forest. At this point we may have to shrink, if so we go to DEFBLOSS where we do so. We may have created a component of $G^+(X)$ containing an odd polygon and no deficient node. If so (Statements 497-505) we find the root R2, store the polygon forming edge J as R2 \rightarrow EDGEDN and call UPSCAN passing it the procedure BLOSSIND. BLOSSIND (Statements 259-264) simply sets the node STATUS indicators correctly.

We have already discussed the shrinking procedure in Section 7.3.

GROWSTEP: (Statements 559-565) J joins an even node P1 of the forest to a node P2 not in the forest. We simply grow the forest. ADDFIX (Statements 235-244) sets the STATUS indicators for the nodes added to the forest. (This corresponds to Step 3b of the blossom algorithm.)

POLYSTEP: (Statements 566-580) J joins an even node P1 of the forest to a node P2 of a component of $G^+(X)$ which contains an odd polygon. First we find the root of this component and hence the polygon forming edge J1. Then we add this component (minus J1) to the forest just as in GROWSTEP. Then we replace J with J1, P1 and P2 with the ends of J1 and go to DXCALC(Statement 382). (This corresponds to Step 3c of the blossom algorithm.)

ODDGROW: (Statements 581-632). Edge J meets an even node P1 of F^1 and an odd node P2 of F^0 . Statements 585-597 add a suitable portion of the tree containing P2 to the tree containing P1. SETYRTO is a procedure used by UPSCAN

to set $P \rightarrow YRTO = YROOTO$ for all nodes scanned. Thus we first set YROOTO correctly. (This corresponds to Step 7 of the blossom algorithm.)

We may now have a tree in the forest containing two deficient nodes P_1 and P_2 . If this is the case, we make a so called "pseudo augmentation" to remedy this (Statements 599-632). These steps also update the forest. (This portion of the code corresponds to Step 7a of the blossom algorithm.)

This completes the description of the main edge processing loop. If we make a complete cycle through the edges without being able to make use of any edge then we go to DUALCHNGE(Statement 636) where we attempt to change the dual variables. FAIL is a one bit switch which is used to indicate whether or not we have an optimal feasible matching. Initially FAIL=F, if we discover a node in a tree of F^1 then FAIL is set equal to T.

In Statements 637-665 we compute EPS2, a bound imposed by the nodes on the amount of dual change that can be made. (EPS2 equals the minimum of ϵ_3, ϵ_4 of Step 9a of the blossom algorithm.)

If FAIL=F, thus the current matching is feasible, we go to CORRECTION(Statement 925) where we correct the matching in the pseudo nodes. If EPS2=0 then we need make no dual variable change; we go to NODEBND(Statement 786) where we either reroot a tree or expand an odd pseudonode of the forest.

Otherwise (Statements 670-705) we compute EPS1, the

bound on the amount of dual change determined by the dual constraints corresponding to the edges. (ESP1 equals the minimum of ϵ_1, ϵ_2 of Step 9a of the blossom algorithm.) Then we let EPS = MIN(EPS1, EPS2). If EPS = 10^{10} (infinity for our purposes) then the forest is Hungarian, no feasible matching exists, we go to CORRECTION and terminate. Otherwise (Statements 706-780) we make a change of dual variables and update the reduced costs accordingly. (For any pseudo node P, $P \rightarrow DCHNG$ is used to ensure that we only change its dual variable once.) If the bound on the dual change was imposed by a constraint corresponding to an edge JX then we can now immediately make use of the edge; we set JCNT equal JX and return to the start of the edge processing loop. (Statement 350).

If the bound on the dual change was imposed by a constraint corresponding to a real node PX of the forest, then we now have $PX \rightarrow Y = 0$. If PX is the root of the tree, we simply reset YRTO for the nodes of the tree. Otherwise we go to AUG(Statement 599) and make a pseudo augmentation. This process corresponds roughly to Step 9d of the blossom algorithm, although in the computer code we do not insist that all trees of F^1 rooted at nodes $i \in V$ for which $y_i = 0$ be moved to F^0 , we simply handle one each time.

If the bound on the dual change was imposed by a constraint corresponding to a pseudonode PX, then $PX \rightarrow Y = 0$ and PX is an odd pseudonode of the forest that has to be

expanded. This we do in Statements 797-924.

The first thing done is to call EXPAND, a procedure (Statements 60-145) that first updates ENDS so as to no longer reflect the existence of pseudonode P and then "corrects" the matching within the blossom corresponding to P so that it is compatible with the matching of the graph containing P. This procedure also forms the nucleus of the final matching correction step (corresponding to Step 12 of the blossom algorithm). Notice that for any calls to AUGMENT in EXPAND we have DESTROY = F, thus the blossom does not have its structure destroyed.

EXPAND set JIN equal to the edge J of the graph incident with P for which X(J) = 1 and sets BROOT equal to the node of the blossom met by JIN. If P \rightarrow EDGEDN = JIN then we have the easier case, JIN is the unique edge of the forest meeting P. This case is handled in Statements 803-842. Otherwise two edges of the forest meet P, this case is handled in Statements 843-924.

ADDBLOS,DEFFIX(Statements 265-295) are routines called by UPSCAN to "unshrink" a blossom and update the status indicators. Their operation is similar to that of UPBLOSS, DNBLOSS. Initially SHRNKNG(BIT(1)) = T. For each node P1 that ADDBLOS is passed, it sets P1 \rightarrow STACKUP = NULL, thereby removing the reference to the pseudonode. Then it proceeds, setting the status of each node to indicate that it belongs to the forest, until it finds a node P1 which would have become an even node of the forest for which

X(P1 EDGEDN) = 0. When this happens PX is set equal to P1 and SHRNKNG is set equal to F. From then on the status of each node encountered is set to indicate that the node does not belong to the forest. DEFFIX breaks the blossom up at edges J for which X(J) = 0. When DEFFIX is passed the node PX it sets SHRNKNG = T and the process continues.

The final part of the program is the step (Statements 925-945) where we correct the matching in the pseudo nodes prior to terminating. For each pseudonode P, P → EXPANDED is used to ensure that we do not try to correct the matching for the pseudonode more than once.

This completes the description of the program.

7.6. Experimental Results

This program was compiled under version 5.2B of the OS/360 PL/I F level compiler, OPT=1 and was tested on a large number of contrived graphs. Then a series of tests on "random graphs" was run to obtain the experimental results described here.

The random graph generator accepted as input the number of nodes and edges desired in the graph together with a range for the degree constraints and a range for the edge costs (integers were used for edge costs in these tests). It generated the graph by successively joining each node of the graph to some other node until sufficient edges had been created. Thus the test graphs had multiple edges but no loops. The random graph generator also accepted

a parameter specifying the desired probability of a node belonging to V^+ .

An option of the random graph generator was to create a file containing the information about each graph in a form suitable as input to BLOSSOM I, the earlier Fortran implementation of the matching algorithm. This enabled comparative tests to be run between the two programs.

The driver program then invoked BLOSSOM to solve the matching problem. Following this a test was made of the matching and dual solution returned by BLOSSOM to ensure that they were feasible solutions satisfying the complementary slackness conditions for optimality.

The results of these tests are listed in Table 7.1. They were run on an IBM/360 model 75 at the University of Waterloo. Thirty two graphs were run on both BLOSSOM I and the code described here, two random graphs were generated with each set of specifications. We required the degree constraint be satisfied with equality at each node. In addition, six "large" graphs were run on the code of this chapter.

One of the most striking observations is that even though the value of the edge costs do not enter into our theoretical bound, the number of different edge costs drastically affects the run time of the code. The reason for this seems to be that the more different edge costs we have, the more dual variable changes that have to be done to obtain an optimal solution, and dual variable changes are practically (although not theoretically) time consuming.

A second observation is that the number of pseudos formed during the course of execution of the code tends to be relatively small. The entries in the table give the total number formed during the execution of the code, the number present at termination is often considerably smaller.

The BLOSSOM program of this chapter does run faster than BLOSSOM I (The ratio of its execution time to that of BLOSSOM I seems to decrease as the number of edges of the graph increases). This is not surprising, however, for BLOSSOM I treats directly a more general form of the matching problem than is treated by the code of this chapter. These more general problems can be reduced to problems solvable with the code of this chapter; however this involves significantly, though algebraically, increasing the number of edges and nodes.

The BLOSSOM procedure itself requires 33K bytes of storage. Storage of the graph requires $28 \times v + 24 \times e + 32 \times p$ bytes of storage, where v is the number of nodes, e is the number of edges and p is the maximum number of pseudonodes present at any one time in the execution. The various PL/1 library routines required to run BLOSSOM add to these storage estimates however. When run with the random graph generator and driver, the [100 node-1000 edge] graphs required 148K bytes of storage, the [1000 node-4000 edge] graphs required 238K bytes of storage.

Since the computer code uses fixed word arithmetic, it may not be able to solve a problem if the number of significant digits of some of the values used becomes too large.

The degree constraints and values of the matching are integers stored as binary half words and so can be no larger than 32767. The value $X(J)$ for any edge J can never be larger than the smaller degree constraint of its ends, so as long as the degree constraints range from 1 to 32767 we will have no difficulty handling these values.

The edge costs and dual variables are stored as hexadecimal (base 16) floating point numbers having six significant hexadecimal digits. Any edge costs stored by the computer are rational numbers. If we multiply all edge costs by a positive constant we do not affect the solution set of the problem. Hence we can assume that the edge costs have been multiplied by a large enough number so that they are all integer. As was shown in the proof of (3.10.7) if our starting dual variables are integer valued then all dual variables computed during the execution of the algorithm will be integer or half integer valued. If the degree constraint of every node is an inequality (that is, $V^{\leq} = \phi$ and $V^{\leq} = V$) then all dual variables are nonnegative and so no dual variable needs to be larger than the largest edge cost. Thus if the edge costs are integers from the range - 1,048,576 to 1,048,576 then the dual variables will be integers and half integers from the same range. These numbers are represented exactly by six hexadecimal digits so we can be sure that the computer code will solve such problems.

In the case that $V^{\leq} \neq \phi$, and consequently some dual variables are allowed to become negative, we may in fact require dual variables considerably larger than the largest edge cost. Consequently the establishment of a bound on the magnitude of

the dual variables is more complicated. For an analysis of a situation of this sort, see Edmonds, Johnson, Lockhart [E7].

If higher precision were required for some problem it would be a straightforward matter to replace all binary half words with full words and all floating point numbers with double precision floating point numbers. Then degree constraints could range from 1 to 2,147,483,648 and if the edge costs were integers from the range -4×10^{15} to 4×10^{15} we could guarantee a correct solution.

TABLE 7.1. TESTS OF BLOSSOM PROGRAM

No. of Nodes	No. of Edges	Range of b_i	Range of c_j	Elapsed Time	Blossom I Elapsed Time	No. of Shrinkings	No. of Dual Variable Change
30	150	1	1-1000	4.9, 6.9 sec.	6.5, 9.2 sec.	1,5	28,39
30	150	1	1-5	0.5, 0.7	1.1, 1.3	0,0	2,3
30	500	1-7	1-500	22.6,19.0	29.4,25.6	2,3	39,33
30	500	1-77	1-500	25.5,24.0	32.0,32.1	6,5	44,41
50	200	1-10	1-10	3.7, 4.6	5.5, 6.2	4,3	10,15
50	500	1-10	1-10	5.8, 5.8	12.0,11.2	5,5	6,7
50	200	1-100	1-10	3.2, 5.7	4.8, 7.4	0,5	10,18
50	200	1-10	1-9999	23.8,16.7	29.4,19.5	11,6	96,66
100	300	1-2	1-10	10.6, 8.1	24.4,23.7	27,19	16,16
100	1000	1-2	1-10	20.3, 8.8	54.7,15.9	26, 6	9,5
100	300	50-150	1-10	9.7,11.5	13.4,16.7	8,13	16,19
100	300	1-2	1-9999	50.5,45.5	62.3,53.8	11, 7	138,124
300	1500	1	1-10	38.1,21.6	65.9,39.8	11, 6	13,7
300	1500	7-77	1-10	44.0,51.3	102.6,112.4	15,13	18,12
300	1500	2	1-100	135.1,125.6	182.4,172.9	0,0	68,70
300	1500	100	1-10	20.5,18.4	36.8,31.9	0,0	8,7
** 500	5000	7	1-10	137.0,123.4		31,28	5,6
** 500	5000	1	1-10	84.2,177.4		36,61	3,5
* 1000	4000	1-2	1-10	143.5,184.8		11,33	14,14

** Run with all nodes i $\in V^{\leq}$.* Run with half nodes in V^{\leq} .

APPENDIX

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

1 BLOSSOM: PROC(NFEDGE,NNODE,NODELST,EDGES,RUNSTAT);

/****** VARIABLE DECLARATIONS *****

2 1 DCL (NFEDGE,NNODE) BIN FIXED(16);
3 1 DCL 1 NODELST(* /*NNODE*/), /* ACTUAL STORAGE FOR NODE VARS */
 2 FILL(7) BIN FIXED(16);
4 1 DCL 1 EDGES(* /*NEDGE*/),
 2 C FLOAT,
 2 X BIN FIXED(15),
 2 STATUS,
 3 FILL BIT(12),
 3 ZER,
 3 EQ,
 3 SHRINK,
 3 FRST) BIT(1),
 2 ENDS(2),
 2 ORIGENDS(2)) PTR;
5 1 DCL RUNSTAT(10) BIN FIXED(15);
 /* RUNSTAT(1)=NO. OF DUAL CHANGES,
 RUNSTAT(2)=NO. OF SHRINKINGS,
 RUNSTAT(3)=DEEPEST NEST OF PSEUDONEEDGES FORMED,
 RUNSTAT(4)=NO. OF EXPANSIONS,
 RUNSTAT(5)=NO. OF TIMES FOREST GROWN,
 RUNSTAT(6)=NO. OF TWO TREE AUGMENTATIONS,
 RUNSTAT(7)=NO. OF ONE TREE AUGMENTATIONS,
 RUNSTAT(8)=NO. OF TIMES POLYGON ADDED TO THE FOREST,
 RUNSTAT(9)=NO. OF PSEUDO AUGMENTATIONS,
 RUNSTAT(10)= 0 IF MATCHING IS FEASIBLE,
 1 IF MATCHING IS NOT FEASIBLE.
 RUNSTAT(10) IS PASSED WITH VALUE 0 IF NO TRACE IS DESIRED,
 WITH VALUE 1 IF A TRACE IS REQUIRED. */
6 1 DCL 1 NODE BASED(P),
 2 BASICS,
 3 DEF BIN FIXED(15),
 3 STATUS,
 4 FILL BIT(7),
 4 REAL,
 4 DCHNG,
 4 YRTO,
 4 INPATH,
 4 EXPANDED,
 4 CONSTEQ,
 4 ODD,
 4 DEFTC,
 4 BLOS)BIT(1),
 3 Y FLOAT,

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
          3 TREE,
          (4 UP,
          4 RT,
          4 DN) PTR,
          3 EDGEON BIN FIXED(16),
          3 STACKUP PTR;
7      1     DCL  1 PSEUDO BASED(P),
          2 BASICS(7) BIN FIXED (16),
          2 ROOT PTR;
8      1     DCL((EPS,FPS1,EPS2,Z) FLOAT, JX BIN FIXED(16)) STATIC;
9      1     DCL SURF ENTRY RETURNS(PTR);
10     1     DCL ((P,P1,P2,P3,R1,R2,R3,PX,BROOT,Q1,Q2,Q3)PTR,
          (I,J1,J2,K,JCNT, LASTJ,JIN,JDN) BIN FIXED(16),
          DELTAX BIN FIXED(15),
          (D1,D2,D3) BIN FIXED(15))STATIC;
11     1     DCL (T INIT ('1'B), F INIT ('0'B)) STATIC BIT (1),
          (ODDB , POLYBIT,SHRNKNG,YROOT0,TRACE,FROMEX,NOCHECK,FAIL)
          STATIC BIT(1);
12     1     FMT: FORMAT(SKIP,A,F(6),A); /* USED FOR TRACING */

/***** GENERAL PURPOSE SUBROUTINES *****/
***** SURF: PROC (P) RETURNS(PTR);
/* PROCEDURE TO FIND HIGHEST LEVEL PSEUDO NODE CONTAINING P. */
/* PNEST IS USED TO COUNT STACK DEPTH. */
14     2     DCL (P,P1 STATIC) PTR;
15     2     DCL PNEST BIN FIXED(15) STATIC;

16     2     PNEST=0;
17     2     P1=P;
18     2     DO WHILE (P1->STACKUP ~=NULL);
          P1=P1->STACKUP;
19     2     1     PNFST=PNEST+1;
20     2     1     END;
21     2     1     IF PNFST > RUNSTAT(3) THEN RUNSTAT(3)=PNEST;
22     2     RETURN(P1);
23     2
24     2
25     2     END SURF;
```

/*THE BLOSSOM ALGORITHM; MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
26   1      FN:PROC(EDGE);
          /* THIS PROCEDURE EVALUATES THE SUM OF THE DUAL VARIABLES
           ON EACH END OF AN EDGE AND ON ODD SETS CONTAINING
           THE EDGE.*/
27   2      DCL (P1,P2) STATIC PTR, EDGE BIN FIXED(16), SUM STATIC FLOAT;
28   2      P1=ORIGENDS(EDGE,1);
29   2      P2=ORIGENDS(EDGE,2);
30   2      SUM=P1->Y + P2->Y;
31   2      IF ~SHRMK(EDGE) THEN GO TO END;
33   2      P1=ENDS(EDGE,1);
34   2      DO WHILE (P1~NULL);
35   2      1      SUM=SUM+P1->Y;
36   2      1      P1=P1->STACKUP;
37   2      1      END;
38   2      END:RETURN(SUM);
39   2      END FN;

40   1      AUGMENT:PROC(P1,R1,DELTAX,DESTROY,ODDB);
          /* THIS PROCEDURE AUGMENTS ALONG THE PATH FROM P1 TO THE ROOT R1
           BY AMOUNT DELTAX. IF DESTROY = T THEN THE TREE GETS BROKEN
           UP AT EDGES J FOR WHICH THE NEW X(J) = 0, IF DESTROY = F
           THEN THIS DOES NOT HAPPEN. WE START AUGMENTING WITH AN
           ADDITION OR A SUBTRACTION DEPENDING ON WHETHER ODD = T OR F.*/
41   2      DCL (P1,R1) PTR,DFLTAX BIN FIXED(15),(DESTROY,ODDB) BIT(1),
          (Q1,Q2) STATIC PTR;

42   2      Q1=P1;
43   2      DO WHILE (Q1~R1);
44   2      1      Q1->INPATH=F;
45   2      1      J1=Q1-> EDGEON;
46   2      1      Q2=Q1->DN;
47   2      1      IF ODDB THEN X(J1)=X(J1)+DELTAX;
49   2      1      ELSE DO;
50   2      2      X(J1)=X(J1)-DFLTAX;
51   2      2      IF DESTROY THEN
52   2      2      1      IF X(J1)=0 THEN CALL UPSCAN(Q1,F,NONDEFIX,T,NONDEFIX);
          /* THIS REMOVES EVRYTHING ABOVE AND SPLITS TREE
           INTO ITS POSITIVE COMPONENTS. */
54   2      2      END;
55   2      1      Q1=Q2;
56   2      1      ODDB=~ODDB;
57   2      1      END;
58   2      2      R1->INPATH=F;
59   2      END AUGMENT;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
60      1      EXPAND:PROC (P);
              /* THIS PROCEDURE EXPANDS A PSEUDONODE.
SPECIFICALLY IT
              1) CORRECTS EDGE ENDS SO THAT THEY NO LONGER REFLECT
                 EXISTENCE OF PSEUDONODE
              2) AUGMENT SO THAT MATCHING CORRECT BUT STACKUP STILL
                 ACKNOWLEDGES PSEUDONODE */
61      2      DCL P PTR, DESTROY BIT(1),((P1,P2) PTR, (I,J) BIN FIXED(16), IN BIT(1))
              STATIC;

62      2      DESTROY=F;
              /* CORRECT EDGE ENDS */
63      2      JIN=0; /* JUST IN CASE THERE IS NO EDGE IN WITH X(J) =1. */
64      2      DO J=1 TO NEDGE;
              IN=F; /* INDICATES PARITY OF NO. OF EDGE ENDS IN P. */
65      2      1      IF ENDS(J,1)=P THEN GO TO P2TEST;
66      2      1      IN=T;
67      2      1      SHRNK(J)=F;
68      2      1      P1=ORTGENDS(J,1);
69      2      1      LPP1: IF P1 -> STACKUP=P THEN DO;
70      2      1              ENDS (J,1)=P1; GO TO P2TEST;
71      2      1              END;
72      2      1              P1=P1 -> STACKUP; GO TO LPP1;
73      2      2      P2TEST: IF ENDS (J,2) =P THEN GO TO LPEND;
74      2      2      IN=NOT IN;
75      2      2      P2=ORTGENDS (J,2);
76      2      1      LPP2: IF P2 -> STACKUP=P THEN DO;
77      2      2              ENDS (J,2)=P2;
78      2      2              GO TO LPEND; END;
79      2      2              P2=P2->STACKUP; GO TO LPP2;
80      2      1      LPEND: IF IN THEN
              IF X(IJ)=1 THEN /* THIS IS THE EDGE INTO THE BLOSSOM*/
              JIN=J;
81      2      1              END;
82      2      1              IF JIN=0 THEN DO; /*CHECK FOR <= NODE WITH Y=0 */
83      2      1                  DO I = 1 TO NNODE;
84      2      2                  P1=ADDR(NODELST(I));
85      2      2                  IF P1->CONSTEQ THEN GO TO ENDSCH;
86      2      2                  IF P1->Y = 0 THEN GO TO ENDSCH;
87      2      2                  /* OTHERWISE WE SEE IF P1 IS CONTAINED IN P. */
88      2      2                  BROOT=P1;
89      2      2                  DO WHILE(BROOT->STACKUP != NULL);
90      2      2                      IF P=BROOT->STACKUP THEN GO TO LAB7;
91      2      2                      /* BROOT IS A SUITABLE NODE FOR RECEIVING A DEFICIENCY*/
92      2      2                      BROOT=BROOT->STACKUP;
93      2      2                      END;
94      2      1      ENDSCH:FND;
              /* P CONTAINS NO <= NODE WITH Y=0, SO CURRENT MATCHING O.K.*/
95      2      1      RETURN;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
109      2   1      LAB7:R1=P->ROOT;
110      2   1      IF BROOT=R1 THEN RETURN; /* CURRENT MATCHING IS CORRECT */
112      2   1      R1->DEF=0;
113      2   1      BROOT->DEF=1;
114      2   1      P1=BROOT;
115      2   1      GO TO AGMNT; /* MATCHING CORRECTION SET UP */
116      2   1      END;
117      2           /* ONE END OF JIN HAS STACKUP=P, THE OTHER DOES NOT. LET
118      2           P1 BE THAT NODE */
119      2           P1=ENDS (JIN,1); IF P1-> STACKUP=P THEN
120      2           P2=ENDS (JIN,2);
121      2           ELSE DO; P2=P1; P1=ENDS (JIN,2); END;
122      2           /* NOW P1 IS THE SURPLUS NODE */
123      2           BROOT=P1; /* VARIABLES RETURNED TO BLOSSOM */
124      2           R1=P->ROOT;
125      2           R1->DEF=0; /* WE WILL CLEAR UP THIS DEFICIENCY */
126      2           IF P1 = R1 THEN RETURN;
127      2
128      2   1      AGMNT:
129      2           DELTAX=1;
130      2           ODDB=F; /* START WITH SUBTRACTION */
131      2           CALL AUGMENT (P1,R1,DELTAX,DESTROY,ODDB);
132      2           IF NOT ODDB THEN /* WE WENT CORRECT DIR'N AROUND POLYGON */
133      2           RETURN;
134      2           J=R1->EDGEON;
135      2           P1=ENDS (J,1);
136      2           P2=ENDS (J,2);
137      2           ODDB=F; /* NORMAL CASE */
138      2           IF P1->ODD THEN ODDB=T; /* ARNORMAL CASE */
139      2           CALL AUGMENT (P1,R1,DELTAX,DESTROY,(ODDB));
140      2           CALL AUGMENT (P2,R1,DELTAX,DESTROY,(ODDB));
141      2           X(J)=X(J)+DELTAX;
142      2           IF ODDB THEN X(J)=X(J)-2*DELTAX; /* CORRECT A BAD GUESS */
143      2           END;
144      2
145      2   1      XOUT:PROC; /* PRINTS CURRENT SOLUTION */
146      2           PUT EDIT('*BLOSS - CURRENT MATCHING :')(SKIP,A);
147      2           PUT EDIT(X)(SKIP,20 F(5));
148      2           END XOUT;
149      2
150      1           YOUT:PROC; /* PRINTS CURRENT DUAL NODE VARS */
151      2           PUT EDIT('*BLOSS - CURRENT NODE DUAL VARIABLES :')(SKIP,A);
152      2           PUT SKIP;
153      2           DO IY = 1 TO NNODE;
154      2           P=ADDR(NODELST(IY));
155      2           PUT EDIT(P->Y)(F(10,2));
156      2           END;
157      2           END YOUT;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
***** GENERAL TREE HANDLING ROUTINES ****
***** REMOVE: PROC (P1);
    /* SUBROUTINE TO REMOVE P1 FROM THE TREE CONTAINING IT. */
    DCL (P1,(P2,P3) STATIC) PTR;
    P2=P1->DN;
    IF P2=NULL THEN GO TO RET /* FOR P1 IS THE ROOT OF ITS TREE */;
    P3=P2->UP;
    IF P3=P1 THEN GO TO EASY;
    DO WHILE (P3->RT =P1);
        P3=P3->RT;
    END;
    /* NOW WE HAVE FOUND P1 */
    P3->RT=P1->RT;
    GO TO RET;
EASY:P2->UP=P1->RT;
RET: P1->RT=NULL;
P1->DN=NULL;
FRST(P1->EDGEDN)=F;
END REMOVE;

REROOT: PROC (P1);
    /* SUBROUTINE WHICH REROOTS THE TREE CONTAINING P1 AT P1. */
    DCL ((P2,P3,PX) PTR,(J,J3) BIN FIXED(16)) STATIC, P1 PTR;
    P2=P1->DN;
    IF P2=NULL THEN RETURN /* FOR P1 IS ALREADY A ROOT. */;
    J=P1->EDGEDN;
    PX=P1;
    CALL RFMOVE (P1);
LP: P3=P2->DN;
    J3=P2->EDGEDN;
    CALL REMOVE (P2);
    CALL ADDON (PX,P2,J);
    IF P3=NULL THEN RETURN /* FOR P2 WAS THE ROOT. */;
    PX=P2;
    P2=P3;
    J=J3;
    GO TO LP;
END REROOT;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
195      1      ADDON:PROC(Q1,Q2,J);
           /* ADDON ATTACHES THE TREE ROOTED AT Q2 TO NODE Q1 BY MEANS
              OF EDGE J.  IT REQUIRFS THAT Q2 BE THE ROOT OF A TREE. */
196      2      DCL (Q1,Q2) PTR, J BIN FIXFD(16);

197      2      Q2->RT=Q1->UP;
198      2      Q2->DN=Q1;
199      2      Q2->EDGEON=J;
200      2      Q1->UP=Q2;
201      2      FRST(J)=T;
202      2      END ADDON;

/****** THE UPSCAN ROUTINES *****/
***** UPSCAN:PROC(P1,UPCALL,SUBRUP,DNCALL,SUBRDN); *****

203      1      UPSCAN:PROC(P1,UPCALL,SUBRUP,DNCALL,SUBRDN);

           /* UPSCAN GOES THROUGH ALL THE NODES ABOVE P1 IN THE TREE
              CONTAINING P1 AND IF UPCALL=T THEN CALLS SUBRUP FOR
              EACH NODE IN THE TREE AS IT REACHES IT COMING UP.
              IF DNCALL = T THEN SUBRDN IS CALLED FOR EACH NODE AS
              IT IS ENCOUNTERED COMING DOWN. */
204      2      DCL (P1,(Q1,Q2) STATIC ) PTR,(UPCALL,DNCALL) BIT(1),
           (SUBRUP,SUBRDN) ENTRY;

205      2      IF UPCALL THEN CALL SUBRUP(P1);
207      2      Q1=P1;
208      2      MVUP:Q2=Q1->UP;
209      2      IF Q2=NULL THEN DO;
211      2      CALLUP: IF UPCALL THEN CALL SUBRUP(Q2);
213      2      1      Q1=Q2;
214      2      1      GO TO MVUP;
215      2      1      END;
216      2      ENDTST: IF Q1=P1 THEN DO;
218      2      1      IF DNCALL THEN CALL SUBRDN(Q1);
220      2      1      RETURN;
221      2      1      END;
222      2      Q2=Q1->RT;
223      2      IF Q2=NULL THEN DO;
225      2      1      IF DNCALL THEN CALL SUBRDN(Q1);
227      2      1      GO TO CALLUP;
228      2      1      END;
229      2      Q2=Q1;
230      2      Q1=Q1->DN;
231      2      IF DNCALL THEN CALL SUBRDN(Q2);
233      2      GO TO ENDTST;
234      2      END UPSCAN;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
235   1      ADDFIX: PROC (Q1);
              /* THIS PROCEDURE SETS ODD,DEFIC AS APPROPRIATE FOR THE NODE
                 Q1. IT DEPENDS ON Q1->TREE.DN. */
236   2      DCL (Q1,Q2 STATIC) PTR;
237   2      Q1->BLOS,Q1->INPATH=F;
238   2      Q2=Q1->DN;
239   2      Q1->DEFIC=Q2->DEFIC;
240   2      Q1->YRTO = Q2->YRTO;
241   2      IF ~Q1->DEFIC THEN Q1->ODD=F;
242   2      ELSE Q1->ODD=~Q2->ODD;
243   2
244   2      END ADDFIX;

245   1      POLYFIX: PROC(P1);
              /* PROC CALLED BY UPSCAN TO
                 1) SET P1->STACKUP = NULL;
                 2) SET P1->INPATH = F. */
246   2      DCL P1 PTR;
247   2      P1->STACKUP=NULL;
248   2      P1->INPATH=F;
249   2
250   1      NONDEFIX: PROC (P1);
              /* THIS IS A PROCEDURE DESIGNED TO BE CALLED BY UPSCAN WHICH CORRECTS
                 THE STATUS INDICATORS AND SPLITS A TREE WITH NON-DEFICIENT ROOT
                 INTO POSITIVE COMPONENTS */
251   2      DCL P1 PTR;
252   2      P1 -> DEFIC, P1 -> ODD = F;
253   2      P1 -> RLOS = F;
254   2      P1 -> YRTO = F;
255   2      /* INDICATORS NOW CORRECT */
256   2      IF P1 -> DN=NULL /* I.E. IT EXISTS */
              THEN IF X(P1->EDGEDN)=0 /* I.E. WE HAVE PLACE FOR
                  DETACHING */
                  THEN CALL REMOVE(P1);
257   2
258   2      END NONDEFIX;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
259      1      BLOSSIND:PROC (P1);
             /* PROCEDURE TO INDICATE THAT P1 IS A NODE IN A NON-DEFICIENT
                BLOSSOM, AND THE EDGE DOWN IS IN A SIMILAR STATE */
260      2      DCL P1 PTR;
261      2      FRST (P1 -> EDGEDN)=T;
262      2      P1 -> ODD, P1 -> DEFIC, P1->INPATH = F;
263      2      P1 -> BLOS=T;
264      2      END;
265      1      ADDBLOS:PROC(P1);
             /* PROCEDURE CALLED WHEN EXPANDED BLOSSOM HAS BEEN ADDED
                TO A DEFIC TREE. IT SETS ODD UNTIL A ZERO EVEN EDGE
                IS FOUND, WHEN IT SETS THINGS UP FOR DEFFIX TO SPLIT
                THINGS INTO NONZERO COMPONENTS. NOCHECK IS USED TO AVOID
                TRYING TO SET DEFIC AND ODD FOR THE ROOT WHEN UPSCAN IS
                STARTED AT THE ROOT OF A TREE. */
266      2      DCL(P1,P2 STATIC)PTR;
             /* PX AND SHRINKNG ARE USED AS EXT. VARS. */
267      2      P1->STACKUP=NULL;
268      2      P1->BLOS,P1->INPATH=F;
269      2      IF NOCHECK THEN DO;
270      2      NOCHECK=F; RETURN; END;
271      2      1      IF ~SHRINKNG THEN GO TO LAB1;
272      2      P2=P1->DN;
273      2      IF P2->ODD THEN
274      2      IF X(P1->EDGEDN)=0 THEN DO; /* DETACH */
275      2      SHRINKNG=F;PX=P1;
276      2      1      LAB1:P1->DEFIC,P1->ODD = F;
277      2      1      RETURN;
278      2      1      FIND;
279      2      1      P1->DEFIC=P2->DEFIC;
280      2      1      P1->ODD=~P2->ODD;
281      2      1      RETURN;
282      2      1      P1->DEFIC=P2->DEFIC;
283      2      1      P1->ODD=~P2->ODD;
284      2      1      RETURN;
285      2      1      P1->DEFIC=P2->DEFIC;
286      2      1      P1->ODD=~P2->ODD;
287      2      1      RETURN;
288      2      DEFFIX:FTRY(P1);
289      2      IF ~SHRINKNG THEN /*POSSIBLE DETACHMENT */
290      2      IF X(P1->EDGEDN)=0 THEN CALL REMOVE(P1);
291      2      IF P1=PX THEN SHRINKNG=T;
292      2      RETURN;
293      2      END ADDBLOS;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
296   1      UPBLOSS:PROC (P1);
        /* UPRLOSS AND DNBLLOSS DO MOST OF THE WORK REQUIRED TO
           SHRINK A BLOSSOM. WE USE PX (PTR) AND SHRNKG (BIT(1)) AS
           DEFINED IN BLOSSOM. WE ASSUME THAT R2 IS THE ROOT OF THE
           BLOSSOM AND P IS THE PSEUDONODE BEING CREATED. */

297   2      DCL P1 PTR;

298   2      IF ~ SHRNKG THEN RETURN;
300   2      IF P1=R2 THEN GO TO BFIX;
302   2      IF ~ P1 -> INPATH THEN
303   2          IF X(P1 -> EDGEDN)=0 THEN DO;
305   2              1      SHRNKG=F; /* STOP SHRINKING */
306   2              1      PX=P1 /* SAVE NODE FOR DNBLLOSS*/;
307   2              1      RETURN;
308   2              1      END;
309   2      BFIX: P1 -> STACKUP=P;
310   2      RETURN;

311   2      DNBLLOSS:ENTRY (P1);
312   2          P1->INPATH=F; /* TURN OFF PATH INDICATOR */
313   2          IF P1~=PX THEN /* NO SHIPPING TO BE DONE, SO */
314   2              RETURN;
315   2              K=P1 -> EDGEDN;
316   2              CALL REMOVE (P1);
317   2              CALL ADDON (P,P1,K);
318   2              SHRNKG=T; /* RESUME SHRINKING */
319   2              RETURN;
320   2              END;

321   1      SETYRTO: PROC(P1);
        /* PROCEDURE CALLED BY UPSCAN TO SET P1->YRT0 EQUAL TO
           THE GLOBAL VARIABLE YROOT0. */
322   2      DCL P1 PTR;

323   2      P1 -> YRT0=YROOT0;
324   2      END SETYRTO;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
*****  
***** INITIALIZATION *****  
*****  
325   1      IF RUNSTAT(10)=1 THEN TRACE=T; ELSE TRACE=F;  
328   1      IF TRACE THEN DO;  
330   1  1      CALL XOUT; CALL YOUT; END;  
   /* GENERATE THE INITIAL EQUALITY SUBGRAPH. */  
333   1  1      IF TRACE THEN DO;  
335   1  1      PUT EDIT('*BLOSS - EDGES IN EQUALITY SUBGRAPH:')(SKIP,A);  
336   1  1      PUT SKIP; END;  
338   1  1      DO J=1 TO NEDGE;  
   C(J) = C(J) - FN(J); /* CALCULATE REDUCED COST */  
339   1  1      IF C(J) = 0 THEN EQ(J)=T; ELSE EQ(J) = F;  
340   1  1      IF TRACE THEN IF EQ(J) THEN PUT EDIT(J)(F(5));  
343   1  1      END;  
   /* END OF EQUALITY SURGRAPH GENERATION */  
347   1      RUNSTAT=0;  
348   1      NOCHECK, FROMEX = F;  
349   1      JCNT,LASTJ=1;  
350   1      A:  
*****  
***** FIRST LEVEL EDGE ANALYSIS *****  
*****  
351   1      IF ~EQ(JCNT) THEN GO TO ENDA;  
352   1      IF ZER(JCNT) THEN GO TO ENDA;  
354   1      IF SHRNK(JCNT) THEN GO TO ENDA;  
356   1      IF FRST(JCNT) THEN GO TO ENDA;  
   /*  
    OTHERWISE WE HAVE AN EDGE WHICH IS IN THE EQUALITY SUB-  
    GRAPH WHICH CAN TAKE ON A NONZERO VALUE AND SO FAR HAS  
    NOT BEEN SHRUNK AND IS NOT IN THE FOREST.*/  
   /*  
    WE NOW ANALYZE THE EDGE. IN ORDER FOR IT TO BE USEFUL  
    ONE END MUST BE AN EVEN NODE OF A DEFICIENT TREE IN THE  
    FOREST FOR WHICH THE ROOT IS NOT A <= NODE WITH Y=0. */  
358   1      P1=ENDS(JCNT,1); P2=ENDS(JCNT,2);  
360   1      IF P1->DEFIC THEN  
361   1          IF ~P1->YRTO THEN  
362   1              IF ~P1->ODD THEN GO TO DEFOUT;  
364   1              IF ~P2->DEFIC THEN GO TO ENDA;  
366   1              IF P2->ODD THEN GO TO ENDA;  
368   1              IF P2->YRTO THEN GO TO ENDA;  
370   1              P3=P2; P2=P1; P1=P3; /* INTERCHANGE POINTERS FOR P2 DEF. OUT ND. */  
373   1              DEFOUT: /* IF THE OTHER END OF THE EDGE IS AN ODD NODE OF  
   THE FOREST THEN THE EDGE IS OF NO USE TO US,  
   UNLESS IT IS IN A TREE WHOSE ROOT IS <= WITH Y = 0. */  
   IF P2->ODD THEN  
       IF ~P2->YRTO THEN GO TO ENDA;  
   ELSE GO TO ODDGROW;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
377   1           J,LASTJ=JCNT; /* FOR WE ARE ABOUT TO ACCOMPLISH SOMETHING*/
378   1           IF P2->BLDS THEN GO TO POLYSTEP;
379   1           IF ~P2->DEFIC THEN GO TO GROWSTEP;
380   1           /* OTHERWISE EDGE(J) JOINS TWO EVEN NODES OF THE FOREST */
381   1           /* FIRST WE SEE IF THEY ARE IN TREES WITH DISTINCT ROOTS,
382   1           IF SO WE CAN SIMPLY AUGMENT, OTHERWISE WE MAY HAVE TO
383   1           SHRINK. AT THE SAME TIME WE COMPUTE HOW MUCH THE VALUES
384   1           ON THE PATH CAN BE CHANGED.*/
385   1
386   1           **** **** **** **** **** **** **** **** **** **** **** **** **** **** ****
387   1           **** **** **** SECOND LEVEL EDGE ANALYSIS **** **** **** **** **** ****
388   1           **** **** **** **** **** **** **** **** **** **** **** **** **** **** ****
389   1           DXCALC: D1,D2,D3=32767;
390   1           /* NOW FIND PATH FROM P1 TO THE ROOT*/
391   1           R1=P1;
392   1           R1->INPATH=T;
393   1           DO WHILE (R1->DN = NULL);
394   1           IF ~R1->ODD THEN D1 = MIN(D1,X(R1->EDGEDN));
395   1           R1=R1->DN;
396   1           R1->INPATH=T;
397   1           END;
398   1           /* SIMILARLY, FIND PATH FROM P2 TO ITS ROOT R2; IF A
399   1           POLYGON IS FORMED, R2 WILL BE THE ROOT OF THE POLYGON. */
400   1           R2=P2;
401   1           DO WHILE(~R2->INPATH);
402   1           R2->INPATH=T;
403   1           IF R2->DN=NULL THEN /* WE ARE AT THE ROOT */ GO TO TWO1REE;
404   1           IF ~R2->ODD THEN D2=MIN(D2,X(R2->EDGEDN));
405   1           R2=R2->DN;
406   1           END;
407   1           /* WE MUST HAVE A COMMON ROOT TO THE TWO TREES SO WE */
408   1           GO TO ONETREE;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
*****  
***** TWO TREE AUGMENTATION *****  
*****  
401   1      TNOTREE:  
        /* IF WE MADE IT TO HERE, R1 AND R2 ARE DIFFERENT SO WE  
           AUGMENT BY AMOUNT */  
        RUNSTAT(6)=RUNSTAT(6)+1;  
        DELTAX=MIN(D1,D2,R1->DEF,R2->DEF);  
402   1      CALL AUGMNT(P1,R1,DELTAX,T,(F));  
403   1      CALL AUGMNT(P2,R2,DELTAX,T,(F));  
404   1      X(J)=X(J)+DELTAX;  
405   1      R1->DEF=R1->DEF - DELTAX;  
406   1      R2->DEF=R2->DEF - DELTAX;  
407   1      IF TRACE THEN DO;  
410   1      1          PUT EDIT('BLOSS - EDGE ',J,' USED FOR 2 TREE AUGMENTATION')  
                      (R(FMT));  
411   1      1          CALL XOUT; END;  
        /* NOW CORRECT STATUS INDICATORS IN THE TREE*/  
413   1      IF R1->DEF = 0 THEN CALL UPSCAN(R1,F,NONDEFIX,T,NONOFFIX);  
415   1      IF R2->DEF = 0 THEN CALL UPSCAN(R2,F,NONOFFIX,T,NONDEFIX);  
        /* FINALLY INCORPORATE J INTO THE FOREST */  
417   1      JADD: IF P1->DEFIC THEN /* P2 CANNOT BE IN A DEFICIENT TREE, ADD  
418   1          ON TO P1*/DO;P3=P2;P2=P1;P1=P3;FND;  
423   1          CALL RFROOT(P1);CALL ADDOM(P2,P1,J);  
425   1          CALL UPSCAN (P1,T,ADDFIX,F);  
426   1          GO TO ENDA;  
  
*****  
***** SINGLE TREE AUGMENTATION *****  
*****  
427   1      ONETREE: /* FIND BOTTLENECK IN STEM OF BLOSSOM */  
        R3=R2;  
428   1      DO WHILE (R2->DN->NULL);  
429   1      1          IF ~R2->ODD THEN DO;  
431   1      2              IF X(R2->EDGEDN) = 1 THEN /* WE HAVE FOUND THE  
432   1      2                  START OF A BLOSSOM, SO*/ GO TO DEFBLOSS;  
433   1      2              D3=MIN(D3,X(R2->EDGEDN));  
434   1      2          END;  
435   1      1              R2=R2->DN;  
436   1      1          END;  
        /* AT THIS POINT, AN AUGMENTATION IS POSSIBLE, SINCE NO EVEN  
           EDGE IN THE STEM HAS X=1, UNLESS R1->DEF = 1.*/  
437   1      IF R1->DEF = 1 THEN GO TO DEFBLOSS;  
        /* OTHERWISE ITS AUGMENTATION TIME.*/  
439   1      RUNSTAT(7)=RUNSTAT(7)+1;  
440   1      DELTAX=MIN(D1,D2,FLOOR(R1->DEF/2),FLOOR(D3/2));  
441   1      ODDB=F; /* NORMAL CASE */  
442   1      IF P1->ODD THEN DO; /* ABNORMAL CASE */  
444   1      1          ODDB=T; DELTAX=MIN(DELTAX,X(J)); END;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
447      1          CALL AUGMENT(P1,R3,DELTAX,T,(ODDB));
448      1          CALL AUGMENT(P2,R3,DELTAX,T,(ODDB));
449      1          X(J)=X(J)+DELTAX;
450      1          IF ODDB THEN X(J) = X(J) -2*DELTAX; /* CORRECT A BAD GUESS */
452      1          POLYBIT=F;
453      1          IF P1->DEFIC THEN
454      1              IF P2->DEFIC THEN
455      1                  IF X(J)>0 THEN POLYBIT=T; /* WE HAVE A NONZERO POLYGON */
457      1                  DELTAX=DELTAX+DELTAX; /* STEM GETS DOUBLE AUGMENTATION */
458      1                  R1->DEF=R1->DEF - DELTAX;
459      1                  ODDB=F;
460      1                  IF R3->ODD THEN ODDB=T;
462      1                  CALL AUGMENT(R3,R1,DELTAX,T,(ODDB));
463      1                  IF TRACE THEN DO;
465      1          1                  PUT EDIT('BLOSS - EDGE ',J,' USED FOR 1 TREE AUGMENTATION')
466      1          1                  (R(FMT));
467      1          1                  CALL XOUT;
468      1          1                  END;
469      1          1                  /* DISASSEMBLE THE TREE IF ROOT NO LONGER DEFICIENT. */
470      1          1                  IF R1->DEF = 0 THEN CALL UPSCAN(R1,F,NONDEFIX,T,NONDEFIX);
471      1          1                  ELSE IF FROMEX THEN DO; /* WE HAVE EXPANDED PSEUDO ODD NODE
472      1          1                      AND MUST ENSURE THAT TREE IS CORRECT. */
473      1          1                      NOCHECK=T; /* IGNORE ROOT, SET F BY ADDBLOS. */
474      1          1                      PX=NULL; SHRNKNG=T; /* GLOBALS FOR ADDBLOS-DEFFIX */
475      1          1                      CALL UPSCAN(R1,T,ADDBLOS,T,DEFFIX);
476      1          1                      END;
477      1          1                      FROMEX=F;
478      1          1                      IF ~POLYBIT THEN /* INCORPORATE J INTO THE FOREST */
479      1          1                          IF X(J)>0 THEN GO TO JADD;
480      1          1                          ELSE DO; FRST(J)=F; GO TO ENDA; END;
481      1          1                          /* OTHERWISE WE HAVE A POLYGON WITH NONZERO EDGES */
482      1          1                          IF R3 -> DEFIC THEN DO;
483      1          1                              R2=R3;
484      1          1                              DO WHILE ((R2->DN)=NULL);
485      1          1                                  IF ~R2 -> ODD THEN
486      1          1                                      IF X(R2->EDGEDN)=1
487      1          1                                          THEN GO TO DEFBLOSS;
488      1          1                                      R2=R2->DN;
489      1          1                                  END;
490      1          1                                  GO TO DEFBLOSS; /* ROOT OF TREE ROOT OF BLOSSOM */
491      1          1                                  END;
492      1          1                              ELSE DO; R2=R3;
493      1          1                                  LB1: IF R2->DN = NULL THEN GO TO LB2;
494      1          1                                  R2=R2->DN;
495      1          1                                  GO TO LB1;
496      1          1                              LB2: R2->EDGEDN=J;
497      1          1                                  CALL UPSCAN (R2,T,BLOSSIND,F); /* SHOW A NONZERO COMPONENT */
498      1          1                                  GO TO ENDA; /* CONTAINING AN ODD POLYGON */
499      1          1                                  END;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
***** PSEUDO NODE CREATION *****
***** DEFBLoss: /* HERE P1 AND P2 ARE THE TWO ENDS OF THE
      EDGE J WHICH FORMS A BLOSSOM, R2 IS THE ROOT OF THE STEM
      AND R3 IS THE ROOT OF THE POLYGON. ALL WE NEED DO IS
      SHRINK IT */
      RUNSTAT(2)=RUNSTAT(2)+1;
      FROMEX=F; /* IN CASE IT WAS SET T BY PSEUDO EXPANSION */
      ALLOCATE PSEUDO; /* CREATE A PSEUDO NODE */
      DEF=1;
      REAL,INPATH,ODD,BLOSS,EXPANDED=F;
      CONSTEQ,DEFIC=T;
      Y=0E0;
      STACKUP=NULL;
      ROOT=R2;
      UP,RT,DN=NULL;
      YRTO=R2->YRTO;
      IF TRACE THEN PUT EDIT('*BLOSS - EDGE ',J,', FORMS PSEUDONODE',
      UNSPEC(P))(SKIP,A,F(6),A,F(10));
      /* INDICATE NODES IN PATHS FROM P2, P1 TO R2 */
      P3=P1;
      DO WHILE (P3!=R2);
          P3->INPATH = T; P3=P3->DN; END;
      R2->INPATH=T;
      P3=P2;
      DO WHILE (!P3->INPATH);
          P3->INPATH = T; P3=P3->DN; END;
      /* TURN OFF INPATH IN UNUSED PART OF STEM */
      P3=R2->DN;
      DO WHILE (P3!=NULL);
          P3->INPATH=F; P3=P3->DN; END;
      SHRINKNG=T; /* WE ARE SHRINKING */
      PX=NULL; /* PREP. FOR CALL OF UPSCAN */
      CALL UPSCAN (R2,T,UPBLOSS,T,DNBLOSS);
      K=R2->EDGEDN; P3=R2->DN;
      CALL REMOVE (R2);
      IF P3 != NULL THEN CALL ADDON (P3,P,K);
      ELSE P->EDGEDN = 0;
      R2->EDGEDN=J; /* THIS IS THE EDGE THAT FORMED THE BLOSSOM;
      NOW FIX ALL EDGES SO THAT ENDS IS CORRECT */
      DO J1=1 TO NEDGE;
          IF SHRNK(J1) THEN GO TO FNDC;
          P1=ENDS(J1,1); P2=ENDS (J1,2);
          IF P1->STACKUP=NULL THEN /* NO CHANGE */
              GO TO ENDXB;
          SHRNK(J1)=T;
          ENDS(J1,1)=P;
      ENDXB: IF P2->STACKUP=NULL THEN
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
555      1   1           ENDS(J1,2)=P;
556      1   1           ELSE SHRINK(J1)=F;
557      1   1           ENDC: END;
558      1           /* NOW SHRINKING IS COMPLETE */
559      1           GO TO ENDA;

*****NORMAL FOREST GROWTH *****
*****GROWSTEP: /* WE GROW TREE BY USING J TO ADD A NONDEFICIENT
560      1           TREE */
561           RUNSTAT(5)=RUNSTAT(5)+1;
562           IF TRACE THEN PUT EDIT('*BLOSS - EDGE ',J,', USED TO GROW FOREST')
563           (R(FMT));
564           CALL REROOT(P2);
565           CALL ADDON(P1,P2,J);
566           CALL UPSCAN (P2,T,ADDFIX,F);
567           GO TO ENDA;

*****ADJUNCTION OF POLYGON TO THE FOREST *****
*****POLYSTEP: /* FIND ROOT OF COMPONENT */
568           RUNSTAT(8)=RUNSTAT(8)+1;
569           IF TRACE THEN PUT EDIT('*BLOSS - EDGE ',J,', USED TO ADD NONZERO PO
570           LYON TO THE FOREST')(R(FMT));
571           P3=P2;
572           DO WHILE (P3->DN=NULL);
573           P3=P3->DN;
574           END;
575           J1=P3->EDGEDN; /*J1 IS THE EDGE WHICH FORMED THE POLYGON */
576           CALL REROOT (P2);
577           /* REROOT THE COMPONENT AND ADD IT TO TREE */
578           CALL ADDON (P1,P2,J);
579           CALL UPSCAN (P2,T,ADDFIX,F);
580           P1=ENDS(J1,1);
581           P2=ENDS(J1,2);
582           J=J1;
583           GO TO DXCALC;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
***** PSEUDO FOREST GROWTH *****
***** ODDGROW: /* AN EDGE J HAS BEEN FOUND JOINING P1 IN F1 TO P2 IN F0*/
      RUNSTAT(5)=RUNSTAT(5)+1;
      J,LASTJ = JCNT;
      IF TRACE THEN PUT EDIT('*BLOSS - EDGE ',J,', USED FOR PSEUDO FOREST
      GROWTH')(R(FMT));
      /* FIND FIRST NODE IN PATH FROM P2 TO ITS ROOT HAVING A ZERO
         DOWN EDGE, OR IF NO SUCH EDGE EXISTS, THEN WE FIND THE
         ROOT OF THE TREE CONTAINING P2, */
      R1=P2;
      DO WHILE(R1->DN ≠ NULL);
          IF X(R1->EDGEDN)=0 THEN GO TO ROOTADD;
          R1=R1->DN;
      END;
      /* R1 IS THE ROOT, ALL EDGES IN PATH HAVE X>0. */
      ROOTADD:
      Q3=R1->DN;
      CALL REMOVE(R1);
      CALL REROOT(P2);
      CALL ADDON(P1,P2,J); /* TREES NOW CONSOLIDATED */
      YROOT0 = F;
      CALL UPSCAN(P2,T,SETYRT0,F);
      IF Q3 ≠ NULL THEN /* WE HAD A ZERO EDGE */GO TO ENDA;
      /* NOW AUGMENT SO AS TO GET DEFICIENCY TO THE ROOT */
      /* Q1 WILL BE THE LAST NODE FOR WHICH THE DOWN EDGE BECOMES 0 */
      AUG: D1=32767;
      RUNSTAT(9)=RUNSTAT(9)+1;
      R2=R1;
      DO WHILE (R2->DN ≠ NULL);
          IF ¬R2->ODD THEN DO;
              J1=R2->EDGEDN;
              IF X(J1) <= D1 THEN DO;
                  D1=X(J1);
                  Q1=R2;
              END;
          END;
          R2=R2->DN;
      END;
      DELTAX=MIN(D1,R2->DEF);
      /* NOW WE AUGMENT */
      CALL AUGMENT(R1,R2,DELTAX,F,(F));
      R2->DEF = R2->DEF - DELTAX;
      R1->DEF=R1->DEF + DELTAX; /* WE INCREASE DEF AT THIS NODE. */
      IF TRACE THEN DO;
          PUT EDIT('*BLOSS - PSEUDO AUGMENTATION')(SKIP,A);
          CALL XOUT;
      END;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
623   1      IF D1 > DELTAX THEN /* NO EDGE IN PATH BECAME ZERO */
624   1          GO TO TADD;
625   1      /* ELSE EVERYTHING ABOVE Q1 GETS REMOVED AND REROOTED AT R1 */
626   1      CALL RFMOVE (Q1);
627   1      YROOT0=F;
628   1      CALL UPSCAN(R2,T,SETYRTO,F);
629   1      IF R2->DEF=0 THEN
630   1          CALL UPSCAN(R2,F,NONDEFIX,T,NONDEFIX); /* ALSO SETS YTR0=F */
631   1      TADD: CALL REROOT(R1);
632   1      YROOT0 = T;
633   1      CALL UPSCAN(R1,T,SETYRTO,F);
634   1      **** END OF MAIN PROCESSING LOOP ****
ENDA:JCNT=1 + MOD(JCNT,EDGE);
IF JCNT=LASTJ THEN /* CONTINUE PROCESSING */ GO TO A;
/* WHENEVER AN EDGE IS MADE USE OF IN THE MAIN LOOP, LASTJ
IS SET EQUAL TO THE INDEX OF THE EDGE. IF JCNT EVER 'CATCHES
UP' WITH LASTJ THEN WE HAVE MADE A COMPLETE CYCLE THROUGH THE
EDGES WITHOUT FINDING ANY EDGES WHICH WE CAN USE SO WE PROCEED
TO ATTEMPT A CHANGE OF DUAL VARIABLES. */
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LFVFL NEST

```
*****  
***** DUAL VARTABLE CHANGE ROUTINE *****  
*****  
636   1      DUALCHNGE:  
       /* NOW EXAMINE NODES, IF NO SURFACE NODE IS IN A DEFIC TREE  
          THEN WE ARE DONE. FAIL IS SET TRUE IF WE DISCOVER THAT  
          THIS IS NOT THE CASE. */  
637   1      FAIL=F;  
       EPS1,EPS2=1E10; /*RIDICULOUSLY LARGE VALUES*/  
*****  
***** DETERMINATION OF NODE BOUND *****  
*****  
638   1      PX=NULL;  
639   1      LF: DO I = 1 TO NNODE;  
640   1      1      P1=ADDR(NODELST(I));  
641   1      1      P2=SURF(P1); /* HIGHEST LEVEL PSEUDONODE CONTAINING P1 */  
642   1      1      IF P2->DEFIC THEN GO TO ENDF;  
643   1      1      IF P2->YRTO THEN GO TO ENDF;  
       /* ELSE WE HAVE NOT YET GOT A FEASIBLE MATCHING, */  
644   1      1      FAIL=T;  
645   1      1      IF P2->ODD THEN  
646   1      1      IF P1->CONSTEQ THEN  
647   1      1      IF P1->Y < EPS2 THEN DO;  
648   1      1      PX=P1;  
649   1      1      EPS2=P1->Y;  
650   1      2      END;  
651   1      2      IF P2->REAL THEN /* NOT IN A PSEUDO NODE */ GO TO ENDF;  
       /* OTHERWISE CHECK THE PSEUDO NODE */  
652   1      2      P2->DCHNG=F; /* NO DUAL CHANGE MADE YET ON THIS NODE */  
653   1      2      IF P2->ODD THEN GO TO ENDF;  
654   1      2      Z=P2->Y / 2E0;  
655   1      2      IF Z<EPS2 THEN DO;  
656   1      2      PX=P2;  
657   1      2      EPS2=Z;  
658   1      2      END;  
659   1      1      ENDF: END LF;  
660   1      1      IF FAIL THEN /* WE ARE FINISHED */ GO TO CORRECTION;  
661   1      1      IF EPS2= 0 THEN GO TO NODEBND; /* NO NEED TO CHECK EDGES,  
662   1      1      WE ALREADY HAVE OUR BOUND, */
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
***** DETERMINATION OF EDGE BOUND *****
/* NOW CHECK EDGES FOR A BOUND ON EPS */
670      1      JX=0;
671      1      LD: DO J=1 TO NEDGE;
672      1      1      IF EQ(J) THEN GO TO ENDD; /* IGNORE EDGES IN EQ SUBGRAPH */
674      1      1      IF SHRMK(J) THEN GO TO ENDD;
676      1      1      IF ZER(J) THEN GO TO ENDD;
678      1      1      P1=ENDS(J,1);
679      1      1      IF ~P1->DEFIC THEN GO TO TRY2;
681      1      1      IF P1->YRTO THEN GO TO TRY2;
683      1      1      P2=ENDS(J,2 );
684      1      1      IF ~P1->ODD THEN GO TO TESTP2;
686      1      1      TRY2:P1=ENDS(J,2 );
687      1      1      IF ~P1->DEFIC THEN GO TO ENDD;
689      1      1      IF P1->YRTO THEN GO TO ENDD;
691      1      1      P2=ENDS(J,1);
692      1      1      IF P1->ODD THEN GO TO ENDD;
/* AT THIS POINT P1 IS AN EVEN NODE OF A DEFIC TREE, AS LONG
AS P2 IS NOT AN ODD NODE WE HAVE FOUND AN EDGE OF INTEREST */
694      1      1      TESTP2:IF P2->ODD THEN
695      1      1          IF ~P2->YRTO THEN GO TO ENDD;
697      1      1          Z=- C(J);
698      1      1          IF P2->DEFIC THEN
699      1      1              IF ~P2->YRTO THEN Z=Z/2E0;
/* ELSE J HAS JUST ONE END IN THE FOREST*/
701      1      1          IF Z>=EPS1 THEN GO TO ENDD;
703      1      1          JX=J;
704      1      1          FPS1=7;
705      1      1      ENDD: END LD;
***** MAKE ACTUAL CHANGE IN DUAL VARS. *****
706      1      EPS=MIN(EPS1,EPS2);
707      1      IF EPS=1E10 THEN /* FOREST IS HUNGARIAN */ DO;
709      1      1          RUNSTAT(10)=1;
710      1      1          GO TO CORRECTION; END;
/* HERE WE GO ON A CHANGE OF DUAL VARIABLES*/
712      1      IF TRACE THEN PUT EDIT('*BLOSS - DUAL VARIABLE CHANGE')(SKIP,A);
714      1      LG : DO I=1 TO NNODE;
715      1      1          P1=ADDR(NODFLST(I));
716      1      1          P2=SURF(P1);
717      1      1          IF ~P2->DEFIC THEN GO TO ENDLG;
719      1      1          IF P2->YRTO THEN GO TO ENDLG;
721      1      1          IF P2->ODD THEN DO;
723      1      2              P1->Y=P1->Y + EPS;
724      1      2              IF ~P2->REAL THEN
725      1      2                  IF ~P2->DCHNG/*P2->Y HAS NOT YET BEEN CHANGED */ THEN DO;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
727    1   3          P2->Y=P2->Y-2E0 * EPS;
728    1   3          P2->DCHNG=T;
729    1   3          IF TRACE THEN PUT EDIT('      PSEUDO ',UNSPEC(P2),
730    1   3          ' DUAL VAR.',P2->Y)(SKIP,A,F(10),A,F(10,1));
731    1   3          END;
732    1   2          END;
733    1   1          ELSE /* P2 IS AN EVEN NODE */ DO;
734    1   2          P1->Y=P1->Y - EPS;
735    1   2          IF ~P2->RFAL THEN
736    1   2          IF ~P2->DCHNG THEN DO;
737    1   3          P2->Y=P2->Y +2E0 * EPS;
738    1   3          P2->DCHNG= T;
739    1   3          IF TRACE THEN PUT EDIT('      PSEUDO ',UNSPEC(P2),
740    1   3          ' DUAL VAR.',P2->Y)(SKIP,A,F(10),A,F(10,1));
741    1   2          END;
742    1   3          END;
743    1   2          END;
744    1   1          ENDLG: END LG;
745    1           RUNSTAT(1)=RUNSTAT(1)+1;
746    1           IF TRACE THEN CALL YOUT;
747    1           IF TRACE THEN PUT EDIT('*BLOSS - EDGES IN EQUALITY SUBGRAPH')
748    1           (SKIP,A);
749    1           /* NOW CALCULATE NEW REDUCED COSTS */
750    1           DO J=1 TO NEDGE;
751    1   1           IF SHRNK(J) THEN GO TO MSG;
752    1   1           IF FRST(J) THEN GO TO NSG;
753    1   1           IF ZER(J) THEN GO TO ENDX;
754    1   1           P1=ENDS(J,1); P2=ENDS(J,2);
755    1   1           IF P1->DEFIC THEN
756    1   2           IF ~P1->YRTO THEN DO;
757    1   2           IF P1->ODD THEN C(J)=C(J)-EPS;
758    1   2           ELSE C(J)=C(J)+EPS;
759    1   2           END;
760    1   1           IF P2->DEFIC THEN
761    1   1           IF ~P2->YRTO THEN DO;
762    1   2           IF P2->ODD THEN C(J) = C(J) - EPS;
763    1   2           ELSE C(J)=C(J) + EPS;
764    1   2           END;
765    1   1           END;
766    1   1           IF P2->DEFIC THEN
767    1   1           IF ~P2->YRTO THEN DO;
768    1   2           IF P2->ODD THEN C(J) = C(J) - EPS;
769    1   2           ELSE C(J)=C(J) + EPS;
770    1   2           END;
771    1   1           MSG; IF C(J)=0 THEN DO;
772    1   2           EQ(J)=T;
773    1   2           IF TRACE THEN PUT EDIT(J)(F(5));
774    1   2           END;
775    1   1           ELSE FO(J)=F;
776    1   1           ENDX:END;
777    1   1           IF EPS1 = EPS THEN DO; JCNT, LASTJ=JX;
778    1   1           GO TO A; /* RETURN TO MAIN LOOP AND ACCOMPLISH SOME -
779    1   1           THING. */
780    1   1           END;
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
***** REROOT A TREE SO ROOT HAS Y=0 ****
***** NODEBND: ****
786   1      IF ~PX -> REAL THEN GO TO PSEUDOEX;
787   1      /* OTHERWISE PX IS A REAL <= EVEN NODE */
788   1      IF TRACE THEN PUT EDIT('*BLOSS - REROOT A TREE')(SKIP,A);
789   1      R1=SURF(PX);
790   1      JCNT,LASTJ=1;
791   1      IF R1->DN~=NULL THEN GO TO AUG; /* FOR R1 IS NOT A ROOT,
792   1      OTHERWISE R1 IS A ROOT. */
793   1      YROOT0 = T;
794   1      CALL UPSCAN (R1,T,SETYRT0,F);
795   1      GO TO A;

***** PSEUDO NODE EXPANSION ROUTINE ****
***** PSEUDOEX: ****
797   1      P=PX;
798   1      RUNSTAT(4)=RUNSTAT(4)+1;
799   1      IF TRACE THEN PUT EDIT('*BLOSS - EXPAND PSEUDONODE ',UNSPEC(P))
              (SKIP,A,F(10));
800   1      CALL EXPAND(P); /* EXPAND THE PSEUDONODE */
801   1      /* R1 IS THE ROOT OF THE BLOSSOM,
802   1      JIN IS THE EDGE FOR WHICH X(JIN)~=0,
803   1      BROOT IS THE EDGE OF JIN IN THE BLOSSOM. */
804   1      J=R1->EDGEDN; /* BLOSSOM FORMING EDGE */

***** CASE 1: 1 EDGE INTO PSEUDO NODE ****
***** EX1: IF P->EDGEDN = JIN THEN DO; /* EASY CASE, ONE EDGE INTO P */
805   1      CALL REROOT(BROOT);
806   1      P1=P->DN;
807   1      CALL REMOVE (P);
808   1      CALL ADDON (P1,BROOT,JIN);
809   1      PX=NULL;SHRNKNG=T; /* GLOBAL VARS FOR ADDBLOS, DEFFIX */
810   1      CALL UPSCAN(BROOT,T,ADDBLOS,T,DEFFIX);
811   1      FREE P->PSEUDO;
812   1      JTST: /* CAN WE TREAT J IN A NORMAL FASHION? */
813   1      JCNT,LASTJ=J;
814   1      IF X(J)=0 THEN GO TO A;
815   1      P1=ENDS(J,1); P2=ENDS(J,2);
816   1      IF P1->DEFIC THEN
817          IF P2->DEFIC THEN /* BOTH ENDS IN DEFIC TREE */
818             GO TO DXCALC;
819          ELSE /* P2 NOT DEFIC */ GO TO GROWSTEP;
820          /* ELSE P1 IS NOT IN A DEFICIENT TREE */
821   1
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
822      1   1           IF P2->DEFIFIC THEN DO; /* SWITCH POINTERS */
824      1   2             P3=P2;P2=P1;P1=P3;
827      1   2             GO TO GROWSTEP;
828      1   2             END;
829      1   1             /* OTHERWISE NEITHER IS IN A DEFICIENT TREE, ARE THEY
830      1   1               IN DIFFERENT NONZERO COMPONENTS? */
831      1   2               P3=P1;
832      1   1               DO WHILE(P3->DN=NULL);
833      1   1                 P3=P3->DN; END;
834      1   1                 R3=P2;
835      1   2                 DO WHILE (R3->DN != NULL);
836      1   2                   R3=R3->DN; END;
837      1   1             IF R3!=P3 THEN /* DIFFERENT CMPNTS */ GO TO GROWSTEP;
838      1   1             /* ELSE WE INDICATE A NONDEFIC BLOSSOM */
839      1   1               R3->EDGEDNEJ;
840      1   1               CALL UPSCAN(R3,T,BLOSSIND,F);
841      1   1               GO TO ENDA;
842      1   1             END; /* OF EASY CASE */

/***** CASE 2: 2 EDGES INTO PSEUDO NODE *****/
/* NOW HARDER CASE : WE HAVE A DOWN EDGE AND AN UP EDGE */
843      1           JDN=P->FDGEDN;
844      1           Q3=P->DN;
845      1           Q1=ENDS(JDN,1); /* FIND EDGE OF JDN IN THE BLOSSOM */
846      1           IF Q3 = Q1 THEN Q1=ENDS(JDN,2);
847      1           /* Q1 IS TO BE THE NEW ROOT OF THE BLOSSOM */
848      1           CALL REROOT(Q1);
849      1           /* REMOVE TOP PART OF TREE */
850      1           Q2=P->UP;
851      1           CALL REMOVE(Q2);
852      1           /* ADD BLOSSOM TO THE TREE */
853      1           CALL REMOVE(P);
854      1           FREE P->PSEUDO;
855      1           CALL ADDON(Q3,Q1,JDN);
856      1           CALL UPSCAN(Q1,T,ADDFIX,F); /* LABEL NODES ODD AND EVEN */
857      1           IF BROOT->ODD THEN DO; /* THINGS WORK OUT EASILY */
858      1             SIMPFIN: PX=NULL; SHRNG=T; /*GLOBAL VARS FOR ADDBLOS-DEFFIX. */
859      1             CALL UPSCAN(Q1,T,ADDBLOS,T,DEFFIX);
860      1             CALL ADDON(BROOT,Q2,JIN); /* ADD THE TOP OF THE TREE */
861      1             GO TO JTST; /* CONTINUE AS IN EASY CASE. */
862      1             END;
863      1             /* OTHERWISE WE MAY HAVE A POLYGON IN THE PATH, OR WE MAY
864      1               JUST NEED J IN THE PATH. FIRST LABEL NODES IN POLYGON */
865      1             P1=ENDS(J,1); P2=ENDS(J,2);
866      1             DO WHILE (P1!=Q3);
867      1               P1->INPATH=T; P1=P1->DN; END;
868      1             DO WHILE(~P2->INPATH);
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
870      1      1          P2->INPATH= T;P2=P2->DN; END;
873      1          R1=P2; /* ROOT OF THE POLYGON */
/* TURN INPATH OFF IN STEM */
874      1          DO WHILE (P2->DN~=Q3);
875      1      1          P2=P2->DN; P2->INPATH=F; END;
/* NOW P->INPATH = T IFF P IS IN THE POLYGON */
/* IF BROOT IS LABELLED EVEN, AND THE PATH FROM BROOT
TO Q1 CONTAINS AT MOST ONE POLYGON NODE THEN POLYGON
IS IN PATH, OTHERWISE NOT. */
878      1          P1=BROOT;
879      1          DO WHILE(¬P1->INPATH);
880      1      1          P1=P1->DN;
881      1      1          IF P1=Q1 THEN /* AT MOST ONE PGON NODE IN PATH */
882      1      1          GO TO POLYCASE;
883      1      1          END;
884      1          P2=P1->DN;
885      1          IF ¬P2->INPATH THEN GO TO POLYCASE;
/* OTHERWISE ALL WE HAVE TO DO IS REMOVE P1->EDGEDN
FROM POLYGON AND REPLACE IT WITH J AND WE CAN TREAT AS
SIMPLE CASE */
886      1          P2=ENDS(J,1);
887      1          P3=P2; /* SEE IF P2 IS END WE WANT FOR ADDON */
888      1          DO WHILE(P3->INPATH);
889      1      1          IF P3=P1 THEN /* CORRECT, SO */ DO;
890      1      2          R3=ENDS(J,2);
891      1      2          GO TO FIN1;
892      1      2          END;
893      1      1          P3=P3->DN;
894      1      1          END;
895      1          /* OTHERWISE WE HAD IT BACKWARDS */
896      1          R3=ENDS(J,1);
897      1          P2=ENDS(J,2);
898      1          FIN1:J1=P1->EDGEDN;
899      1          CALL REMOVE(P1);
900      1          CALL REROOT(P2);
901      1          CALL ADDON (R3,P2,J);
902      1          J=J1;
903      1          GO TO SIMPFIN;
904      1          POLYCASE: /* HERE WE HAVE A POLYGON IN PATH, LABEL PATH FROM
905      1          BROOT TO POLYGON OR STEM CORRECTLY,
906      1          FIRST MARK NODES IN STEM. */
907      1          DO WHILE(R1~=Q3);
908      1      1          R1->INPATH=T; R1=R1->DN; END;
909      1          IF BROOT->INPATH THEN /* NO FIXING NECESSARY */ GO TO WINDUP;
910      1          P1=BROOT;
911      1          DO WHILE (¬P1->INPATH);
912      1      1          P2=P1; P1=P1->DN; END;
913      1      1          P1->ODD=¬P1->ODD;
914      1          CALL UPSCAN(P2,T,ADDFIX,F);
```

/*THE BLOSSOM ALGORITHM: MAIN PROCEDURE. 16-03-73 */

STMT LEVEL NEST

```
918      1          P1->ODD=>P1->ODD;
919      1          WINDUP: CALL ADDON (BROOT,Q2,JIN);
920      1          P1=ENDS(J,1); P2=ENDS(J,2);
921          /* SET UP FOR RETURN TO MAIN LOOP. */
922      1          CALL UPSCAN(Q1,T,POLYFIX,F);
923      1          FROMEX=T;
924      1          GO TO DXCALC;

/***** FINAL CORRECTION OF MATCHING IN PSEUDOS *****/
***** CORRFCTION: IF TRACE THEN PUT EDIT('*BLOSS ~ CORRECT MATCHING IN PSEUDO
925      1          NODES')(SKIP,A);
926          IF TRACE THEN CALL XOUT;
927          DO I=1 TO NNODE;
928              P1=ADDR(NODELST(I));
929          EXP1: IF P1->STACKUP=NULL THEN GO TO EXPEND;
930          P2=P1->STACKUP;
931          IF P2-> EXPANDED THEN GO TO EXPEND;
932          P3=P2->STACKUP;
933          DO WHILE((P3=NULL)&(~ P3->EXPANDED));
934              P2=P3; P3=P3->STACKUP;
935          END;
936          CALL EXPAND (P2); /* EXPAND AND KEEP THE BLOSSOM */
937          P2->EXPANDED=T;
938          GO TO EXP1;
939          EXPEND: END;
940      END BLOSSOM;
/***** END OF BLOSSOM ALGORITHM *****/
*****
```

References

- [B1] M.L. Balinski, K. Spielberg, "Methods for Integer Programming: Algebraic, Combinatorial, and Enumerative", in Progress in Operations Research Vol. III, J. Aronofsky (ed.), Wiley, New York, N.Y. 195-292 (1969).
- [B2] C. Berge, "Sur le couplage maximum d'un graph", C.R. Acad. Sci. Paris 247, 285-259 (1958).
- [B3] C. Berge, The Theory of Graphs and Its Applications, Methuen, London, England (1962).
- [B4] G. Birkhoff, S. MacLean, A Survey of Modern Algebra, Third ed., Macmillan, New York, N.Y. (1965).
- [B5] R.G. Busacker, T.L. Saaty, Finite Graphs and Networks, McGraw-Hill, New York, N.Y. (1965).
- [C1] C. Carathéodory, "Über den Variabilitätsbereich der Koeffizienten von Potenzreihen, die gegebene Werte nicht annehmen", Math. Ann. 64, 95-115 (1907).
- [D1] G. B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, N.J. (1963).
- [E1] J. Edmonds, "Paths, Trees and Flowers", Canadian J. Math. 17, 449-467 (1965).
- [E2] J. Edmonds, "Maximum Matching and a Polyhedron with 0, 1-vertices", J. Res. Nat. Bur. of Standards 69B (Math. and Math. Phys.) No. 1, 125-130 (1965).
- [E3] J. Edmonds, "An Introduction to Matching" Notes on lectures given at Ann Arbor, Michigan (1967)

- [E4] J. Edmonds, "Optimum Matchings", in manuscript.
- [E5] J. Edmonds, E.L. Johnson, "Matching: A Well-Solved Class of Integer Linear Programs", preprint: summary appears in Combinatorial Structures and their Applications, Gordon and Breach, New York, N.Y. 89-92 (1970).
- [E6] J. Edmonds, E.L. Johnson, "Matching, Euler Tours and the Chinese Postman", I.B.M. Research Report RC 3783 (1972), to appear in Math. Programming.
- [E7] J. Edmonds, E.L. Johnson, S. Lockhart, "Blossom I: A Computer Code for the Matching Problem", to appear.
- [G1] B. Grünbaum, Convex Polytopes, Interscience, London, England (1967).
- [H1] G. Hadley, Linear Programming, Addison-Wesley, Reading, Mass. (1962).
- [H2] F. Harary, Graph Theory, Addison-Wesley, Reading, Mass. (1969).
- [I1] E. Isaacson, H. Keller, Analysis of Numerical Methods, John Wiley and Sons, New York, N.Y. (1966).
- [I2] I.B.M. Systems 1360 Operating System, PL/1(F) Language Reference Manual, C28-8201 (1970)
- [J1] E.L. Johnson, "Programming in Networks and Graphs", Univ. of Calif., Berkely Research Report ORC 65-1 (1965).
- [J2] E.L. Johnson, "Networks and Basic Solutions", Operations Research 14, 619-623 (1966).
- [K1] V. Klee, C. Witzgall, "Facets and Vertices of Transportation Polytopes", Boeing Scientific Rsch. Lab. Doc. DL-82-0662, (1967).

- [K2] D. König, Theorie der endlichen und unendlichen Graphen,
Acad. Verl. M.B.H., Leipzig (1936). Reprint,
Chelsea Publishing Company, New York, N.Y. (1950).
- [K3] D. E. Knuth, The Art of Computer Programming, Vol. 1,
Fundamental Algorithms, Addison-Wesley, Reading,
Mass. (1968).
- [R1] R.T. Rockafellar, Convex Analysis, Princeton University
Press, Princeton, N.J. (1969).
- [S1] J. Stoer, C. Witzgall, Convexity and Optimization in
Finite Dimensions I, Springer-Verlag, Berlin,
Heidelberg (1970).
- [T1] W.T. Tutte, "The Factorization of Linear Graphs", J.
London Math. Soc. 22, 107-111 (1947).
- [T2] W.T. Tutte, "The Factors of Graphs", Canadian J. Math.
4, 314-328 (1952).
- [T3] W.T. Tutte, "A Short Proof of the Factor Theorem for
Finite Graphs", Canadian J. Math. 6, 347-352
(1954).