

**THE SUBTOUR POLYTOPE OF THE  
TRAVELLING SALESMAN PROBLEM**

By Sylvia Boyd

PHD THESIS

UNIVERSITY OF WATERLOO

1986

## ABSTRACT

Let  $K_n = (V, E)$  denote the complete graph on  $n$  nodes. A tour of  $K_n$  is the 0-1 incidence vector of the edge set of a hamilton cycle in  $K_n$ . Let  $T_n$  denote the set of all tours of  $K_n$ . Then, given edge weights  $c_e \in \mathbb{R}$  for all  $e \in E$ , the symmetric travelling salesman problem (henceforth denoted by TSP) is to find a member of  $T_n$  of minimum weight, i.e.

$$\min \{cx \mid x \in T_n\}.$$

An important linear programming relaxation of the TSP is the subtour polytope  $Q_S^n$  which is defined by the non-negativity, upper bound, degree and subtour constraints of the TSP. By removing the subtour constraints, a weaker relaxation is obtained, called the fractional 2-factor polytope, and denoted by  $Q_F^n$ .

In Chapter 4 we describe a primal simplex method for optimizing over  $Q_F^n$  and related problems. The algorithm is a specialized version of a simplex method for the generalized network flow problem which in general (and specifically for optimizing over  $Q_F^n$ ) is not guaranteed to be finite. However, we ensure finiteness of our method.

In Chapter 6 we introduce a finite dual simplex method for optimizing over  $Q_S^n$  and related problems. In general each pivot of this method requires an exponential number of steps, however in many

applications (including optimizing over  $Q_S^n$ ) the pivots can be performed in a polynomial number of steps.

In Chapter 7 we discuss computational results obtained from implementations of these simplex methods, showing that these methods, in combination with a novel method for predicting important subtour constraints, provide good lower bounds for the TSP.

In Chapter 5 we describe new results on  $Q_S^n$ , establishing some properties of its vertices, and optimizing a class of objective functions obtained from the clique tree inequalities over  $Q_S^n$ .

Finally, in Chapter 2 we present a useful result on deriving facet-inducing inequalities for a polytope from those of a related polytope.

### ACKNOWLEDGEMENTS

I would first like to express my sincere thanks to my supervisor, Dr. W.R. Pulleyblank, who has contributed greatly to my development as a mathematician. It was he who first introduced me to the subjects discussed in the thesis. He also provided me with guidance and encouragement during the writing of the thesis, in spite of his hectic schedule. His assistance in all aspects of this research is greatly appreciated.

Appreciation must also be expressed to my typist and friend Susie Bell for her excellent workmanship in the typing of the thesis, and to her family for their support in this endeavour. Without her considerable flexibility and cooperation, the thesis would not have been completed on time.

I am very grateful for the encouragement, love and support provided, especially during difficult times, by my good friends Jim, Colette, Cynthia and Ian. The support and prayers of others are also gratefully acknowledged.

Above all, a very special thanks must go to my husband Ian for his endless love and assistance during all stages of the degree, but especially during the writing of the thesis.

Financial assistance in the form of teaching and research assistantships from the C and O department at the University of Waterloo as well as a postgraduate scholarship from the Natural

Sciences and Engineering Research Council of Canada are acknowledged with appreciation.

TABLE OF CONTENTS

	<u>Page</u>		<u>Page</u>
ABSTRACT			
ACKNOWLEDGEMENTS			
CHAPTER 1. Introduction and Notation	1.1		
1.1 Introduction	1.1		
1.2 General Notation	1.12		
1.3 Graph Theory	1.14		
1.4 Linear Programming	1.16		
1.5 Simplex Method	1.21		
1.6 The Dual Simplex Method	1.26		
CHAPTER 2. Basic Polyhedral Theory and Facet Generating Techniques	2.1		
2.1 Polyhedra and Their Faces	2.1		
2.2 Facet-Inducing Inequalities	2.4		
2.3 Facets for Polytopes Extended by a Cone	2.8		
CHAPTER 3 The Travelling Salesman Polytope and its Facets	3.1		
3.1 The Travelling Salesman Polytope $Q_T^n$	3.2		
3.2 Simple Classes of Facets for $Q_T^n$	3.4		
3.3 Clique Tree Inequalities	3.6		
CHAPTER 4 A Simplex Method for the Fractional 2-Factor Problem	4.1		
4.1 The Node-Edge Incidence Matrix of $K_n$	4.4		
4.2 A Simplex Method for Linear Program (4.0.2)	4.16		
		4.3 Ensuring Finiteness	4.24
		4.4 Solving the Fractional 2-Factor Problem	4.30
		CHAPTER 5 The Subtour Polytope	5.1
		5.1 The Structure of the Vertices of the Subtour Polytope	5.2
		5.2 Clique Tree Inequalities and $Q_S^n$	5.14
		CHAPTER 6 A Simplex Method for the Subtour Problem	6.1
		6.1 A Dual Simplex Method for Linear Program (6.0.1)	6.4
		6.2 Ensuring Finiteness	6.29
		6.3 A Simplex Method for the Subtour Problem	6.34
		6.4 A Simplex Method for the Perfect b-Matching Problem	6.37
		CHAPTER 7 Computer Implementations and Results	7.1
		7.1 An Important Set of Cuts	7.1
		7.2 Computer Implementations	7.4
		7.3 Computational Results	7.6
		REFERENCES	R.1

## CHAPTER 1

Introduction and Notation1.1 INTRODUCTION

Consider the following problem. A salesman wishes to visit each of a prescribed set of cities exactly once, then return to the city at which he started. Find the shortest possible route for the salesman to travel.

The above problem is known as the travelling salesman problem, and is of interest both practically and theoretically. It appears to have first been formulated about 55 years ago [Menger, 1932], and has been extensively studied since its formulation as a 0-1 linear program in 1954 by [Dantzig, Fulkerson and Johnson, 1954]. Some of its practical applications include scheduling, production management, and routing. Also it has been shown [Karp, 1972] that the travelling salesman problem belongs to the class of NP-complete or "hard" combinatorial problems, and thus is of theoretical interest.

Mathematically the (symmetric) travelling salesman problem (henceforth denoted by TSP) is defined as follows: given a graph  $G = (V, E)$  and a vector  $c \in \mathbb{R}^E$  of edge costs, find a hamilton cycle in  $G$  of minimum edge cost. As is often the case, we restrict ourselves to problems in which  $G$  is the complete graph (denoted by  $K_{|V|}$ ) since any edge not existing in  $G$  can be included with a

sufficiently large cost without affecting the optimum solution.

We study the TSP from a polyhedral approach, which consists of associating a polytope with the problem. This polytope, referred to as the travelling salesman polytope and henceforth denoted by  $Q_T^n$ , is the convex hull of all 0-1 incidence vectors of edge sets of hamilton cycles in  $K_n$ . Then to solve the TSP we consider the problem

$$\min \{cx \mid x \in Q_T^n\}.$$

An extensive survey of the polyhedral aspects of the TSP can be found in [Grötschel and Padberg, 1985a] and [Grötschel and Padberg, 1985b].

In order to apply standard linear programming techniques to the problem of optimizing over  $Q_T^n$  we would like to have a complete linear description of it. Some of the necessary inequalities for  $Q_T^n$  are known. However, since optimizing over  $Q_T^n$  is an NP-hard problem the following theorem shows it unlikely we will ever find a complete description.

(1.1.1) Theorem. [Karp and Papadimitriou, 1980]. If there exists an NP-description of a class of valid inequalities that induce every facet of the polytopes of a class of NP-complete problems, then  $NP = coNP$ .

It is considered unlikely by most researchers that

NP = coNP. See [Garey and Johnson, 1979] and [Papadimitriou, 1982] for good introductions to complexity theory.

Most bounds for NP-complete combinatorial optimization problems are obtained from relaxations of the problem which are easier to solve. These bounds in turn are useful in solution methods, such as branch and bound and cutting plane methods, for these NP-complete problems.

When studying the TSP we investigate optimizing over a polytope  $Q$  which is defined by a subset of the necessary inequalities known for  $Q_T^n$ . This provides a lower bound for the TSP, and in some cases an optimal solution. Such lower bounds can prove very useful when used as a part of branch and bound techniques used to solve TSP's, or simply for bounding the error of a heuristic solution.

In general optimizing over a partial description of  $Q_T^n$  is done by a linear programming cutting plane approach. Typically in this approach a commercial linear programming code is used to optimize the objective function  $cx$  over a small subset of the necessary constraints known for  $Q_T^n$ . If the optimal solution  $x^*$  obtained corresponds to a solution for the TSP, then  $x^*$  corresponds to an optimal solution for the TSP. Otherwise we try to find a valid constraint for  $Q_T^n$  which is not satisfied by  $x^*$ , add it to our present set of constraints, and repeat.

This approach was first introduced by [Dantzig, Fulkerson

and Johnson, 1954] who used it to prove optimality for a 49-city TSP. Later it was developed and used successfully by [Grötschel, 1980] and [Padberg and Hong, 1980]. At present, the largest real world TSP solved to optimality is a 318 city problem [Crowder and Padberg, 1980]. This problem was introduced by [Lin and Kernighan, 1973] and arose from the routing of a laser drilling machine.

Let  $K_n = (V, E)$  be the complete graph on  $n$  nodes. The system of constraints most often used as a starting system for the cutting plane approach for the TSP is the following:

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 2 \quad \text{for all } i \in V$$

$$0 \leq x_{ij} \leq 1 \quad \text{for all } ij \in E.$$

We call the polytope associated with the above constraints the fractional 2-factor polytope and denote it by  $Q_F^n$ . In general, commercial linear programming packages are used to optimize over  $Q_F^n$ . However, optimizing over  $Q_F^n$  can also be solved in polynomial time as an instance of bipartite  $b$ -matching or by solving it directly using a primal-dual algorithm (see Section 4.4).

Optimizing over  $Q_F^n$  can result in solutions which are fractional. Moreover, the integer solutions may contain subtours, i.e. the edges in  $K_n$  corresponding to integer components of a solution may form cycles in  $K_n$  of size less than  $n$ . Such solutions can be "cut-off" by adding the so-called subtour elimination

constraints to the linear system for  $Q_F^n$ , thus obtaining the system

$$\begin{aligned} \sum_{j \in V \setminus \{i\}} x_{ij} &= 2 && \text{for all } i \in V, \\ 0 \leq x_{ij} &\leq 1 && \text{for all } ij \in E, \\ \sum_{i,j \in S} x_{ij} &\leq |S| - 1 && \text{for all } S \subseteq V, 3 \leq |S| \leq n - 3. \end{aligned}$$

We call the associated polytope the subtour polytope and denote it by  $Q_S^n$ .

The results of [Dantzig, Fulkerson and Johnson, 1954], [Grötschel, 1980], and [Padberg and Hong, 1980] indicate that the linear system for  $Q_S^n$  would provide a much better starting point than  $Q_F^n$  for the cutting-plane approach for the TSP. In particular,  $Q_S^n$  provides the basis for a branch and bound technique for the TSP used by [Held and Karp, 1970]. This technique, based on 1-trees, uses a Lagrangian relaxation of the TSP to obtain a lower bound for the optimal solution, and solving over this Lagrangian relaxation is equivalent to minimizing over  $Q_S^n$ . Furthermore, it was shown empirically by [Christofides, 1979] that optimizing over this Lagrangian relaxation gave approximately 99% of the optimal value for the TSP on randomly generated problems. However, no efficient means is known for solving the Lagrangian relaxation.

The separation problem can be solved in polynomial time for  $Q_S^n$ , i.e. given any  $x \in \mathbb{R}^n$  it can be determined in polynomial

time if  $x \in Q_S^n$ , and if not, an inequality in the defining system of  $Q_S^n$  which is violated by  $x$  is provided. Therefore we can optimize over  $Q_S^n$  in polynomial time by means of the ellipsoid algorithm (see [Grötschel, Lovász and Schrijver, 1981] for more details). However, there does not yet exist a direct, i.e. nonellipsoidal polynomial time algorithm to solve the problem of optimizing over  $Q_S^n$ . Also note that, unlike  $Q_F^n$ , the number of constraints in the linear system for  $Q_S^n$  is exponential in the size of the problem. Consequently, no standard implementation of the simplex method can be used to optimize over  $Q_S^n$ .

The simplex method, introduced by [Dantzig, 1947], is a method for optimizing a linear objective function subject to a finite number of linear constraints. The general idea of the method is to attempt to find a sequence of basic feasible solutions such that the corresponding objective function values are always improving until finally the optimal value is achieved. Each iteration of the simplex method consists of pivoting from the present basic feasible solution to the next one in the sequence.

In theory the number of these iterations can be exponential in the number of constraints. However, in practice the simplex method is very efficient, and it has been observed that the number of iterations required in practice grows linearly with the number of constraints (see [Kuhn and Quandt, 1963], and [Dantzig, 1963], p. 160).

In the case of combinatorial optimization problems, it is sometimes possible to design very efficient versions of the simplex method which take advantage of the special structure of the particular problem. Perhaps one of the most successful examples of this is in the case of the so-called transshipment or network flow problems (see [Chvátal, 1983]) for which the resulting simplified simplex method is called the network simplex method. Finiteness of this algorithm is guaranteed by an elegant pivot rule due to [Cunningham, 1976] which is easily implemented.

Another example of a combinatorial problem with a special structure is the perfect b-matching problem. Unlike the transshipment problem, the number of constraints in the linear system for this problem is exponential in the size of the problem. Hence any standard implementation of the simplex method is impossible. However, in [Koch, 1979], a primal simplex method is described in which all the bases encountered have special structures which allow easy computations in the corresponding pivots. Unfortunately, the method has no anti-cycling rule, and none of the general anti-cycling methods can be applied without destroying the basis structure required. Hence the method is not guaranteed to be finite.

The submodular flow problem is an example of a problem with an exponential number of constraints for which there exists a combinatorial primal simplex method which is finite. This method,

due to [F. Barahona, W. Cunningham, 1984], requires an oracle which can minimize a submodular function. In general this oracle is available only via the ellipsoid algorithm, but in several applications it is provided by an efficient combinatorial procedure.

An outline of the revised simplex method, both in the primal and dual forms, is provided in Chapter 1, along with outlines of the corresponding lexicographical methods for ensuring finiteness. All of these methods are presented in the forms most useful for our applications later. In Chapter 1 we also introduce some linear programming theory and our basic notation.

In Chapter 2 we briefly discuss the polyhedral theory and notation required for the thesis. Also included in Chapter 2 is a method for generating facets of a polytope  $\bar{P}$  from those of a polytope  $P$ , where  $\bar{P}$  is the polytope obtained by extending  $P$  by the negative of some cone  $C$  and then intersecting the resulting polyhedron with another cone  $D$ . A special instance of such a polytope is in the case where  $\bar{P}$  is the so-called monotone completion of  $P$ .

Generally in the past, the facets of a polytope  $P$  and its monotone completion  $\bar{P}$  have been studied separately, despite their close relationship. This is true for the travelling salesman polytope  $Q_T^n$  and the monotone travelling salesman polytope  $\bar{Q}_T^n$ , as well as for the linear ordering polytope  $P_{LO}^n$  and its monotone completion  $P_{AC}^n$ , the acyclic subgraph polytope (see Section 2.3).



However, we show that for any facet-inducing inequality for  $P$  which is valid for  $\bar{P}$ , there is an equivalent inequality which is also facet-inducing for  $\bar{P}$ . Note that this has previously been proven for the case  $P = Q_T^n$  in [Grötschel and Pulleyblank, 1981].

In Chapter 3, we introduce the travelling salesman polytope  $Q_T^n$  as well as several large classes of the facet-inducing inequalities for it. These classes are those which have proven most useful in cutting plane algorithms for the travelling salesman problem. We also discuss in which instances the separation problem can be solved in polynomial time for the inequalities introduced.

In Chapter 4 we introduce the fractional 2-factor problem and its associated polytope  $Q_F^n$ . Recall that optimizing over  $Q_F^n$  provides a lower bound for the TSP, which in turn is useful in branch and bound and cutting plane techniques used for solving the TSP.

We then describe a primal simplex method for a generalization of the fractional 2-factor problem. The success of the network simplex method for the upper-bounded transshipment problem indicates that such a method is desirable. At each iteration of the method we solve the original system of  $O(|V|^2)$  constraints and variables by using the structure of the bases to efficiently solve a system of  $|V|$  constraints. Note that this problem is a special case of the network flow with gains problem (or generalized network flow problem) and the simplex method

presented is a specialized implementation of a primal simplex method for the network flow with gains problem (see [Kennington and Helgason, 1980]). However, this general algorithm is at present known to be finite only for the case of positive gains (see [Elam, Glover and Klingman, 1979]) while the problem we address is an instance of network flow with negative gains. We ensure finiteness of our method by using a simplification of the lexicographical anti-cycling method as it applies to this problem.

In Chapter 5 we introduce the subtour problem and its associated polytope  $Q_S^n$ . Recall that optimizing over  $Q_S^n$  provides a lower bound for the TSP which is better than the one provided by optimizing over  $Q_F^n$ . In the first part of the chapter we examine the structure of the vertices of  $Q_S^n$ , and in the latter part we optimize a class of objective functions obtained from the so-called clique tree inequalities over  $Q_S^n$ .

In Chapter 6 we describe a dual simplex method for a generalization of the subtour and perfect b-matching problems. For these particular problems (and perhaps also for other instances of the general problem) each iteration of the method can be performed in a number of steps polynomial in the size of the problem. This is accomplished by taking advantage of the structure of the bases for these problems and also keeping the tight cuts nested at each iteration. Note that at present we know of no technique for performing primal simplex iterations in polynomial time for the

subtour problem. We ensure finiteness of the method by using a simplification of the lexicographical anti-cycling method as it applies to this problem.

The primal simplex method of [Koch, 1979] for the perfect b-matching problem is also discussed briefly in Chapter 6, along with an explanation of why it is not possible to generalize the specially structured bases he uses to the subtour problem.

Finally, in Chapter 7 we discuss computational results obtained from an implementation of the simplex method for the fractional 2-factor problem (described in Chapter 4), and an implementation of the dual simplex method for the subtour problem (described in Chapter 6).

A novel feature of the latter implementation is that when it attempts to optimize over  $Q_S^n$ , instead of starting initially with the linear system of  $Q_F^n$  and adding cuts one at a time, it predicts a set of "important" subtour constraints which it adds to the initial system all at once. It thus optimizes over a polytope  $\bar{P}$  satisfying  $Q_S^n \subseteq \bar{P} \subseteq Q_F^n$ . If the optimal solution obtained is not a member of  $Q_S^n$ , we proceed to introduce cuts in a standard fashion. These final cuts are found using an efficient heuristic procedure, and were found sufficient for obtaining an optimal solution to the subtour problem in almost all problems tested.

## 1.2 GENERAL NOTATION

For any finite set  $X$  we let  $|X|$  denote the cardinality of  $X$ . Given another set  $Y \subseteq X$ , we let  $X \setminus Y$  denote the members of  $X$  which are not members of  $Y$ .

We let  $\mathbb{R}$  denote the set of real numbers. For any  $t \in \mathbb{R}$  we use the notation  $\lfloor t \rfloor$  to denote the largest integer  $z$  such that  $z \leq t$ , and  $\lceil t \rceil$  to denote the smallest integer  $z$  such that  $z \geq t$ .

For any finite set  $E$  we let  $\mathbb{R}^E$  denote the set of all real vectors indexed by  $E$ . We let  $0$  represent the vector which is zero in all components.

Given a vector  $x \in \mathbb{R}^E$  and a subset  $J \subseteq E$  for some finite set  $E$ , we use the notation  $x_J$  to denote the vector whose components are those of  $x$  indexed by  $J$ . If  $J = \{i\}$  we use  $x_i$  rather than  $x_{\{i\}}$ . For  $a \in \mathbb{R}^E$  and  $b \in \mathbb{R}^E$  we say  $a \leq b$  if  $a_e \leq b_e$  for all  $e \in E$ .

For any finite set  $E$  the incidence vector of  $F \subseteq E$  is the vector  $x \in \mathbb{R}^E$  defined by

$$x_e = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{if } e \notin F. \end{cases}$$

For any  $y \in \mathbb{R}^E$  and  $J \subseteq E$  we let  $y(J) = \sum_{i \in J} y_i$ .

A set  $X \subseteq \mathbb{R}^E$  is linearly independent if whenever

$\sum_{x \in X} \alpha_x x = 0$  for some  $\alpha \in \mathbb{R}^X$  we have  $\alpha = 0$ . Otherwise,  $X$  is linearly dependent.

For any two finite sets  $J$  and  $K$  we let  $\mathbb{R}^{J \times K}$  denote the set of all real matrices whose rows are indexed by  $J$  and whose columns are indexed by  $K$ . For any  $B \subseteq K$  we let  $A_B$  denote the matrix whose columns are those of  $A$  indexed by  $B$ . If  $B = \{i\}$  we use  $A_j$  rather than  $A_{\{i\}}$ .

The rank or linear rank of a matrix  $A \in \mathbb{R}^{J \times K}$  is the cardinality of a maximal independent subset of the columns of  $A$ . We say  $A$  is nonsingular if the rank of  $A$  equals  $|K|$  and  $|K| = |J|$ . The row rank of  $A$  is the cardinality of a maximal independent set of rows of  $A$ . A column basis for  $A$  is a maximal independent subset of the columns of  $A$ .

For any positive integer  $n$ , the identity matrix of size  $n$ , denoted by  $I^n$ , is the matrix with  $n$  rows and  $n$  columns such that the diagonal entries have value one and all other entries have value zero. For finite sets  $J$  and  $K$  and any nonsingular matrix  $A \in \mathbb{R}^{J \times K}$  the inverse of  $A$ , denoted by  $A^{-1}$ , is defined by  $A^{-1}A = I^{|K|}$ .

Finally, for any finite sets  $J$  and  $K$  and matrix  $A \in \mathbb{R}^{J \times K}$ , the transpose of  $A$ , denoted by  $A^t$ , is the matrix whose columns are the rows of  $A$  and whose rows are the columns of  $A$ .

### 1.3 GRAPH THEORY

Some standard references on graph theory are [Bondy and Murty, 1976] and [Harary, 1969].

For our purposes a graph  $G$  is an ordered pair  $(V, E)$ , where  $V$  is a finite set of elements called nodes, and  $E$  is a finite set of elements called edges such that every edge  $e \in E$  corresponds to two distinct nodes in  $V$ , called the ends of  $e$ . An edge  $e \in E$  with ends  $u$  and  $v$  is sometimes denoted by  $uv$ , and we say  $e$  joins  $u$  and  $v$ . Two nodes  $u, v \in V$  are said to be adjacent if  $uv \in E$ , and if  $e = uv$ ,  $e$  is said to be incident with  $u$  and  $v$ .

If  $G$  is any graph, we let  $V(G)$  and  $E(G)$  denote the node set and edge set of  $G$  respectively. For any  $S \subseteq V(G)$  we use  $\gamma(S)$  to denote the set of edges in  $E(G)$  with both ends in  $S$ , and we use  $\delta(S)$  to denote the set of edges in  $E(G)$  with exactly one end in  $S$ . For any node  $v \in V(G)$  the degree of  $v$  is  $|\delta(v)|$ .

A graph  $G$  is complete if every pair of nodes is joined by exactly one edge. The complete graph on  $n$  nodes is denoted by  $K_n$ .

A graph  $H$  is called a subgraph of  $G$  if  $V(H) \subseteq V(G)$ ,  $E(H) \subseteq E(G)$  and every  $e \in E(H)$  has the same ends in  $H$  as in  $G$ . If  $V(H) \subseteq V(G)$  then  $H$  is a spanning subgraph, and if every edge of  $G$  in  $\gamma(V(H))$  is in  $E(H)$  then  $H$  is an induced subgraph of  $G$ . For any  $S \subseteq V(G)$  we use  $\langle S \rangle$  to denote the subgraph of  $G$  induced by  $S$ .

A path in a graph  $G$  is a finite non-null sequence  $v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$  whose terms are alternately nodes and edges such that the nodes are distinct, and for  $1 \leq i \leq k$  the ends of  $e_i$  are

$v_{i-1}$  and  $v_i$ . We say path  $P$  joins  $v_0$  and  $v_n$ , and sometimes we denote  $P$  by the node sequence  $v_0, v_1, \dots, v_n$ , where  $V(P) = \{v_0, v_1, \dots, v_n\}$  and  $v_{i-1}v_i \in E(P)$  for  $i = 1, 2, \dots, n$ . The length of  $P$  is  $|E(P)|$ , and a hamilton path of  $G$  is a path  $P$  in  $G$  of length  $|V(G)| - 1$ .

A graph  $G$  is connected if every two nodes in  $G$  are joined by a path. A component of a graph  $G$  is any maximal connected subgraph of  $G$ .

A cycle in  $G$  is a connected subgraph  $C$  of  $G$  such that every node in  $V(C)$  has degree 2 in  $C$ . Sometimes we denote  $P$  by the node sequence  $v_0, v_1, \dots, v_n$  where  $V(P) = \{v_0, v_1, \dots, v_n\}$ ,  $v_0v_n \in E(G)$  and  $v_{i-1}v_i \in E(G)$  for  $i = 1, 2, \dots, n$ . The length of  $C$  is  $|E(C)|$ , and a hamilton cycle of  $G$  is a cycle  $C$  in  $G$  of length  $|V(G)|$ .

A clique in a graph  $G$  is a set  $W \subseteq V(G)$  of nodes such that  $\langle W \rangle$  is complete. An articulation set of  $G$  is a set  $C$  of nodes such that removing  $C$  from  $G$  disconnects  $G$ . A cut is a set of edges  $\delta(S)$  for some  $S \subseteq V(G)$ .

A forest is a graph which contains no cycles. A tree is a connected forest. All trees  $T$  have the property that  $|E(T)| = |V(T)| - 1$ .

A graph  $G$  is bipartite if  $V(G)$  can be partitioned into two sets  $V_1$  and  $V_2$  such that  $E(G) = \delta(V_1) = \delta(V_2)$ , i.e. all edges in  $E(G)$  have one end in  $V_1$  and the other in  $V_2$ .

The node-edge incidence matrix of a graph  $G$  is a matrix  $A \in \mathbb{R}^{V(G) \times E(G)}$  such that the entry of  $A$  indexed by node  $v$  and edge  $e$  has value 1 if  $v$  is an end of  $e$ , otherwise it has value 0. Also,

for any vector  $x \in \mathbb{R}^{E(G)}$ , the support of  $x$  is the graph whose node set is  $V(G)$ , and whose edge set consists of all edges  $e \in E(G)$  such that  $x_e \neq 0$ .

A matching of a graph  $G$  is a set of edges  $M \subseteq E(G)$  such that every node  $v \in V(G)$  is the end of at most one edge in  $M$ . We say  $M$  misses  $v \in V(G)$  if  $v$  is not the end of any edge in  $M$ . A near-perfect matching of  $G$  is a matching which misses exactly one node in  $V(G)$ , and a perfect matching of  $G$  is one in which no nodes are missed.

A digraph is a graph whose edges are each given a direction. Thus each edge  $uv$  becomes an ordered pair  $(u, v)$ , and these ordered pairs are called arcs. For any arc  $(u, v)$  we call  $u$  the tail of the arc and  $v$  the head of the arc.

A dicycle in a digraph  $D$  is a non-null sequence  $v_0 a_1 v_1 a_2 v_2 \dots a_k$  whose terms are alternately nodes and edges such that the nodes are distinct,  $a_k = (v_{k-1}, v_0)$  and for  $1 \leq i \leq k - 1$  we have  $a_i = (v_{i-1}, v_i)$ . We say a digraph is acyclic if it contains no dicycles.

#### 1.4 LINEAR PROGRAMMING

A linear programming problem involves optimizing a linear objective function subject to a finite number of linear constraints. For our purposes, it is convenient to consider linear programming problems having the following form:

$$\begin{array}{ll}
 \text{minimize} & cx \\
 \text{subject to} & Ax = b \\
 & Dx \geq d \\
 & x \geq 0.
 \end{array} \tag{1.4.1}$$

Note that any general linear programming problem can be transformed into the above form (see [Dantzig, 1963], p. 85-89) and all theorems in this section can be extended in the appropriate way.

A vector  $x$  is a feasible solution to (1.4.1) if it satisfies all of the given constraints. A feasible solution is an optimal solution for (1.4.1) if it minimizes  $cx$  for all feasible solutions  $x$ .

The following is a consequence of the fundamental theorem of linear programming (see [Dantzig, 1963], p. 120).

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

- i) There exists no feasible solution.
- ii) The objective function is unbounded subject to the constraints.
- iii) There is an optimal feasible solution.

With any linear programming problem we can associate a dual linear programming problem. The original is then referred to as the primal problem. The primal and dual problems have the

property that the value of the objective function at a feasible solution in one bounds the optimal objective value in the other.

The following is the dual linear programming problem associated with (1.4.1):

$$\begin{array}{ll}
 \text{maximize} & yb + wd \\
 \text{subject to} & yA + wD \leq c \\
 & w \geq 0, y \text{ unrestricted.}
 \end{array} \tag{1.4.3}$$

Note that every constraint in the primal problem corresponds to a variable in the dual problem and vice versa.

The following three theorems describe the relationship between (1.4.1) and (1.4.3).

(1.4.4) Weak L.P. Duality Theorem (see [Dantzig, 1963], p. 130).

If  $\bar{x}$  is a feasible solution to the primal problem (1.4.1) and  $(\bar{y}, \bar{w})$  is a feasible solution to the dual problem (1.4.3) then  $c\bar{x} \geq \bar{y}b + \bar{w}d$ .

(1.4.5) Strong Duality Theorem (see [Dantzig, 1963], p. 129, 134).

If the primal problem (1.4.1) has an optimal solution  $x^*$  then the dual problem (1.4.3) has an optimal solution  $(y^*, w^*)$  and  $cx^* = y^*b + w^*d$ .

(1.4.6) Complementary Slackness Theorem (see [Dantzig, 1963],

p. 135-136). A feasible solution  $\bar{x}$  to the primal problem (1.4.1) and a feasible solution  $(\bar{y}, \bar{w})$  to the dual problem (1.4.3) are optimal if and only if

- i)  $\bar{x}_j > 0$  implies that  $(\bar{y}, \bar{w})$  satisfies the corresponding dual constraint with equality, and
- ii)  $\bar{w}_i > 0$  implies that  $\bar{x}$  satisfies the corresponding primal constraint with equality.

We sometimes refer to an inequality  $ax \leq a_0$  as tight with respect to a solution  $\bar{x}$  if  $a\bar{x} = a_0$  and slack if  $a\bar{x} < a_0$ .

The conditions i) and ii) in (1.4.6) are called the complementary slackness conditions. This theorem shows that one easy way to prove some feasible primal solution is optimal for (1.4.1) is to provide a feasible dual solution for (1.4.3) which satisfies the complementary slackness conditions.

We say that a linear programming problem is in equality form if all of its constraints are equations, i.e. it has the form

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array} \quad (1.4.7)$$

and  $A$  has full row rank. It is possible to transform (1.4.1) into the form shown in (1.4.7) by subtracting a non-negative slack variable  $s_i$  from each inequality in (1.4.1). Thus any linear programming problem can be put into

equality form.

The dual problem for (1.4.7) is as follows:

$$\begin{array}{ll} \text{maximize} & yb \\ \text{subject to} & yA \leq c \\ & y \text{ unrestricted.} \end{array} \quad (1.4.8)$$

A basis for (1.4.7) is defined by a set of column indices  $B$  corresponding to a maximal independent subset of the columns of  $A$ . The variables  $x_B$  are called basic variables, and the columns of  $A_B$  are called basic columns. If we let  $N$  represent the set of indices of non-basic columns then the unique basic primal solution for (1.4.7) corresponding to  $B$  is defined by

$$A_B x_B = b, \quad x_N = 0 \quad (1.4.9)$$

and the unique basic dual solution for (1.4.8) is defined by

$$yA_B = c_B. \quad (1.4.10)$$

Note that these basic solutions are not necessarily feasible solutions for their respective problems, but they do satisfy the complementary slackness conditions. Hence if they are both feasible they are optimal solutions as well by the Complementary Slackness Theorem (1.4.6).

## 1.5 THE SIMPLEX METHOD

The simplex method [Dantzig, 1947] is an algorithm for solving linear programming problems which is usually expressed for problems in equality form. When it is performed on the primal linear program it is called the primal simplex method, but it can also be applied to the dual, in which case it is called the dual simplex method. In this section we describe the so-called revised simplex method performed on the primal linear program (1.4.7). In Section 1.6 we outline the corresponding procedure for applying this method to the associated dual problem.

The general idea of the simplex method is to attempt to find a sequence of basic feasible solutions such that the corresponding objective function values are always improving until finally the optimal value is achieved. The algorithm goes from one primal feasible basis to the next one in the sequence by carefully choosing the index of some non-basic column to enter the basis and the index of some basic column to leave it. The corresponding variables are called the entering variable and the leaving variable respectively, and performing the exchange is called a pivot. The choice of entering variable is motivated by the desire to improve, i.e. decrease our objective function value. The choice of leaving variable is generally forced by the requirement that the new primal basic solution be feasible, i.e. we wish to keep  $x \geq 0$ .

To start the simplex method we need to find an initial

primal feasible basis, if one exists. There are general procedures used for this, however they are not required for our specific applications and hence are not discussed here.

Once we have an initial feasible basis we begin pivoting. At each iteration we are given the current primal feasible basis  $B$ . Letting  $N$  be the set of indices of non-basic columns we can express our objective function value  $z$  in terms of the non-basic variables  $x_N$  as follows:

$$z = c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N. \quad (1.5.1)$$

Since  $x_N = 0$  for our present primal basic solution, our current objective value is  $c_B A_B^{-1} b$ .

Given a primal feasible basis  $B$ , the next step in the simplex method is to calculate the corresponding dual solution  $\bar{y}$  as in (1.4.10). We check to see if  $\bar{y}$  is dual feasible by calculating  $\bar{c}_N = c_N - \bar{y} A_N$ , called the reduced cost for the iteration. If  $\bar{c}_N \geq 0$  then we have dual feasibility and hence optimality for both the primal and the dual problems by the Complementary Slackness Theorem (1.4.6). Otherwise there exists some  $j$  such that  $\bar{c}_j < 0$  and we choose the corresponding  $x_j$  as our entering variable and add  $j$  to our basis. We then let the value of  $x_j$  increase from its present value of 0 up to some value  $t \geq 0$ . This results in a decrease of  $-\bar{c}_j t$  in our objective function value, as can be seen by substituting  $\bar{c}_N = c_N - c_B A_B^{-1} A_N$  into (1.5.1).

The requirement that we remain primal feasible limits how large we can make  $t$ . Thus we require

$$x_B = A_B^{-1}b - t(A_B^{-1}A_j) \geq 0. \quad (1.5.2)$$

Calculating  $\bar{b} = A_B^{-1}b$  and  $\bar{a} = A_B^{-1}A_j$ , we satisfy (1.5.2) by defining  $t$  as follows:

$$t = \min \left\{ \frac{\bar{b}_i}{\bar{a}_i} \mid \bar{a}_i > 0, i \in B \right\}.$$

This portion of the simplex method is commonly referred to as the ratio test.

If  $t$  has no limit, i.e.  $\bar{a}_i \leq 0$  for all  $i$ , then our objective function can be made arbitrarily negative and thus our primal linear program is unbounded. Otherwise at least one basic variable  $x_i$  is forced to have value zero by our choice of  $t$ , and we choose such an  $x_i$  as the leaving variable for the pivot. The new primal feasible basis  $B'$  is then defined by  $B' = (B \cup \{j\}) \setminus \{i\}$ .

If we have  $t > 0$  for every iteration, our objective values are strictly decreasing, guaranteeing each basis encountered is different from the rest. Such a guarantee implies termination of the algorithm since the number of possible bases is clearly finite.

It may instead happen that  $t = 0$  for some sequence of consecutive pivots. Such pivots are called degenerate, and can

cause the algorithm to repeat some basis from a previous iteration. This phenomenon is known as cycling, and if it occurs the simplex method may fail to terminate.

Fortunately there are various ways of preventing the occurrence of cycling. For example, the perturbation method and the Lexicographic method guarantee no basis is repeated by forcing the choice of leaving variable at each iteration of the simplex method.

In the perturbation method for preventing cycling we perturb the right-hand sides  $b_1, b_2, \dots, b_m$  in (1.4.7) by small positive amounts  $\epsilon_1, \epsilon_2, \dots, \epsilon_m$  which are chosen different enough to guarantee every pivot will be nondegenerate, yet small enough for the problem to be considered unchanged for all practical purposes. We then apply the simplex method to the linear program (1.4.7) with  $b$  replaced by  $b' = b + e$ , where  $e$  is the vector whose  $i^{\text{th}}$  component is  $\epsilon_i$ . Consequently, given a basis  $B$  for (1.4.7) and an entering column  $k$ , the corresponding value of  $x_B$  as given in (1.5.2) becomes

$$x_B = A_B^{-1}(b + e) - t A_B^{-1}k.$$

For suitable choices of  $\epsilon_1, \epsilon_2, \dots, \epsilon_m$  the leaving variable  $x_j$  will be uniquely determined, for there will be a unique  $j \in \{1, 2, \dots, m\}$  such that

$$\frac{(A_B^{-1}b)_j + \bar{a}^j e}{(A_B^{-1}k)_j} = \min_i \left\{ \frac{(A_B^{-1}b)_i + \bar{a}^i e}{(A_B^{-1}k)_i} \mid (A_B^{-1}k)_i > 0 \right\}, \quad (1.5.3)$$



where  $\bar{a}^i$  represents the  $i^{\text{th}}$  row of  $A_B^{-1}$ .

The lexicographic method for preventing cycling makes use of a lexicographic rule for ordering vectors. Given two vectors  $a$  and  $b$  of length  $m$ , we say  $a$  is lexicographically less than  $b$  if there exists  $l \in \{1, 2, \dots, m\}$  such that  $a_i = b_i$  for  $i = 1, 2, \dots, l-1$  and  $a_l < b_l$ .

The lexicographic method is equivalent to the implementation of the perturbation method in which we define  $\epsilon_i = \epsilon^i$  for some fixed  $\epsilon > 0$  which is sufficiently small. However, rather than actually compute a suitable value for  $\epsilon$ , we treat the perturbed right-hand sides of (1.4.7) as formal polynomials in  $\epsilon$  and keep track of the corresponding coefficient vectors as we pivot.

Given a basis  $B$  for (1.4.7) and an entering column  $k$ , the corresponding  $i^{\text{th}}$  coefficient vector of  $e = (\epsilon, \epsilon^2, \epsilon^3, \dots, \epsilon^m)$  is given by

$$d^i = \frac{1}{(A_B^{-1}k)_i} ((A_B^{-1}b)_i, \bar{a}^i), \quad (1.5.4)$$

where  $\bar{a}^i$  denotes the  $i^{\text{th}}$  row of  $A_B^{-1}$ . It follows that, for  $\epsilon$  sufficiently small, finding  $j \in \{1, 2, \dots, m\}$  such that (1.5.3) holds is equivalent to finding the unique lexicographically minimum vector among  $\{d^i \mid (A_B^{-1}k)_i > 0, i = 1, 2, 3, \dots, m\}$  for  $d^i$  as defined in (1.5.4). We then choose  $x_j$  to leave the basis.

Note that if there is a unique  $i$  for which  $\frac{1}{(A_B^{-1}k)_i} (A_B^{-1}b)_i$

is minimum, the lexicographic method chooses the same leaving variable as the usual simplex method. Thus we need only implement this method for choosing the leaving variable if there is a tie between two or more variables in the pivot being performed. If  $T \subseteq \{1, 2, \dots, m\}$  is the set of indices of tied variables, we break the tie by finding the lexicographically minimum vector among  $\{\frac{1}{(A_B^{-1}k)_i} \bar{a}^i \mid i \in T\}$ . Thus we need only calculate the rows of  $A_B^{-1}$  corresponding to  $T$ .

For more information on the simplex algorithm or more details about the perturbation and lexicographic anti-cycling methods, see [Chvátal, 1983].

## 1.6 THE DUAL SIMPLEX METHOD

In this section we describe the so-called dual revised simplex method performed on the linear program (1.4.7) shown below.

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & Ax = b \\ & x \geq 0. \end{array} \quad (1.4.7)$$

The dual linear program for (1.4.7) is as follows:

$$\begin{array}{ll} \text{maximize} & yb \\ \text{subject to} & yA \leq c \\ & y \text{ unrestricted.} \end{array} \quad (1.4.8)$$

Recall that the solution  $\bar{y}$  for (1.4.8) corresponding to a basis  $B$  is given by  $\bar{y} = c_B A_B^{-1}$ , and  $B$  is called dual feasible if  $\bar{y}A \leq c$ .

As mentioned earlier, the dual simplex method is simply the revised simplex method described in Section 1.5 performed on dual linear program (1.4.8). However, for our applications of the method it is useful to have an outline of the specific steps involved.

The general idea of the dual simplex method is to attempt to find a sequence of basic dual feasible solutions such that the corresponding dual objective function values are always improving until finally the optimal value is achieved. The algorithm goes from the present dual feasible basis  $B$  to the next one in the sequence by carefully choosing the index of one of the basic variables  $x_B$  to leave the basis and the index of one of the non-basic variables to enter it. These variables are called the leaving variable and the entering variable respectively. The choice of leaving variable is motivated by the desire to improve, i.e. increase our dual objective function value. The choice of entering variable is generally forced by the requirement that the new basic dual solution be feasible.

We start the method by finding an initial dual feasible solution, if one exists. Then at each iteration of the algorithm we have a current dual feasible basis  $B$ . Letting  $\bar{c} = c - yA$  represent the reduced costs for any feasible solution  $y$  for (1.4.8), we can express our dual objective function  $z$  in terms of  $\bar{c}_B$  as follows:

$$z = c_B A_B^{-1} b - \bar{c}_B A_B^{-1} b. \quad (1.6.1)$$

Since  $\bar{c}_B = 0$  for our present basic dual solution, our current objective value is  $c_B A_B^{-1} b$ .

Letting  $N$  be the set of indices of non-basic columns, we find the basic primal solution  $(\bar{x}_B, \bar{x}_N)$  corresponding to the current basis  $B$  as in (1.4.9), and we find the current basic dual solution  $\bar{y} = c_B A_B^{-1}$ . If  $\bar{x}_B \geq 0$  then we have primal feasibility and hence optimality for both the primal and the dual problems by the Complementary Slackness Theorem (1.4.6). Otherwise  $\bar{x}_i < 0$  for some  $i \in B$ , and we choose  $x_i$  as our leaving variable. We then modify  $y$  in such a way that the value of the reduced cost  $\bar{c}_i = c_i - yA_i$  increases from its present value of 0 up to some value  $t \geq 0$ . This results in an increase of  $-\bar{x}_i t$  in the dual objective value, as can be seen by substituting  $\bar{x}_B = A_B^{-1} b$  into (1.6.1). It also results in a new dual solution  $y$  which is defined by

$$y = \bar{y} - t\bar{a} \quad (1.6.2)$$

where  $\bar{a}$  is the row of  $A_B^{-1}$  corresponding to the leaving variable  $x_i$ .

The requirement that we remain dual feasible limits how large we can make  $t$ . Thus we require our new reduced costs  $\bar{c}_N = c_N - yA_N$  to be non-negative, i.e. we require

$$\bar{y}A_N - t\bar{a}A_N \leq c_N. \quad (1.6.3)$$

We satisfy (1.6.3) by defining  $t$  as

$$t = \min \left\{ \frac{c_j - \bar{y}A_j}{-\bar{a}A_j} \mid j \in N, \bar{a}A_j < 0 \right\}. \quad (1.6.4)$$

If  $t$  has no limit, then the dual linear program (1.4.8) is unbounded, and hence the primal linear program (1.4.7) has no feasible solution. Otherwise some reduced cost  $\bar{c}_j$  is forced to have value zero by our choice of  $t$ , and we choose the corresponding nonbasic variable  $x_j$  as the entering variable for the pivot. The new dual feasible basis  $B'$  is then defined by  $B' = (B \cup \{j\}) \setminus \{i\}$ .

For more details on the revised dual simplex algorithm, see [Chvátal, 1983].

We can ensure finiteness of the dual simplex method by applying a dual lexicographic method for choosing the entering variable. This method is simply the lexicographic method described in Section 1.5 in the case where the simplex method is applied to the dual linear program (1.4.8). However, to make it more convenient to implement the method later, we describe the specific steps involved.

In the dual lexicographic method for preventing cycling, we replace each right-hand side  $c_i$  of the dual linear program (1.4.8) by  $c_i + \epsilon_i$ , where  $\epsilon_i = \epsilon^i$  for some fixed  $\epsilon > 0$  which is assumed to be sufficiently small in value for our purposes. Then if there is a tie for the entering variable as we pivot, we break the

tie by lexicographically comparing the corresponding coefficient vectors of  $e$  in (1.6.4).

More specifically, let  $B$  be any basis for (1.4.7), and let  $N$  be the indices of the non-basic columns. The corresponding solution for (1.4.8) with  $c$  replaced by  $c + e$  is

$$\bar{y} = c_B A_B^{-1} + e_B A_B^{-1}.$$

Substituting this into (1.6.4) gives

$$t = \min \left\{ \frac{(c_j - c_B A_B^{-1} A_j) + (e_j - e_B A_B^{-1} A_j)}{-\bar{a} A_j} \mid j \in N, \bar{a} A_j < 0 \right\},$$

where  $\bar{a}$  is the row of  $A_B^{-1}$  corresponding to the leaving variable  $x_i$ . Thus if there existed a tie in (1.6.4) between the indices  $T \subseteq N$ , we would break the tie for the entering variable by finding the unique lexicographically minimum vector among

$$\left\{ \frac{1}{-\bar{a} A_j} k^j \mid j \in T \right\}, \quad (1.6.5)$$

where  $k^j, j \in N$  is defined by

$$k_m^j = \begin{cases} 1 & \text{if } m = j \\ (-A_B^{-1} A_j)_m & \text{if } m \in B \\ 0 & \text{otherwise.} \end{cases}$$

If the minimum vector is  $k^n$ , the corresponding variable  $x_n$  is then chosen to enter the basis.

## CHAPTER 2

Basic Polyhedral Theory and Facet Generating Techniques

In the first part of this chapter we define the basic terminology of polyhedral theory and discuss some of the important results in this area. In the latter portion of the chapter we examine methods for generating valid inequalities for a set  $S \subseteq \mathbb{R}^E$  and prove some general results concerning these.

Our discussion of polyhedral theory is brief and covers only what is essential for later sections. More detailed treatments of the subject can be found in [Bachem and Grötschel, 1982], [Rockafellar, 1970] and [Stoer and Witzgall, 1970].

2.1 POLYHEDRA AND THEIR FACES

For any finite set  $E$  let  $\mathbb{R}^E$  denote the set of all real vectors indexed by  $E$ . For any  $X \subseteq \mathbb{R}^E$  the convex hull of  $X$ , denoted by  $\text{conv}(X)$ , is the set of all  $y \in \mathbb{R}^E$  such that  $y$  can be expressed as a convex combination of a finite subset of the members of  $X$ , i.e.  $\text{conv}(X) = \{y \in \mathbb{R}^E \mid y = \sum(\lambda_x x \mid x \in \bar{X}) \text{ for some finite } \bar{X} \subseteq X \text{ and some } \lambda \in \mathbb{R}^{\bar{X}} \text{ such that } \sum(\lambda_x \mid x \in \bar{X}) = 1, \lambda \geq 0\}$ .

A halfspace is a set  $H \subseteq \mathbb{R}^E$  of the form  $\{ax \leq a_0\}$  and a hyperplane is a set  $L \subseteq \mathbb{R}^E$  of the form  $\{ax = a_0\}$  for some  $a \in \mathbb{R}^E \setminus \{0\}$  and  $a_0 \in \mathbb{R}$ . We say that  $H$  is defined by the inequality  $ax \leq a_0$  and  $L$  is defined by the equation  $ax = a_0$ . An inequality

$ax \leq a_0$  is valid for some  $S \subseteq \mathbb{R}^E$  if  $S$  is contained in the halfspace defined by  $ax \leq a_0$ . This valid inequality is called supporting for  $S$  if the intersection of  $S$  with the hyperplane defined by  $ax = a_0$  is non-empty.

A polyhedron  $P \subseteq \mathbb{R}^E$  is the intersection of finitely many halfspaces. Equivalently,  $P$  is the solution set of a finite system of linear equations and inequalities, and can be expressed in the form  $P = \{x \in \mathbb{R}^E \mid Dx = d, Ax \leq b\}$ . A polytope is a polyhedron which is bounded.

A polyhedron  $C$  is called a cone if it is the solution set of a homogeneous linear system, i.e. if it can be expressed in the form  $C = \{w \in \mathbb{R}^E \mid wA \geq 0\}$ . We say  $C$  is pointed if  $\{w \in C \mid wA = 0\} = \{0\}$ . For any  $X \subseteq \mathbb{R}^E$  the cone of  $X$ , denoted by  $\text{cone}(X)$ , is the set of all non-negative linear combinations of members of  $X$ , i.e.  $\text{cone}(X) = \{y \in \mathbb{R}^E \mid y = \sum(\alpha_x x \mid x \in X) \text{ for some } \alpha \in \mathbb{R}^X \text{ such that } \alpha \geq 0\}$ . The following result is due to [Weyl, 1935].

(2.1.1) Theorem. For every finite  $X \subseteq \mathbb{R}^E$ ,  $\text{cone}(X)$  is a cone.

The converse of (2.1.1) is due to Minkowski (see [Bachem and Grötschel, 1982]).

(2.1.2) Theorem. For every cone  $C \subseteq \mathbb{R}^E$  there exists a finite set  $X \subseteq \mathbb{R}^E$  such that  $C = \text{cone}(X)$ .

For any cone  $C \subseteq \mathbb{R}^E$  we call  $X \subseteq \mathbb{R}^E$  a generating set for  $C$  if  $C = \text{cone}(X)$ . We then say  $X$  generates  $C$ , and call the points in  $X$  generators for  $C$ .

The following is a fundamental result which is due to [Goldman, 1956] and deducible from (2.1.1) and (2.1.2). The necessary portion of the theorem also follows from [Motzkin, 1936].

For convenience, given  $P \subseteq \mathbb{R}^E$  and  $C \subseteq \mathbb{R}^E$  we use the notation  $P + C$  to denote  $\{x \in \mathbb{R}^E \mid x = v + r \text{ for some } v \in P \text{ and } r \in C\}$  and define  $P - C$  in a similar fashion.

(2.1.3) Theorem.  $P \subseteq \mathbb{R}^E$  is a polyhedron if and only if there exist finite sets  $V$  and  $R \subseteq \mathbb{R}^E$  such that  $P = \text{conv}(V) + \text{cone}(R)$ .

Note that  $P$  is a polytope if and only if the set  $R$  above is empty, i.e. if and only if  $P = \text{conv}(V)$  for some  $V \subseteq \mathbb{R}^E$ .

A subset  $F$  of a polyhedron  $P$  is a face of  $P$  if  $F$  is either the empty set or else the polyhedron obtained by taking the linear system which defines  $P$  and replacing some of the inequalities with the corresponding equations. A face  $F$  of  $P$  is called proper if  $F \neq P$ . A point  $x \in P$  is called an interior point of  $P$  if  $x$  belongs to no proper face of  $P$ .

Linear programming duality can be used to prove the following (see [Stoer and Witzgall, 1970], p. 43).

(2.1.4) Theorem. Let  $P \subseteq \mathbb{R}^E$  be a polyhedron and let  $F$  be a nonempty subset of  $P$ . Then  $F$  is a face of  $P$  if and only if there exists  $a \in \mathbb{R}^E$  and  $a_0 \in \mathbb{R}$  such that  $F = \{x \in P \mid ax = a_0\}$ .

If  $ax \leq a_0$  is a valid inequality for  $P$  then we say the face  $F = \{x \in P \mid ax = a_0\}$  is induced by  $ax \leq a_0$ .

Finally, two valid inequalities  $ax \leq a_0$  and  $bx \leq b_0$  for  $P$  are called equivalent with respect to  $P$  if they induce the same face, i.e.  $\{x \in P \mid ax = a_0\} = \{x \in P \mid bx = b_0\}$ .

## 2.2 FACET INDUCING INEQUALITIES

A finite set  $X \subseteq \mathbb{R}^E$  is affinely independent if whenever  $\sum(\lambda_x x \mid x \in X) = 0$  and  $\sum(\lambda_x \mid x \in X) = 0$  for some  $\lambda \in \mathbb{R}^X$  we also have  $\lambda = 0$ .

Given  $X \subseteq \mathbb{R}^E$  the affine hull of  $X$ , denoted by  $\text{aff}(X)$ , is the set of all  $y \in \mathbb{R}^E$  for which there exists a finite set  $\tilde{X} \subseteq X$  such that  $y = \sum(\lambda_x x \mid x \in \tilde{X})$  and  $\sum(\lambda_x \mid x \in \tilde{X}) = 1$  for some  $\lambda \in \mathbb{R}^{\tilde{X}}$ . Note that for any  $S \subseteq \mathbb{R}^E$  there exists a set  $X \subseteq S$  such that  $S = \text{aff}(X)$ ; in particular,  $S$  is the affine hull of a largest affinely independent subset of itself.

Unlike linear independence, affine independence has the

property that it is invariant under translations of the origin. The affine rank of a set  $S \subseteq \mathbb{R}^E$  is the cardinality of a largest affinely independent subset of  $S$  and is denoted by  $r_a(S)$ . It is related to the linear rank of  $S$ , denoted  $r_\ell(S)$ , in the following way.

(2.2.1) Proposition. For any  $S \subseteq \mathbb{R}^E$ , if  $0 \in \text{aff}(S)$  then  $r_a(S) = r_\ell(S) + 1$ , otherwise  $r_a(S) = r_\ell(S)$ .

The dimension of a polyhedron  $P \subseteq \mathbb{R}^E$  is defined as  $r_a(P) - 1$  and denoted by  $\dim(P)$ . We say  $P$  is of full dimension if  $\dim(P) = |E|$  or equivalently if there does not exist some linear equation  $ax = a$  satisfied by all  $x \in P$ . Given a linear system defining  $P$ , the set of constraints which are satisfied with equality by all  $x \in P$  is called the equality set or equation system and is related to the dimension of  $P$  in the following way (see [Pulleyblank, 1973], Section 2.2).

(2.2.2) Theorem. Let  $P \subseteq \mathbb{R}^E$  be a non-empty polyhedron and let  $Ax = b$  represent the equality set for  $P$ . Then if the linear rank of  $A$  is  $\rho$ , we have  $\dim(P) = n - \rho$ .

Any proper face of a non-empty polyhedron  $P$  is called an edge if it has dimension 1, and a vertex if it has dimension 0 (i.e. if it consists of a single element). The set  $X$  of vertices of a

polytope  $P$  have the property that  $P = \text{conv}(X)$ ; thus optimizing some objective function  $cx$  over polytope  $P$  is equivalent to optimizing  $cx$  over  $X$ .

Often in combinatorial optimization the description of a polyhedron  $P$  is given in terms of its vertices, precluding the use of linear programming techniques on the associated problem. Thus a major problem is to find a finite linear system which defines  $P$ , hopefully having as few constraints as possible. To do this requires the finding of inequalities which induce maximal nonempty proper faces of  $P$ , called the facets of  $P$ . The inequalities which induce facets are called facet-inducing inequalities. The following gives the relationship between facets and a minimal defining linear system for a polyhedron  $P$  (see [Pulleyblank, 1973], Section 2.3).

(2.2.3) Theorem. Let  $P \subseteq \mathbb{R}^E$  be a polyhedron and suppose  $P = \{x \in \mathbb{R}^E \mid Ax \leq b, Dx = d\}$ . Then this is a minimal linear system sufficient to define  $P$  if and only if

- i) the rows of  $D$  are linearly independent and  $Dx = d$  forms the equality set, and
- ii) each constraint of  $Ax \leq b$  induces a distinct facet of  $P$ .

Finding a complete minimal linear system which defines a polyhedron  $P$  is often a difficult problem, but even a partial description can be useful for combinatorial optimization problems.

Thus, the problem of identifying facet-inducing inequalities for  $P$  is an important one. The following theorem provides two basic methods for proving an inequality is facet inducing (see [Pulleyblank, 1973], Chapter 2).

(2.2.4) Theorem. Let  $F$  be a non-empty proper face of  $P = \{x \in \mathbb{R}^E \mid Ax \leq b, Dx = d\}$ , let  $Dx = d$  be an equality set of  $P$  and let the rows of  $D$  be indexed by a finite set  $I$ . Then the following statements are equivalent:

- i)  $F$  is a facet of  $P$ .
- ii)  $\dim(F) = \dim(P) - 1$ .
- iii) For any  $a, \bar{a} \in \mathbb{R}^E$  and  $\alpha, \bar{\alpha} \in \mathbb{R}$  satisfying  $F = \{x \in P \mid ax = \alpha\} = \{x \in P \mid \bar{a}x = \bar{\alpha}\}$  there exist  $\lambda \in \mathbb{R}^I$  and positive  $\gamma \in \mathbb{R}$  such that  $\bar{a} = \gamma a + \lambda D$  and  $\bar{\alpha} = \gamma \alpha + \lambda d$ .

Using Theorem (2.2.4), one method of showing a valid inequality  $ax \leq \alpha$  for polyhedron  $P$  is facet-inducing is to exhibit a set of  $k = \dim(P)$  affinely independent vectors  $x_1, x_2, \dots, x_k \in P$  which satisfy  $ax_i = \alpha$  for  $i = 1, 2, \dots, k$ , and show  $ax \leq \alpha$  does not induce  $P$ . The second method involves assuming there exists a valid inequality  $\bar{a}x \leq \bar{\alpha}$  for  $P$  such that

$$\emptyset \neq F = \{x \in P \mid ax = \alpha\} \subsetneq \{x \in P \mid \bar{a}x = \bar{\alpha}\} \neq P.$$

Then using the properties of the points in  $F$  we show it must be the case that  $\bar{a} = \gamma a + \lambda D$  for some  $\lambda \in \mathbb{R}^I$  and positive scalar  $\gamma$ . This implies that  $\bar{a} = \gamma a + \lambda d$  since  $\bar{a}x \leq \bar{\alpha}$  is both valid and supporting.

### 2.3 FACETS FOR POLYTOPES EXTENDED BY A CONE

Let  $P \subseteq \mathbb{R}^E$  be a polytope which lies in the non-negative orthant of  $\mathbb{R}^E$ . The monotone completion  $\bar{P}$  of  $P$  is defined by  $\bar{P} = \{x \in \mathbb{R}^E \mid 0 \leq x \leq x' \text{ for some } x' \in P\}$ . Letting  $C$  be the non-negative orthant of  $\mathbb{R}^E$ , we can equivalently describe  $\bar{P}$  as  $(P - C) \cap C$ , i.e.  $P$  extended by the negative of cone  $C$  and then intersected with  $C$ . The resulting  $\bar{P}$  is clearly a polytope.

Often when  $P$  is a polytope arising from a combinatorial problem, the monotone completion of  $P$  also represents a problem of interest. For example, consider the linear ordering problem which can be defined as follows. Let  $D_n = (V, A_n)$  be the complete digraph on  $n$  nodes. A tournament in  $D_n$  is a subgraph  $D = (V, A)$  of  $D_n$  such that for every two nodes  $u, v \in V$  there exists exactly one arc in  $A$  with endnodes  $u$  and  $v$ . Given a positive integer  $n$  and a vector  $c \in \mathbb{R}^{A_n}$  of real arc weights, the linear ordering problem is to find an acyclic tournament  $(V, T)$  in  $D_n$  such that  $c(T)$  is maximized. The linear ordering polytope is denoted by  $P_{LO}^n$  in [Grötschel, Jünger and Reinelt, 1982a] and defined as

$$P_{LO}^n = \text{conv}\{x^T \in \mathbb{R}^{A_n} \mid T \text{ is the arc set of an acyclic tournament of } D_n\}.$$

The monotone completion of  $P_{LO}^n$  is the so called acyclic subgraph polytope denoted by  $P_{AC}^n$  in [Grötschel, Jünger and Reinelt, 1982b] and defined by

$$P_{AC}^n = \text{conv}\{x^A \in \mathbb{R}^{A_n} \mid A \text{ is the arc set of an acyclic subgraph of } D_n\}.$$

Another example is the travelling salesman problem whose associated polytope  $Q_T^n$  has as its monotone completion the so-called monotone travelling salesman polytope, denoted by  $\bar{Q}_T^n$  and defined as  $\bar{Q}_T^n = \{x^T \in \mathbb{R}^E \mid T \text{ is a subset of the edge set of a hamilton cycle in } K_n = (V, E)\}$ .

Generally in the past the facets of a polytope  $P$  and its monotone completion  $\bar{P}$  have been studied separately, despite their close relationship. Although this was originally the case for  $Q_T^n$  and  $\bar{Q}_T^n$ , it was shown in [Grötschel and Pulleyblank, 1981] that for any facet-inducing inequality of  $Q_T^n$  which is valid for  $\bar{Q}_T^n$  there is an equivalent inequality which is also facet-inducing for  $\bar{Q}_T^n$ . In this section we prove a more general result of the same nature using some similar techniques.

Let  $P \subseteq \mathbb{R}^E$  be a polytope and let  $C \subseteq \mathbb{R}^E$ ,  $D \subseteq \mathbb{R}^E$  be cones such that  $D$  contains  $P$  and  $C$  is pointed. We can generalize the notion of monotone completion by considering  $P$  extended by the negative of cone  $C$  and then intersected with cone  $D$ , i.e. we consider  $Q = (P - C) \cap D$ . In particular we are interested in the cases in which  $P$  is not of full dimension and  $Q$  is of full dimension, and wish to determine the conditions under which a facet-inducing inequality for such a  $P$  will also be facet-inducing for such a  $Q$ . Before answering this question we require some preliminaries.

For polytope  $P \subseteq \mathbb{R}^E$  not of full dimension let  $Ax = c$  be a

minimal equation system for  $P$  and for any  $b \in \mathbb{R}^E$  and finite  $K \subseteq \mathbb{R}^E$  define  $K^0(b, K) = \{g \in K \mid bg=0\}$ . Let  $G \in \mathbb{R}^{E \times K}$  be the matrix whose columns are the elements of  $K$ . Then we call a facet-inducing inequality  $ax \leq a_0$  for  $P$  support reduced with respect to  $K$  if  $\{Ag \mid g \in K^0(a, K)\}$  contains a column basis of the matrix  $AG$ . Note that if  $P$  is of full dimension then any facet-inducing inequality for  $P$  is support reduced with respect to  $K$ .

(2.3.1) Lemma. Let  $P \subseteq \mathbb{R}^E$  be a polytope not of full dimension with minimal equation system  $Ax = c$ ,  $A \in \mathbb{R}^{L \times E}$ ,  $c \in \mathbb{R}^L$ . Let  $C \subseteq \mathbb{R}^E$  be a pointed cone with generating set  $K$ , and let  $G \in \mathbb{R}^{E \times K}$  be the matrix whose columns are the elements of  $K$ . Then  $P - C$  is of full dimension if and only if  $AG$  has full row rank.

Proof. Suppose matrix  $AG$  does not have full row rank. Then there exists  $\lambda \in \mathbb{R}^L$ ,  $\lambda \neq 0$  such that  $\lambda AG = 0$ . Let  $a = \lambda A$ , and consider  $az$  for any  $z \in P - C$ . Since  $z \in P - C$ ,  $z = x - Gy$  for some  $x \in P$  and  $y \in \mathbb{R}^K$ ,  $y \geq 0$ . Thus

$$\begin{aligned} az &= ax - aGy \\ &= \lambda Ax - \lambda AGy \\ &= c \end{aligned}$$

and it follows from (2.2.2) that  $P - C$  is not of full dimension

Conversely, suppose  $P - C$  is not of full dimension. Then by (2.2.2) there exists a non-zero vector  $a \in \mathbb{R}^E$  and  $a_0 \in \mathbb{R}$  such



that  $ax = a_0$  for all  $x \in P-C$ . Since  $0 \in C$ ,  $ax = a_0$  for all  $x \in P$  and thus  $a = \lambda A$  for some  $\lambda \neq 0$ ,  $\lambda \in \mathbb{R}$  (since  $Ax = 0$  is a minimal equation system for  $P$ ). Since  $ax = c$  for all  $x \in P - C$ , it thus follows that  $-aGy = 0$  for all  $y \geq 0$ ,  $y \in \mathbb{R}^K$  and consequently  $\lambda AG = 0$ . Therefore  $AG$  does not have full row rank.  $\square$

The following is the main result of this section.

(2.3.2) Theorem. Let  $P \subseteq \mathbb{R}^E$  be a polytope and let  $C \subseteq \mathbb{R}^E$ ,  $D \subseteq \mathbb{R}^E$  be cones such that  $C$  is pointed and  $D$  contains  $P$ . Suppose  $(P - C) \cap D$  is of full dimension, and let  $ax \leq a_0$  be a facet-inducing inequality for  $P$  which is valid for  $(P - C) \cap D$ , support reduced with respect to a generating set  $K$  for  $C$ , and for which the corresponding facet of  $P$  is not contained in any facet of  $D$ . Then  $ax \leq a_0$  is facet-inducing for  $(P - C) \cap D$ .

Proof. Let  $A \in \mathbb{R}^{L \times E}$  and  $c \in \mathbb{R}^L$  be such that  $Ax = c$  is a minimal equation system for  $P$ , and let  $G \in \mathbb{R}^{E \times K}$  be the matrix whose columns are the elements of  $K$ . Since  $a$  is support reduced with respect to  $K$ , there exists  $B \subseteq K^0(a, K) \subseteq K$  such that  $\{Ag \mid g \in B\}$  is a column basis of  $AG$ . Since  $ax \leq a_0$  is facet-inducing for  $P$  there exists a set of  $\dim(P)$  affinely independent points in  $P$  which satisfy  $ax = a_0$  and we let  $\bar{X}$  denote such a set. Note that  $(P - C) \cap D$  being of full dimension implies  $D$  and  $P - C$  are also of

full dimension, and thus  $|B| = |L|$  by (2.3.1). Also note that  $\bar{X} \subseteq (P - C) \cap D$  since  $P \subseteq D$  and  $0 \in C$ , and  $|\bar{X}| = |E| - |L|$  by (2.2.2).

Let  $\{d_i x \geq 0, i = 1, 2, \dots, r\}$  be a minimal defining system for cone  $D$ . Since the facet of  $P$  induced by  $ax \leq a_0$  is assumed not to be contained in any facet of  $D$ , it follows that for all  $i \in \{1, 2, \dots, r\}$  there exists an  $x^i \in \bar{X}$  such that  $d_i x^i > 0$ , and we let  $\bar{x} = \frac{1}{r} \sum_{i=1}^r x^i$ . Clearly  $a\bar{x} = a_0$  and  $d_i \bar{x} > 0$  for all  $i \in \{1, 2, \dots, r\}$ . Next, for all  $g \in B$  define  $\delta_g = \min \left\{ \frac{d_i \bar{x}}{|d_i g|} \mid i = 1, 2, \dots, r \right\}$ , let  $\hat{x}^g = \bar{x} - \delta_g g$  and let  $\hat{X} = \{\hat{x}^g \mid g \in B\}$ .

Clearly  $\hat{X} \subseteq P - C$ . We also have  $\hat{X} \subseteq D$  since for all  $g \in B$  and  $i \in \{1, 2, \dots, r\}$ ,

$$\begin{aligned} d_i \hat{x}^g &= d_i \bar{x} - \delta_g d_i g \\ &\geq d_i \bar{x} - d_i \bar{x} \\ &= 0 \end{aligned}$$

and thus  $\hat{X} \subseteq (P - C) \cap D$ . We have  $a\hat{x}^g = a_0$  for all  $\hat{x}^g \in \hat{X}$  since  $ag = 0$  for all  $g \in B$ . Hence  $\bar{X} \cup \hat{X}$  gives us  $\dim((P - C) \cap D)$  points in  $(P - C) \cap D$  which satisfy  $ax = a_0$ . We can complete the proof that  $ax \leq a_0$  is facet-inducing for  $(P - C) \cap D$  by showing that these points are affinely independent.

Let  $\alpha \in \mathbb{R}^{\bar{X} \cup \hat{X}}$  be such that

$$i) \sum \{\alpha_x \mid x \in \bar{X}\} + \sum \{\alpha_g \mid g \in B\} = 0, \text{ and}$$

$$\text{ii) } \Sigma(\alpha_x x \mid x \in \tilde{X}) + \Sigma(\alpha_g \tilde{x}^g \mid g \in B) = 0.$$

Substituting  $\tilde{x}^g = \bar{x} - \delta_g g$  into ii) and multiplying through by matrix  $A$  gives

$$c(\Sigma(\alpha_x \mid x \in \tilde{X}) + \Sigma(\alpha_g \mid g \in B)) - \Sigma(\alpha_g \delta_g A g \mid g \in B) = 0.$$

Using i) it then follows that

$$\Sigma(\alpha_g \delta_g A g \mid g \in B) = 0$$

which implies  $\alpha_g = 0$  for all  $g \in B$  since  $\{A g \mid g \in B\}$  is a column basis for  $AG$ . It then follows from i) that  $\alpha = 0$  and thus  $\tilde{X} \cup \tilde{X}$  is an affinely independent set.  $\square$

Given a facet of  $P$  induced by  $ax \leq a_0$ , there exist many equivalent inequalities which induce it when  $P$  is not of full dimension. We provide below the necessary and sufficient conditions for the existence of an inequality equivalent to  $ax \leq a_0$  which satisfies the conditions of (2.3.2).

The following lemma is equivalent to Farkas' Lemma [Farkas, 1902].

**2.3.3 Lemma.** Given  $b \in \mathbb{R}^J$ ,  $B \in \mathbb{R}^{I \times J}$  for some finite sets  $I$  and  $J$ , the system  $Bx = 0$ ,  $bx < 0$ ,  $x \geq 0$  is unsolvable if and only if the system  $\lambda b \leq b$  is solvable.

**2.3.4 Theorem.** Let  $P \subseteq \mathbb{R}^E$  be a polytope not of full dimension, let  $Ax = c$  be a minimal equation system for  $P$  and let  $C \subseteq \mathbb{R}^E$  be a pointed cone. Then a facet-inducing inequality  $ax \leq a_0$  for  $P$  has an equivalent form which is valid for  $P - C$  if and only if there does not exist  $x \in C$  such that  $Ax = 0$  and  $ax < 0$ .

Proof. Let  $K \subseteq \mathbb{R}^E$  be a generating set for  $C$  and let  $G \in \mathbb{R}^{E \times K}$  be the matrix whose columns are the elements of  $K$ . An inequality  $\bar{a}x \leq \bar{a}_0$  which is valid for  $P$  is also valid for  $P - C$  if and only if  $\bar{a}(x - Gy) \leq \bar{a}_0$  for all  $x \in P$  and  $y \in \mathbb{R}^K$  such that  $y \geq 0$ ; or equivalently, if and only if  $\bar{a}G \geq 0$ . Thus by (2.2.4) there exists an inequality  $\bar{a}x \leq \bar{a}_0$  which is equivalent to  $ax \leq a_0$  with respect to  $P$  and valid for  $P - C$  if and only if there exists a scalar  $\gamma > 0$  and  $\lambda \in \mathbb{R}^k$  such that

$$\gamma a G + \lambda A G \geq 0. \quad (2.3.5)$$

Dividing inequality (2.3.5) by  $\gamma$ , it follows from (2.3.3) that there exist such a  $\gamma$  and  $\lambda$  if and only if there does not exist a vector  $y \geq 0$  such that  $A Gy = 0$  and  $a Gy < 0$ ; i.e. if and only if there does not exist  $x \in C$  such that  $Ax = 0$  and  $ax < 0$ , as required.  $\square$

Given an inequality  $ax \leq a_0$  which is facet-inducing for  $P$  and valid for  $P - C$ , the following algorithm finds an equivalent

inequality  $\bar{a}x \leq \bar{a}_0$  which is support reduced with respect to a generating set  $K$  for  $C$ .

(2.3.6) Support Reducing Algorithm

Let  $P \subseteq \mathbb{R}^E$  be a polytope not of full dimension, let  $C \subseteq \mathbb{R}^E$  be a pointed cone, and let  $ax \leq a_0$  be a facet-inducing inequality for  $P$  which is valid for  $P - C$ . Let  $Ax = c$ ,  $A \in \mathbb{R}^{L \times E}$  and  $c \in \mathbb{R}^L$ , be a minimal equation system for  $P$ , let  $K \subseteq \mathbb{R}^E$  be a generating set for  $C$ , and let  $G \in \mathbb{R}^{E \times K}$  be the matrix whose columns are the elements of  $K$ .

- (1) If  $\{Ag \mid g \in K_0(a, K)\}$  contains a column basis of  $AG$  then let  $\bar{a} = a$ ,  $\bar{a}_0 = a_0$ , and stop. Otherwise go to step (2).
- (2) Find  $\lambda \in \mathbb{R}^L$  such that  $\lambda(AG) \neq 0$  but  $\lambda Ag = 0$  for all  $g \in K^0(a, K)$ , and let  $\beta = \min\{\frac{a_0 g}{\lambda Ag} \mid g \in K \text{ and } \lambda Ag > 0\}$ . Let  $\hat{a} = a - \beta \lambda A$  and let  $\hat{a}_0 = a_0 - \beta \lambda c$ . Replace  $a$  with  $\hat{a}$  and  $a_0$  with  $\hat{a}_0$  and go to step (1).

To show the validity of (2.3.6) we make use of the following lemma (see [Whitney, 1935]).

(2.3.7) Lemma. For finite sets  $L$ ,  $E$ , and  $K$  let  $A \in \mathbb{R}^{L \times E}$  and  $G \in \mathbb{R}^{E \times K}$ . Then  $\{Ag \mid g \in S \subseteq K\}$  contains no column basis of  $AG$  if and only if there exists  $\gamma \in \mathbb{R}^L$  such that  $\gamma(AG) \neq 0$  but  $\gamma Ag = 0$  for all  $g \in S$ .

(2.3.8) Theorem. Let  $P$ ,  $C$ ,  $ax \leq a_0$ ,  $Ax = c$ ,  $K$ , and  $G$  be as in (2.3.6). Then the support reducing algorithm (2.3.6) finds, in a finite number of steps, an inequality  $\bar{a}x \leq \bar{a}_0$  which is equivalent to  $ax \leq a_0$ , valid for  $P - C$ , and support reduced with respect to  $K$ .

Proof. If  $\{Ag \mid g \in K^0(a, K)\}$  does not contain a column basis for  $AG$  then by (2.3.7) there exists  $\gamma \in \mathbb{R}^L$  such that  $\gamma(AG) \neq 0$  and  $\gamma Ag = 0$  for all  $g \in K^0(a, K)$ . If  $\gamma(AG) < 0$  then let  $\lambda = -\gamma$ , otherwise let  $\lambda = \gamma$  and use  $\lambda$  to obtain  $\hat{a}$  and  $\hat{a}_0$  as in step (2) of the algorithm.

Certainly  $\hat{a}x \leq \hat{a}_0$  induces the same facet of  $P$  as  $ax \leq a_0$ . Also, since an inequality  $fx \leq f_0$  which is valid for  $P$  is valid for  $P - C$  if and only if  $fG \geq 0$ , our choices of  $\beta$  and  $\lambda$  in step (2) of the algorithm guarantees  $\hat{a}x \leq \hat{a}_0$  is valid for  $P - C$  and  $K^0(a, K) \subset K^0(\hat{a}, K)$ . Thus after at most  $|E|$  repetitions of steps (1) and (2) the algorithm obtains the inequality  $\bar{a}x \leq \bar{a}_0$  as required.  $\square$

Combining the results of (2.3.2), (2.3.4) and (2.3.8) gives the following result.

(2.3.9) Corollary. Let  $P \subseteq \mathbb{R}^E$  be a polytope not of full dimension with minimal equation system  $Ax = c$ , and let  $C \subseteq \mathbb{R}^E$ ,  $D \subseteq \mathbb{R}^E$  be cones such that  $C$  is pointed,  $D$  contains  $P$ , and  $(P - C) \cap D$  is of full dimension. Then a facet-inducing inequality  $ax \leq a_0$  for  $P$  has an equivalent form which is facet-inducing for  $(P - C) \cap D$  if

$ax \geq 0$  for all  $x \in C$  such that  $Ax = 0$ .

When the extension cone  $C$  is the non-negative orthant of  $\mathbb{R}^E$ , as is the case for the monotone completion  $\bar{P} = (P - C) \cap C$  of  $P$ , we can prove the following result which is a specialization of (2.3.2).

(2.3.10) Theorem. Let  $P \subseteq \mathbb{R}^E$  be a polytope which lies in the non-negative orthant  $C$  of  $\mathbb{R}^E$ , and let  $Ax = c$  be a minimal equation system for  $P$ . Then a facet-inducing inequality  $ax \leq a_0$  for  $P$  is facet-inducing for the monotone completion  $\bar{P} = (P - C) \cap C$  of  $P$  if  $a \geq 0$ ,  $A_S$  contains a column basis of  $A$  for  $S = \{e \in E \mid a_e = 0\}$ , and for all  $e \in E$  there exists  $x \in P$  such that  $ax = a_0$  and  $x_e \neq 0$ .

Proof. The set  $K$  composed of the columns of the identity matrix  $I^{|E|}$  provides a generating set for  $C$ . Thus  $a \geq 0$  implies  $ax \leq a_0$  is valid for  $\bar{P}$ .

For every  $e \in E$  let  $x^e$  be a point in  $P$  such that  $x_e^e > 0$ , and let  $\bar{x} = \frac{1}{|E|} \sum x^e$ . Clearly  $\bar{x} \in P$  and  $\bar{x} > 0$ . Thus  $\{x^g \in \mathbb{R}^E \mid x^g = \bar{x} - \sum_{k \in K \setminus g} (\bar{x}_k)k, g \in K\} \cup \{0\}$  contains  $|E| + 1$  affinely independent points in  $\bar{P}$  and  $\bar{P}$  is of full dimension.

For  $S = \{e \in E \mid a_e = 0\}$  we have  $A_S$  contains a column basis for  $A$ , and hence  $ax \leq a_0$  is support reduced with respect to  $K$ . By (2.3.2) it follows that  $ax \leq a_0$  is facet-inducing for  $\bar{P}$ .  $\square$

We conclude this section by examining the implications of (2.3.10) for the acyclic subgraph polytope  $P_{AC}^n$ . In particular we can use (2.3.10) to show that all four classes of inequalities which are shown to be facet-inducing for the linear ordering polytope  $P_{LO}^n$  in [Grötschel, Jünger and Reinelt, 1982a] are facet-inducing for  $P_{AC}^n$ . Note that these inequalities are among those shown to be facet-inducing for  $P_{AC}^n$  in [Grötschel, Jünger and Reinelt, 1982b].

Let  $D_n = (V, A_n)$  be the complete graph on  $n$  nodes. In [Grötschel, Jünger and Reinelt, 1982a] it is shown that  $\dim(P_{LO}^n) = \binom{n}{2}$ , and that

$$x_{(i,j)} + x_{(j,i)} = 1 \quad \text{for all } (i,j) \in A_n \quad (2.3.11)$$

is a minimal equation system for  $P_{LO}^n$ . Letting  $Ax = c$ ,  $A \in \mathbb{R}^{L \times A_n}$ ,  $c \in \mathbb{R}^{A_n}$  represent the system (2.3.11), then clearly for any  $S \subseteq A_n$ ,  $\{A_e \mid e \in S\}$  contains a column basis for  $A$  if and only if  $A_n \setminus S$  contains at most one of the arcs  $(i,j)$  and  $(j,i)$  for all  $(i,j) \in A_n$ . Thus given an inequality  $ax \leq a_0$  and the set  $S = \{e \in A_n \mid a_e = 0\}$ ,  $\{A_e \mid e \in S\}$  contains a column basis for  $A$  if  $S$  satisfies the above. This is easily verified for all four classes of inequalities shown to be facet-inducing for  $P_{LO}^n$  in [Grötschel, Jünger and Reinelt, 1982a]. Also it is trivial to show these inequalities are all valid for  $P_{AC}^n$ . Thus by (2.3.10) they are all facet-inducing for  $P_{AC}^n$ .

## CHAPTER 3

The Travelling Salesman Polytope and its Facets

In this chapter we introduce the travelling salesman problem and its associated polytope  $Q_T^n$ . Since optimizing over  $Q_T^n$  is an NP-hard problem ([Karp, 1972]), it is unlikely we will ever find a complete linear description of  $Q_T^n$  (see [Karp and Papadimitriou, 1980]). However, some of the necessary constraints are known, and the main purpose of this chapter is to introduce several large classes of these facet-inducing inequalities for  $Q_T^n$ . This will show that to give a complete linear description of  $Q_T^n$  would require a very large number of constraints.

It is easy to see that  $Q_T^n$  is not of full dimension. Thus it is possible to have two distinct inequalities which induce the same facet of  $Q_T^n$ . In order to find a minimal linear description of a polytope, such redundant inequalities must be removed. Therefore we discuss in this chapter which of the inequalities introduced induce distinct facets of  $Q_T^n$ .

We also discuss in which instances the separation problem can be solved in polynomial time for the inequalities introduced. For a class of inequalities and a given point  $x$ , the separation problem is to find an inequality in the class which is violated by  $x$ , or show no such inequality exists. If there exists a polynomial algorithm to solve the separation problem for all constraints in a

linear description of a polytope  $P$ , then there exists a polynomial algorithm for optimizing over  $P$  using the ellipsoid algorithm (see [Grötschel, Lovász and Schrijver, 1981]). Thus the separation problem is useful in determining over which partial descriptions of  $Q_T^n$  we can optimize in polynomial time.

The classes of facet-inducing inequalities discussed in this chapter are those which have proven most useful in cutting plane algorithms for the travelling salesman problem. There are other classes of facets for  $Q_T^n$  which are related to hypohamiltonian or hypotractable subgraphs of  $K_n$  (see [Maurras, 1976], [Grötschel, 1980], and [Cornuéjols and Pulleyblank, 1982]). These facets are complicated to describe, and the cutting plane approach has been used successfully to find optimal or close to optimal solutions without them. Consequently, they are not discussed here.

For an extensive overview of the polyhedral aspects of the travelling salesman problem, see [Grötschel and Padberg, 1985a,b].

3.1 THE TRAVELLING SALESMAN POLYTOPE  $Q_T^n$ 

Let  $K_n = (V, E)$  denote the complete graph on  $n$  nodes. A tour of  $K_n$  is the 0-1 incidence vector of the edge set of a hamilton cycle in  $K_n$ . Then, given a vector  $c \in \mathbb{R}^E$  of edge weights, the (symmetric) travelling salesman problem (henceforth denoted by TSP) is to find a tour of  $K_n$  of minimum weight, i.e.

$$\min\{cx \mid x \text{ is a tour of } K_n\}. \quad (3.1.1)$$

In the polyhedral approach to studying the TSP we consider the associated travelling salesman polytope, denoted by  $Q_T^n$ , whose vertices are the tours of  $K_n$ ; i.e.  $Q_T^n$  is the convex hull of all tours of  $K_n$ . Then to solve the TSP we consider the problem

$$\min\{cx \mid x \in Q_T^n\}.$$

Since every node in  $K_n$  must be incident with exactly two edges of any hamilton cycle, any  $x \in Q_T^n$  must satisfy

$$x(\delta(v)) = 2 \text{ for all } v \in V. \quad (3.1.2)$$

These  $n$  equations are called the degree constraints for  $Q_T^n$ . They indicate that  $Q_T^n$  is not of full dimension. In fact, since the equations (3.1.2) are linearly independent it follows by (2.2.2) that  $\dim(Q_T^n) \leq |E| - n = n/2(n - 3)$ . By actually constructing  $n/2(n - 3) + 1$  affinely independent tours, [Grötschel and Padberg, 1979a] proved the following result.

(3.1.3) Theorem. The dimension of  $Q_T^n$  is  $n/2(n - 3)$  for  $n \geq 3$ .

The following is a direct consequence of (2.2.2) and (3.1.3).

(3.1.4) Corollary. Let  $A \in \mathbb{R}^{V \times E}$  be the node-edge incidence matrix for  $K_n$ . Then  $Ax = 2$  forms a minimal equation system for  $Q_T^n$  for  $n \geq 3$ .

### 3.2 SIMPLE CLASSES OF FACETS FOR $Q_T^n$

For this section let  $V$  and  $E$  represent the node set and edge set of  $K_n$ .

There are several large classes of necessary inequalities known for  $Q_T^n$ . Clearly for any  $x \in Q_T^n$  we have  $x \geq 0$ , or equivalently

$$-x_e \leq 0 \text{ for all } e \in E. \quad (3.2.1)$$

These are called the trivial or non-negativity constraints.

(3.2.2) Theorem [Grötschel and Padberg, 1979a]. For all  $n \geq 5$ , the inequalities  $x_e \geq 0$ ,  $e \in E$ , induce distinct facets of  $Q_T^n$ .

Since a tour of  $K_n$  cannot contain a subtour, i.e. cannot contain the 0-1 incidence vector of a cycle of length  $w < n$ , every  $x \in Q_T^n$  must satisfy

$$x(\gamma(S)) \leq |S| - 1 \text{ for all } S \subseteq V, 2 \leq |S| \leq n - 2. \quad (3.2.3)$$

These constraints are called the subtour elimination constraints, and were first introduced by [Dantzig, Fulkerson and Johnson, 1954]. Note that if we take any constraint  $x(\gamma(S)) \leq |S| - 1$  in (3.2.3) and

subtract half the sum of the all degree constraints for  $v \in S$  we obtain  $-1/2(x(\delta(S))) \leq -1$ . Thus the subtour elimination constraints (3.2.3) can equivalently be described as

$$x(\delta(S)) \geq 2 \text{ for all } S \subseteq V, 2 \leq |S| \leq n - 2, \quad (3.2.4)$$

which is known as the cut form of these constraints. It is clear from this form that we need only consider these inequalities for  $|S| \leq \lfloor n/2 \rfloor$  since  $S$  and  $V \setminus S$  result in the same inequality.

(3.2.5) Theorem [Grötschel and Padberg, 1979b]. For all  $n \geq 4$  and  $S \subseteq V$  satisfying  $2 \leq |S| \leq \lfloor n/2 \rfloor$ , the subtour elimination constraints (3.2.3) induce distinct facets of  $Q_1^n$ .

Note that none of the subtour elimination constraints induce trivial facets. Also note that by taking  $|S| = 2$  in the inequalities (3.2.3), we obtain the so-called upper-bound constraints

$$x_e \leq 1 \text{ for all } e \in E. \quad (3.2.6)$$

The separation problem for the non-negativity constraints (3.2.1) and the upper-bound constraints (3.2.6) can be solved in polynomial time simply by checking each of the constraints. Such a method would not be satisfactory for the subtour elimination constraints (3.2.3) as there exists an exponential number of them. However, it is possible to solve the separation problem for the

subtour elimination constraints by using the polynomially-bounded algorithm of [Gomory and Hu, 1961] for finding the minimum cost cut in a graph. Taking the current solution  $x^*$  as the edge costs, the algorithm is used to find a minimum cost cut  $\delta(S)$ ,  $S \subseteq V$ . If  $x^*(\delta(S)) \geq 2$  then  $x^*$  satisfies all subtour elimination constraints (3.2.4), otherwise  $x(\delta(S)) \geq 2$  is a most-violated constraint. In the above separation routine we are required to solve  $|V| - 1$  maximum flow problems, thus it requires  $O(n^4)$  steps to implement. As a result, it is not generally used in the cutting plane approach to the TSP.

### 3.3 CLIQUE TREE INEQUALITIES

A class of facet inducing inequalities for  $Q_1^n$  called clique tree inequalities was introduced by [Grötschel and Pulleyblank, 1981]. A clique tree is a connected graph  $C$  whose maximal cliques partition into two sets, the set of handles and the set of teeth, which satisfy the following properties:

- (1) No two teeth intersect.
- (2) No two handles intersect.
- (3) Each tooth contains at least two and at most  $n - 2$  nodes, and at least one node belonging to no handle.
- (4) The number of teeth intersecting each handle is odd and at least three.

(5) If a tooth  $T$  and a handle  $H$  have a nonempty intersection, then  $H \cap T$  is an articulation set of the clique tree.

An example of a clique tree is shown below in Figure

3.3.1. The ellipses indicate cliques, and the teeth and handles are indicated by  $T$  and  $H$  respectively.

For any clique tree we use  $H(C)$  respectively  $T(C)$  to denote the set of handles respectively teeth of  $C$ . A tooth of  $C$  is called pendent if it intersects at most one handle in  $C$ . We denote the set of pendent teeth of  $C$  by  $T_p(C)$ . All other teeth are called nonpendent, and the set of these teeth is denoted by  $T_{np}(C)$ .

Given a clique tree  $C$ , the clique tree inequality for  $C$  is

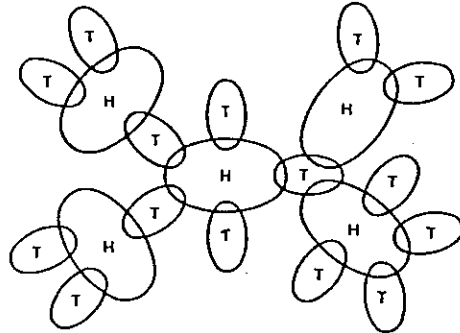


Figure 3.3.1

$$\begin{aligned} & \sum(x(\gamma(H)) | H \in H(C)) + \sum(x(\gamma(T)) | T \in T(C)) \\ & \leq \sum(|H| | H \in H(C)) + \sum(|T| - t_T | T \in T(C)) - \frac{(|T(C)| + 1)}{2} \end{aligned} \quad (3.3.1)$$

where for  $T \in T(C)$ ,  $t_T$  denotes the number of  $H \in H(C)$  which intersect  $T$ . For convenience, we represent inequality (3.3.1) by  $a_C x \leq \alpha_C$ . A simple clique tree is a clique tree  $C$  such that  $|H \cap T| \leq 1$  for all  $T \in T(C)$  and  $H \in H(C)$  which intersect in  $C$ . Note that a simple clique tree produces inequalities in which all coefficients have value zero or one, while the more general clique trees produce inequalities whose coefficients belong to  $\{0, 1, 2\}$ .

A comb is a clique tree with one handle. The inequalities corresponding to simple combs were introduced by [Chvátal, 1973b] and later generalized to the inequalities corresponding to general combs by [Grötschel and Padberg, 1979a]. The clique trees consisting of a single tooth provide us with the subtour elimination constraints. Hence the clique tree inequalities include as special cases both the comb and the subtour elimination constraints.

Let  $C$  be a comb with  $H(C) = \{H\}$  and  $T(C) = \{T_1, T_2, \dots, T_k\}$ . Then  $C'$  with  $H(C') = \{V \setminus H\}$  and  $T(C') = \{T_1, T_2, \dots, T_k\}$  is also a comb. Furthermore, if we take the comb inequality  $a_C x \leq \alpha_C$  for  $C$ , subtract half the sum of the degree constraints for  $v \in H$  and add half the sum of the degree constraints for  $v \in V \setminus H$ , we obtain the comb inequality for comb  $C'$ . Hence these two combs induce the same facet of  $Q_T^n$ . [Grötschel, 1977] showed that, except for this case,



the comb inequalities induce distinct facets of  $Q_T^n$ , and in all cases these facets are distinct from those induced by the non-negativity constraints (3.2.1) and the subtour elimination constraints (3.2.3).

(3.3.2) Theorem [Grötschel and Pulleyblank, 1981]. Each clique tree inequality is facet inducing for  $Q_T^n$ . Moreover, distinct clique trees induce distinct facets except for the comb and subtour elimination cases previously described.

Simple combs generalize the 2-factor constraints

$$x(\gamma(S)) + x(H) \leq |S| + \frac{|H| - 1}{2} \quad \text{for all } S \subseteq V, H \subseteq \delta(S), \\ |H| \geq 3 \text{ and odd} \quad (3.3.3)$$

which were first introduced by [Edmonds, 1965]. [Padberg and Rao, 1982] showed that the separation problem can be solved for these constraints by using a modification of the minimum cut algorithm [Gomory and Hu, 1961]. However, the 2-factor constraints (3.3.3) and the subtour elimination constraints (3.2.3) are the only classes of clique tree inequalities for which a polynomial time algorithm is known for the separation problem, and an outstanding problem is to find other classes of clique trees for which separation can be solved in polynomial time.

## CHAPTER 4

### A Simplex Method for the Fractional 2-Factor Problem

Given  $K_n = (V, E)$  and a vector  $c \in \mathbb{R}^E$  of edges costs, the fractional 2-factor problem is to find  $x \in \mathbb{R}^E$  which minimizes  $cx$  subject to  $x$  satisfying the degree constraints (3.1.2), non-negativity constraints (3.2.1) and the upper-bound constraints (3.2.6) for  $Q_T^n$ . Thus it is the problem

$$\begin{aligned} \text{minimize} \quad & cx \\ \text{subject to} \quad & x(\delta(v)) = 2 \text{ for all } v \in V \\ & 0 \leq x \leq 1. \end{aligned} \quad (4.0.1)$$

The corresponding polytope, which we denote by  $Q_F^n$ , is defined by  $Q_F^n = \{x \in \mathbb{R}^E \mid Ax = 2, 0 \leq x \leq 1\}$ , where  $A \in \mathbb{R}^{V \times E}$  is the node-edge incidence matrix of  $K_n$ . The vertices of this polytope have a special structure: any vertex  $\hat{x}$  of  $Q_F^n$  is half-integer, and the set of edges  $J \subseteq E$  such that  $\hat{x}_j = \frac{1}{2}$  for all  $j \in J$  partitions into the edge sets of an even number of odd disjoint cycles (see [Balinski, 1970]).

The polytope  $Q_F^n$  contains  $Q_T^n$ . Since all the defining constraints for  $Q_F^n$  are valid for  $Q_T^n$ . Thus we have

$$\min\{cx \mid x \in Q_F^n\} \leq \min\{cx \mid x \in Q_T^n\}.$$

Furthermore, if the solution to a fractional 2-factor problem is a tour, it is necessarily an optimal tour for the related TSP.

It is possible to solve the fractional 2-factor problem in

polynomial time either by transforming it into an instance of bipartite b-matching or by solving it directly using a primal-dual algorithm (see Section 4.4), or by a polynomial algorithm for linear programming. However, the success of the network simplex algorithm for the upper-bounded transshipment problem and the fact that it was found in practice to be as fast or faster than the primal-dual method (see [Glover, Karney, and Klingman, 1977]) for large problems indicates it would be desirable to find a simplex method for the fractional 2-factor problem.

In this chapter we present a finite combinatorial simplex method for the following linear program:

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & Ax = b \\ & 0 \leq x \leq u \end{array} \quad (4.0.2)$$

where  $A \in \mathbb{R}^{V \times E}$  is the node-edge incidence matrix of  $K_n = (V, E)$ ,  $c \in \mathbb{R}^E$ , and  $b \in \mathbb{R}^V$ ,  $u \in \mathbb{R}^E$  satisfy  $b \geq 0$  and  $u > 0$  (see Section 1.5 for a description of the revised simplex method). Note that the fractional 2-factor problem is a special case of (4.0.2) in which  $b = 2$  and  $u = 1$ .

Problem (4.0.2) is a special case of the network flow with gains problem (or generalized network flow problem) and the simplex method presented in this chapter is a specialized implementation of a primal simplex method for the network flow with gains problem (see [Kennington and Helgason, 1980]). However, the general algorithm is at present only finite in the case of positive gains (see [Elam,

Glover and Klingman, 1979]) while (4.0.2) is an instance of network flow with negative gains.

The simplex method presented here ensures finiteness by providing a combinatorial interpretation of lexicography for the linear program (4.0.2). Although this method is easy to implement, it would be more desirable to find a set of "strongly feasible" bases along with a pivot rule which would ensure finiteness, as [Cunningham, 1976] did for the network simplex method and [Barr, Glover and Klingman, 1977] did for the assignment problem. We do not know such a set of bases and pivot rule for (4.0.2) or even for the fractional 2-factor problem, but we believe they exist.

There are other linear programs which can be transformed into the form of (4.0.2). For instance, consider the linear program

$$\begin{array}{ll} \text{minimize} & c'x \\ \text{subject to} & Ax = b' \\ & \ell' \leq x \leq u' \end{array} \quad (4.0.3)$$

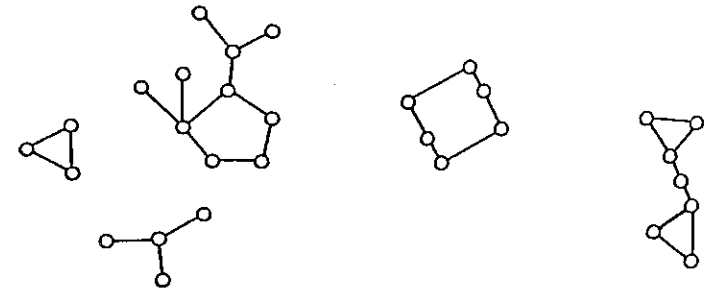
where  $A \in \mathbb{R}^{V \times E}$  is the node-edge incidence matrix of  $K_n = (V, E)$ ,  $c' \in \mathbb{R}^E$ , and  $u' \in \mathbb{R}^E$ ,  $\ell' \in \mathbb{R}^E$  satisfy  $u' \geq \ell'$ . This linear program can be transformed into the form of (4.0.2) in the following way. Let  $u = u' - \ell'$ , let  $b_v = b'_v - \ell'(\delta(v))$  for all  $v \in V$ , let  $c_e = c'_e$  for all  $e \in E$  such that  $u'_e > \ell'_e$ , and let  $c_e = +\infty$  if  $u'_e = \ell'_e$ . Then it is easy to verify that  $x$  is an optimal solution for (4.0.2) if and only if  $x' = x + \ell'$  is an optimal solution for (4.0.3). Thus the method we describe for solving (4.0.2) could also be used to solve (4.0.3).

Before presenting the simplex method for linear program (4.0.2), we discuss some properties of the node-edge incidence matrix  $A$  of  $K_n$  which will prove useful.

#### 4.1 THE NODE-EDGE INCIDENCE MATRIX OF $K_n$

Let  $G = (V, E)$  be a graph with node-edge incidence matrix  $B \in \mathbb{R}^{V \times E}$ , and let  $I$  be the family of all subsets of  $E$  such that the corresponding columns of matrix  $B$  are linearly independent. Then  $M(G) = (E, I)$  is a matroid defined on the set of edges  $E$  which is sometimes called the real matroid of  $G$  (see [Grötschel and Pulleyblank, 1981]). The independent sets of  $M(G)$  (i.e. those subsets  $J$  of  $E$  for which the corresponding columns of  $B$  are linearly independent) can easily be shown to be the set of all  $T \subseteq E$  such that each component of  $G' = (V, T)$  contains no even cycles, and at most one odd cycle. The circuits, or minimal dependent sets of  $M(G)$  are the sets  $C \subseteq E$  such that  $C$  is either the edge set of an even cycle in  $G$  or else the edge set of two disjoint odd cycles joined by a path (see Figure 4.1.1). We use the term dumbbell to denote a graph consisting of two disjoint odd cycles joined by a path.

(4.1.1) Theorem. Let  $A \in \mathbb{R}^{V \times E}$  be the node-edge incidence matrix of  $K_n = (V, E)$ . Then  $T \subseteq E$  is a basis of  $A$  if and only if every component of  $G' = (V, T)$  contains no even cycle and exactly one odd cycle. Also, a set  $C \subseteq E$  indexes a minimal dependent set of columns of  $A$  if and only if  $C$  is the edge set of an even cycle or a dumbbell in  $K_n$ .



An independent set of edges of  $M(G)$ . Two circuits of  $M(G)$ .

Figure 4.1.1

A connected graph which contains exactly one cycle is composed of a tree with one edge added. If the cycle is odd we call such a graph an odd 1-tree. If all the components of a graph  $G$  are odd 1-trees, we call  $G$  an odd 1-forest. Thus  $T \subseteq E$  is a basis of the node-edge incidence matrix  $A$  of  $K_n = (V, E)$  if and only if  $G' = (V, T)$  is an odd 1-forest.

For the remainder of this section let  $A \in \mathbb{R}^{V \times E}$  be the node-edge incidence matrix of  $K_n = (V, E)$  and let  $T \subseteq E$  be a column basis of  $A$ .

It follows from Theorem (4.1.1) that the system  $A_T x = d$  has a unique solution for any  $d \in \mathbb{R}^V$ ; i.e. there is a unique  $x \in \mathbb{R}^T$  satisfying  $x(\delta(v)) = d_v$  for all  $v \in V$  in the graph  $G' = (V, T)$ . This solution  $x$  can be obtained easily from the 1-forest structure of  $G'$  using the algorithm described below. The

general idea of this algorithm is to consider each 1-tree in  $G'$  to be rooted at its odd cycle, then calculate  $x$  progressively for each 1-tree by starting with the edges on the outside of the tree and working in towards its root.

(4.1.2) Algorithm to solve  $A_T x = d$  for the odd 1-forest  $G' = (V, T)$ .

Without loss of generality we assume  $G'$  consists of a single odd 1-tree, since each component of  $G'$  can be treated independently. Let  $C$  be the single odd cycle in  $G'$ , and at any stage of the algorithm let  $U$  represent the edges  $e$  in  $T \setminus E(C)$  for which  $x_e$  has not yet been calculated.

- (1) Let  $U = T \setminus E(C)$ .
- (2) If  $U = \emptyset$  then go to (4).
- (3) Choose  $u \in V \setminus V(C)$  such that  $u$  has exactly one edge  $uv \in U$  incident with it. Let  $x_{uv} = d_u - x(\delta(u) \setminus U)$ , and let  $U = U \setminus \{uv\}$ . Go to step (2).
- (4) For every  $v \in V(C)$  let  $d'_v = d_v - x(\delta(v) \setminus E(C))$ .
- (5) For each  $e = uv \in E(C)$  let  $(X_e, Y_e)$  be the bipartition of the nodes of  $C$  in the bipartite graph obtained by removing  $e$  from  $C$ . Since  $C$  is an odd cycle, both  $u$  and  $v$  are on the same side, say  $X_e$ , of this partition. Let  $x_e = 1/2 (\sum(d'_w | w \in X_e) - \sum(d'_w | w \in Y_e))$  for every  $e \in E(C)$ .

It also follows from (4.1.1) that the system  $yA_T = c$  has a unique solution for any vector  $c \in \mathbb{R}^T$ ; i.e. there is a unique  $y \in \mathbb{R}^V$  such that  $y_u + y_v = c_{uv}$  for any edge  $uv \in T$ . Again, the structure of the 1-forest  $G' = (V, T)$  can be used to obtain  $y$ . The general idea of the method is to consider each 1-tree as rooted at its odd cycle and calculate  $y$  for the nodes of each of the 1-trees progressively from the root outwards.

(4.1.3) Algorithm to solve  $yA_T = c$  for the odd 1-forest  $G' = (V, T)$ .

Without loss of generality assume  $G'$  consists of a single odd 1-tree, and let  $C$  be the odd cycle in  $G'$ . At any stage of the algorithm let  $U$  represent the nodes  $v$  in  $V \setminus V(C)$  for which  $y_v$  has not yet been calculated.

- (1) Let  $U = V \setminus V(C)$ .
- (2) For any node  $v \in V(C)$ , let  $M_v$  be the matching in  $C$  which misses only node  $v$ , and let  $N_v = E(C) \setminus M_v$ . Then for each  $v \in V(C)$  let  $y_v = 1/2 (c(N_v) - c(M_v))$ .
- (3) If  $U = \emptyset$ , then stop.
- (4) Choose edge  $uv \in T \setminus E(C)$  such that  $u \in U$ ,  $v \notin U$ . Let  $y_u = c_{uv} - y_v$ , and let  $U = U \setminus \{u\}$ . Go to step (3).

The properties of matrix  $A$  and edge set  $T$  can also be used to find  $A_T^{-1}A_L \in \mathbb{R}^{T \times L}$  for any  $L \subseteq ET$ . This is a necessary part of the simplex methods presented in this chapter and in Chapter 6. In

particular, we need to calculate specific columns of  $A_T^{-1}A_L$  and specific rows of  $A_T^{-1}$ . We discuss the methods for doing this in the remainder of this section.

Let  $k^e \in \mathbb{R}^T$  denote the column of  $A_T^{-1}A_L$  indexed by edge  $e \in L$ ; i.e. let  $k^e = A_T^{-1}A_e$ . Then  $k^e$  is the unique solution to

$$A_T k^e = A_e. \tag{4.1.4}$$

This could be solved using the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.2). However, we can also solve (4.1.4) by making use of the set of edges  $C \subseteq T \cup \{e\}$  which index a minimal dependent set of columns in  $A$ . It follows from Theorem (4.1.1) that the edges in  $C$  either form an even cycle or a dumbbell in  $K_n$ . By taking advantage of these structures, we can calculate  $k^{uv}$  for any edge  $uv \in L$  by letting  $k_j^{uv} = 0$  for all  $j \in T \setminus C$ , and ensuring

$$k^{uv} (\delta(w) \cap (C \setminus \{uv\})) = \begin{cases} 1 & \text{if } w = u \text{ or } w = v \\ 0 & \text{otherwise.} \end{cases} \tag{4.1.5}$$

It is easily verified that  $k^e$  as calculated by Algorithm (4.1.6) below satisfies (4.1.5).

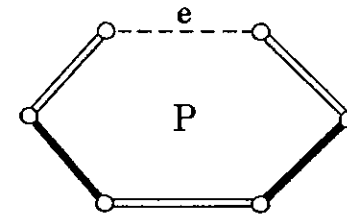
(4.1.6) Algorithm to calculate  $k^e = A_T^{-1}A_e$ ,  $e \in E \setminus T$ , using the circuit formed by  $C$ .

Case 1.  $C$  forms an even cycle  $P$  in  $K_n$ .

We can uniquely partition  $C$  into disjoint perfect matchings  $M_1$  and  $M_2$  of  $P$  such that the edge  $e$  being added to  $T$  is in  $M_1$ . Then  $k^e \in \mathbb{R}^T$  is defined by

$$k_j^e = \begin{cases} -1 & \text{if } j \in M_1 \\ 1 & \text{if } j \in M_2 \\ 0 & \text{otherwise} \end{cases}$$

(see Figure 4.1.2).



- corresponds to  $k_j^e = 1$
- corresponds to  $k_j^e = -1$
- edges  $j \in T$  with  $k_j^e = 0$  are not shown

Figure 4.1.2

Case 2.  $C$  forms a dumbbell  $P$  in  $K_n$ .

Let the two odd cycles in  $P$  be  $Q_1$  and  $Q_2$ , and let the path joining them be  $R$ . Define  $M_1$  to be the near-perfect matching of  $Q_1$

deficient at the node where  $R$  joins  $Q_1$ , and define the near-perfect matching  $M_2$  of  $Q_2$  similarly.

(a) The length of  $R$  is odd.

Let  $M_3$  be the perfect matching of  $R$ , and define  $b \in \mathbb{R}^T$  by

$$b_j = \begin{cases} -1 & \text{if } j \in M_3 \\ 1 & \text{if } j \in E(R) \setminus M_3 \\ -1/2 & \text{if } j \in M_1 \cup M_2 \\ 1/2 & \text{if } j \in (E(Q_1) \setminus M_1) \cup (E(Q_2) \setminus M_2) \\ 0 & \text{otherwise.} \end{cases}$$

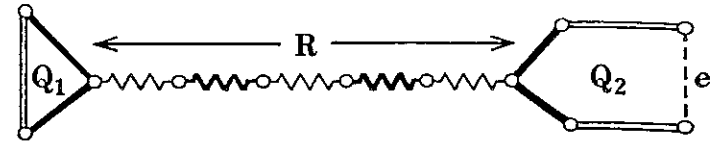
Then  $k^e \in \mathbb{R}^T$  is defined by

$$k^e = \begin{cases} b & \text{if } e \in M_3 \\ -b & \text{if } e \in E(R) \setminus M_3 \\ 2b & \text{if } e \in M_1 \cup M_2 \\ -2b & \text{if } e \in (E(Q_1) \setminus M_1) \cup (E(Q_2) \setminus M_2) \end{cases}$$

(see Figure 4.1.3).

(b) The length of  $R$  is even.

Let  $M_3$  be the near-perfect matching of  $R$  deficient at the node where  $R$  joins  $Q_1$ , and define  $b \in \mathbb{R}^T$  by



— corresponds to  $k_j^e = 1$

- - - corresponds to  $k_j^e = -1$

〰 corresponds to  $k_j^e = 2$

〰 corresponds to  $k_j^e = -2$

- edges  $j \in T$  with  $k_j^e = 0$  are not shown

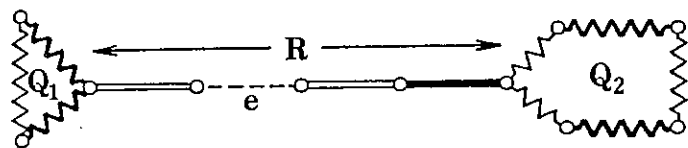
Figure 4.1.3

$$b_j = \begin{cases} -1 & \text{if } j \in M_3 \\ 1 & \text{if } j \in E(R) \setminus M_3 \\ -1/2 & \text{if } j \in M_2 \cup (E(Q_1) \setminus M_1) \\ 1/2 & \text{if } j \in M_1 \cup (E(Q_2) \setminus M_2) \\ 0 & \text{otherwise.} \end{cases}$$

Then  $k^e \in \mathbb{R}^T$  is defined by

$$k^e = \begin{cases} b & \text{if } e \in M_3 \\ -b & \text{if } e \in E(R) \setminus M_3 \\ 2b & \text{if } e \in M_2 \cup (E(Q_1) \setminus M_1) \\ -2b & \text{if } e \in M_1 \cup (E(Q_2) \setminus M_2) \end{cases}$$

(see Figure 4.1.4).



- corresponds to  $k_j^e = 1$
- corresponds to  $k_j^e = -1$
- corresponds to  $k_j^e = 1/2$
- corresponds to  $k_j^e = -1/2$

edges  $j \in T$  with  $k_j^e = 0$  are not shown

Figure 4.1.4

Finding the row of  $A_T^{-1}A_L \in \mathbb{R}^{T \times L}$  indexed by  $e \in T$  is equivalent to calculating  $a^e A_L$ , where  $a^e$  denotes the row of  $A_T^{-1}$

indexed by  $e$ . We can calculate  $a^e \in \mathbb{R}^V$  by using the odd 1-forest algorithm (4.1.3) to solve

$$a^e A_T = t, \tag{4.1.7}$$

where  $t \in \mathbb{R}^T$  is defined by

$$t_f = \begin{cases} 1 & \text{if } f = e \\ 0 & \text{otherwise.} \end{cases}$$

However, we can also solve (4.1.7) by making use of the structure of the odd 1-forest  $G' = (V, T)$ . When any edge  $e$  is removed from a 1-tree, a tree  $H$  is formed. Using the bipartite graph  $H$  we calculate  $a^e$  by letting  $a_v^e = 0$  for all  $v \in V \setminus V(H)$ , and ensuring  $a_{V(H)}^e$  satisfies

$$a_u^e + a_v^e = \begin{cases} 1 & \text{if } e = uv \\ 0 & \text{otherwise.} \end{cases}$$

for all edges  $uv \in E(H) \cup \{e\}$ . This can be accomplished as follows.

(4.1.8) Algorithm to calculate  $a^e$ , the row of  $A_T^{-1}$  indexed by  $e \in T$ .

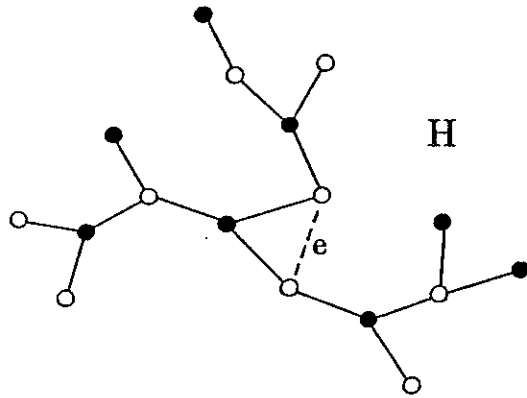
Let  $a^e \in \mathbb{R}^V$  denote the row of  $A_T^{-1} \in \mathbb{R}^{T \times V}$  indexed by  $e \in T$ , and let  $H$  be the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$ . Then  $H$  forms a bipartite graph, and we let  $V_1, V_2$  be the corresponding unique bipartition of  $V(H)$  such that at least one end of  $e$  is in  $V_1$ .

Case 1.  $e \in E(C)$  for some odd cycle  $C$  in the odd 1-forest  $G' = (V, T)$ .

In this case,  $a^e$  is defined by

$$a_w^e = \begin{cases} 1/2 & \text{if } w \in V_1 \\ -1/2 & \text{if } w \in V_2 \\ 0 & \text{otherwise} \end{cases}$$

(see Figure 4.1.5).



○ corresponds to  $a_w^e = 1/2$

● corresponds to  $a_w^e = -1/2$

- nodes with  $a_w^e = 0$  are not shown

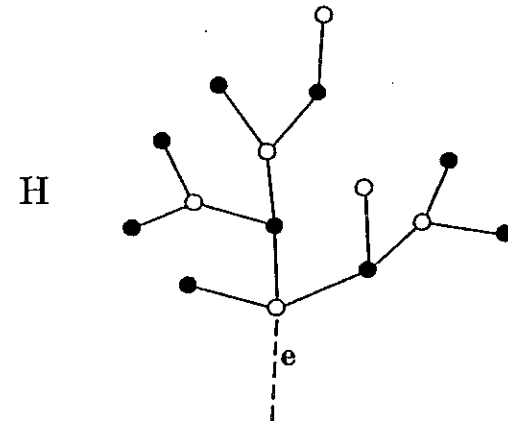
Figure 4.1.5

Case 2.  $e \notin C$  for any odd cycle  $C$  in the odd 1-forest  $G' = (V, T)$ .

In this case  $a^e$  is defined by

$$a_w^e = \begin{cases} 1 & \text{if } w \in V_1 \\ -1 & \text{if } w \in V_2 \\ 0 & \text{otherwise} \end{cases}$$

(see Figure 4.1.6)



○ corresponds to  $a_w^e = 1$

● corresponds to  $a_w^e = -1$

- nodes with  $a_w^e = 0$  are not shown

Figure 4.1.6



4.2 A SIMPLEX METHOD FOR LINEAR PROGRAM (4.0.2)

Recall that we wish to solve the linear program

$$\begin{aligned} &\text{minimize} && cx && (4.0.2) \\ &\text{subject to} && Ax = b \\ & && 0 \leq x \leq u \end{aligned}$$

where  $A \in \mathbb{R}^{V \times E}$  is the node-edge incidence matrix of  $K_n = (V, E)$ ,  $c \in \mathbb{R}^E$ , and  $b \in \mathbb{R}^V$ ,  $u \in \mathbb{R}^E$  satisfy  $b \geq 0$  and  $u > 0$ . We can write this linear program equivalently as

$$\begin{aligned} &\text{minimize} && \sum_{e \in E} c_e x_e && (4.2.1) \\ &\text{subject to} && x(\delta(v)) = b_v \text{ for all } v \in V, \\ & && -x_e \geq -u_e \text{ for all } e \in E, \\ & && x_e \geq 0 \text{ for all } e \in E. \end{aligned}$$

The associated dual linear program is

$$\begin{aligned} &\text{maximize} && \sum_{v \in V} y_v b_v - \sum_{e \in E} \mu_e u_e && (4.2.2) \\ &\text{subject to} && y_u + y_v - \mu_{uv} \leq c_{uv} \text{ for all } uv \in E, \\ & && \mu_e \geq 0 \text{ for all } e \in E. \end{aligned}$$

In order to obtain a simplex method for (4.2.1) we must do the following:

- i) determine what a basis for the linear program looks like,
- ii) find a method for obtaining the corresponding basic primal solution,

- iii) find a method for obtaining the corresponding basic dual solution,
- iv) describe how to perform a simplex pivot, and
- v) describe how to find an initial primal feasible basis.

In addition to the above, we must ensure the sequence of pivots performed is finite. This is discussed in Section 4.3.

i) The Bases

By introducing slack variables  $s_e$  for every edge  $e \in E$ , we can rewrite the linear program (4.2.1) in equality form as follows:

$$\begin{aligned} &\text{minimize} && \sum_{e \in E} c_e x_e && (4.2.3) \\ &\text{subject to} && x(\delta(v)) = b_v \text{ for all } v \in V, \\ & && -x_e - s_e = -u_e \text{ for all } e \in E, \\ & && x_e, s_e \geq 0 \text{ for all } e \in E. \end{aligned}$$

The coefficient matrix and right-hand side for (4.2.3) have the following form:

$$\begin{array}{l} \text{rows corresponding} \\ \text{to degree} \\ \text{constraints} \end{array} \left\{ \begin{array}{|c|c|} \hline A & 0 \\ \hline \end{array} \right. \begin{array}{l} \text{columns} \\ \text{corresponding} \\ \text{to } x_e \end{array} \quad \begin{array}{l} \text{rows corresponding} \\ \text{to upper-bound} \\ \text{constraints} \end{array} \left\{ \begin{array}{|c|c|} \hline -I|E| & -I|E| \\ \hline \end{array} \right. \begin{array}{l} \text{columns} \\ \text{corresponding} \\ \text{to } s_e \end{array} \quad \begin{array}{|c|} \hline b \\ \hline \\ \hline -u \\ \hline \end{array} \quad (4.2.4)$$

Coefficient Matrix                      Right-Hand Side

It follows from Theorem (4.1.1) that matrix A has rank  $|V|$ , thus the coefficient matrix above has rank  $|V| + |E|$ . Consequently, a basis corresponds to specifying  $|V| + |E|$  of the variables  $x_e, s_e, e \in E$  as basic variables. For each  $e \in E$  there are the following possibilities:

- (a) both  $x_e$  and  $s_e$  are basic;
- (b)  $x_e$  is basic,  $s_e$  is nonbasic;
- (c)  $x_e$  is nonbasic,  $s_e$  is basic;
- (d) both  $x_e$  and  $s_e$  are nonbasic.

In the corresponding basic solution we require  $x_e = s_e = 0$  for edges  $e$  of type (d), implying  $u_e = 0$  for these edges. Since we assume  $u > 0$ , it follows that possibilities (a), (b), and (c) partition the edges of  $K_n$  into three sets, say T, B and Z respectively. Edges  $e$  in B must have  $x_e = u_e$  and  $s_e = 0$  in the corresponding basic solution, and edges  $e$  in Z must have  $x_e = 0, s_e = u_e$ .

Since there are  $|V| + |E|$  basic variables in our set, it follows that  $2|T| + |B| + |Z| = |E| + |V|$ , and thus  $|T| = |V|$ . Furthermore, the columns corresponding to all  $x_e$  and  $s_e, e \in T$ , must be linearly independent. Clearly this is true if and only if the columns of A indexed by T are also linearly independent, hence the edges in T must form an odd 1-forest in  $K_n$  by Theorem (4.1.1).

In summary, selecting a basis for (4.2.3) is equivalent to specifying a partition (T,B,Z) of E such that the graph  $G' = (V,T)$

is an odd 1-forest. The set of basic variables will then consist of  $\{x_e, s_e | e \in T\} \cup \{x_e | e \in B\} \cup \{s_e | e \in Z\}$ .

In the basis (T,B,Z), B represents the edges  $e$  for which  $x_e$  is at its upper bound (i.e.  $x_B = u_B$ ), and Z represents the edges  $e$  for which  $x_e$  is at its lower bound of zero (i.e.  $x_Z = 0$ ).

ii) Finding the Corresponding Basic Primal Solution

Let the edge partition (T,B,Z) define a basis for the linear program (4.2.3). In the corresponding basic primal solution  $(\bar{x}, \bar{s}), \bar{x} \in \mathbb{R}^E, \bar{s} \in \mathbb{R}^E$ , we require the nonbasic variables  $\bar{x}_Z$  and  $\bar{s}_B$  to equal zero, and the basic variables  $\bar{x}_T, \bar{x}_B, \bar{s}_T, \bar{s}_Z$  to satisfy

(4.2.5)

degree constraints	$A_T$	$A_B$	0	0	=	$\bar{x}_T$	=	b	
upper-bound constraints for $e \in B$	0	$-I^{ B }$	0	0		$\bar{x}_B$		-	$u_B$
upper-bound constraints for $e \in T$	$-I^n$	0	$-I^n$	0		$\bar{s}_T$		-	$u_T$
upper-bound constraints $e \in Z$	0	0	0	$-I^{ Z }$		$\bar{s}_Z$		-	$u_Z$

columns corresponding to  $x_T$

↑

columns corresponding to  $x_B$

↑

columns corresponding to  $s_T$

↑

columns corresponding to  $s_Z$

↑

where  $A$  is the node-edge incidence matrix of  $K_n$ .

Thus we can calculate  $\bar{x}$  and  $\bar{s}$  as follows.

(4.2.6) Algorithm to calculate primal solution  $(\bar{x}, \bar{s})$  for basis  $(T, B, Z)$ .

- (1) Let  $\bar{x}_B = u_B$ , and  $\bar{x}_Z = 0$ .
- (2) Let  $b' \in \mathbb{R}^V$  be defined by  $b'_v = b_v - u_B(\delta(v) \cap B)$ . Then  $\bar{x}_T$  is the unique solution to  $A_T \bar{x}_T = b'$ , which can be found using the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.2).
- (3) Let  $\bar{s}_Z = u_Z$ ,  $\bar{s}_B = 0$ , and  $\bar{s}_T = u_T - \bar{x}_T$ .

iii) Finding the Corresponding Basic Dual Solution

Given the basis of (4.2.3) defined by  $(T, B, Z)$ , the corresponding basic dual solution  $(\bar{y}, \bar{\mu})$ ,  $\bar{y} \in \mathbb{R}^V$ ,  $\bar{\mu} \in \mathbb{R}^E$ , must satisfy  $(\bar{y}, \bar{\mu})A' = [c_T, c_B, 0]$ , where  $A'$  is the coefficient matrix of (4.2.5). Thus we require  $(\bar{y}, \bar{\mu})$  to satisfy

$$\bar{y}_u + \bar{y}_v - \bar{\mu}_{uv} = c_{uv} \quad \text{for all edges } uv \in T \cup B,$$

$$\bar{\mu}_e = 0 \quad \text{for all } e \in T \cup Z.$$

Hence we can calculate  $\bar{y}$  and  $\bar{\mu}$  as follows.

(4.2.7) Algorithm to calculate dual solution  $(\bar{y}, \bar{\mu})$  for basis  $(T, B, Z)$ .

- (1) Let  $\bar{\mu}_e = 0$  for all  $e \in T \cup Z$ .
- (2) The vector  $\bar{y}$  is the unique solution to  $\bar{y}A_T = c_T$ , which can be found using the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.3).
- (3) For all  $uv \in B$  let  $\bar{\mu}_{uv} = \bar{y}_u + \bar{y}_v - c_{uv}$ .

iv) Performing a Pivot

At each iteration of our simplex method we must check to see if the current basis is dual feasible, i.e. check to see if the corresponding basic dual solution satisfies the linear program (4.2.2). If it does, the current basic primal and dual solutions are optimal for their respective problems. Otherwise one of the following necessarily occurs:

- (a)  $y_u + y_v > c_{uv}$  for some edge  $uv \in Z$ .
- (b)  $\mu_{uv} < 0$  for some  $uv \in B$ , or equivalently  $y_u + y_v < c_{uv}$  for some  $uv \in B$ .

In case (a) we would choose the nonbasic variable  $x_{uv}$  as our entering variable, and in case (b) we would choose the nonbasic variable  $s_{uv}$  as the entering variable. In either case, adding the corresponding column to the basis is equivalent to adding  $uv$  to  $T$ ; i.e. adding  $uv$  to the odd 1-forest  $G' = (V, T)$ . By Theorem (4.1.1), this results in a set of edges  $C \subseteq T \cup \{uv\}$  which forms either an even cycle or a dumbbell in  $G'' = (V, T \cup \{uv\})$ .

Let  $(\bar{x}, \bar{s})$  be the current basic primal solution. Then by (4.2.6)  $\bar{x}_T$  satisfies  $A_T \bar{x}_T = b'$ , where  $b' \in \mathbb{R}^V$  is defined by  $b'_V = b_V - u_B(\delta(v) \cap B)$ . If  $x_e$  is the entering variable for some  $e \in Z$ , we wish to raise its value from zero to some value  $\alpha \in \mathbb{R}$ ,  $\alpha \geq 0$ . Then  $x_T$  is the unique solution to

$$A_T x_T + \alpha A_e = b' \quad (4.2.8)$$

or equivalently,

$$x_T = \bar{x}_T - \alpha k^e \quad (4.2.9)$$

where  $k^e = A_T^{-1} A_e$ . By taking advantage of the structure of the subgraph of  $K_n$  formed by the minimal dependent set of edges  $C$ , we can calculate  $k^e$  using Algorithm (4.1.6).

If the entering variable chosen is  $s_e$  for some edge  $e \in B$ , increasing  $s_e$  from its present value of 0 up to some value  $\alpha$  corresponds to lowering  $x_e$  from its present value of  $u_e$  to  $u_e - \alpha$ . In this case, the corresponding value of  $x_T$  is given by

$$x_T = \bar{x}_T + \alpha k^e \quad (4.2.10)$$

The requirement that we preserve primal feasibility restricts how much we increase the value of the entering variable  $x_e$  or  $s_e$ . In both cases, this is accomplished by increasing  $\alpha$  until either  $x_f = u_f$  or  $x_f = 0$  for some edge  $f \in T \cup \{e\}$ , where  $x_T$  is given by (4.2.9) or (4.2.10). In the former case we remove edge  $f$

from  $T$  and add it to  $U$ , and variable  $s_e$  leaves the basis. In the latter case we remove  $f$  from  $T$  and add it to  $Z$ , and  $x_e$  leaves the basis.

Given a primal feasible basis  $(T, B, Z)$  for the linear program (4.2.1), the steps of a simplex pivot can be summarized as follows:

- (1) Calculate the values of the primal variables  $\bar{x}_T$  corresponding to  $(T, B, Z)$  as in (4.2.6), and calculate the values of the dual variables  $\bar{y}$  corresponding to  $(T, B, Z)$  as in (4.2.7).
- (2) If  $\bar{y}_u + \bar{y}_v \leq c_{uv}$  for all  $uv \in Z$  and  $\bar{y}_u + \bar{y}_v \geq c_{uv}$  for all  $uv \in B$  then stop; the current basis is an optimal basis. Otherwise, either choose  $e = uv \in Z$  such that  $\bar{y}_u + \bar{y}_v > c_{uv}$  and go to step (3), or else choose  $e = uv \in B$  such that  $\bar{y}_u + \bar{y}_v < c_{uv}$  and go to step (4).
- (3) Calculate  $k^e$  using the circuit algorithm (4.1.6) and let  $x_T = \bar{x}_T - \alpha k^e$ . Let  $T' = T \cup \{e\}$ ,  $B' = B$ , and  $Z' = Z \setminus \{e\}$ . Go to step (5).
- (4) Calculate  $k^e$  using the circuit algorithm (4.1.6) and let  $x_T = \bar{x}_T + \alpha k^e$ . Let  $T' = T \cup \{e\}$ ,  $B' = B \setminus \{e\}$ , and  $Z' = Z$ .
- (5) Increase the value of  $\alpha$  until  $x_f = u_f$  or  $x_f = 0$  for some edge  $f \in T'$ . In the former case let  $T'' = T' \setminus \{f\}$ ,  $B'' = B' \cup \{f\}$  and  $Z'' = Z'$ . In the latter case let  $T'' = T' \setminus \{f\}$ ,  $B'' = B'$  and  $Z'' = Z' \cup \{f\}$ . The new basis is then given by  $(T'', B'', Z'')$  and the pivot is complete.

v) Finding an Initial Primal Feasible Basis.

An initial feasible basis is easily constructed for some instances of linear program (4.0.2), and in particular for the fractional 2-factor problem (see Section 4.4). In general, the two-phase method for the simplex algorithm can be used to find such a basis, although a combinatorial method would be more desirable (see [Chvátal, 1983] for a description of the two-phase method for the simplex algorithm).

### 4.3 ENSURING FINITENESS

We can ensure finiteness of the simplex method described in Section 4.2 for the linear program (4.2.1) by applying the lexicographic anti-cycling method described in Section 1.5.

Given the current basis  $(T, B, Z)$  for the linear program (4.2.1), consider the system of equations as shown in (4.2.5) which we must solve in order to obtain the current primal basic solution  $(\bar{x}, \bar{s})$ . Let  $A'$  be the coefficient matrix of (4.2.5), let  $\bar{a}$  be the column of the coefficient matrix of (4.2.4) corresponding to the entering variable chosen multiplied by  $(A')^{-1}$ , and let  $L$  be the matrix whose  $i$ th row corresponds to the  $i$ th row of  $(A')^{-1}$  divided by  $\bar{a}_i$ . Now whenever we have a tie for the leaving variable, we avoid degeneracy by calculating the lexicographically minimum vector among the corresponding rows of  $L$ , using an ordering of the indices  $V \cup E$

of the columns of  $L$  chosen before beginning the simplex method. We show how to calculate these rows combinatorially. We also show that the rows of  $L_V$  are sufficient for calculating the lexicographically minimum row of  $L$  if we index the first  $|V|$  columns of  $L$  with  $V$  in the prescribed ordering; i.e. we need only compare the first  $|V|$  entries of any tied rows in order to break the tie. Note that this technique of shortening of the lexicography rows can be done for any problem with upper bounds on the variables (see [Murty, 1983]).

We first consider how to calculate a specific row of  $(A')^{-1}$ . Finding row  $i$  of  $(A')^{-1}$  is equivalent to finding the solution  $(\bar{y}, \bar{\mu})$  for

$$(y, \mu)A' = e^i \quad (4.3.1)$$

where  $y \in \mathbb{R}^V$ ,  $\mu \in \mathbb{R}^E$ , and  $e^i$  represents row  $i$  of  $I^{|V|+|E|}$ . Using the structure of the basis  $(T, B, Z)$ , this can be done using a method similar to Algorithm (4.2.7) for finding the basic dual solution. First we calculate  $\bar{y}$  by solving

$$yA_T = w \quad (4.3.2)$$

for a particular right-hand side  $w \in \mathbb{R}^T$ . We then calculate the corresponding value of  $\bar{\mu}$  as in step (3) of Algorithm (4.2.7).

The rows of  $(A')^{-1}$  correspond to the basic variables  $x_T$ ,  $s_T$ ,  $x_B$  and  $s_Z$ . For each of these four types of rows the vector  $w \in \mathbb{R}^T$  for (4.3.2) and the corresponding values of  $\bar{y}$  and  $\bar{\mu}$  are easily calculated from (4.3.1). They are as follows.

Case 1. Row  $i$  of  $(A')^{-1}$  corresponds to basic variable  $x_e$ ,  $e \in T$ .

The vector  $w \in \mathbb{R}^T$  for (4.3.2) is defined by

$$w_j = \begin{cases} 1 & \text{for } j = e \\ 0 & \text{otherwise.} \end{cases}$$

Thus the solution  $\bar{y}$  for (4.3.2) is the row of  $A_T^{-1}$  corresponding to edge  $e$ . This can be found using the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.8).

The corresponding value of  $\bar{\mu} \in \mathbb{R}^E$  is defined by

$$\bar{\mu}_{uv} = \begin{cases} \bar{y}_u + \bar{y}_v & \text{for } uv \in B \\ 0 & \text{otherwise.} \end{cases}$$

Case 2. Row  $i$  of  $(A')^{-1}$  corresponds to basic variable  $s_e$ ,  $e \in T$ .

The vector  $w \in \mathbb{R}^T$  for (4.3.2) is defined by

$$w_j = \begin{cases} -1 & \text{for } j = e \\ 0 & \text{otherwise.} \end{cases}$$

Thus the solution  $y$  for (4.3.2) is the negative of the row of  $A_T^{-1}$  corresponding to edge  $e$ , which can be found using Algorithm (4.1.8).

The corresponding value of  $\bar{\mu} \in \mathbb{R}^E$  is defined by

$$\bar{\mu}_{uv} = \begin{cases} \bar{y}_u + \bar{y}_v & \text{for } uv \in B \\ -1 & \text{for } uv = e \\ 0 & \text{otherwise.} \end{cases}$$

Case 3. Row  $i$  of  $(A')^{-1}$  corresponds to basic variable  $x_e$ ,  $e \in B$ .

The vector  $w \in \mathbb{R}^T$  for (4.3.2) is defined by  $w = 0$ , thus  $\bar{y} = 0$  in this case. The corresponding value of  $\bar{\mu} \in \mathbb{R}^E$  is defined by

$$\bar{\mu}_j = \begin{cases} -1 & \text{for } j = e \\ 0 & \text{otherwise.} \end{cases}$$

Case 4. Row  $i$  of  $(A')^{-1}$  corresponds to basic variable  $s_e$ ,  $e \in Z$ .

The vector  $w \in \mathbb{R}^T$  for (4.3.2) is zero, and thus  $\bar{y} = 0$ .

The corresponding value of  $\bar{\mu} \in \mathbb{R}^E$  is defined by

$$\bar{\mu}_j = \begin{cases} -1 & \text{for } j = e \\ 0 & \text{otherwise.} \end{cases}$$

In order to now calculate a specific row of matrix  $L$  we take the corresponding row of  $(A')^{-1}$  and divide it by the corresponding component of  $\bar{a}$ . We can calculate  $\bar{a}$  by solving for  $[x_T, x_B, s_T, s_Z]$  in

$$A' \begin{bmatrix} x_T \\ x_B \\ s_T \\ s_Z \end{bmatrix} = a$$

where  $a$  is the column of the coefficient matrix in (4.2.4) corresponding to the entering variable  $x_e$ ,  $e \in Z$  or  $s_e$ ,  $e \in B$ . Calculate  $k_e = A_T^{-1}A_e$  using the even cycle or dumbbell formed when  $e$  is added to  $T$ , and Algorithm (4.1.6). Then we have the following two cases.

Case 1. The entering variable is  $x_e$ ,  $e \in Z$ .

In this case  $\bar{a} = [x_T, x_B, s_T, s_Z]$  where  $x_T = k^e$ ,  $x_B = 0$ ,  $s_T = -k^e$ , and  $s_Z$  is defined by

$$s_Z = \begin{cases} 1 & \text{in the component corresponding to } e \\ 0 & \text{elsewhere.} \end{cases}$$

Case 2. The entering variable is  $s_e$ ,  $e \in B$ .

In this case  $\bar{a} = [x_T, x_B, s_T, s_Z]$  where  $x_T = -k^e$ ,  $s_T = k^e$ ,  $s_Z = 0$  and  $x_B$  is defined by

$$x_B = \begin{cases} 1 & \text{in the component corresponding to } e \\ 0 & \text{elsewhere.} \end{cases}$$

Now consider the set  $K$  of possible leaving variables among which we may have to break a tie. If the edge indexing the entering variable is  $e$ , then  $K \subseteq x_T \cup s_T \cup \{s_e\} \cup \{x_e\}$ . Furthermore  $K$  contains at most one of  $x_j$  and  $s_j$  for  $j \in T \cup \{e\}$ . Thus if  $\ell$  and  $m$  are two rows of matrix  $L$  corresponding to variables in  $K$ , it is easily verified from our previous calculations that  $\ell_V \neq m_V$ . If we assume the first  $|V|$  columns of  $L$  are indexed by  $V$ , it follows that the rows of  $L_V$  are sufficient for determining which of the rows of  $L$  corresponding to the variables in  $K$  is lexicographically minimum.

In summary, we can choose a leaving variable lexicographically in our simplex method for the linear program (4.2.1) as follows.

- (1) Given the current basis  $(T, B, Z)$  for (4.2.1) and entering variable  $x_e$ ,  $e \in Z$  or  $s_e$ ,  $e \in B$ , determine the set of variables  $K$  which are the possible leaving variables found by the simplex method.
- (2) If  $|K| = 1$ , pivot as usual. Otherwise, let  $k^e \in \mathbb{R}^T$  be as described in the circuit algorithm (4.1.6) (note that  $k^e$  will have already been calculated earlier in the pivot operation). Then each row  $\ell$  of  $L_V$  corresponding to a variable in  $K$  is found as follows:
  - (a) If the variable is  $s_j$  for some  $j \in Z$  or  $x_j$  for some  $j \in B$  then  $\ell = 0$ .
  - (b) If the variable is  $s_j$  or  $x_j$  for some edge  $j \in T$  then find the row of  $A_T^{-1}$  corresponding to  $j$ , denoted  $a^j$ , using the tree formed when edge  $j$  is removed from the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.8). Then  $\ell$  is defined by
 
$$\ell = \begin{cases} a^j/k^e & \text{if } x_e, e \in Z \text{ was the entering} \\ & \text{variable} \\ -a^j/k^e & \text{if } s_e, e \in B \text{ was the entering} \\ & \text{variable.} \end{cases}$$
- (3) Using a prescribed ordering  $v_1, v_2, \dots, v_n$  of the node set  $V = \{v_i \mid i = 1, 2, \dots, n\}$  (chosen before beginning the simplex method), find the variable in  $K$  for which the corresponding row in  $L_V$  is lexicographically minimum. Choose this variable as the leaving variable.

4.4 SOLVING THE FRACTIONAL 2-FACTOR PROBLEM

The fractional 2-factor problem is a special case of linear program (4.0.2) in which  $b = 2$  and  $u = 1$ . As a result we can solve this problem using the finite simplex method described in Sections 4.2 and 4.3. An initial feasible basis  $(T, B, Z)$  for the problem is easily constructed from any hamilton cycle  $C$  of  $K_n = (V, E)$  as follows. Choose an edge  $i \in E(C)$ , and an edge  $j \in (E \setminus E(C)) \cup \{i\}$  which creates an odd cycle when added to  $G' = (V, E(C) \setminus \{i\})$ . Then let  $T = (E(C) \setminus \{i\}) \cup \{j\}$ , let  $B = \{i\}$  if  $i \neq j$  or let  $B = \emptyset$  if  $i = j$ , and let  $Z = E \setminus (T \cup B)$ .

There also exist other efficient methods for solving the fractional 2-factor problem. For example, we can solve this problem in polynomial time by transforming it into an instance of an uncapacitated bipartite b-matching problem as follows. First form a new graph  $\bar{G} = (\bar{V}, \bar{E})$  from  $K_n = (V, E)$  by replacing each edge  $e = uv \in E$  by a  $uv$ -path  $u, u', v', v$ . Let the new edges replacing  $e$  be  $e_1 = uu'$ ,  $e_2 = u'v'$ , and  $e_3 = v'v$  (see Figure 4.4.1), and consider the problem

$$\begin{aligned} &\text{minimize} && \bar{c}x && (4.4.1) \\ &\text{subject to} && \bar{A}x = \bar{b} \\ & && x \geq 0 \end{aligned}$$

where  $\bar{A}$  is the node-edge incidence matrix of  $\bar{G}$ ,  $\bar{b} \in \mathbb{R}^{\bar{V}}$  is defined by

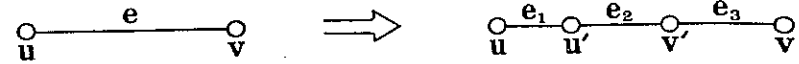


Figure 4.4.1

$$\bar{b}_v = \begin{cases} 2 & \text{if } v \in V \\ 1 & \text{otherwise,} \end{cases}$$

and  $\bar{c} \in \mathbb{R}^{\bar{E}}$  is defined by

$$\bar{c}_f = \begin{cases} c_e & \text{if } f = e_1 \text{ for some } e \in E \\ 0 & \text{otherwise.} \end{cases}$$

Given any solution  $x$  for the fractional 2-factor problem for  $K_n$ , we can construct a solution  $\bar{x} \in \mathbb{R}^{\bar{E}}$  for (4.4.1) by letting  $\bar{x}_{e_1} = x_e$ ,  $\bar{x}_{e_3} = x_e$ , and  $\bar{x}_{e_2} = 1 - x_e$  for every  $e \in E$ . Conversely, a solution  $\bar{x}$  for (4.4.1) corresponds to a solution  $x$  for the fractional 2-factor problem in which  $x_e = \bar{x}_{e_1}$  for all  $e \in E$ .

We then transform problem (4.4.1) into an instance of bipartite b-matching by splitting each node  $v \in \bar{V}$  into nodes  $v_1$  and  $v_2$  and replacing each edge  $e = uv \in \bar{E}$  by edges  $e_1 = u_1v_2$  and  $e_2 = u_2v_1$ . Letting  $G' = (V_1 \cup V_2, E')$  be the resulting bipartite graph, we consider the bipartite b-matching problem



$$\begin{aligned} & \text{minimize} && c'x && (4.4.2) \\ & \text{subject to} && A'x = b' \\ & && x \geq 0 \end{aligned}$$

where  $A'$  is the node-edge incidence matrix of  $G'$ ,  $b' \in \mathbb{R}^{V_1 \cup V_2}$  is defined by  $b'_{v_1} = b'_{v_2} = b_v$  for all  $v \in \bar{V}$ , and  $c' \in \mathbb{R}^E$  is defined by  $c'_{u_1 v_2} = c'_{v_1 u_2} = \frac{1}{2} c_{uv}$  for all  $uv \in \bar{E}$ . A solution  $\bar{x}$  for (4.4.1) corresponds to the solution  $x'$  in (4.4.2) in which  $x'_{u_1 v_2} = x'_{u_2 v_1} = \bar{x}_{uv}$  for all  $uv \in \bar{E}$ . Conversely, a solution  $x'$  for (4.4.2) corresponds to the solution  $\bar{x}$  for (4.4.1) in which  $\bar{x}_{uv} = \frac{1}{2} (x'_{u_1 v_2} + x'_{u_2 v_1})$  for all  $uv \in \bar{E}$ . Problem (4.4.2) can be solved in polynomial time using network flows (see [Lawler, 1976]).

The fractional 2-factor problem can also be solved directly in polynomial time by a primal-dual method. This method is a simplified version of the primal-dual method developed by [Edmonds and Johnson, 1970] and described in [Havel, 1982] for the integer 2-factor problem. In the fractional case, the fact that we allow values of  $1/2$  in the solution makes the shrinking of blossoms unnecessary as we can augment instead. The details of the algorithm are easily deduced from [Havel, 1982] and thus are not included here.

## CHAPTER 5

The Subtour Polytope

Given  $K_n = (V, E)$  and a vector  $c \in \mathbb{R}^E$  of edge costs, the subtour problem is to find  $x \in \mathbb{R}^E$  which minimizes  $cx$  subject to  $x$  satisfying the degree constraints (3.1.2), non-negativity constraints (3.2.1), and the subtour elimination constraints (3.2.3) for the travelling salesman polytope  $Q_1^n$ ; i.e. it is the linear program

$$\begin{aligned} & \text{minimize} && cx && (5.0.1) \\ & \text{subject to} && x(\delta(v)) = 2 && \text{for all } v \in V \\ & && x(\delta(S)) \leq |S| - 1 && \text{for all } S \subseteq V, 2 \leq |S| \leq n - 2, \\ & && x_e \geq 0 && \text{for all } e \in E. \end{aligned}$$

The associated subtour polytope, which we denote by  $Q_S^n$ , is defined by

$$Q_S^n = \{x \in \mathbb{R}^E \mid Ax = 2, Bx \leq b\}, \quad (5.0.2)$$

where  $A$  is the node-edge incidence matrix for  $K_n$ , and  $Bx \leq b$  consists of all subtour elimination constraints (3.2.3) and non-negativity constraints (3.2.1).

In the first part of this chapter we examine the structure of the vertices of  $Q_S^n$ . In particular, we look at the types of

fractions which can occur as components of a vertex  $x$  of  $Q_S^n$ , as well as discuss the structure of the support of  $x$ .

In the latter part of this chapter we optimize a class of objective functions obtained from the clique tree inequalities (3.3.1) over  $Q_S^n$ .

### 5.1 THE STRUCTURE OF THE VERTICES OF THE SUBTOUR POLYTOPE

The vertices of some polytopes have a very specific structure, and knowledge of this structure can prove useful when developing combinatorial algorithms. For instance, consider the fractional 2-factor polytope  $Q_F^n$ . As discussed in Chapter 4, if  $\hat{x}$  is a vertex of  $Q_F^n$ , then  $\hat{x}$  is half-integer, and the set of edges  $J$  such that  $\hat{x}_j = \frac{1}{2}$  for all  $j \in J$  partitions into the edge sets of an even number of odd disjoint cycles (see [Balinski, 1970]).

The following result shows that, in general, the vertices of the subtour polytope  $Q_S^n$  have a much more complicated structure.

For any  $F \subseteq Q_S^n$ , let  $B_F = \{j \in E(K_n) \mid x_j = 0 \text{ for all } x \in F\}$  and let  $D_F = \{S \subseteq V(K_n) \mid 2 \leq |S| \leq n - 2 \text{ and } x(\delta(S)) = 2 \text{ for all } x \in F\}$ .

(5.1.1) Theorem. For any positive integer  $k \geq 3$  there exists a vertex of  $Q_S^{2k+4}$  which contains components of the form  $\frac{a}{k}$ , where  $\gcd(a, k) = 1$ .

Proof. Consider the point  $\bar{x} \in Q_S^{2k+4}$  corresponding to Figure 5.1.1, where the support of  $\bar{x}$  is a ladder-like structure with  $k$  "rungs"  $e_1, e_2, \dots, e_k$ . Figure 5.1.1 shows  $x$  for  $k$  odd. For  $k$  even, the two triangles would be on the same side of the figure.

We show that  $\bar{x}$  is a vertex of  $Q_S^{2k+4}$  by showing that  $F = \{\bar{x}\}$ , where

$$F = \{x \in Q_S^{2k+4} \mid x_e = 0 \text{ for all } e \in B_{\bar{x}}, x(\delta(S)) = 2 \text{ for all } S \in D_{\bar{x}}\}$$

Suppose  $\bar{x} \in F$ . Then  $\bar{x}_e = 0$  for all  $e \in B_{\bar{x}}$ , and  $\bar{x}_e = 1$  whenever  $\bar{x}_{(a,b)} = \lambda$ , where  $a$  and  $b$  are shown in Figure 5.1.1. Then in order for  $\bar{x}$  to satisfy degree constraints, the remaining components of  $\bar{x}$  must have the values shown in Figure 5.1.2.

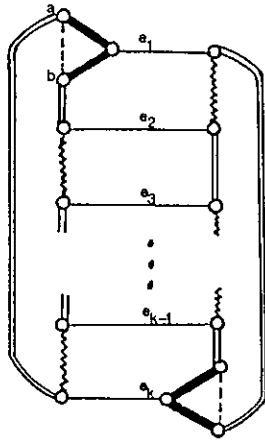
Consider  $S \subseteq V(K_{2k+4})$  such that  $\delta(S) = \{e_1, e_2, \dots, e_k\}$ . Since  $S \in D_{\bar{x}}$  it follows that

$$k(2\lambda) = \bar{x}(\delta(S)) = 2$$

and  $\lambda = \frac{1}{k}$ . Thus  $\bar{x} = \bar{x}$ .  $\square$

Before proving some results concerning the structure of the support of vertices of  $Q_S^n$ , we establish two preliminary results.

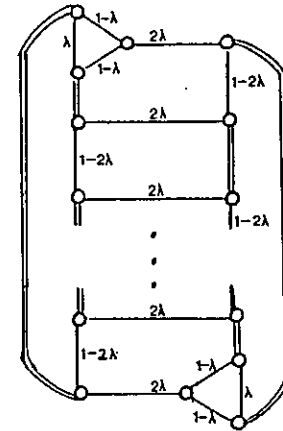
Let  $V$  be a non-empty set and let  $C = \{C_1, C_2, \dots, C_k\}$  where  $C_i \subseteq V$ ,  $i = 1, 2, \dots, k$ . We say sets  $C_i, C_j$  cross if  $C_i \cap C_j \neq \emptyset$  and neither  $C_i \subseteq C_j$  nor  $C_j \subseteq C_i$ . We call  $C$  a nested family if it contains no pairs of sets which cross.



- corresponds to  $\bar{x}_e = 1$
- corresponds to  $\bar{x}_e = \frac{2}{k}$
- corresponds to  $\bar{x}_e = \frac{k-2}{k}$
- corresponds to  $\bar{x}_e = \frac{k-1}{k}$
- corresponds to  $\bar{x}_e = \frac{1}{k}$

- edges with  $\bar{x}_e = 0$  are not shown.

Figure 5.1.1



- corresponds to  $\bar{x}_e = 1$

- edges with  $\bar{x}_e = 0$  are not shown.

Figure 5.1.2

(5.1.2) Theorem. Let  $F$  be any face of  $Q_S^n$ . Then  $F$  can be defined as

$$F = \{x \in Q_S^n \mid x_e = 0 \text{ for all } e \in B_F, \text{ and } x(\delta(S)) = 2 \text{ for all } S \in D^*\}$$

for some  $D^* \subseteq D_F$  such that  $D^*$  is a nested family.

Proof. By definition,

$$F = \{x \in Q_S^n \mid x_e = 0 \text{ for all } e \in B_F, \text{ and } x(\delta(S)) = 2 \text{ for all } S \in D\}$$

for some  $D \subseteq D_F$ . Suppose  $D$  contains a pair of sets  $S_1, S_2$  which cross. Let  $C = \{e \in E(K_n) \mid e \in (\delta(S_1) \cap \delta(S_2)) \setminus \delta(S_1 \cap S_2)\}$ , and define  $F' = \{x \in Q_S^n \mid x_e = 0 \text{ for all } e \in B_F, \text{ and } x(\delta(S)) = 2 \text{ for all } S \in D'\}$ , where  $D' = (D \cup \{S_1 \cup S_2, S_1 \cap S_2\}) \setminus \{S_1, S_2\}$ . We will show  $F = F'$ .

First observe that for any  $x \in \mathbb{R}^{E(K_n)}$ ,

$$x(\delta(S_1 \cap S_2)) + x(\delta(S_1 \cup S_2)) + 2(x(C)) = x(\delta(S_1)) + x(\delta(S_2)). \quad (5.1.3)$$

Hence for  $x \in F$  we have

$$\begin{aligned} 4 &= x(\delta(S_1)) + x(\delta(S_2)) \\ &= x(\delta(S_1 \cap S_2)) + x(\delta(S_1 \cup S_2)) + 2(x(C)) \\ &\geq 4 + 2(x(C)). \end{aligned}$$

It follows that  $C \subseteq B_F$ ,  $x(\delta(S_1 \cup S_2)) = 2$ ,  $x(\delta(S_1 \cap S_2)) = 2$  and hence  $x \in F'$ .

Now, suppose  $x \in F'$ . Using (5.1.3) and the above it follows

that

$$\begin{aligned} 4 &= x(\delta(S_1 \cap S_2)) + x(\delta(S_1 \cup S_2)) + 2(x(C)) \\ &= x(\delta(S_1)) + x(\delta(S_2)) \\ &\geq 4. \end{aligned}$$

Hence  $x(\delta(S_1)) = 2$ ,  $x(\delta(S_2)) = 2$ , and thus  $x \in F$ .

From the above we can conclude  $F = F'$ . Furthermore, if either  $S_1 \cup S_2 \in D$  or  $S_1 \cap S_2 \in D$  then  $|D'| < |D|$ , and if  $S_1 \cup S_2 \notin D$  and  $S_1 \cap S_2 \notin D$  then

$$\sum_{S_i \in D'} |S_i|^2 = \sum_{S_i \in D} |S_i|^2 + 2(|S_1 \setminus S_2| + |S_2 \setminus S_1|) > \sum_{S_i \in D} |S_i|^2.$$

Since  $|D|$  is finite and  $\sum_{S_i \in K} |S_i|^2$  is bounded above for any set  $K$  of subsets of  $V(K_n)$ , repeatedly applying the "uncrossing" procedure described above must eventually result in  $D^*$  as required.  $\square$

Theorem (5.1.2) also appears in [Cornuéjols, Fonlupt and Naddef, 1983], Theorem (3.11).

(5.1.4) Lemma. Let  $C$  be any nested family of distinct subsets of a non-empty set  $V$ . Then  $|C| \leq 2|V| - 1$ .

Proof. The proof follows easily by induction on  $|V|$ . For details see [Pulleyblank, 1973].  $\square$

For any  $\bar{x} \in Q_S^n$ , let  $G_{\bar{x}}$  denote the support of  $\bar{x}$ . Given such a graph  $G_{\bar{x}}$  we let  $V_i$  represent the set of nodes of degree  $i$  in  $G_{\bar{x}}$ , and  $\bar{V}_i$  represent the set of nodes of degree  $i$  not incident with an edge  $j$  for which  $\bar{x}_j = 1$ .

Let  $C$  be a cycle of length three in  $G_{\bar{x}}$ . If  $V(C) \subseteq \bar{V}_3$  and  $\bar{x}(\delta(V(C))) = 2$ , then we call  $C$  a triangle configuration. Thus if  $C$  is a triangle configuration in  $G_{\bar{x}}$ , the three components of  $\bar{x}$  corresponding to the edges of  $\delta(V(C))$  must have values  $\alpha_1, \alpha_2$ , and  $\alpha_3$  such that  $0 < \alpha_i < 1$  for  $i \in \{1,2,3\}$  and  $\sum_{i=1}^3 \alpha_i = 2$ . Because degree constraints must be satisfied, it follows that the components of  $\bar{x}$  corresponding to edges in  $E(C)$  must also have values  $\alpha_1, \alpha_2$ , and  $\alpha_3$  as shown in Figure 5.1.3.

Given  $\bar{x} \in Q_S^n$  such that  $G_{\bar{x}}$  contains a triangle configuration  $C$ , let  $\bar{x}$  be a vector obtained from  $\bar{x}$  by removing the components corresponding to  $E(C)$ . We denote  $\bar{x}$  by  $\bar{x}+C$ .

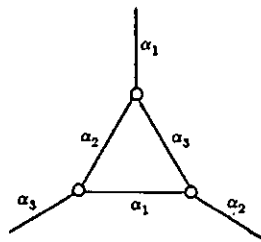


Figure 5.1.3

(5.1.5) Lemma. Let  $\bar{x} \in Q_S^n$  for some  $n \geq 5$  and let  $C$  be a triangle configuration in  $G_{\bar{x}}$ . Then  $\bar{x}+C \in Q_S^{n-2}$ .

Proof. If all edges of  $\delta(V(C))$  have a common endpoint then  $G_{\bar{x}} \cong K_4$  implying  $\bar{x} \in Q_S^4$ . If exactly two edges of  $\delta(V(C))$  have a common endpoint  $u$ , then  $\bar{x}(\delta(V(C) \cup \{u\})) < 2$  implying  $\bar{x} \in Q_S^n$ . Hence each edge of  $\delta(V(C))$  must have a distinct endpoint, and contracting  $C$  in  $G_{\bar{x}}$  creates no multiple edges. Since  $\bar{x}+C$  also satisfies all the constraints for  $Q_S^{n-2}$ , it follows that  $\bar{x}+C \in Q_S^{n-2}$ .  $\square$

Given  $\bar{x} \in Q_S^{n-2}$  and  $v \in \bar{V}_3$  in  $G_{\bar{x}}$ , we can replace  $v$  by a triangle configuration  $C$  to obtain a corresponding vector  $\bar{x}$ , where the values of the components of  $\bar{x}$  corresponding to  $E(C)$  are uniquely determined by those of  $\delta(v)$  in  $\bar{x}$  as in Figure 5.1.3. We denote  $\bar{x}$  by  $\bar{x}+v$ .

(5.1.6) Lemma. Let  $\bar{x} \in Q_S^n$  and let  $v \in \bar{V}_3$  in  $G_{\bar{x}}$ . Then  $\bar{x}+v \in Q_S^{n+2}$ .

Proof. Let  $\bar{x} = \bar{x}+v$ . Clearly  $\bar{x}$  satisfies all the degree and non-negativity constraints for  $Q_S^{n+2}$ . Furthermore, if there exists  $S \subseteq V(G_{\bar{x}})$  such that  $\bar{x}(\delta(S)) < 2$ , we can assume  $|\delta(S) \cap E(C)| = 2$ . But then either  $\bar{x}(\delta(S \setminus V(C)) \cup \{v\}) < 2$  or  $\bar{x}(\delta(S \setminus V(C))) < 2$ , contradicting the fact that  $\bar{x} \in Q_S^n$ . Hence  $\bar{x}$  must satisfy all subtour elimination constraints for  $Q_S^{n+2}$  and thus  $\bar{x} \in Q_S^{n+2}$ .  $\square$

(5.1.7) Corollary. Let  $\bar{x} \in Q_S^n$  and let  $C$  be a triangle configuration in  $G_x$ . Then  $\bar{x}$  is a vertex of  $Q_S^n$  if and only if  $\bar{x}+C$  is a vertex of  $Q_S^{n-2}$ .

Proof. Noting that  $\bar{V}_3 = \emptyset$  for  $x \in Q_S^n$ ,  $n \leq 5$ , the above result follows from Lemmas (5.1.5) and (5.1.6) and the fact that  $x$  is not a vertex of  $Q_S^n$  if and only if  $x$  can be expressed as a convex combination of  $k$  distinct points of  $Q_S^n$ ,  $k \geq 2$ .  $\square$

Using Theorem (5.1.2) along with Lemma (5.1.4) and Corollary (5.1.7), it is possible to obtain an upper bound on the number of tight subtour elimination constraints which are necessary to define a vertex of  $Q_S^n$ .

(5.1.8) Theorem. Let  $\bar{x}$  be a vertex of  $Q_S^n$ ,  $n \geq 3$ . Then there exists  $D \subseteq D_x$  such that  $\{\bar{x}\} = \{x \in Q_S^n \mid x_e = 0 \text{ for all } e \in B_x, \text{ and } x(\delta(S)) = 2 \text{ for all } S \in D\}$  and  $|D| \leq n - \frac{|\bar{V}_3|}{2} - 3$ .

Proof. We prove the result by induction on  $n$ . If  $n = 3$  then  $\bar{x} = (1,1,1)$  and  $D = \emptyset$  suffices.

Now assume the theorem is true for  $3 \leq n \leq k-1$  and let  $\bar{x}$  be a vertex of  $Q_S^k$ . If there exists a triangle configuration  $C$  in  $G_x$  then  $\bar{x} = \bar{x}+C$  is a vertex of  $Q_S^{k-2}$  by Corollary (5.1.7).

Hence by the induction hypothesis there exists  $D' \subseteq D_x$  such that  $\{\bar{x}\} = \{x \in Q_S^{k-2} \mid x_e = 0 \text{ for all } e \in B_x \text{ and } x(\delta(S)) = 2 \text{ for all } S \in D'\}$  and  $|D'| \leq k - \frac{|\bar{V}_3|}{2} - 4$ . Letting  $v \in V(K_{k-2})$  be the node resulting from the contraction of  $C$ , define  $D = \{V(C)\} \cup \{S' \subseteq V(K_n) \mid S' = (S \setminus \{v\}) \cup V(C) \text{ for some } S \in D'\}$ . The result now follows for  $\bar{x}$  and  $D$ .

Suppose  $G_x$  contains no triangle configuration. (In this case we do not actually require induction.) By Theorem (5.1.2) there exists  $D \subseteq D_x$  sufficient to define  $\bar{x}$  such that  $D$  is a nested family. Also, since subtour elimination constraints are equivalent for  $S \subseteq V(K_n)$  and  $\bar{S} = V(K_n) \setminus S$  (see [Grötschel and Padberg, 1979b]), we can assume  $S \in D$  implies  $\bar{S} \notin D$ . Let  $D' = D \cup \{\bar{S}\}$  for some maximal  $S \in D$ , and let  $C = D' \cup \{\{v\} \mid v \in V(K_n)\} \cup \{V(K_n)\}$ . Then  $C$  is a nested family, and  $|C| = |D| + n + 2$ . We now show that there exists  $F \subseteq \{S \subseteq V(K_n)\} \setminus C$  such that  $|F| \geq \frac{|\bar{V}_3|}{2}$  and  $F \cup C$  is a nested family.

For each  $v \in \bar{V}_3$  there is a unique minimal  $S \in D'$  such that  $v \in S$ . If  $S$  properly contains at least one member of  $D'$  then, letting  $T = \{v \in S \mid v \notin S' \text{ for any } S' \subset S\}$ , we define  $\left\lfloor \frac{|T|}{2} \right\rfloor$  subsets of  $V(K_n)$  different from those in  $C$  as follows. First form  $\left\lfloor \frac{|T|}{2} \right\rfloor$  sets which consist of disjoint pairs of nodes in  $T$ . Then, if  $T$  is odd, add a set consisting of the singleton not included in the previous pairs plus any  $S' \in D'$  such that  $S' \subset S$ . Note that, in the case  $T = \{v\}$ ,  $S \setminus T \notin D'$  since  $v \in \bar{V}_3$  and  $\bar{x}(\delta(S)) = 2$  implies

$\bar{x}(\delta(S \setminus T)) > 2$ , and thus  $S \cup \{v\} \neq S$ .

If  $S$  does not properly contain any members of  $D'$  then  $|S| \geq 2$ . In fact, since  $S$  contains  $v \in \bar{V}_3$  and  $\bar{x}(\delta(S)) = 2$ ,  $|S| \geq 3$ . If  $|S| = 3$  then  $|S \cap \bar{V}_3| \leq 2$  since, by assumption,  $S$  does not form a triangle configuration in  $G_{\bar{x}}$ . In this case we define one subset of  $V(K_n)$  different from those in  $C$  by choosing any pair of nodes in  $S$ . If  $|S| \geq 4$  then we define  $\lfloor \frac{|S|}{2} \rfloor$  new subsets of  $V(K_n)$  as follows. First form  $\lfloor \frac{|S|}{2} \rfloor$  sets which consist of disjoint pairs of nodes in  $S$ . Then, if  $|S|$  is odd, add a set consisting of one of the pairs and the singleton not previously included in the pairs.

Let  $F$  be the set of all subsets defined above, considering each minimal  $S \in D'$  containing some  $v \in \bar{V}_3$ . Since  $F$  satisfies the required properties,  $F \cup C$  is a nested family such that  $|F \cup C| \geq |D| + n + \frac{|\bar{V}_3|}{2} + 2$ . By Lemma (5.1.4),  $|F \cup C| \leq 2n - 1$  and thus  $|D| \leq n - \frac{|\bar{V}_3|}{2} - 3$  as required.  $\square$

From Theorem (5.1.8) we obtain the following two corollaries concerning vertices of  $Q_S^n$ .

(5.1.9) Corollary. Let  $\bar{x}$  be a vertex of  $Q_S^n$ ,  $n \geq 3$ . Then  $|E(G_{\bar{x}})| + \frac{|\bar{V}_3|}{2} \leq 2n - 3$ .

Proof. Let  $D$  be as in Theorem (5.1.8). Then, since  $\bar{x}$  is a vertex of  $Q_S^n$ ,

$$|B_{\bar{x}}| + |D| \geq \dim(Q_S^n) = \binom{n}{2} - n.$$

Hence

$$|E(G_{\bar{x}})| = \binom{n}{2} - |B_{\bar{x}}| \leq |D| + n.$$

By Theorem (5.1.8),  $|D| \leq n - \frac{|\bar{V}_3|}{2} - 3$  and thus  $|E(G_{\bar{x}})| \leq 2n - \frac{|\bar{V}_3|}{2} - 3$  as required.  $\square$

(5.1.10) Corollary. Let  $\bar{x}$  be any vertex of  $Q_S^n$ ,  $n \geq 3$ . Then  $|\{j \in E(K_n) \mid \bar{x}_j = 1\}| \geq 3$ .

Proof. Since  $2|E(G_{\bar{x}})| = \sum_{i=1}^n i|V_i|$ , it follows that  $|E(G_{\bar{x}})| \geq |V_2| + \frac{3}{2}|V_3| + 2(n - |V_2| - |V_3|)$ . Thus by Corollary (5.1.9),

$$|V_2| + \frac{|V_3 \setminus \bar{V}_3|}{2} \geq 3. \quad (5.1.11)$$

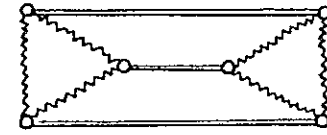
Every node in  $V_2$  is incident with two edges  $j$  for which  $\bar{x}_j = 1$  and every node in  $V_3 \setminus \bar{V}_3$  is incident with one such edge. Thus  $|\{j \in E(K_n) \mid \bar{x}_j = 1\}| \geq |V_2| + \frac{|V_3 \setminus \bar{V}_3|}{2}$  and the result follows from (5.1.11).  $\square$

The bounds in Corollaries (5.1.9) and (5.1.10) are both tight for the vertex of  $Q_S^6$  shown in Figure 5.1.4.

We know of no infinite family of vertices for which either bound is always tight. However, consider the vertex  $x$  of  $Q_S^{10}$  shown in Figure 5.1.5 which is obtained from Figure 5.1.1. By repeatedly expanding nodes  $v_1$  and/or  $v_2$  into triangle configurations we obtain an infinite family of vertices  $\bar{x}$  for which  $|E(G_{\bar{x}})| = 2n - \frac{|\bar{V}_3|}{2} - 4$  and  $|\{j \in E(K_n) \mid \bar{x}_j = 1\}| = 4$ .

5.2 CLIQUE TREE INEQUALITIES AND  $Q_S^n$

For an introduction to clique trees and clique tree inequalities see Section 3.3. For the remainder of this section we represent the clique tree inequality (3.3.1) by  $a_C x \leq \alpha_C$ .





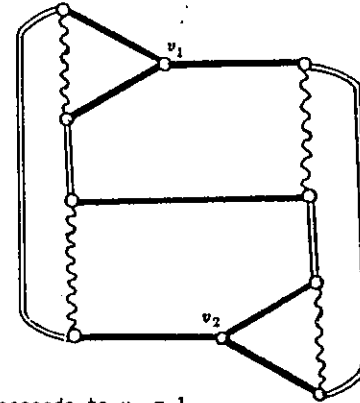
-  - corresponds to  $x_e = 1$
-  - corresponds to  $x_e = \frac{1}{2}$
- edges for which  $x_e = 0$  are not shown.

Figure 5.1.4






-  - corresponds to  $x_e = 1$
-  - corresponds to  $x_e = \frac{2}{3}$
-  - corresponds to  $x_e = \frac{1}{3}$
- edges for which  $x_e = 0$  are not shown.

Figure 5.1.5



For any clique tree  $C$  define

$$\text{GAP}(C) = \max\{\alpha_C x \mid x \in Q_S^n\} - \alpha_C.$$

Thus  $\text{GAP}(C)$  is the amount by which the right-hand side of the clique tree inequality corresponding to  $C$  is less than it would have to be if the inequality were to be valid for  $Q_S^n$ . Therefore, in a sense, it measures the amount of  $Q_S^n$  "cut off" by the inequality.

Theorem (5.2.2) below shows that  $\text{GAP}(C)$  is determined by the number of non-pendent teeth in  $C$ . First we require the following lemma.

(5.2.1) Lemma. Let  $K$  be a clique in a graph  $G$ , let  $R \subseteq K$  satisfy  $|R| \geq 3$ , and let  $S = K \setminus R$ . Then there exists  $\bar{x}^{(K)} \in \mathbb{R}^{\gamma(K)}$  satisfying

- (1)  $\bar{x}^{(K)}(\delta(v)) = 2$  for all  $v \in S$ .
- (2)  $\bar{x}^{(K)}(\delta(v)) = 1$  for all  $v \in R$ .
- (3)  $\bar{x}^{(K)} \geq 0$ , and
- (4)  $\bar{x}^{(K)}(\delta(Q)) \geq 2$  for all nonempty  $Q \subseteq S$ .

Proof (by construction). Let  $S = \{s_1, s_2, \dots, s_{|S|}\}$  and  $R = \{r_1, r_2, \dots, r_{|R|}\}$ .

Case 1.  $|S|$  is even.

Construct a hamilton cycle  $D$  in  $\langle K \rangle$  that visits the nodes

in the order  $r_1, s_1, s_2, \dots, s_{\frac{|S|}{2}}, r_2, s_{\frac{|S|}{2}+1}, \dots, s_{|S|}, r_3, r_4, \dots, r_{|R|}$

(see Figure 5.2.1). Then  $\bar{x}^{(K)}$  defined by

$$\bar{x}_i^{(K)} = \begin{cases} \frac{1}{2} & \text{for edges } i \in D \\ 1 & \text{for edges } i \text{ of the form } (s_t, s_{|S|-t+1}) \text{ for} \\ & t = 1, 2, \dots, \frac{|S|}{2} \\ 0 & \text{otherwise} \end{cases}$$

satisfies the required properties.

Case 2.  $|S|$  is odd.

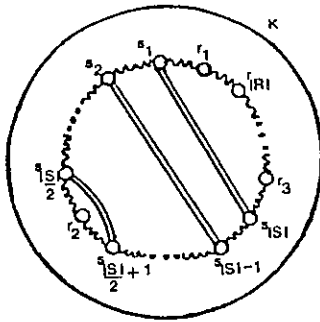
Construct a hamilton cycle  $D$  in  $\langle K \setminus \{r_{|R|}\} \rangle$  that visits the nodes in the order  $r_1, s_1, s_2, \dots, s_{\frac{|S|-1}{2}}, r_2, s_{\frac{|S|-1}{2}+1}, \dots, s_{|S|}, r_3, r_4, \dots, r_{|R|-1}$  (see Figure 5.2.2). Then  $\bar{x}^{(K)}$  defined by

$$\bar{x}_i^{(K)} = \begin{cases} \frac{1}{2} & \text{for edges } i \in D \\ 1 & \text{for edges } i \text{ of the form } (s_t, s_{|S|-t}) \\ & \text{for } t = 1, 2, \dots, \frac{|S|-1}{2}, \text{ and edge } (s_{|S|}, r_{|R|}) \\ 0 & \text{otherwise} \end{cases}$$

satisfies the required properties.  $\square$

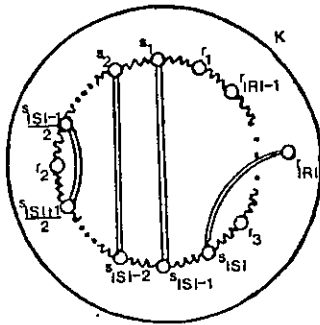
(5.2.2) Theorem. For any clique tree  $C$  for which  $H(C) \neq \emptyset$ ,

$$\text{GAP}(C) = \frac{|T_{\text{NP}}(C)| + 1}{2}.$$



$\text{wavy line}$  corresponds to  $\bar{x}_e^{(K)} = \frac{1}{2}$   
 $\text{straight line}$  corresponds to  $\bar{x}_e^{(K)} = 1$   
 - edges  $e$  with  $\bar{x}_e^{(K)} = 0$  are not shown.

Figure 5.2.1



$\text{wavy line}$  corresponds to  $\bar{x}_e^{(K)} = \frac{1}{2}$   
 $\text{straight line}$  corresponds to  $\bar{x}_e^{(K)} = 1$   
 - edges  $e$  with  $\bar{x}_e^{(K)} = 0$  are not shown.

Figure 5.2.2

Proof. Let the defining linear system for  $Q_S^n$  be

$$Ax = 2, Bx \leq b$$

where  $Ax = 2$  represents all degree constraints (3.1.2), and  $Bx \leq b$  consists of all subtour elimination constraints (3.2.3) and non-negativity constraints (3.2.1).

We wish to solve the linear program

$$(P) \quad \max\{a_c x \mid Ax = 2, Bx \leq b, x \in \mathbb{R}^E\}.$$

The dual of P is

$$(D) \quad \min\{\lambda 2 + \gamma b \mid \lambda A + \gamma B = a_c, \gamma \geq 0\}.$$

By the Complementary Slackness Theorem (1.4.6), feasible solutions  $\bar{x}$  and  $(\bar{\lambda}, \bar{\gamma})$  for P and D respectively are optimal if and only if  $B^i \bar{x} = b_i$  whenever  $\bar{\gamma}_i > 0$ , where  $B^i$  represents the  $i$ th row of B.

For any  $T \in T(C)$  define  $\hat{T} = T \setminus \{ U \mid T \cap H \}$  and for any  $H \in H(C)$  define  $\hat{H} = H \setminus \{ U \mid T \cap H \}$ . Let  $(\bar{\lambda}, \bar{\gamma})$  be defined by

$\bar{\lambda}_i = \begin{cases} \frac{1}{2} & \text{whenever } A^i x = 2 \text{ is a degree constraint for } v \notin \hat{T} \text{ for some } t \in T_p(C) \\ 0 & \text{otherwise} \end{cases}$

$$\bar{\lambda}_i = \begin{cases} \frac{1}{2} & \text{whenever } A^i x = 2 \text{ is a degree constraint for } v \notin \hat{T} \text{ for some } t \in T_p(C) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and } \bar{y}_i = \begin{cases} \frac{1}{2} & \text{whenever } B^i x \leq b_i \text{ is} \\ & \begin{cases} 1) \text{ a subtour elimination constraint for } T, \bar{T}, \text{ or} \\ & T \cap H \text{ for some } T \in T_P(C) \text{ and } H \in H(C) \text{ which} \\ & \text{intersects } T \text{ in } C. \\ 2) \text{ a non-negativity constraint for an edge} \\ & e \in \delta(H) \setminus E(C) \text{ for some } H \in H(C), \text{ or} \\ 3) \text{ a non-negativity constraint for an edge} \\ & e \in \delta(\bar{T}) \setminus E(C) \text{ for some } T \in T_{NP}(C). \end{cases} \\ 1 & \text{whenever } B^i x \leq b_i \text{ is a subtour elimination} \\ & \text{constraint for } T \cap H \text{ for some } T \in T_{NP}(C) \text{ and} \\ & H \in H(C) \text{ which intersects } T. \\ 0 & \text{otherwise.} \end{cases}$$

It can be checked by direct calculation that  $(\bar{x}, \bar{y})$  is a feasible solution for  $D$ . We show that it is also an optimal solution by constructing a feasible solution  $\bar{x}$  for  $P$  which satisfies the complementary slackness conditions as follows:

- (1) For each  $T \in T_P(C)$ , construct a hamilton path in  $\langle T \rangle$  with ends  $w_T \in \bar{T}$  and  $u_{T,H} \in T \cap H$ , where  $H$  is the unique handle in  $H(C)$  intersecting  $T$ . Construct this path so that it is also a hamilton path in  $\langle T \cap H \rangle$  and  $\langle \bar{T} \rangle$ .
- (2) For each  $T \in T_{NP}(C)$  construct a hamilton path in  $\langle T \cap H \rangle$  for each  $H \in H(C)$  which intersects  $T$ . Let the ends of this path be  $u_{T,H}$  and  $v_{T,H}$ .

- (3) For each  $T \in T_{NP}(C)$  for which  $t_T = 2$ , construct a hamilton path in  $\langle \bar{T} \cup \{v_{T,H_1}, v_{T,H_2}\} \rangle$  with ends  $v_{T,H_1}, v_{T,H_2}$ , where  $\langle H_1 \rangle$  and  $\langle H_2 \rangle$  are the two handles in  $C$  which intersect  $T$ .
- (4) For each  $H \in H(C)$ , use Lemma (5.2.1) to obtain  $\bar{x}^{(\bar{H})}$  for the clique  $\bar{H} = S \cup R$ , where  $S = \bar{H}$  and  $R = \{u_{T,H} \mid T \text{ intersects } H\}$ .
- (5) For each  $T \in T_{NP}(C)$  with  $t_T \geq 3$ , use Lemma (5.2.1) to obtain  $\bar{x}^{(\bar{T})}$  for the clique  $\bar{T} = S \cup R$ , where  $S = \bar{T}$  and  $R = \{v_{T,H} \mid H \text{ intersects } T\}$ .
- (6) Use Lemma (5.2.1) to obtain  $\bar{x}^{(\bar{V})}$  for clique  $\bar{V} = S_{\bar{V}} \cup R_{\bar{V}}$  where  $S_{\bar{V}} = V(K_n) \setminus V(C)$  and  $R_{\bar{V}} = \{w_T \mid T \in T_P(C)\}$ .

Now let  $\bar{x}$  be defined by

$$\bar{x}_i = \begin{cases} 1 & \text{for all edges } i \text{ in the hamilton paths of (1), (2)} \\ & \text{or (3)} \\ \bar{x}_i^{(\bar{H})} & \text{for each edge } i \in \gamma(\bar{H}), H \in H(C) \\ \bar{x}_i^{(\bar{T})} & \text{for each edge } i \in \gamma(\bar{T}), T \in T_{NP}(C), t_T \geq 3 \\ \bar{x}_i^{(\bar{V})} & \text{for each edge } i \in \gamma(\bar{V}) \\ 0 & \text{otherwise.} \end{cases}$$

It can easily be checked that  $\bar{x}$  satisfies degree constraints, non-negativity constraints, and complementary slackness conditions. It remains to be shown that  $\bar{x}$  satisfies the subtour elimination constraints, i.e.  $x(\delta(S)) \geq 2$  for all  $S \subset V(K_n)$  such that  $2 \leq |S| \leq n - 2$ . We will consider three cases.

Case 1.  $S \subseteq \bar{H}$ ,  $S \subseteq \bar{T}$ , or  $S \subseteq \bar{V}$ .

For such  $S$  it follows by Lemma (5.2.1) and the construction of  $\bar{x}$  that  $\bar{x}(\delta(S)) \geq 2$ .

Case 2.  $S \subseteq V(K_n) \setminus \bar{V}$ .

Clearly for any maximal clique  $K$  in  $C$  and any cut  $\delta(M)$  of  $\langle K \rangle$  for nonempty  $M, K$ , we have  $\bar{x}(\delta(M)) \geq 1$ . If  $S$  satisfies the conditions of Case 1, we are done. Otherwise,  $\delta(S)$  must contain at least two cuts of the form  $\delta(M)$  as described above and hence  $\bar{x}(\delta(S)) \geq 2$ .

Case 3.  $S \subseteq V(K_n)$  satisfies  $S \cap V \neq \emptyset$ ,  $S \cap (V(K_n) \setminus \bar{V}) \neq \emptyset$ .

Again we have  $\bar{x}(\delta(M)) \geq 1$  for any cut  $\delta(M)$  of  $\langle \bar{V} \rangle$  for nonempty  $M \subseteq \bar{V}$ . Since  $\delta(S)$  must contain at least one cut of  $\langle \bar{V} \rangle$  and one cut of  $\langle K \rangle$  for a maximal clique in  $C$ , it follows that  $\bar{x}(\delta(S)) \geq 2$ .

In all cases,  $\bar{x}(\delta(S)) \geq 2$ . Thus  $\bar{x}$  and  $(\bar{\lambda}, \bar{\gamma})$  are optimal solutions for  $P$  and  $D$  respectively. It now follows that

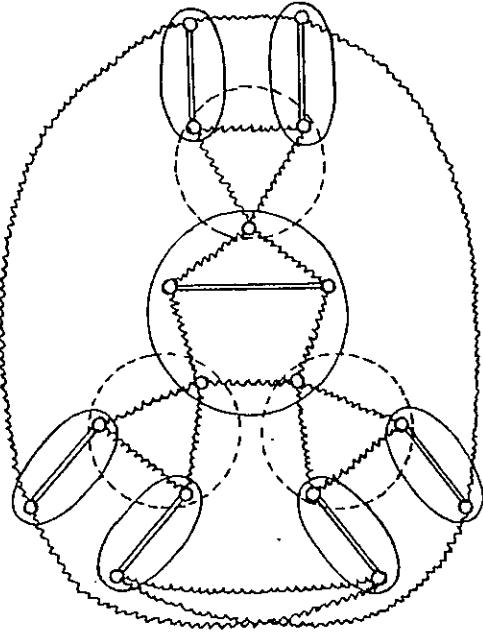
$$\begin{aligned} \text{GAP}(C) &= a_C \bar{x} - \alpha_C \\ &= \bar{\lambda} 2 + \bar{\gamma} b - \alpha_C \\ &= \sum(|H|: H \in \mathcal{H}(C)) + \sum(|\bar{T}|: T \in \mathcal{T}_{NP}(C)) + \sum(|T| - \frac{3}{2}: T \in \mathcal{T}_P(C)) \\ &\quad + \sum(\sum(|T \cap H| - 1: H \in \mathcal{H}(C), H \cap T \neq \emptyset) : T \in \mathcal{T}_{NP}(C)) - \alpha_C \\ &= \sum(|H|: H \in \mathcal{H}(C)) + \sum(|T| - t_T: T \in \mathcal{T}(C)) - \frac{1}{2}|\mathcal{T}_P(C)| - \alpha_C \\ &= \frac{|\mathcal{T}_{NP}(C)| + 1}{2} \quad \square \end{aligned}$$


In general, for any clique tree  $C$  there exists a vertex of  $Q_S^n$  which maximizes  $a_C x$  over  $Q_S^n$ . Theorem (5.2.2) constructs  $\bar{x} \in Q_S^n$  which maximizes  $a_C x$  over  $Q_S^n$ , however this  $\bar{x}$  is not necessarily a vertex of  $Q_S^n$ . For example, consider the vector  $\bar{x}$  corresponding to Figure 5.2.3. Then  $\bar{x}$  is of the form described in Theorem (5.2.2). However,


$$\bar{x} = \frac{1}{2} \bar{x} + \frac{1}{2} \bar{\bar{x}}$$

where  $\bar{x}$  and  $\bar{\bar{x}}$  are the vectors corresponding to Figure 5.2.4 and 5.2.5 respectively. Since  $\bar{x}, \bar{\bar{x}} \in Q_S^n$  we have  $\bar{x}$  is a convex combination of other points in  $Q_S^n$  and thus cannot be a vertex.

It is possible to modify the construction used in Theorem (5.2.2) so that vector  $\bar{x}$  constructed is a vertex of  $Q_S^n$ . Notice that  $\bar{x}$  is half-integer, i.e.  $\bar{x}_i \in \{0, \frac{1}{2}, 1\}$ . Hence the edges of the



 corresponds to  $\bar{x}_e = \frac{1}{2}$

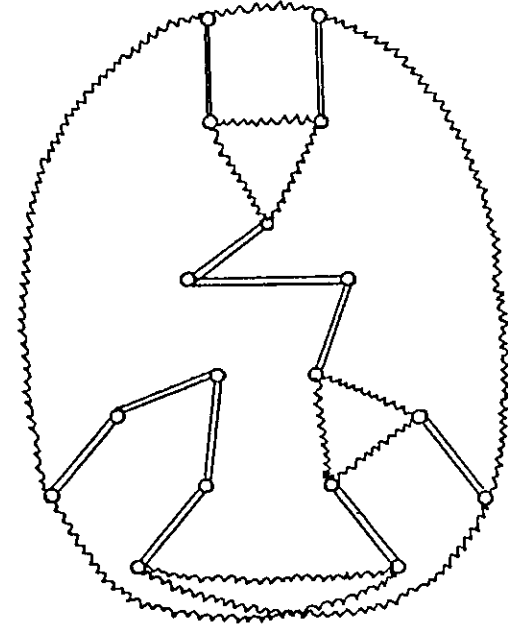
 corresponds to  $\bar{x}_e = 1$


- edges with  $\bar{x}_e = 0$  are not shown


- corresponds to teeth

- corresponds to handles

Figure 5.2.3

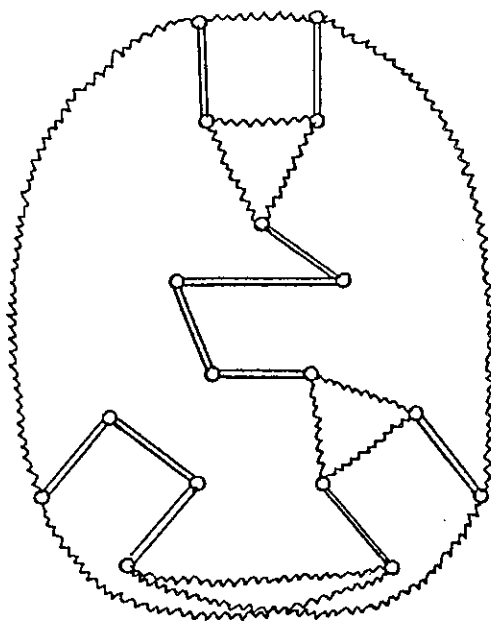


 corresponds to  $\bar{x}_e = \frac{1}{2}$

 corresponds to  $\bar{x}_e = 1$

- edges with  $\bar{x}_e = 0$  are not shown.

Figure 5.2.4



~~~~~ corresponds to  $\bar{x}_e = \frac{1}{2}$

—•— corresponds to  $\bar{x}_e = 1$

- edges with  $\bar{x}_e = 0$  are not shown.

Figure 5.2.5

support of  $\bar{x}$  for which  $\bar{x}_i = \frac{1}{2}$  partition into edge disjoint cycles which we will call 1/2-cycles in  $\bar{x}$ .

We have the following theorem.

(5.2.3) Theorem. Let  $\bar{x}$  be constructed as in Theorem (5.2.2). Then if the support of  $\bar{x}$  is planar,  $\bar{x}$  is a vertex of  $Q_S^n$ .

Proof. We show that such an  $\bar{x}$  is the unique point of  $Q_S^n$  satisfying a set of subtour elimination constraints and non-negativity constraints with equality, and hence is a vertex of  $Q_S^n$ .

Let  $B = \{e \in E(K_n) \mid \bar{x}_e = 0\}$  and  $D = \{S \subseteq V(K_n) \mid \bar{x}(\delta(S)) = 2, 2 \leq |S| \leq n-2\}$  and let  $F = \{x \in Q_S^n \mid x_e = 0 \text{ for all } e \in B, x(\delta(S)) = 2 \text{ for all } S \in D\}$ . Suppose  $\bar{x} \in F$ . It trivially follows that  $\bar{x}_e = \bar{x}_e$  for all  $e$  such that  $\bar{x}_e = 0$  or  $\bar{x}_e = 1$ . We wish to show  $\bar{x}_e = \frac{1}{2}$  whenever  $\bar{x}_e = \frac{1}{2}$ . Given any clique tree  $C$  and  $T \in T(C)$ ,  $H \in H(C)$ , we adopt the notation of Theorem (5.2.2) for  $w_T, \bar{V}, \bar{H}, S_{\bar{V}}$  and  $R_{\bar{V}}$  and given any clique  $K$  we adopt the notation of Lemma (5.2.1) for  $\bar{x}^{(K)}$ .

Claim 1. For the 1/2-cycle  $J$  in  $\bar{x}^{(\bar{V})}$ ,  $\bar{x}_e = \frac{1}{2}$  for all  $e \in E(J)$ .

Proof of Claim 1. For all  $v \in V(J)$  there exists an edge  $e = (v, w)$  such that  $\bar{x}_e = 1$ . Hence for degree constraints to be satisfied we must have  $\bar{x}_{e_1} + \bar{x}_{e_2} = 1$  for any pair  $e_1, e_2$  of adjacent edges in  $J$ .

Claim 1 now follows if  $J$  is an odd cycle, which is the case if  $C$  is a comb. Otherwise, due to the tree structure of clique tree  $C$ , there exists a handle  $H$  of  $C$  which has exactly one non-pendent tooth  $T$  which intersects it. Let  $T_H = \{T_1, T_2, \dots, T_{2k}\}$  be the pendent teeth intersecting  $H$ .

Consider  $S_1 \subseteq V(K_n)$  where

$$S_1 = T_H \cup \bar{H} \cup Y \cup Z$$

$$\text{and } Y = \begin{cases} S_{\bar{V}} & \text{if } w_T = r_2 \text{ for some } T \in T_H, \text{ and } |S_{\bar{V}}| \text{ even} \\ S_{\bar{V}} \setminus \{s_{|S_{\bar{V}}|}\} & \text{if } w_T = r_2 \text{ for some } T \in T_H \text{ and } |S_{\bar{V}}| \text{ odd} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\text{and } Z = \begin{cases} \{s_{|S_{\bar{V}}|}\} & \text{if } w_T = r_{|\bar{R}_{\bar{V}}|} \text{ for some } T \in T_H \text{ and } |S_{\bar{V}}| \text{ odd} \\ \emptyset & \text{otherwise} \end{cases}$$

(see Figure 5.2.6).

Let  $\delta(S_1) \cap E(J) = \{e_1, e_2\}$  where  $\bar{x}_{e_1} = \bar{x}_{e_2}$ . Furthermore  $S_1 \in \mathcal{D}$ . Thus if  $|H \cap T| > 1$  or  $t_T = 2$  then

$$2 = \bar{x}(\delta(S_1)) = \bar{x}_{e_1} + \bar{x}_{e_2} + 1$$

which implies  $\bar{x}_{e_1} = \frac{1}{2}$  and Claim 1 holds. Otherwise  $H \cap T = \{v\}$  where  $v$  is the common node of two 1/2-cycles in  $\bar{x}$ . Let  $S_2 = S_1 \setminus \{v\}$ . Noting that  $S_2 \in \mathcal{D}$  we have

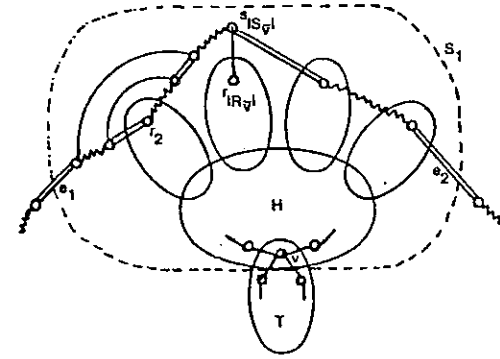


Figure 5.2.6

$$2 = \bar{x}(\delta(S_1)) + \bar{x}(\delta(S_2)) - \bar{x}(\delta(v)) = 4\bar{x}_{e_1}$$

which implies  $\bar{x}_{e_1} = \frac{1}{2}$ , and again Claim 1 holds.

Using Claim 1 we can show

Claim 2. For any edges  $e_1 = (u,v)$  and  $e_2 = (v,w)$  adjacent in some 1/2-cycle in  $\bar{x}$ ,  $\bar{x}_{e_1} + \bar{x}_{e_2} = 1$ .

A Simplex Method for the Subtour Problem

Proof of Claim 2. The only situation in which Claim 2 does not trivially hold is when  $v$  is a common node of two 1/2-cycles in  $\bar{x}$ . In this case  $\{v\}$  is an articulation set for  $C$ . Hence, using the planarity of the support of  $\bar{x}$ , we can find  $S \in D$  such that  $\delta(S) = \{e_1, e_2, x, y\}$ ,  $x, y \in E(J)$ . Thus, using Claim 1,  $\bar{x}_{e_1} + \bar{x}_{e_2} = 1$ .

It follows from Claim 2 that  $\bar{x}_e = \frac{1}{2}$  for all odd 1/2-cycles in  $\bar{x}$ . Note that all 1/2-cycles for handles of  $C$  are odd, but we can have even 1/2-cycles occurring in non-pendent teeth  $T$  for which  $t_T \geq 3$ .

Let  $D$  be the 1/2-cycle in such a tooth  $T$ . Using methods similar to those used in Claim 1 it is possible to find  $S \in D$  such that  $\delta(S) = \{e_1, e_2, x, y\}$  where  $x, y \in J$ ,  $e_1, e_2 \in D$  and  $\bar{x}_{e_1} = \bar{x}_{e_2}$ . Hence

$$2 = \bar{x}(\delta(S)) = 2\bar{x}_{e_1} + 1$$

and  $\bar{x}_e = \frac{1}{2}$ .  $\square$

It is always possible to construct  $\bar{x}$  as in Theorem (5.2.2) so that the support of  $\bar{x}$  is planar by forming the support in a planar fashion one clique at a time. Thus it follows from Theorem (5.2.3) that we can always perform the construction used in Theorem (5.2.2) so as to obtain a vertex of  $Q_S^n$ .

Since  $Q_T^n \subseteq Q_S^n \subseteq Q_F^n$ , the subtour problem (5.0.1) is related to the fractional 2-factor problem (4.0.1) and the TSP (3.1.1) in the following way:

$$\min\{cx \mid x \in Q_F^n\} \leq \min\{cx \mid x \in Q_S^n\} \leq \min\{cx \mid x \in Q_T^n\}.$$

Thus optimizing over  $Q_S^n$  provides a lower bound on the optimal value of the related TSP. Furthermore, if the optimal solution for the subtour problem is integer, it is an optimal tour for the related TSP. The lower bound provided by the subtour problem is, in general, a substantially better lower bound than that provided by the fractional 2-factor problem, but is more difficult to obtain.

The separation problem can be solved in polynomial time for  $Q_S^n$  (see Chapter 3), therefore we can optimize over  $Q_S^n$  in polynomial time by means of the ellipsoid algorithm (see [Grötschel, Lovász and Schrijver, 1981] for more details). However, there does not yet exist a direct, i.e. nonellipsoidal polynomial time algorithm to solve the problem of optimizing over  $Q_S^n$ . Also note that, unlike  $Q_F^n$ , the number of constraints in the linear system for  $Q_S^n$  is exponential in the size of the problem. Consequently, no standard implementation of the simplex method can be used to solve the subtour problem.

In this chapter we describe a finite dual simplex method for the following linear program:



$$\begin{array}{ll}
 \text{minimize} & cx \\
 \text{subject to} & Ax = b, Dx \geq d \\
 & 0 \leq x \leq u
 \end{array} \tag{6.0.1}$$

where  $A \in \mathbb{R}^{V \times E}$  denotes the node-edge incidence matrix of  $K_n = (V, E)$ ,  $D \in \mathbb{R}^{S \times E}$  denotes the cut-edge incidence matrix of the cuts  $\delta(S)$  defined by the members  $S$  of a family  $\mathcal{S}$  of subsets of  $V$ ,  $c \in \mathbb{R}^E$ , and  $d \in \mathbb{R}^S$ ,  $u \in \mathbb{R}^E$  and  $b \in \mathbb{R}^V$  satisfy  $b \geq 0$ ,  $d \geq 0$ , and  $u > 0$  (see Section 1.6 for a description of the revised dual simplex method). At each iteration of the method, the basis is broken into blocks, which is a common practice in combinatorial simplex methods (for instance see the generalized upper bound simplex method described in [Chvátal, 1983]). Finiteness of the method is ensured by applying dual lexicography in a form which is considerably simplified computationally for this problem.

For the remainder of this chapter, let  $A \in \mathbb{R}^{V \times E}$ ,  $D \in \mathbb{R}^{S \times E}$ ,  $c \in \mathbb{R}^E$ ,  $d \in \mathbb{R}^S$ ,  $b \in \mathbb{R}^V$  and  $u \in \mathbb{R}^E$  be defined as above.

Note that the subtour problem (5.0.1) is a special case of (6.0.1) for which  $b = 2$ ,  $d = 2$ ,  $u = 1$ , and  $\mathcal{S} = \{S \subseteq V \mid 3 \leq |S| \leq n - 3\}$ .

Another special case of (6.0.1) is the perfect b-matching problem for which  $u = +\infty$ ,  $d = 1$ , and  $\mathcal{S} = \{S \subseteq V \mid b(S) \text{ is odd}\}$ . Thus it is also a problem which, in general, has an exponential number of constraints. This problem can be solved in polynomial time by applying scaling techniques described in [Marsh, 1978] to the primal-dual algorithm developed by J. Edmonds (see [Pulleyblank, 1973]). Also, in [Koch, 1979] a primal simplex method for the perfect b-matching problem is described in which all the bases encountered have a special structure

which allows easy computations in the corresponding pivots.

Unfortunately, the method has no anti-cycling pivot rule to ensure finiteness. Moreover, the specially-structured bases do not generalize to the subtour problem (see Section 6.4).

The dual simplex method described here for (6.0.1) has two major drawbacks in the general case. First of all,  $|\mathcal{S}|$  may be exponential in  $n$ , as is often the case when  $\mathcal{S}$  is described implicitly, like " $\mathcal{S}$  is the set of all cuts". Thus checking the primal feasibility of some basic solution for the cut constraints of (6.0.1) may, in principle, require an exponential number of steps. However, for both the subtour problem and the perfect b-matching problem there exists a separation routine which, in polynomial time, can find an infeasible cut constraint for a given basic solution if there exists one (see Sections 6.3 and 6.4). Thus each iteration of the dual simplex method presented here in general requires an exponential number of steps, but can be performed in a polynomial number of steps in the case of the subtour and b-matching problems. Note that at present no technique for performing primal simplex iterations in a polynomial number of steps is known for the subtour problem.

The second drawback is that this dual simplex method is not completely combinatorial. At any iteration it may be necessary to solve a linear system with  $k$  equations and  $k$  unknowns where, in general,  $k \leq |E| - |V|$ . Note that this system is small compared to the original system of  $|V| + |E| + |\mathcal{S}|$  equations and unknowns, yet can be large

enough to prove difficult to solve computationally. However, it is possible to ensure  $k \leq |V| - 5$  for the subtour and b-matching problems by keeping the sets corresponding to active cut constraints nested (see Sections 6.3 and 6.4).

Finally, note that any linear program having the same form as (6.0.1) but with general lower bounds  $x \leq u$  for some  $u \in \mathbb{R}^E$ ,  $l \leq x$ , can be transformed into an instance of (6.0.1) using a transformation similar to the one described in the introduction of Chapter 4. Thus the method we describe for solving (6.0.1) can also be used on such problems.

#### 6.1 A DUAL SIMPLEX METHOD FOR LINEAR PROGRAM (6.0.1)

Recall that we wish to solve the linear program

$$\begin{aligned} \text{minimize} \quad & cx & (6.0.1) \\ \text{subject to} \quad & Ax = b \\ & Dx \geq d \\ & 0 \leq x \leq u \end{aligned}$$

This linear program can be written equivalently as

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e & (6.1.1) \\ \text{subject to} \quad & x(\delta(v)) = b_v & \text{for all } v \in V, \\ & x(\delta(S)) \geq d_S & \text{for all } S \in \mathcal{S} \\ & -x_e \geq -u_e & \text{for all } e \in E, \\ & x_e \geq 0 & \text{for all } e \in E. \end{aligned}$$

The dual linear program is

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} y_v b_v - \sum_{e \in E} \mu_e u_e + \sum_{S \in \mathcal{S}} \delta_S d_S & (6.1.2) \\ \text{subject to} \quad & y_u + y_v - \mu_{uv} + \sum(\delta_S | S \in \mathcal{S}, uv \in \delta(S)) \leq c_{uv} \\ & \text{for all } uv \in E, \\ & \mu_e \geq 0 & \text{for all } e \in E, \\ & \delta_S \geq 0 & \text{for all } S \in \mathcal{S} \end{aligned}$$

We refer to the three types of constraints in (6.1.2) as the dual edge constraints, and the non-negativity constraints for  $\mu$  and  $\delta$  respectively.

In order to develop the dual simplex method for (6.1.1) we must do the following:

- i) determine what a basis for the linear program looks like,
- ii) find a method for obtaining the corresponding basic primal solution,
- iii) find a method for obtaining the corresponding basic dual solution,
- iv) describe how to perform a dual simplex pivot, and
- v) describe how to find an initial dual feasible basis.

In addition to the above, we must ensure the sequence of pivots performed is finite. This is discussed in Section 6.2.

#### i) The Bases

By introducing slack variables  $s_e$  for every edge  $e \in E$  and  $\gamma_S$  for every  $S \in \mathcal{S}$ , we can rewrite the linear program (6.1.1) in equality form as follows:

$$\begin{aligned}
 &\text{minimize} && \sum_{e \in E} c_e x_e && (6.1.3) \\
 &\text{subject to} && x(\delta(v)) = b_v \text{ for all } v \in V, \\
 &&& -x_e - s_e = -u_e \text{ for all } e \in E, \\
 &&& x(\delta(S)) - \gamma_S = d_S \text{ for all } S \in \mathcal{S}, \\
 &&& x_e, s_e \geq 0 \text{ for all } e \in E, \\
 &&& \gamma_S \geq 0 \text{ for all } S \in \mathcal{S}.
 \end{aligned}$$

The coefficient matrix and right-hand side for (6.1.3) have the form shown in Figure (6.1.1).

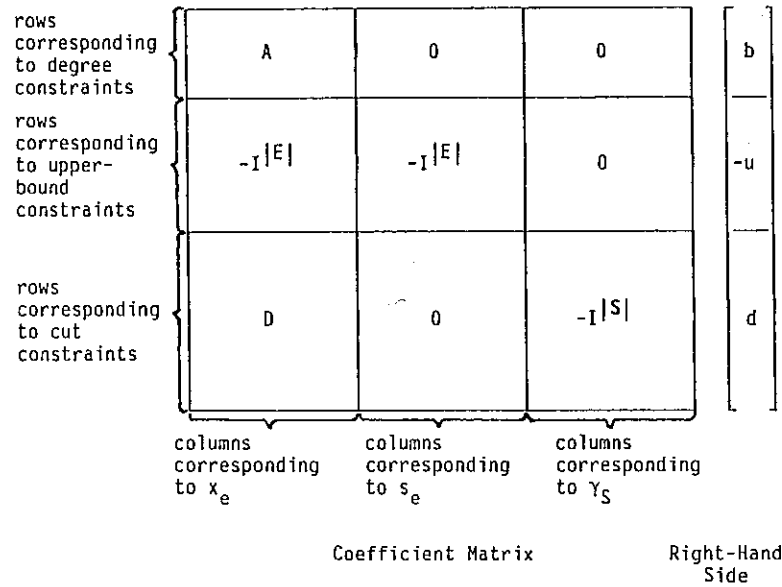


Figure 6.1.1

It follows from (4.1.1) that matrix A has rank  $|V|$ , thus the coefficient matrix above has rank  $|V| + |E| + |S|$ . Consequently a basis corresponds to specifying  $|V| + |E| + |S|$  of the variables  $s_e, x_e, e \in E$ , and  $\gamma_S, S \in \mathcal{S}$  to be basic. As in the case of the fractional 2-factor problem, there are the following possibilities for each  $e \in E$ :

- (a) both  $x_e$  and  $s_e$  are basic;
- (b)  $x_e$  is basic,  $s_e$  is nonbasic;
- (c)  $x_e$  is nonbasic,  $s_e$  is basic;
- (d) both  $x_e$  and  $s_e$  are nonbasic.

In the corresponding basic solution we require  $x_e = s_e = 0$  for edges of type (d), implying  $u_e = 0$  for these edges. Since we assume  $u > 0$ , it follows that possibilities (a), (b), and (c) partition the edges of  $K_n$  into three sets, which we denote by J, B, and Z respectively. Edges  $e$  in B must have  $x_e = u_e, s_e = 0$  in the corresponding basic solution, i.e., must have their values at upper bound. Edges  $e$  in Z must have  $x_e = 0, s_e = u_e$ .

Let  $K \subseteq \mathcal{S}$  be the set of subsets in  $\mathcal{S}$  such that the corresponding slack variables are nonbasic, i.e. such that  $\gamma_{S \setminus K}$  are basic. Since there are  $|V| + |E| + |S|$  basic variables in total, it follows that  $2|J| + |B| + |Z| + |S| - |K| = |V| + |E| + |S|$ . Thus  $|J| = |V| + |K|$ , since  $|J| + |B| + |Z| = |E|$ . Furthermore, the columns corresponding to  $x_e, s_e$  for  $e \in J$  and  $\gamma_S$  for  $S \in S \setminus K$  must be linearly independent. This is true if and only if the matrix

$$\begin{array}{|c|} \hline A_J \\ \hline \bar{D}_J \\ \hline \end{array} \quad (6.1.4)$$

is nonsingular, where  $\bar{D} \in \mathbb{R}^{K \times E}$  is the matrix composed of the rows of  $D$  corresponding to cut constraints for  $K$ . Thus  $J$  can be partitioned into edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that  $T$  is the edge set of an odd 1-forest in graph  $K_n$ .

In summary, selecting a basis for the linear program (6.1.3) is equivalent to specifying a quadruple  $(K, J, B, Z)$  which satisfies the following:

- (a)  $K \subseteq S$ ;
- (b)  $\{J, B, Z\}$  is a partition of  $E$ ;
- (c) The matrix of (6.1.4) is nonsingular. Moreover,  $J$  can be partitioned into edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that the graph  $G' = (V, T)$  is an odd 1-forest.

The set of basic variables will then consist of

$$\{y_S \mid S \in S \setminus K\} \cup \{x_e, s_e \mid e \in J\} \cup \{x_e \mid e \in B\} \cup \{s_e \mid e \in Z\}.$$

In a basis defined by  $(K, J, B, Z)$ ,  $K$  represents the set of tight cuts (i.e.  $x(\delta(S)) = d_S$  for all  $S \in K$ ),  $B$  represents the edges  $e$  for which  $x_e$  is at its upper bound (i.e.  $x_B = u_B$ ), and  $Z$  represents the edges  $e$  for which  $x_e$  is at its lower bound of zero (i.e.  $x_Z = 0$ ).

It is useful to note that for any basis  $(K, J, B, Z)$  of (6.1.3) we must have  $|K| \leq |E| - |V|$  since  $|K| = |J| - |V|$  and  $J \subseteq E$ . In other words, the number of "tight cuts" required to define a basic solution will never be greater than  $|E| - |V|$ .

#### ii) Finding the Corresponding Basic Primal Solution.

For the remainder of the section let  $\bar{D} \in \mathbb{R}^{K \times E}$  denote the cut-edge incidence matrix for cuts  $\delta(S)$ ,  $S \in K$ , and let  $\bar{D} \in \mathbb{R}^{S \setminus K \times E}$  denote the cut-edge incidence matrix for cuts  $\delta(S)$ ,  $S \in S \setminus K$ .

Let the quadruple  $(K, J, B, Z)$  define a basis for the linear program (6.1.3). In the corresponding basic primal solution  $(\bar{x}, \bar{s}, \bar{y})$ ,  $\bar{x} \in \mathbb{R}^E$ ,  $\bar{s} \in \mathbb{R}^E$ ,  $\bar{y} \in \mathbb{R}^S$ , we require the nonbasic variables  $\bar{x}_Z$ ,  $\bar{s}_B$ , and  $\bar{y}_K$  to equal zero, and the basic variables  $\bar{x}_J$ ,  $\bar{x}_B$ ,  $\bar{s}_J$ ,  $\bar{s}_Z$ ,  $\bar{y}_{S \setminus K}$  to satisfy the following:

(6.1.5)

|                                           |             |             |         |         |               |       |     |                          |                     |
|-------------------------------------------|-------------|-------------|---------|---------|---------------|-------|-----|--------------------------|---------------------|
| degree constraints                        | $A_J$       | $A_B$       | $0$     | $0$     | $0$           | $x_J$ | $b$ |                          |                     |
| cut constraints for $S \in K$             | $\bar{D}_J$ | $\bar{D}_B$ | $0$     | $0$     | $0$           |       |     | $x_B$                    | $d_K$               |
| upper-bound constraints for $e \in B$     | $0$         | $-I B $     | $0$     | $0$     | $0$           |       |     |                          |                     |
| upper-bound constraints for $e \in J$     | $-I J $     | $0$         | $-I J $ | $0$     | $0$           |       |     | $s_J$                    | $-u$                |
| upper-bound constraints for $e \in Z$     | $0$         | $0$         | $0$     | $-I Z $ | $0$           |       |     |                          |                     |
| cut constraints for $S \in S \setminus K$ | $\bar{D}_J$ | $\bar{D}_B$ | $0$     | $0$     | $-I S  -  K $ |       |     | $\gamma_{S \setminus K}$ | $d_{S \setminus K}$ |

columns corresponding to  $x_J$       columns corresponding to  $x_B$       columns corresponding to  $s_J$       columns corresponding to  $s_Z$       columns corresponding to  $\gamma_{S \setminus K}$

Thus  $(\bar{x}_J, \bar{x}_B, \bar{s}_J, \bar{s}_Z, \bar{\gamma}_{S \setminus K})$  must satisfy

$$\begin{aligned} \bar{x}_J(\delta(v) \cap J) + \bar{x}_B(\delta(v) \cap B) &= b_v \text{ for all } v \in V, \\ \bar{x}_J(\delta(S) \cap J) + \bar{x}_B(\delta(S) \cap B) &= d_S \text{ for all } S \in K, \\ \bar{x}_e &= u_e \text{ for all } e \in B, \\ \bar{x}_e + \bar{s}_e &= u_e \text{ for all } e \in J, \\ \bar{s}_e &= u_e \text{ for all } e \in Z, \\ \bar{x}_J(\delta(S) \cap J) + \bar{x}_B(\delta(S) \cap B) - \bar{\gamma}_S &= d_S \text{ for all } S \in S \setminus K. \end{aligned}$$

As with the fractional 2-factor problem, we solve (6.1.5) by decomposing the basis into blocks, then solving for the basic variables a block at a time. In this way we are able to take advantage of the combinatorial structure of the basis  $(K, J, B, Z)$ .

From (6.1.5) we easily obtain  $\bar{s}_Z = u_Z$  and  $\bar{x}_B = u_B$ . Then letting  $b' \in \mathbb{R}^V$  be defined by  $b'_v = b_v - u_B(\delta(v) \cap B)$  for all  $v \in V$  and letting  $d' \in \mathbb{R}^S$  be defined by  $d'_S = d_S - u_B(\delta(S) \cap B)$  for all  $S \in S$ , it follows that  $\bar{x}_J$  is the unique solution to

$$\begin{bmatrix} A_J \\ \bar{D}_J \end{bmatrix} \begin{bmatrix} x_J \\ \gamma_{S \setminus K} \end{bmatrix} = \begin{bmatrix} b' \\ d'_K \end{bmatrix} \tag{6.1.6}$$

We solve for  $\bar{x}_J$  by further decomposing the basis. Partition  $J$  into edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that the graph  $G^* = (V, T)$  is an odd 1-forest. Then we solve

$$\begin{bmatrix} A_T & A_L \\ \bar{D}_T & \bar{D}_L \end{bmatrix} \begin{bmatrix} x_T \\ x_L \end{bmatrix} = \begin{bmatrix} b' \\ d'_K \end{bmatrix}$$

or equivalently

$$\begin{bmatrix} I|T| & A_T^{-1}A_L \\ \bar{D}_T & \bar{D}_L \end{bmatrix} \begin{bmatrix} x_T \\ x_L \end{bmatrix} = \begin{bmatrix} A_T^{-1}b' \\ d'_K \end{bmatrix} \tag{6.1.7}$$

From the top  $|V|$  equations of (6.1.7) we obtain

$$x_T = A_T^{-1}b' - A_T^{-1}A_Lx_L. \quad (6.1.8)$$

Substituting (6.1.8) into the bottom  $L$  equations gives

$$(\bar{D}_L - \bar{D}_T(A_T^{-1}A_L))x_L = d'_K - \bar{D}_T A_T^{-1}b'.$$

Letting  $B = \bar{D}_L - \bar{D}_T(A_T^{-1}A_L)$  and  $p = d'_K - \bar{D}_T A_T^{-1}b'$ , we then find  $\bar{x}_L$  by solving  $Bx_L = p$ . Then we use  $\bar{x}_L$  in (6.1.8) in order to obtain  $\bar{x}_T$ . Finally, from  $\bar{x}_J = (\bar{x}_T, \bar{x}_L)$  we can calculate  $\bar{s}_J$  and  $\bar{y}_{S \setminus K}$ .

In order to solve  $Bx_L = p$  we must find  $B$  and  $p$ . We obtain  $w = A_T^{-1}b'$  by solving  $A_T w = b'$  using the odd 1-forest structure of  $T$  and Algorithm (4.1.2). Then  $p \in \mathbb{R}^K$  is easily obtained by calculating  $p_S = d'_S - w(\delta(S) \cap T)$  for all  $S \in K$ .

To obtain matrix  $B \in \mathbb{R}^{K \times L}$  we must calculate each of the entries in  $\bar{D}_T(A_T^{-1}A_L)$ . Each column  $k^e = A_T^{-1}A_e$ ,  $e \in L$ , of  $A_T^{-1}A_L$  can be calculated using Algorithm (4.2.10) and the even cycle or dumbbell formed when edge  $e$  is added to the odd 1-forest  $G' = (V, T)$ . Then for each  $e \in L$ , the entry of  $\bar{D}_T k^e$  corresponding to  $S \in K$  is simply  $k^{(e)}(\delta(S) \cap T)$ .

Once we have found  $\bar{x}_L$  by solving  $Bx_L = p$ , we find  $\bar{x}_T$  by solving  $A_T x_T = f$ , where  $f \in \mathbb{R}^V$  is defined by  $f_v = b'_v - x_L(\delta(v) \cap L)$  for all  $v \in V$ . This can be done using Algorithm (4.1.2) and the odd 1-forest  $G' = (V, T)$ .

In summary, we calculate the basic primal solution corresponding to the basis  $(K, J, B, Z)$  as follows.

(6.1.9) Algorithm to calculate primal solution  $(\bar{x}, \bar{s}, \bar{y})$  for basis  $(K, J, B, Z)$

- (1) Let  $\bar{x}_B = u_B$  and  $\bar{x}_L = 0$ .
- (2) Partition  $J$  into edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that  $G' = (V, T)$  forms an odd 1-forest.
- (3) Let  $b' \in \mathbb{R}^V$  be defined by  $b'_v = b_v - u_B(\delta(v) \cap B)$  for all  $v \in V$  and let  $w \in \mathbb{R}^T$  be the unique solution to  $A_T w = b'$  found by using Algorithm (4.1.2) and the odd 1-forest  $G' = (V, T)$ . Then define  $p \in \mathbb{R}^K$  by  $p_S = d'_S - u_B(\delta(S) \cap B) - w(\delta(S) \cap T)$  for all  $S \in K$ .
- (4) For every  $e \in L$  calculate  $k^e = A_T^{-1}A_e$  by using the circuit formed when  $e$  is added to  $T$  and Algorithm (4.2.10).
- (5) For all  $S \in K$  and  $e \in L$ , define each entry  $a_{S,e}$  of matrix  $B \in \mathbb{R}^{K \times L}$  by  $a_{S,e} = \theta - k^e(\delta(S) \cap T)$ , where
 
$$\theta = \begin{cases} 1 & \text{if } e \in \delta(S) \\ 0 & \text{otherwise.} \end{cases}$$
- (6) Calculate the unique solution  $\bar{x}_L$  to  $Bx_L = p$ .
- (7) Define  $f \in \mathbb{R}^V$  by  $f_v = b'_v - u_B(\delta(v) \cap B) - \bar{x}_L(\delta(v) \cap L)$  for all  $v \in V$ . Then calculate the unique solution  $\bar{x}_T$  to  $A_T x_T = f$  using the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.2).

(8) Let  $\bar{s}_J = u_J - \bar{x}_J$ ,  $\bar{s}_B = 0$ ,  $\bar{s}_Z = u_Z$ ,  $\bar{y}_K = 0$ , and let  $\bar{y}_S = d_S - u_B(\delta(S) \cap B) - \bar{x}_J(\delta(S) \cap J)$  for all  $S \in S \setminus K$ .

Note that in the above algorithm, the values of all basic values corresponding to a basis  $(K, J, B, Z)$  are found combinatorially using the structure of the basis with the exception of  $\bar{x}_L$ , determined in step (6). We calculate  $\bar{x}_L$  by solving the system  $Bx_L = p$  which has  $|K|$  equations and  $|K|$  unknowns. In general, it is possible for  $|K|$  to be as large as  $|E| - |V|$ . However, for some specific instances of linear program (6.1.3), and in particular for the subtour problem, it is possible to restrict the size of  $K$  to be at most  $|V| - 5$  by ensuring  $K$  is a nested family (see Section 6.3).

Also note that, in general, the size of  $S$  may be exponential in terms of the size of  $V$ , and thus calculating  $\bar{y}_{S \setminus K}$  requires an exponential number of calculations. However, in practice it is not generally necessary to calculate  $\bar{y}_{S \setminus K}$ .

iii) Finding the Corresponding Basic Dual Solution.

Given the basis of (6.1.3) defined by  $(K, J, B, Z)$ , the corresponding basic dual solution  $(\bar{y}, \bar{\mu}, \bar{\delta})$ ,  $\bar{y} \in \mathbb{R}^V$ ,  $\bar{\mu} \in \mathbb{R}^E$ ,  $\bar{\delta} \in \mathbb{R}^S$ , must satisfy

(6.1.10)

|                                                               |         |               |            |            |            |                  |                          |       |
|---------------------------------------------------------------|---------|---------------|------------|------------|------------|------------------|--------------------------|-------|
| tight dual constraints for edges $e \in J$                    | $A_J^t$ | $\bar{D}_J^t$ | 0          | $-I^{ J }$ | 0          | $\bar{D}_J^t$    | $y$                      | $c_J$ |
| tight dual constraints for edges $e \in B$                    | $A_B^t$ | $\bar{D}_B^t$ | $-I^{ B }$ | 0          | 0          | $\bar{D}_B^t$    | $\delta_K$               | $c_B$ |
| tight non-negativity constraints for $\mu_J$                  | 0       | 0             | 0          | $-I^{ J }$ | 0          | 0                | $\mu_J$                  | = 0   |
| tight non-negativity constraints for $\mu_Z$                  | 0       | 0             | 0          | 0          | $-I^{ Z }$ | 0                | $\mu_Z$                  | 0     |
| tight non-negativity constraints for $\delta_{S \setminus K}$ | 0       | 0             | 0          | 0          | 0          | $-I^{ S  -  K }$ | $\delta_{S \setminus K}$ | 0     |

columns corresponding to  $y$       columns corresponding to  $\delta_K$       columns corresponding to  $\mu_B$       columns corresponding to  $\mu_J$       columns corresponding to  $\mu_Z$       columns corresponding to  $\delta_{S \setminus K}$

Thus we require  $(\bar{y}, \bar{\mu}, \bar{\delta})$  to satisfy

$$\bar{y}_u + \bar{y}_v - \bar{\mu}_{uv} + \sum (\bar{\delta}_S | S \in K, uv \in \delta(S)) = c_{uv} \text{ for all } uv \in J \cup B,$$

$$\bar{\mu}_j = 0 \text{ for all } j \in J \cup Z, \text{ and}$$

$$\bar{\delta}_S = 0 \text{ for all } S \in S \setminus K.$$

It follows that  $(\bar{y}, \bar{\delta}_K)$  can be calculated by solving

$$\begin{bmatrix} A_J^t & \bar{D}_J^t \end{bmatrix} \begin{bmatrix} y \\ \delta_K \end{bmatrix} = \begin{bmatrix} c_J \end{bmatrix} \tag{6.1.11}$$

after which we can calculate  $\bar{\mu}_B$  using

$$\bar{\mu}_{uv} = \bar{y}_u + \bar{y}_v + \sum(\bar{\delta}_S | S \in K, uv \in \delta(S)) - c_{uv} \text{ for all } uv \in B.$$

In order to solve (6.1.11) we once again take advantage of the fact that we can partition  $J$  into the edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that  $G' = (V, T)$  is an odd 1-forest. Thus we solve

$$\begin{bmatrix} A_T^t & \bar{D}_T^t \\ A_L^t & \bar{D}_L^t \end{bmatrix} \begin{bmatrix} y \\ \delta_K \end{bmatrix} = \begin{bmatrix} c_T \\ c_L \end{bmatrix}$$

or equivalently

$$\begin{bmatrix} I^{|T|} & (A_T^t)^{-1} \bar{D}_T^t \\ A_L^t & \bar{D}_L^t \end{bmatrix} \begin{bmatrix} y \\ \delta_K \end{bmatrix} = \begin{bmatrix} (A_T^t)^{-1} c_T \\ c_L \end{bmatrix} \quad (6.1.12)$$

From the top  $|T|$  equations of (6.1.12) we obtain

$$y = c_T A_T^{-1} - \delta_K \bar{D}_T A_T^{-1}. \quad (6.1.13)$$

Substituting (6.1.13) into the bottom  $|K|$  equations gives

$$\delta_K (\bar{D}_L - \bar{D}_T A_T^{-1} A_L) = c_L - c_T A_T^{-1} A_L.$$

Letting  $B = \bar{D}_L - \bar{D}_T A_T^{-1} A_L$  and  $p = c_L - c_T A_T^{-1} A_L$ , we then find  $\bar{\delta}_K$  by solving  $\delta_K B = p$ . Then we use  $\bar{\delta}_K$  in (6.1.13) in order to obtain  $\bar{y}$ .

In order to solve  $\delta_K B = p$  we must find  $B$  and  $p$ . We obtain  $w = c_T A_T^{-1}$  by solving  $w A_T = c_T$  using the odd 1-forest structure of  $T$  and Algorithm (4.1.3). Then  $p \in \mathbb{R}^L$  is easily obtained by calculating  $p_{uv} = c_{uv} - w_u - w_v$  for all edges  $uv \in L$ .

One method for obtaining  $B \in \mathbb{R}^{K \times L}$  has already been described in Algorithm (6.1.9), which is used to find the current basic primal solution. We can also obtain  $B$  by first calculating each of the rows of  $\bar{D}_T A_T^{-1} \in \mathbb{R}^{K \times V}$ . For each  $S \in K$ , let  $r^S$  denote the row of  $\bar{D}_T A_T^{-1}$  indexed by  $S$ . Then for every edge  $uv \in L$ , the entry  $r^S A_L$  indexed by  $uv$  is simply  $r_u^S + r_v^S$ .

We can calculate  $r^S$  for each  $S \in K$  in one of the following two ways. First, we can find  $r^S$  by solving  $r^S A_T = \bar{d}^S$ , where  $\bar{d}^S$  is defined by

$$\bar{d}_e^S = \begin{cases} 1 & \text{if } e \in \delta(S) \cap T \\ 0 & \text{otherwise.} \end{cases}$$

This can be done using the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.3).

Alternatively, we can find  $r^S$  by first calculating each of the rows of  $A_T^{-1} \in \mathbb{R}^{T \times V}$  indexed by edges  $e \in \delta(S) \cap T$ . Each of these can be found using Algorithm (4.1.8) and the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$ . Then we have

$$r^S = \sum(a^e | e \in \delta(S) \cap T),$$



where  $a^e$  denotes the row of  $A_T^{-1}$  indexed by edge  $e$ .

Once we have found  $\bar{\delta}_K$  by solving  $\delta_K B = p$ , we find  $\bar{y}$  by solving  $yA_T = f$ , where  $f$  is defined by  $f_e = c_e - \sum(\bar{\delta}_S | e \in \delta(S))$  for all  $e \in T$ . This can be done using Algorithm (4.1.3) and the odd 1-forest  $G' = (V, T)$ .

In summary, we calculate the basic dual solution corresponding to the basis  $(K, J, B, Z)$  as follows.

(6.1.14) Algorithm to calculate dual solution  $(\bar{y}, \bar{u}, \bar{\delta})$  for basis  $(K, J, B, Z)$ .

- (1) Let  $\bar{u}_{J \cup Z} = 0$ , and  $\delta_{S, K} = 0$ .
- (2) Partition  $J$  into edge sets  $T$  and  $L$  of size  $|V|$  and  $|K|$  respectively such that  $G'(V, T)$  forms an odd 1-forest.
- (3) Let  $w \in \mathbb{R}^V$  be the unique solution to  $wA_T = c_T$  found by using Algorithm (4.1.3) and the odd 1-forest  $G' = (V, T)$ . Then define  $p \in \mathbb{R}^L$  by  $p_{uv} = c_{uv} - w_u - w_v$  for all edges  $uv \in L$ .
- (4) Let  $a^e \in \mathbb{R}^V$  denote the row of  $A_T^{-1} \in \mathbb{R}^{T \times V}$  indexed by edge  $e \in T$ . Then for every  $e \in T$ , calculate  $a^e$  using Algorithm (4.1.8) and the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$ .
- (5) For all  $S \in K$ , calculate  $r^S \in \mathbb{R}^V$  defined by  $r^S = \sum(a^e | e \in \delta(S) \cap T)$ .
- (6) For all  $S \in K$  and  $uv \in L$ , define each entry  $\beta_{S, uv}$  of matrix

$B \in \mathbb{R}^{K \times L}$  by  $\beta_{S, uv} = \theta - (r_u^S + r_v^S)$ , where

$$\theta = \begin{cases} 1 & \text{if } uv \in \delta(S) \\ 0 & \text{otherwise.} \end{cases}$$

- (7) Calculate the unique solution  $\bar{\delta}_K$  to  $\delta_K B = p$ .
- (8) Define  $f \in \mathbb{R}^T$  by  $f_e = c_e - \sum(\bar{\delta}_S | S \in K, e \in \delta(S))$  for all  $e \in T$ . Then calculate the unique solution  $\bar{y}$  to  $yA_T = f$  using Algorithm (4.1.3) and the odd 1-forest  $G' = (V, T)$ .
- (9) For every edge  $uv \in B$  calculate  $\bar{u}_{uv}$  using 
$$\bar{u}_{uv} = \bar{y}_u + \bar{y}_v + \sum(\bar{\delta}_S | S \in K, uv \in \delta(S)) - c_{uv}.$$

iv) Performing a Pivot.

At each iteration of the dual simplex method we are given a dual feasible basis for linear program (6.1.3) which is defined by the quadruple  $(K, J, B, Z)$ . We will also assume we are given a partition of the edge set  $J$  into the sets  $T$  and  $L$  respectively such that  $T$  is the edge set of an odd 1-forest. As part of the pivot procedure we will obtain such a partition for the next iteration.

First we must check to see if the current basis  $(K, J, B, Z)$  is primal feasible, i.e. check to see if the corresponding primal solution as given in Algorithm (6.1.9) satisfies linear program (6.1.3). If it does, then the current basic primal and dual solutions are optimal for their respective problems. Otherwise one of the following necessarily occurs:

- (a)  $x_e < 0$  for some edge  $e \in J$ .  
 (b)  $s_e < 0$  for some edge  $e \in J$ , or equivalently  $x_e > u_e$  for some  $e \in J$ .  
 (c)  $\gamma_S < 0$  for some subset  $S \in S \setminus K$ , or equivalently  $x(\delta(S)) < d_S$  for some  $S \in S \setminus K$ .

In case (a) we would choose the basic variable  $x_e$  as our leaving variable, and in case (b) we would choose the basic variable  $s_e$  as our leaving variable. In either case, this corresponds to edge  $e$  leaving  $J$ .

In case (c) we would choose the basic variable  $\gamma_S$  to leave the basis. This corresponds to  $S$  joining the set  $K$  of tight cuts. In all three of the above cases we then allow the dual constraint corresponding to the leaving variable (which is currently tight) to become slack by some amount  $t \geq 0$ . Letting  $(\bar{y}, \bar{\mu}, \bar{\delta})$  be the current dual solution and letting  $A'$  denote the coefficient matrix of (6.1.10), the new dual solution in terms of  $t$  is defined by

$$[y, \delta_K, \mu, \delta_{S \setminus K}] = [\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}] - t\bar{a}$$

where  $\bar{a}$  denotes the row of  $(A')^{-1}$  corresponding to the leaving variable chosen.

We calculate  $\bar{a} = [\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}]$  by solving

$$A' \begin{bmatrix} \bar{y} \\ \bar{\delta}_K \\ \bar{\mu}_B \\ \bar{\mu}_J \\ \bar{\mu}_Z \\ \bar{\delta}_{S \setminus K} \end{bmatrix} = \begin{bmatrix} q \\ 0 \\ r \\ 0 \\ z \end{bmatrix} \quad (6.1.15)$$

where  $q \in \mathbb{R}^J$ ,  $r \in \mathbb{R}^J$ , and  $z \in \mathbb{R}^{S \setminus K}$  are defined by

$$q_e = \begin{cases} 1 & \text{if } x_e \text{ is the leaving variable} \\ 0 & \text{otherwise,} \end{cases}$$

$$r_e = \begin{cases} 1 & \text{if } s_e \text{ is the leaving variable} \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{and } z_S = \begin{cases} 1 & \text{if } \gamma_S \text{ is the leaving variable} \\ 0 & \text{otherwise} \end{cases}$$

(i.e. the right-hand side of (6.1.15) has value 1 in the component corresponding to the dual constraint being made slack and value 0 elsewhere).

We solve (6.1.15) a block at a time. From the structure of  $A'$  it follows immediately that  $\bar{\mu}_J = r$ ,  $\bar{\mu}_Z = 0$ , and  $\bar{\delta}_{S \setminus K} = z$ . We then find  $(\bar{y}, \bar{\delta}_K)$  by solving

$$\begin{bmatrix} A_J^t & \bar{D}_J^t \end{bmatrix} \begin{bmatrix} \bar{y} \\ \bar{\delta}_K^t \end{bmatrix} = \begin{bmatrix} h \end{bmatrix} \quad (6.1.16)$$

where  $h = q - r + (\bar{D}_J^t)z$ . Thus  $h = q$  if  $x_e$  is the leaving variable,  $h = -r$  if  $s_e$  is the leaving variable, and if  $\gamma_S$  is the leaving variable  $h$  is the column of  $\bar{D}_J^t$  corresponding to  $S$ , i.e.  $h \in \mathbb{R}^J$  is defined by

$$h_e = \begin{cases} 1 & \text{if } e \in \delta(S) \\ 0 & \text{otherwise.} \end{cases}$$

We solve (6.1.16) using the partition  $T, L$  of  $J$  which is given, and steps (3)-(8) of the dual solution algorithm (6.1.14) with the right-hand sides  $c_T$  and  $c_L$  replaced with  $h_T$  and  $h_L$  respectively. We can then calculate  $\bar{u}_B$  as in step (9) of Algorithm (6.1.14) using  $c_B = 0$ .

Once we have found  $\bar{a} = [\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}]$ , the new dual solution in terms of  $t$  is given by

$$[\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}] = [\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}] - t[\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}]$$

where  $[\bar{y}, \bar{\delta}_K, \bar{\mu}, \bar{\delta}_{S \setminus K}]$  is the current basic dual solution. We then raise the value of  $t$  as high as possible while still maintaining dual feasibility. There are the following three types of dual constraints which may restrict  $t$ :

- (i) an edge constraint for some edge  $e \in Z$
- (ii) a non-negativity constraint for some variable  $\mu_e$ ,  $e \in B$ .
- (iii) a non-negativity constraint for some variable  $\delta_S$ ,  $S \in K$ .

Defining the reduced cost vector  $\bar{c} \in \mathbb{R}^E$  by

$$\bar{c}_{uv} = c_{uv} - (\bar{y}_u + \bar{y}_v + \sum (\bar{\delta}_S | S \in K, uv \in \delta(S))),$$

it follows that  $(y, \mu, \delta)$  will be dual feasible if and only if  $t$  satisfies

- (i)  $\bar{c}_{uv} \geq -t(\bar{y}_u + \bar{y}_v + \sum (\bar{\delta}_S | S \in K, uv \in \delta(S)))$  for all  $uv \in Z$ ,
- (ii)  $\bar{c}_{uv} \leq -t(\bar{y}_u + \bar{y}_v + \sum (\bar{\delta}_S | S \in K, uv \in \delta(S)))$  for all  $uv \in B$ , and
- (iii)  $\bar{\delta}_S \geq t\bar{\delta}_S$  for all  $S \in K$ .

We choose  $t$  as large as possible, subject to the above restrictions. If  $t$  is restricted by an edge constraint for an edge  $e \in Z$ , we choose  $x_e$  as the variable to enter the basis and we add  $e$  to  $J$ . If  $t$  is restricted by a non-negativity constraint for a variable  $\mu_e$ ,  $e \in B$ , we choose  $s_e$  as the variable to enter the basis and we add  $e$  to  $J$ . Finally, if  $t$  is restricted by a non-negativity constraint for a variable  $\delta_S$ ,  $S \in K$ , we choose  $\gamma_S$  as the variable to enter the basis and  $S$  is added to  $S \setminus K$ , or equivalently  $S$  leaves  $K$  and the corresponding cut constraint is no longer required to be tight.

The only part of the pivot procedure which remains to be discussed is the maintenance of the odd 1-forest  $G' = (V, T)$ . This odd 1-forest is only affected during the pivot if the leaving

variable chosen is  $x_e$  or  $s_e$  for some  $e \in T$ , in which case the corresponding edge  $e$  is removed from  $T$ . We deal with this immediately after such a leaving variable is chosen rather than at the end of the pivot as this simplifies the calculation of  $\bar{y}$  and  $\bar{\delta}_K$  in (6.1.16) in some cases.

If the leaving variable is  $x_e$  or  $s_e$  for some  $e \in T$ , we try to find an edge  $f \in L$  such that  $(T \setminus \{e\}) \cup f$  forms the edge set of an odd 1-forest, i.e. we try to find an edge in  $L$  which we can "swap" with  $e$ . To do this, we first calculate  $a^e$ , the row of  $A_T^{-1}$  corresponding to  $e$ . We can find  $a^e$  using Algorithm (4.1.8) and the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$ . Then it is clear from the alternating structure of  $a^e$  (see Figures 4.1.5 and 4.1.6) that edge  $uv \in L$  will form the edge set of an odd 1-forest when added to  $T \setminus \{e\}$  if and only if  $a_u^e + a_v^e \neq 0$ , i.e. if and only if  $a_{uv}^e \neq 0$ .

Consequently we calculate  $a_{uv}^e = a_u^e + a_v^e$  for each  $uv \in L$ . If we find some edge  $uv \in L$  such that  $a_{uv}^e \neq 0$ , we replace the partition  $T, L$  of  $J$  with the partition  $T', L'$ , where  $T' = (T \setminus \{e\}) \cup \{uv\}$  and  $L' = (L \setminus \{uv\}) \cup \{e\}$ . Thus our leaving variable stays the same, but the corresponding edge leaving  $J$  is now in  $L$ , not  $T$ .

If we cannot find an edge  $f \in L$  to swap with the leaving edge  $e \in T$ , then  $a_{eL}^e = 0$ . In this case solving  $\bar{\delta}_K B = p$  becomes unnecessary since  $p = 0 - a_{eL}^e = 0$ . Hence  $\bar{\delta}_K = 0$ , and the

non-negativity constraints for the variables  $\delta_S, S \in K$  will not restrict the value of  $t$  in the pivot. Consequently the entering variable will be  $x_f$  or  $s_f$  for some edge  $f \in Z \cup B$ , and  $J$  becomes  $(J \setminus \{e\}) \cup \{f\}$ . Since it is always possible to find a subset of edges in  $J$  which forms an odd 1-forest, it follows that we can replace  $T$  with  $(T \setminus \{e\}) \cup \{f\}$  in our partition of  $J$ .

Given a dual feasible basis  $(K, J, B, Z)$  for linear program (6.1.3) and a partition  $T, L$  of edge set  $J$  such that  $|T| = |V|$  and  $G' = (V, T)$  is an odd 1-forest, the steps of a dual simplex pivot can be summarized as follows.

(6.1.17) Algorithm to Perform a Pivot.

(1) Calculate the values of the primal variables  $\bar{x}_J$  corresponding to basis  $(K, J, B, Z)$  as in Algorithm (6.1.9), steps (3)-(7), and calculate the values of the dual variables  $\bar{y}$  and  $\bar{\delta}_K$  corresponding to basis  $(K, J, B, Z)$  as in Algorithm (6.1.14), steps (3)-(8).

(2) If

i)  $\bar{x}_e \geq 0$  for all  $e \in J$ ,

ii)  $\bar{x}_e \leq u_e$  for all  $e \in J$ ,

and iii)  $\bar{x}_J(\delta(S) \cap J) + u_B(\delta(S) \cap B) \geq d_S$  for all  $S \in S$

then stop; the current basis is optimal. Otherwise do one of the following:

- i) Choose edge  $e \in J$  such that  $\bar{x}_e < 0$ , and let  $J' = J \setminus \{e\}$ ,  $B' = B$ ,  $Z' = Z \cup \{e\}$ , and  $K' = K$ . Go to step (3).
- ii) Choose edge  $e \in J$  such that  $\bar{x}_e > u_e$ , and let  $J' = J \setminus \{e\}$ ,  $B' = B \cup \{e\}$ ,  $Z' = Z$ , and  $K' = K$ . Go to step (3).
- iii) Choose  $M \in S \setminus K$  such that  $\bar{x}_j(\delta(M) \cap J) + u_B(\delta(M) \cap B) < d_M$ , and let  $J' = J$ ,  $B' = B$ ,  $Z' = Z$ , and  $K' = K \cup \{M\}$ . Let  $T' = T$ ,  $L' = L$ , define  $h \in \mathbb{R}^J$  by

$$h_j = \begin{cases} 1 & \text{if edge } j \in \delta(M) \\ 0 & \text{otherwise} \end{cases}$$

and go to step (5) to calculate  $\bar{y}$  and  $\bar{\delta}_K$ .

- (3) If  $e \in L$  then let  $T' = T$ ,  $L' = L \setminus \{e\}$ , define  $h \in \mathbb{R}^J$  by

$$h_j = \begin{cases} 1 & \text{if } j = e \\ 0 & \text{otherwise,} \end{cases}$$

and go to step (5) to calculate  $\bar{y}$  and  $\bar{\delta}_K$ . Otherwise,  $e \in T$ , and we let  $a^e \in \mathbb{R}^T$  denote the row of  $A_T^{-1}$  indexed by  $e$ . Calculate  $a^e$  using the tree formed when edge  $e$  is removed from the odd 1-forest  $G' = (V, T)$  and Algorithm (4.1.8).

- (4) If  $a_u^e + a_v^e \neq 0$  for some edge  $uv \in L$ , let  $T = (T \setminus \{e\}) \cup \{uv\}$ , let  $L = (L \setminus \{uv\}) \cup \{e\}$  and go to step (3). Otherwise, let  $\bar{\delta}_K = 0$ ,  $\bar{y} = a^e$ ,  $T' = T \setminus \{e\}$ ,  $L' = L$ , and go to step (6) to choose the value of  $t$ .

- (5) Find  $B \in \mathbb{R}^{K \times L}$  and  $p \in \mathbb{R}^L$  as in Algorithm (6.1.14), steps (3)-(6), with  $c_J$  replaced by  $h$ . Calculate  $\bar{\delta}_K$  by solving  $\bar{\delta}_K B = p$ , then calculate  $\bar{y}$  by solving  $\bar{y} A_T = f$ , where  $f \in \mathbb{R}^T$  is defined by  $f_j = h_j - \sum(\bar{\delta}_S | S \in K, j \in \delta(S))$  for all edges  $j \in T$ . Go to step (6) to calculate the value of  $t$ .

- (6) For all edges  $uv \in J \cup B$  calculate  $\bar{c}_{uv}$  and  $\tilde{c}_{uv}$  defined by

$$\bar{c}_{uv} = c_{uv} - (\bar{y}_u + \bar{y}_v + \sum(\bar{\delta}_S | S \in K, uv \in \delta(S)))$$

$$\text{and } \tilde{c}_{uv} = (\bar{y}_u + \bar{y}_v + \sum(\bar{\delta}_S | S \in K, uv \in \delta(S))).$$

Then calculate

$$t_1 = \min \left\{ \frac{-\bar{c}_j}{\tilde{c}_j} \mid j \in Z, \tilde{c}_j < 0 \right\},$$

$$t_2 = \min \left\{ \frac{-\bar{c}_j}{\tilde{c}_j} \mid j \in B, \tilde{c}_j > 0 \right\},$$

$$\text{and } t_3 = \min \left\{ \frac{\bar{\delta}_S}{\tilde{\delta}_S} \mid S \in K, \tilde{\delta}_S > 0 \right\}.$$

Let  $t = \min \{t_1, t_2, t_3\}$ .

- (7) Do one of the following.

- i) Choose an edge  $f \in Z$  such that  $t = \frac{-\bar{c}_f}{\tilde{c}_f}$ . Let  $J'' = J' \cup \{f\}$ ,  $B'' = B'$ ,  $Z'' = Z' \setminus \{f\}$ , and  $K'' = K'$ . If  $|T'| = |V| - 1$ , then let  $T'' = T' \cup \{f\}$  and  $L'' = L'$ . Otherwise let  $T'' = T'$ , and  $L'' = L' \cup \{f\}$ .

ii) Choose an edge  $f \in B$  such that  $t = \frac{-\bar{c}_f}{c_f}$ . Let  $J'' = J' \cup \{f\}$ ,  $B'' = B' \setminus \{f\}$ ,  $Z'' = Z'$ , and  $K'' = K'$ . If  $|T'| = |V| - 1$ , then let  $T'' = T' \cup \{f\}$  and  $L'' = L'$ . Otherwise let  $T'' = T'$ , and  $L'' = L' \cup \{f\}$ .

iii) Choose a subset of nodes  $N \in K$  such that  $t = \frac{\bar{\delta}_N}{\delta_N}$ . Let  $J'' = J'$ ,  $B'' = B'$ ,  $Z'' = Z'$ , and  $K'' = K' \setminus \{N\}$ .

(8) Define the new dual feasible basis to be  $(K'', J'', B'', Z'')$ , and the new partition of  $J''$  to be  $T'', L''$  where  $|T''| = |V|$  and  $G' = (V, T'')$  is an odd 1-forest. The pivot is now complete.

Note that in step (2) of the pivot operation we may need to check primal feasibility for an exponential number of constraints. However, for many specific instances of linear program (6.1.13) and in particular for the subtour problem, it is possible to check this in polynomial time using a separation routine (see Section 6.3).

Also note that in steps (1) and (5) of the pivot we are required to solve a system of  $|K|$  equations and  $|K|$  unknowns. In general,  $|K| \leq |E| - |V|$ . However, for some specific instances of linear program (6.1.13), and in particular for the subtour problem, it is possible to restrict the size of  $K$  to be at most  $|V| - 5$  by ensuring that  $K$  is always a nested family (see Section 6.3).

#### v) Finding an Initial Dual Feasible Basis

To find an initial dual feasible basis for linear program (6.1.3), we first solve linear program (4.2.1) using the simplex method described in Chapter 4. Let the optimal basis found for (4.2.1) be defined by the edge partition  $(T, B, Z)$ . Then letting  $K = \emptyset$  and  $J = T$ , the basis defined by the quadruple  $(K, J, B, Z)$  is a dual feasible basis for (6.1.3). The corresponding initial partition  $T, L$  of  $J$  is defined by  $T = J, L = \emptyset$ .

### 6.2 ENSURING FINITENESS

We can ensure finiteness of the dual simplex method described in Section 6.1 for the linear program (6.1.1) by applying the dual lexicographic anti-cycling method described in Section 1.6. We apply this method to the dual linear program (6.1.2), which can be represented by

$$[y, \nu, \delta] P \leq [c, 0, 0],$$

where  $P$  is the coefficient matrix shown in Figure 6.1.1.

When we perturb the right-hand side of (6.1.2), the corresponding coefficient vectors will have length  $|E| + |E| + |S|$ . We assume the first  $|E|$  components of these vectors correspond to the dual edge constraints, or equivalently, we assume the first  $|E|$  components are indexed by  $E$  in some prescribed ordering chosen at

the beginning of the dual simplex method. Then given any basis for (6.1.1) and a leaving variable, we describe how to calculate these  $|E|$  components for the corresponding coefficient vectors and show they are sufficient for determining which coefficient vector is lexicographically minimum. Note that shortening these vectors by  $|E|$  components is possible for any linear program with upper bounds (see [Murty, 1983]).

Let the current basis be defined by the quadruple  $(K, J, B, Z)$  and let  $A'$  be the corresponding coefficient matrix of (6.1.5). Note that the rows of  $(A')^{-1}$  correspond to the basic primal variables. We let  $\bar{a}$  represent the row of  $(A')^{-1}$  corresponding to the current leaving variable. Then it follows from (1.6.5) that the first  $|E|$  components of the coefficient vector corresponding to a particular dual constraint  $j$  are given by

$$\frac{1}{-\bar{a}P_j} f^j, \quad (6.2.1)$$

where  $P$  is the coefficient matrix shown in Figure 6.1.1, and  $f^j \in \mathbb{R}^E$  is defined by

$f^j_J =$  the components of  $(A')^{-1}P$  corresponding to the basic variables  $x_J$ ,

$f^j_B =$  the components of  $(A')^{-1}P_j$  corresponding to the basic variables  $x_B$ ,

and for all edges  $z \in Z$ ,

$$f^j_Z = \begin{cases} 1 & \text{if } j \text{ is the dual edge constraint for } z \\ 0 & \text{otherwise.} \end{cases}$$

We describe how to calculate  $f^j$  for each of the three types of dual constraints  $j$  which correspond to non-basic primal variables.

Case 1.  $j$  is a dual edge constraint for some edge  $e \in Z$ .

To find  $f^j$  we must calculate the components of  $(A')^{-1}P_j$  corresponding to  $x_J$  and  $x_B$ . These are found by solving

$$\begin{array}{|c|c|} \hline A_J & A_B \\ \hline \bar{D}_J & \bar{D}_B \\ \hline 0 & -I^{|B|} \\ \hline \end{array} \begin{array}{|c|} \hline x_J \\ \hline x_B \\ \hline \end{array} = \begin{array}{|c|} \hline h \\ \hline t \\ \hline g \\ \hline \end{array} \quad (6.2.2)$$

where  $\bar{D} \in \mathbb{R}^{K \times E}$  is the cut-edge incidence matrix for cuts corresponding to  $K$ , and  $h \in \mathbb{R}^V$ ,  $t \in \mathbb{R}^K$ , and  $g \in \mathbb{R}^B$  are defined by  $h = A_e$ ,  $t = \bar{D}_e$ , and  $g = 0$ .

Clearly  $x_B = 0$ . We solve for  $x_J$  by using the current partition  $T, L$  of  $J$  and steps (3)-(7) of the primal solution algorithm (6.1.9) with  $u_B$ ,  $b$ , and  $d_K$  replaced by 0,  $h$ , and  $t$  respectively. Thus  $f^j \in \mathbb{R}^E$  is defined by

$$f^j_i = \begin{cases} 1 & \text{if } i = e \\ -x_i & \text{if } i \in J \\ 0 & \text{otherwise} \end{cases}$$

for all  $i \in E$ .

Case 2.  $j$  is a non-negativity constraint for  $u_e$ ,  $e \in B$ . In this case we find the components of  $(A')^{-1}p_j$  corresponding to  $x_j$  and  $x_B$  by solving (6.2.2) with right-hand sides  $h = 0$ ,  $t = 0$ , and  $g$  defined by

$$g_i = \begin{cases} -1 & \text{if } i = e \\ 0 & \text{otherwise.} \end{cases}$$

for all  $i \in B$ .

Clearly  $x_B = -g$ . We solve for  $x_j$  by using the current partition  $T, L$  of  $J$  and steps (3)-(7) of the primal solution algorithm (6.1.9) with  $u_B$ ,  $b$ , and  $d_K$  replaced with  $g$ ,  $0$ , and  $0$  respectively. Thus  $f^j \in \mathbb{R}^E$  is defined by

$$f_i^j = \begin{cases} -1 & \text{if } i = e \\ -x_i & \text{if } i \in J \\ 0 & \text{otherwise} \end{cases}$$

for all  $i \in E$ .

Case 3.  $j$  is a non-negativity constraint for  $\delta_M$ ,  $M \in K$ .

In this case, we find the components of  $(A')^{-1}p_j$  corresponding to  $x_j$  and  $x_B$  by solving (6.2.2) with right-hand sides  $h = 0$ ,  $g = 0$ , and  $t$  defined by

$$t_S = \begin{cases} -1 & \text{if } S = M \\ 0 & \text{otherwise} \end{cases}$$

for all  $S \in K$ .

Clearly  $x_B = 0$ . We solve for  $x_j$  by using the current partition  $T, L$  of  $J$  and steps (3)-(7) of the primal algorithm (6.1.9) with  $u_B$ ,  $d$ ,  $b_K$  replaced with  $0$ ,  $0$ ,  $t$  respectively. Thus  $f^j \in \mathbb{R}^E$  is defined by

$$f_i^j = \begin{cases} -x_i & \text{if } i \in J \\ 0 & \text{otherwise} \end{cases}$$

for all  $i \in E$ .

After calculating  $f^j$ , we find the vectors defined in (6.2.1) by finding  $\bar{a}p_j$  and then calculating  $\frac{1}{-\bar{a}p_j} f^j$ . We have already calculated  $\bar{a}p_j$  as part of the pivot algorithm (6.1.17). The vector  $\bar{a} = [\bar{y}, \bar{\delta}_K, \bar{u}, \bar{\delta}_{S \setminus K}]$  as calculated in Section 6.1, and thus  $\bar{a}p_j$  is defined by

$$\bar{a}p_j = \begin{cases} \bar{c}_{uv} & \text{if } j \text{ is the dual edge constraint for } uv, uv \in Z \\ -\bar{c}_{uv} & \text{if } j \text{ is the non-negativity constraint for } u_{uv}, uv \in B \\ -\bar{\delta}_S & \text{if } j \text{ is the non-negativity constraint for } \delta_S, S \in K \end{cases}$$

where for all  $uv \in J \cup B$ ,  $\bar{c}_{uv} = \bar{y}_u + \bar{y}_v - \sum(\bar{\delta}_S | S \in K, uv \in \delta(S))$ .

Since the coefficient matrix of (6.2.2) is nonsingular, and no two of the right-hand sides  $z^j = \begin{bmatrix} h \\ t \\ g \end{bmatrix}$  for dual constraint  $j$  are multiples of each other, it follows that the vectors  $\frac{1}{-\bar{a}p_j} f_{J \cup B}^j$  are distinct for all dual constraints  $j$  corresponding to non-basic



primal variables. Thus the first  $|E|$  components of the coefficient vectors given by (6.2.1) are sufficient for determining the entering variable using the dual lexicographic method in the case of a tie.

### 6.3 A SIMPLEX METHOD FOR THE SUBTOUR PROBLEM

The subtour problem (5.0.1) is a special case of linear program (6.0.1) for which  $b = 2$ ,  $d = 2$ ,  $u = 1$ , and  $S = \{S \subseteq V \mid 3 \leq |S| \leq n - 3\}$ . Thus the dual simplex method described in Sections 6.1 - 6.2 can be applied to this problem. Primal feasibility can be checked in polynomial time at each iteration of the method using a separation routine for subtour constraints (see Section 3.3). Moreover, by keeping the set  $K$  which corresponds to tight subtour constraints nested at every iteration, it is possible to improve the computational effectiveness of this dual simplex method for the subtour problem (see Section 5.1 for an introduction to nested families of subsets).

In our initial dual feasible basis we have  $K = \emptyset$  which is trivially nested. Then to ensure  $K$  stays nested at each iteration, we choose the leaving variable in step (2) of the pivot algorithm (6.1.17) more carefully.

Recall that for a given basis  $(K, J, B, Z)$  with corresponding primal solution  $\bar{x}$ , there are three types of primal inequalities which can be violated:

- i)  $\bar{x}_e \geq 0$  for some edge  $e \in J$ ,
- ii)  $\bar{x}_e \leq 1$  for some edge  $e \in J$ ,
- iii)  $x(\delta(S)) \geq 2$  for some  $S \in S$ .

We choose the leaving variable for the current iteration by choosing a violation of one of the above, if one exists.

If we choose a set  $S$  for which iii) is violated, we must add the set  $S$  to  $K$ . Thus we desire to find such a violation for which  $S \cup K$  is nested. This can be obtained from any violation of iii) by repeated applications of the following uncrossing algorithm, which requires that  $0 \leq x_j \leq 1$  (i.e. it requires there be no violated inequalities of types i) or ii)).

#### (6.3.1) Uncrossing Algorithm

Let the basic primal solution  $\bar{x}$  corresponding to the current basis  $(K, J, B, Z)$  satisfy  $0 \leq \bar{x}_j \leq 1$ , let  $S_1 \in S$  violate iii), and suppose  $S_1$  crosses some set  $S_2 \in K$ .

Calculate  $\alpha_1 = \bar{x}(\delta(S_1 \cap S_2))$  and  $\alpha_2 = \bar{x}(\delta(S_1 \cup S_2))$ .

$$\begin{aligned} \text{Since } \alpha_1 + \alpha_2 &= \bar{x}(\delta(S_1 \cap S_2)) + \bar{x}(\delta(S_1 \cup S_2)) \\ &= \bar{x}(\delta(S_1)) + \bar{x}(\delta(S_2)) - 2\bar{x}((\delta(S_1) \cap \delta(S_2)) \setminus \delta(S_1 \cap S_2)) \\ &\leq \bar{x}(\delta(S_1)) + 2 \\ &< 4, \end{aligned}$$

it follows that either  $\alpha_1 < 2$ ,  $\alpha_2 < 2$ , or both. If  $\alpha_1 < 2$ , then let  $S^* = S_1 \cap S_2$ , otherwise let  $S^* = S_1 \cup S_2$ .

Note that the subset  $S^*$  found in Algorithm (6.3.1) satisfies  $3 \leq |S^*| \leq n - 3$  since  $\bar{x} \leq 1$ . Thus  $S^* \in S$  and violates iii). Moreover, the number of sets in  $K$  crossed by  $S^*$  is strictly less than the number of sets in  $K$  crossed by  $S_1$ . Consequently, after at most  $|K|$  repetitions of Algorithm (6.3.1) we will obtain  $S \in S$  such that  $S$  violates iii), and  $K \cup S$  is a nested family.

The separation routine for subtour constraints runs in time  $O(n^4)$  (see Section 3.3) and for general problems,  $|K| \leq |E| - |V|$ . Thus checking primal feasibility for the subtour problem where  $K$  is required to remain nested requires time  $O(n^4)$ , where  $n = |V|$ .

The following theorem shows that if  $K$  is a nested family of the sort used in the algorithm, its size is restricted.

(6.3.2) Theorem. Let  $(K, J, B, Z)$  be a basis for linear program (6.0.1) such that  $K \subseteq \{S \subseteq V \mid 3 \leq |S| \leq n - 3\}$  and  $K$  is a nested family. Then  $|K| \leq n - 5$ , where  $n = |V|$ .

Proof. Since cut constraints are equivalent for  $S \subseteq V$  and  $\bar{S} = V \setminus S$ , we can assume  $S \in K$  implies  $\bar{S} \notin K$ . Let  $K' = K \cup \{\bar{S}\}$  for some maximal  $S \in K$ . Then  $K'$  must contain at least two minimal members  $S_1$  and  $S_2$ . It follows that  $|S_1| = |S_2| = 3$ , and we let  $S_1'$  and  $S_2'$  be subsets of cardinality two in  $S_1$  and  $S_2$  respectively.

Let  $C = K' \cup \{S_1, S_2\} \cup \{V\} \cup \{\{v\} \mid v \in V\}$ . Then  $C$  is a nested family such that  $|C| = |K| + n + 4$ . Thus it follows that  $|K| \leq n - 5$  by Lemma (5.1.4).

Recall that in the dual simplex method for linear program (6.0.1), we are required to solve systems of  $|K|$  equations and  $|K|$  unknowns. In general  $|K| \leq |E| - |V|$ , but for the subtour problem we can ensure  $|K| \leq |V| - 5$  by keeping  $K$  nested, thus improving the computational effectiveness of the method.

#### 6.4 A SIMPLEX METHOD FOR THE PERFECT b-MATCHING PROBLEM

The perfect b-matching problem is a special case of linear program (6.0.1) in which  $d = 1$ ,  $u = +\infty$  (i.e. there are no upper-bound constraints),  $b > 0$  and integer, and  $S = \{S \subseteq V \mid b(S) \text{ is odd, } 3 \leq |S| \leq n - 3\}$ . Thus the dual simplex method described in Sections 6.1 - 6.2 can be applied to this problem, where for every basis  $(K, J, B, Z)$  we have  $J = \emptyset$ . Primal feasibility of the corresponding cut constraints can be checked at each iteration in polynomial time using a separation routine developed by [Padberg and Rao, 1982] which is based on a modification of the procedure of [Gomory and Hu, 1961] for finding a minimum cost cut in a graph.

As in the case of the subtour problem, we can improve the computational effectiveness of this dual simplex method for the

perfect b-matching problem by ensuring the set  $K$  corresponding to tight cut constraints stays nested at every iteration (see Section 5.1 for an introduction to nested families of subsets). In the initial feasible basis we have  $K = \emptyset$  which is trivially nested. Then for any given basis  $(K, J, \emptyset, Z)$  with corresponding primal solution  $\bar{x}$ , there are two types of primal constraints which can be violated:

- i)  $\bar{x}_e \geq 0$  for some edge  $e$ , and
- ii)  $\bar{x}(\delta(S)) \geq 1$  for some  $S \in \mathcal{S}$ .

We choose the leaving variable for the current iteration by choosing a violation of one of the above, if one exists.

Assuming our current set  $K$  is a nested family, we ensure it remains nested in the next iteration by only choosing violations of ii) for which the corresponding set  $S \in \mathcal{S}$  does not cross any members of  $K$ . Given any violation of ii) we can obtain one for which  $S \cup K$  is nested by repeated applications of the following uncrossing algorithm, which requires  $\bar{x}_j \geq 0$  (i.e. it requires there be no violations of i)).

#### 6.4.1 Uncrossing Algorithm

Let the basic primal solution  $\bar{x}$  corresponding to the current basis  $(K, J, \emptyset, Z)$  for the perfect b-matching problem satisfy  $\bar{x}_j \geq 0$ , let  $S \in \mathcal{S}$  violate ii), and suppose  $S$  crosses some set  $T \in K$ .

Since  $b(S)$  and  $b(T)$  are both odd, it follows that either  $b(S \cap T)$  and  $b(S \cup T)$  are both odd, or else  $b(S \setminus T)$  and  $b(T \setminus S)$  are both odd.

Case 1.  $b(S \cap T)$  and  $b(S \cup T)$  are odd.

$$\text{Calculate } \alpha_1 = \bar{x}(\delta(S \cap T)) \text{ and } \alpha_2 = \bar{x}(\delta(S \cup T)).$$

Since

$$\begin{aligned} \alpha_1 + \alpha_2 &= \bar{x}(\delta(S \cap T)) + \bar{x}(\delta(S \cup T)) \\ &= \bar{x}(\delta(S)) + \bar{x}(\delta(T)) - 2\bar{x}((\delta(S) \cap \delta(T)) \setminus \delta(S \cap T)) \\ &\leq \bar{x}(\delta(S)) + 1 \\ &< 2, \end{aligned}$$

it follows that either  $\alpha_1 < 1$ ,  $\alpha_2 < 1$ , or both. If  $\alpha_1 < 1$  then let  $S^* = S \cap T$ , otherwise let  $S^* = S \cup T$ .

Case 2.  $b(S \setminus T)$  and  $b(T \setminus S)$  are odd.

$$\text{Calculate } \alpha_1 = \bar{x}(\delta(S \setminus T)) \text{ and } \alpha_2 = \bar{x}(\delta(T \setminus S)).$$

Since

$$\begin{aligned} \alpha_1 + \alpha_2 &= \bar{x}(\delta(S \setminus T)) + \bar{x}(\delta(T \setminus S)) \\ &= \bar{x}(\delta(S)) + \bar{x}(\delta(T)) - 2\bar{x}(\delta(S \cap T) \setminus (\delta(S \setminus T) \cup \delta(T \setminus S))) \\ &\leq \bar{x}(\delta(S)) + 1 \\ &< 2, \end{aligned}$$

it follows that either  $\alpha_1 < 1$ ,  $\alpha_2 < 1$ , or both. If  $\alpha_1 < 1$  then let  $S^* = S \setminus T$ , otherwise let  $S^* = T \setminus S$ .

Since the subset  $S^*$  found in Algorithm (6.4.1) violates ii) and  $b(S^*)$  is odd, it follows that  $3 \leq |S^*| \leq n - 3$  and thus  $S^* \in S$ . Moreover, the number of sets in  $K$  crossed by  $S^*$  is strictly less than the number of sets in  $K$  crossed by  $S$ , since  $K$  is nested. Consequently, after at most  $|K|$  repetitions of Algorithm (6.4.1) we will obtain  $\bar{S} \in S$  such that  $\bar{S}$  violates ii), and  $K \cup \bar{S}$  is a nested family.

Recall that in the dual simplex method for linear program (6.0.1) we are required to solve systems of  $|K|$  equations and  $|K|$  unknowns at each iteration. In general, if  $K$  is not nested then  $|K| \leq |E| - |V|$ . However, in [Pulleyblank, 1973] it is shown that if  $K$  is a nested family of subsets of the sort used in the dual simplex method for the perfect b-matching problem, then  $|K| \leq \frac{|V| - 1}{2}$ . Thus by ensuring  $K$  is always a nested family, we can improve the computational effectiveness of this method.

Another simplex method for the perfect b-matching problem appears in [Koch, 1979]. This method is a primal simplex method which encounters only specially structured bases, called bundles, which allow easy computations in the corresponding pivots. Unfortunately, the method has no anti-cycling pivot rule to ensure finiteness, and none of the general anti-cycling methods can be applied without destroying the basis structure required.

Let  $F \subseteq S$  be a nested family, and for any  $M \in F$  let  $S_1, S_2, \dots, S_t$  be the maximal members of  $F$  contained in  $M$ . Then the

surface graph,  $G_F$  of  $F$  is the graph obtained from  $G' = (F, E \cap \gamma(F))$  by "shrinking" each  $S \in \{S_1, S_2, \dots, S_t\}$  into a node  $v_S$  called a pseudo-node; i.e.  $V(G_F) = (M \setminus (S_1 \cup S_2 \cup \dots \cup S_t)) \cup \{v_{S_1}, v_{S_2}, \dots, v_{S_t}\}$ ,  $E(G_F) = (E \cap \gamma(M)) \setminus (\gamma(S_1) \cup \gamma(S_2) \cup \dots \cup \gamma(S_t))$ , and  $\delta(v_S)$  in  $G_F$  equals  $\delta(S)$  in  $G'$  for all  $S \in \{S_1, S_2, \dots, S_t\}$ .

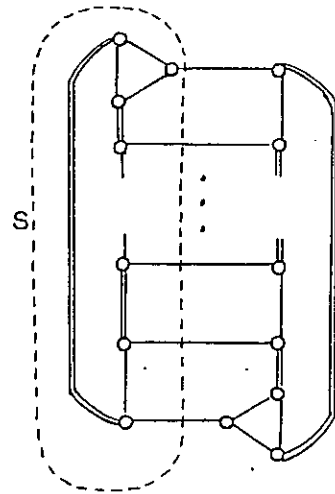
A bundle corresponds to a basis  $(K, J, \emptyset, Z)$  for the perfect b-matching problem which satisfies the following:

- i)  $K$  is nested.
- ii) For all  $S \in K \cup \{V\}$ , the graph  $G = (V(G_S), J \cap E(G_S))$  is an odd 1-forest.

In [Koch, 1979] it is shown how the structure of a bundle can be used to find the corresponding primal and dual basic solution recursively. It is also shown that

- (a) given any non-optimal feasible bundle, there always exists a primal simplex pivot to another feasible bundle, and
- (b) for any objective function  $cx$ ,  $c \in \mathbb{R}^E$ , there exists a bundle for the associated perfect b-matching problem such that the corresponding basic primal and dual solutions are optimal.

It is not always possible to find an optimal basis for the subtour problem which corresponds to a bundle. Shown in Figure 5.1.1 is a vertex of  $Q_S^{2k+4}$ , where  $k \geq 3$  is a positive integer (see Theorem (5.1.1)). This vertex has a unique corresponding basis  $(\{S\}, J, B, Z)$  shown in Figure 6.4.1. The surface graph  $G_V$  for this basis is shown in Figure 6.4.2, and is clearly not an odd 1-forest.



○—○ corresponds to  $e \in J$   
 ○=○ corresponds to  $e \in B$   
 - edges  $e \in I$  are not shown

Figure 6.4.1

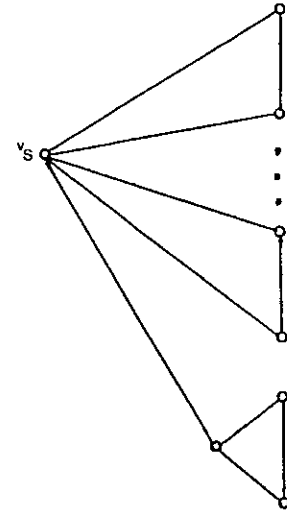


Figure 6.4.2

## CHAPTER 7

Computer Implementations and Results

In this chapter we present computational results obtained from an implementation of the simplex method for the fractional 2-factor problem described in Chapter 4. We also present the computational results of an implementation of the dual simplex method for the subtour problem (described in Chapter 6) in which, rather than including all the subtour constraints, we instead include only a small subset of them which are predicted to be "important" cuts. If the optimal solution obtained is not a member of  $Q_S^n$ , we proceed to add subtour constraints which we obtain using an efficient heuristic procedure with very favourable results.

7.1 AN IMPORTANT SET OF CUTS

To solve the subtour problem using the dual simplex method described in Chapter 6, we first obtain the optimal solution to the corresponding fractional 2-factor problem, then add all the subtour constraints to our linear system and apply the dual simplex method. At each iteration of the method we check primal feasibility using, when necessary, a separation routine for subtour constraints which requires  $O(n^4)$  steps, where  $n$  is the number of nodes (see Section 3.2). In order to avoid having to use this separation routine, we instead seek to find a small subset of subtour constraints which are

easily obtained, and likely to be violated by our initial primal solution and subsequent infeasible basic primal solutions encountered.

We obtain such a set of subtour constraints by finding a minimum cost spanning tree in  $K_n$ , using the same edge costs as those for the subtour problem. Such a spanning tree can be found in  $O(n^2)$  steps (see [Papadimitriou and Steiglitz, 1982]).

Consider any infeasible basic primal solution encountered during the dual simplex method. If there exists a violated subtour constraint for some  $S \in \mathcal{S}$ , it indicates that the cost of any edge in  $\delta(S)$  is high. Hence the minimum cost spanning tree is likely to only have one edge across such a cut. Consequently, as our initial set of subtour constraints we use the cut constraints corresponding to all sets  $S \subseteq V$ ,  $3 \leq |S| \leq n - 3$  such that  $S$  is one side of a node partition obtained by removing some edge from the minimum cost spanning tree. Letting  $S_{\text{MIN}}$  represent the set of all such node subsets  $S$ , we then use the dual simplex method described in Chapter 6 to solve linear program (6.0.1) in the case where  $u = 1$ ,  $b = 2$ ,  $d = 2$ , and  $S = S_{\text{MIN}}$ . Note that  $|S_{\text{MIN}}| \leq n - 5$ , where  $n$  is the number of nodes.

If the optimal solution obtained is not in  $Q_S^n$ , we look for a violated subtour constraint to add to our current linear system, and then continue the dual simplex method. Again, to avoid making

use of the separation routine which requires  $O(n^4)$  steps, we use a heuristic procedure to find such a violation.

In this heuristic procedure we first find a maximum cost spanning tree using the costs  $\bar{x}_e$  for all  $e \in E$ , where  $\bar{x}$  is the current basic primal solution. Then letting  $S_{MAX}$  be the set of all  $S \subseteq V$ ,  $3 \leq |S| \leq n - 3$  such that  $S$  is one side of a node partition obtained by removing some edge from the maximum cost spanning tree, we simply check all the subtour constraints corresponding to sets  $S \in S_{MAX}$  for feasibility. This method does not necessarily find an existing violated subtour constraint, but we do have the following result.

(7.1.1) Theorem. Let  $\bar{x} \in \mathbb{R}^E$  satisfy the constraints of the 2-factor problem, let  $G_{\bar{x}}$  represent the support of  $\bar{x}$ , and let  $T$  be a maximum cost spanning tree of  $K_n = (V, E)$  with respect to the costs  $\bar{x}_e$  for all  $e \in E$ . Then if  $\bar{x}(\delta(S)) \geq 2$  for every  $S \subseteq V$ ,  $3 \leq |S| \leq n - 3$  such that  $S$  is one side of a node partition obtained by removing some edge from  $T$ , it follows that  $G_{\bar{x}}$  is 2-edge connected.

Proof. Suppose  $G_{\bar{x}}$  is not 2-edge connected. Then there exists  $S \subseteq V$  such that  $|\delta(S) \cap E(G_{\bar{x}})| < 2$ , and thus  $\bar{x}(\delta(S)) < 2$  since  $\bar{x} \leq 1$ . Consequently,  $|\delta(S) \cap E(T)| \geq 2$ , and for at least one edge  $e \in \delta(S) \cap E(T)$  we have  $\bar{x}_e = 0$ .

Consider  $\bar{S} \subseteq V$  obtained by taking one side of the node partition formed when edge  $e$  is removed from  $T$ . By assumption,  $\bar{x}(\delta(\bar{S})) \geq 2$ . Thus there exists some edge  $f$  such that  $f \in \delta(\bar{S})$  and  $\bar{x}_f > 0$ . Consequently, the spanning tree  $T' = T \setminus \{e\} \cup \{f\}$  has a greater cost than  $T$ , contradicting the fact that  $T$  is a maximum cost spanning tree.  $\square$

## 7.2 COMPUTER IMPLEMENTATIONS

All our computer implementations are run on an IBM PC using Basic 2.0.

The first algorithm implemented was the simplex method described in Chapter 4 for the fractional 2-factor problem on  $K_n = (V, E)$ . We then used the optimal basis obtained to find an initial dual feasible basis for the following problem:

$$\begin{aligned} & \text{minimize} && cx && (7.2.1) \\ & \text{subject to} && x(\delta(v)) = 2 && \text{for all } v \in V, \\ & && x(\delta(S)) \geq 2 && \text{for all } S \in S_{MIN}, \\ & && 0 \leq x \leq 1 \end{aligned}$$

where  $S_{MIN}$  is the set of all  $S \subseteq V$ ,  $3 \leq |S| \leq n - 3$  obtained from a minimum cost spanning tree of  $K_n$  using edge costs  $c$  (see Section 7.1). We solve (7.2.1) using the dual simplex method described in Chapter 6. Note that the polytope  $\bar{P}$  associated with

(7.2.1) satisfies  $Q_S^n \subseteq \bar{P} \subseteq Q_F^n$ , thus solving (7.2.1) provides a better lower bound for the TSP than the fractional 2-factor problem. We use TREETOPE to denote the polytope  $\bar{P}$ .

If the optimal solution  $\bar{x}$  obtained for (7.2.1) is not in  $Q_S^n$ , we grow a maximum cost spanning tree  $T$  with respect to the edge costs  $\bar{x}$ , and let  $S_{MAX}$  represent all the  $S \subseteq V$ ,  $3 \leq |S| \leq n - 3$  obtained from  $T$  (see Section 7.1). If all the subtour constraints corresponding to  $S \in S_{MAX}$  are satisfied by  $\bar{x}$  we stop. Otherwise we add a violated cut constraint to our current linear system and continue the dual simplex method.

The next time we reach optimality we check the feasibility of the cuts corresponding to the old  $S_{MAX}$  before growing a new maximum cost spanning tree with respect to the current basic primal solution  $\bar{x}$ . In this way we only grow a maximum cost spanning tree when necessary.

Once we have stopped, our linear system consists of the constraints of (7.2.1) plus several more subtour constraints. Note that the polytope  $P$  corresponding to this linear system satisfies  $Q_S^n \subseteq P \subseteq \text{TREETOPE}$ , thus we have now obtained a better lower bound for the TSP. We use TREETOPE(X) to denote the polytope  $P$ .

In all implementations, the odd 1-forests are stored by making use of the "triply linked" structure (see [Koch, 1979]). The

problems used are either Euclidean problems generated from a random number seed, or else problems for which the corresponding cost or distance matrix is read into the program. In the case of Euclidean problems, the data for the problem requires  $O(n)$  bytes of storage, for all we need store are the locations of the cities. Our implementation of the fractional 2-factor problem also requires, in principle,  $O(n)$  storage. However, for reasons of efficiency, we actually compute the full distance matrix at the beginning of the program, and hence we use  $O(n^2)$  bytes of storage. (This is our only requirement of  $O(n^2)$  bytes of storage, in this case.) Of course, for the case of non-Euclidean problems, storage of this distance matrix is essential. The algorithms for TREETOPE and TREETOPE(X) require  $O(n^2)$  bytes of storage.

### 7.3 COMPUTATIONAL RESULTS

For all implementations we ran a set of 35 randomly generated Euclidean problems. Of these 35 problems, 10 were 40-city problems, 10 were 60-city problems, 10 were 80-city problems, and 5 were 100-city problems. We also ran the 120-city problem discussed in [Grötschel, 1980] and the 48-city problem discussed in [Rinaldi and Yarrow, 1985]. The results for the fractional 2-factor problem are shown in Table 7.3.1, the results for optimizing over TREETOPE are shown in 7.3.2, and the results for optimizing over TREETOPE(X) are shown in Table 7.3.3.



To each randomly generated problem we applied various standard TSP heuristics available to us. Letting HCOST represent the cost of the best tour found using the heuristics, we then calculated

$$\text{GAP} = 100 \left( \frac{\text{HCOST} - \text{COST}}{\text{HCOST}} \right)$$

for each of the three implementations, where COST represents the cost of the optimal solution found by the implementation. For the 48-city problem and the 120-city problems we let HCOST represent the cost of an optimal tour.

For each TSP tested, GAP represents the percent difference between the best heuristic solution and the lower bound found by each implementation. Note that a large GAP for a problem may indicate that the heuristics performed poorly, rather than indicating the lower bound obtained is poor. For the fractional 2-factor problem the average GAP over all 37 problems tested was 6.9%, while for optimizing over TREETOPE and TREETOPE(X) it was 2.3% and .7% respectively. Of the solutions found by optimizing over TREETOPE(X), all but 4 were also optimal solutions for the corresponding subtour problem, and 5 of the solutions were tours. The total elapsed times required by all these routines ranged from around five minutes on a problem of 40 to 50 cities to seventy minutes for the 120 city problem.

RESULTS FOR THE FRACTIONAL 2-FACTOR PROBLEM

| Number of Cities | Average Number of Pivots | Average Number of Degenerate Pivots | Average Gap | Maximum Gap | Minimum Gap |
|------------------|--------------------------|-------------------------------------|-------------|-------------|-------------|
| 40               | 47                       | 41                                  | 7.4%        | 15.3%       | 2.1%        |
| 60               | 76                       | 67                                  | 7.6%        | 13.8%       | 3.3%        |
| 80               | 93                       | 81                                  | 6.6%        | 10.5%       | 3.7%        |
| 100              | 126                      | 112                                 | 6.1%        | 7.9%        | 3.2%        |
| 48-city problem  | 65                       | 54                                  | 5.4%        | -           | -           |
| 120-city problem | 142                      | 130                                 | 4.0%        | -           | -           |

Table 7.3.1

RESULTS FOR OPTIMIZING OVER TREETOPE

| Number of Cities | Average Number of Pivots | Average Number of Degenerate Pivots | Average Number of Tight Subtour Constraints | Average Gap | Maximum Gap | Minimum Gap |
|------------------|--------------------------|-------------------------------------|---------------------------------------------|-------------|-------------|-------------|
| 40               | 32                       | 6                                   | 5                                           | 1.9%        | 4.6%        | 0%          |
| 60               | 38                       | 11                                  | 7                                           | 2.3%        | 4.2%        | .5%         |
| 80               | 46                       | 12                                  | 8                                           | 2.6%        | 4.1%        | 1.1%        |
| 100              | 47                       | 16                                  | 8                                           | 3.0%        | 4.0%        | 1.7%        |
| 48-city problem  | 36                       | 1                                   | 8                                           | .7%         | -           | -           |
| 120-city problem | 37                       | 2                                   | 9                                           | 1.4%        | -           | -           |

Table 7.3.2

RESULTS FOR OPTIMIZING OVER TREETOPE(X)

| Number of Cities | Average Number of Pivots | Average Number of Degenerate Pivots | Average Number of Tight Subtour Constraints | Average Gap | Maximum Gap | Minimum Gap |
|------------------|--------------------------|-------------------------------------|---------------------------------------------|-------------|-------------|-------------|
| 40               | 32                       | 7                                   | 7                                           | .5%         | 2.1%        | 0%          |
| 60               | 64                       | 21                                  | 12                                          | .7%         | 1.8%        | 0%          |
| 80               | 84                       | 33                                  | 16                                          | .8%         | 1.6%        | 0%          |
| 100              | 98                       | 37                                  | 21                                          | 1.1%        | 2.2%        | .6%         |
| 48-city problem  | 45                       | 3                                   | 11                                          | .3%         | -           | -           |
| 120-city problem | 85                       | 6                                   | 21                                          | .4%         | -           | -           |

Table 7.3.3

## REFERENCES

1. [A. Bachem and M. Grötschel, 1982], "New aspects of polyhedral theory", in B. Korte (ed.) Modern Applied Mathematics - Optimization and Operations Research, North Holland, 51-106 (1982).
2. [M. Balinski, 1970], "On recent developments in integer programming", in H.W. Kuhn (ed.), Proceedings of the Princeton Symposium on Mathematical Programming, Princeton University Press, N.J., 267-302 (1970).
3. [F. Barahona and W. Cunningham, 1984], "A submodular network simplex method", Math. Programming Study 22, 9-31 (1984).
4. [R. Barr, F. Glover, and D. Klingman, 1977], "The alternating basis algorithm for assignment problems", Math. Programming 13, 1-13 (1977).
5. [J.A. Bondy and U.S.R. Murty, 1976], Graph Theory with Applications, Elsevier, New York (1976).
6. [N. Christofides, 1979], "The travelling salesman problem," in N. Christofides et al (eds.), Combinatorial Optimization, J. Wiley, New York, 131-149 (1979).
7. [V. Chvátal, 1973], "Edmonds polytopes and weakly hamiltonian graphs", Mathematical Programming 5, 29-40 (1973).
8. [V. Chvátal, 1983], Linear Programming, W.H. Freeman and Company, New York (1983).
9. [G. Cornuéjols and W. Pulleyblank, 1982], "The travelling salesman polytope and  $\{0,2\}$ -matchings", in A. Bachem et al (eds.), Bonn Workshop on Combinatorial Optimization, North-Holland, Annals of Discrete Mathematics 16, 27-55 (1982).
10. [H. Crowder and M. Padberg, 1980], "Solving large-scale symmetric travelling salesman problem", Operations Research 2, 393-410 (1954).
11. [W. Cunningham, 1976], "A network simplex method", Math. Programming 11, 105-116 (1976).
12. [G. Dantzig, 1948], "Programming in a linear structure", Comptroller, USAF, Washington, D.C. (1948).
13. [G. Dantzig, 1963], Linear Programming and Extensions, Princeton University Press, N.J. (1963).
14. [G. Dantzig, D. Fulkerson and S. Johnson, 1954], "Solution of a large-scale travelling salesman problem", Operations Research 2, 393-410 (1954).
15. [J. Edmonds, 1965], "Paths, trees, and flowers", Canadian Journal of Mathematics 17, 449-467 (1965).
16. [J. Edmonds and E. Johnson, 1970], "Matching: a well-solved class of integer linear programs", in R. Guy et al (eds.), Combinatorial Structures and Their Applications, Gordon-Breach, New York (1970).
17. [J. Elam, F. Glover, and D. Klingman, 1979], "A strongly convergent primal simplex algorithm for generalized networks", Mathematics of Operations Research 4, 39-59 (1979).
18. [J. Farkas, 1902], "Theorie der einfachen Ungleichungen", Journal für die Reine und Angewandte Mathematik 124, 1-27 (1902).
19. [M. Garey and D. Johnson, 1979], Computers and Intractability, W.H. Freeman and Company, San Francisco (1979).
20. [F. Glover, D. Karney, and D. Klingman, 1974], "Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems", Networks 4, 191-212 (1974).
21. [A.J. Goldman, 1956]. "Resolution and separation theorems for polyhedral convex sets", in H.W. Kuhn and A.W. Tucker (eds.), Linear Inequalities and Related Systems, Annals of Mathematics Study #38, 41-51 (1956).
22. [R. Gomory and T. Hu, 1961], "Multiterminal network flows", Journal of S.I.A.M. 9, 551-570 (1961).
23. [M. Grötschel, 1977], Polyedrische Charakterisierungen Kombinatorischer Optimierungsprobleme, Verlag Anton Hain, Meisenheim am Glan, 1977.
24. [M. Grötschel, 1980], "On the symmetric travelling salesman problem: solution of a 120 city problem", Mathematical Programming Study 12, 61-77 (1980).

25. [M. Grötschel and M. Padberg, 1979a], "On the symmetric travelling salesman problem I: inequalities", Mathematical Programming 16, 265-280 (1979).
26. [M. Grötschel and M. Padberg, 1979b], "On the symmetric travelling salesman problem II: lifting theorems and facets", Mathematical Programming 16, 281-302 (1979).
27. [M. Grötschel and M. Padberg, 1985a], "Polyhedral aspects of the travelling salesman problem I: theory", in E. Lawler, J. Lenstra, A. Rinnooy Kan (eds.) The Travelling Salesman Problem, Wiley (1985).
28. [M. Grötschel and M. Padberg, 1985b], "Polyhedral aspects of the travelling salesman problem II: computation", in E. Lawler, J. Lenstra, A. Rinnooy Kan (eds.) The Travelling Salesman Problem, Wiley (1985).
29. [M. Grötschel and W. Pulleyblank, 1981], "Clique tree inequalities and the symmetric travelling salesman problem", Report No. 81196-OR, Institut für Ökonometrie und Operations Research, Universität Bonn, 1981, to appear in Math. of Operations Research.
30. [M. Grötschel, M. Jünger and G. Reinelt, 1982a], "Facets of the linear ordering polytope", Research Report 82217-OR, Institut für Operations Research, Universität Bonn, (1982).
31. [M. Grötschel, M. Jünger and G. Reinelt, 1982b], "On the acyclic subgraph polytope", Research Report 82215-OR, Institut für Operations Research, Universität Bonn, (1982).
32. [M. Grötschel, L. Lovász, and A. Schrijver, 1981], "The ellipsoid method and its consequences in combinatorial optimization", Combinatorica 1, 169-197 (1981).
33. [F. Harary, 1969], Graph Theory, Addison-Wesley, Reading, Mass. (1969).
34. [T. Havel, 1982], "The Combinatorial Distance Geometry Approach to the Calculation of Molecular Conformation", Doctoral Thesis, University of California, Berkeley, United States, 1982.
35. [M. Held and R. Karp, 1970], "The travelling salesman problem and minimum spanning trees", Operations Research 18, 1138-1162 (1970).

36. [R. Karp, 1972], "Reducibility among combinatorial problems", in R. Miller and J. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, 85-103 (1972).
37. [R. Karp and C. Papadimitriou, 1980], "On linear characterizations of combinatorial optimization problems", Proceedings of the Twenty-first Annual Symposium on the Foundations of Computer Science, IEEE, 1-9 (1980).
38. [J. Kennington and R. Helgason, 1980], Algorithms for Network Programming, Wiley, New York (1980).
39. [E. Koch, 1979], "Ramifications of Matching Theory", Doctoral Thesis, University of Waterloo, Waterloo, Canada, 1979.
40. [H. Kuhn and R. Quandt, 1963], "An experimental study of the simplex method", Proceedings of Symposia in Applied Mathematics 15, 107-124 (1963).
41. [S. Lin and B. Kernighan, 1973], "An effective heuristic algorithm for the travelling salesman problem", Operations Research 21, 498-516 (1973).
42. [B. Marsh, 1976], Doctoral Thesis, John Hopkins University, Baltimore, MD, U.S.A., 1976.
43. [J. Maurras, 1975], "Some results on the convex hull of the hamiltonian cycles of symmetric complete graphs", in B. Roy (ed.), Combinatorial Programming: Methods and Applications, Reidel, Dordrecht, 1975.
44. [K. Menger, 1930], "Botenproblem", in K. Menger, (ed.), Ergebnisse Eines Mathematischen Kolloquiums, Heft 2, Leipzig, 11-12 (1930).
45. [T.S. Motzkin, 1936], "Beiträge zur Theorie der linearen Ungleichungen", (Dissertation, Basel, 1933) Jerusalem (1936).
46. [K. Murty, 1983], Linear Programming, Wiley, New York (1983).
47. [M. Padberg and S. Hong, 1980], "On the symmetric travelling salesman problem: a combinatorial study", Mathematical Programming Study 12, 78-107 (1980).

48. [M. Padberg and M. Rao, 1980], "Odd minimum cut-sets and b-matchings", Working Paper, Graduate School of Business Administration, New York University, revised version, August 1980.
49. [C. Papadimitriou and K. Steiglitz, 1982], Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall Inc., N.J. (1982).
50. [W.R. Pulleyblank, 1973], "Faces of Matching Polyhedra", Doctoral Thesis, University of Waterloo, Waterloo, Canada, 1973.
51. [G. Rinaldi and L. Yarrow, 1985], "Optimizing a 48 city problem: a case study in combinatorial problem solving", Working Paper, New York University, May 1985.
52. [T. Rockafellar, 1970], Convex Analysis, Princeton University Press (1970).
53. [E. Stiemke, 1915], "Über positive Lösungen homogener linearer gleichungen," Mathematische Annalen 76, 340-342.
54. [J. Stoer and C. Witzgall, 1970], Convexity and Optimization in Finite Dimensions, Springer, Berlin (1970).
55. [H. Weyl, 1935], "Elementare theorie der konvexen polyeder", Commentarii Mathematici Helvetici 7, 290-306 (1935).
56. [H. Whitney, 1935], "On the abstract properties of linear dependence", American Journal of Mathematics 57, 509-533 (1935).