INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Solving structured 0/1 integer programs arising from truck dispatching scheduling problems

Lee, Eva Kwok-Yin, Ph.D.

Rice University, 1993
RICE UNIVERSITY

SOLVING STRUCTURED 0/1 INTEGER PROGRAMS
ARISING FROM TRUCK DISPATCHING SCHEDULING PROBLEMS

by

EVA K. LEE

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
DOCTOR OF PHILOSOPHY

APPROVED, THESIS COMMITTEE

[Signatures and names of committee members]

Houston, Texas
April, 1993
Abstract

Solving Structured 0/1 Integer Programs
Arising from Truck Dispatching Scheduling Problems

by

Eva K. Lee

A branch-and-cut IP solver is developed for a class of structured 0/1 integer programs arising from a truck dispatching scheduling problem. This problem is characterized by a group of set partitioning constraints and a group of knapsack equality constraints of a specific form. Families of facets for the polytopes associated with individual knapsack constraints are identified, and in some cases, a complete characterization of a polytope is obtained. In addition, a notion of “conflict graph” is introduced and utilized to obtain an approximating node-packing polytope for the convex hull of all 0/1 solutions. The branch-and-cut solver generates cuts based on both the knapsack constraints and the approximating node-packing polytope, and incorporates these cuts into a tree-search algorithm that uses problem reformulation and linear programming-based heuristics at each node in the search tree to assist in the solution process. Numerical experiments are performed on large-scale real instances supplied by Texaco Trading & Transportation, Inc. The optimal schedules obtained correspond to cost savings for the company and greater job satisfaction for drivers due to more balanced work schedules and income distribution. It is noteworthy that this is apparently the first time that branch-and-cut has been applied to an equality constrained problem in which the entries in the constraint matrix and right hand side are not purely 0/1.


Acknowledgements

I wish to express my deepest gratitude to my thesis advisor, Professor Robert Bixby, who introduced me to the fascinating area of linear and combinatorial optimization, and gave me much encouragement and advice during the course of my research. With great patience, he spent time discussing my work with me, reading my research reports, and giving me suggestions for improvements. With his experience in this area, he has given me valuable ideas regarding my implementation; and moreover, he has been an excellent model of what it takes to be a good researcher and a good teacher.

I would like to thank my committee members, Professor Danny Sorensen, Professor Mark Krentel, Professor Andrew Boyd, and Professor Karla Hoffman, for caring about this work and agreeing to serve on the committee. In particular, Professor Hoffman shared with me her rich experience in this area, and pointed out my oversights on earlier drafts of the thesis.

I would also like to thank Donna McDougald of Texaco, Inc. for taking the time to explain to me the truck dispatching system and obtaining test problems for me, and Texaco Trading & Transportation, Inc. for allowing the test problems to be released.

I also wish to single out my gratitude to Professor Richard Tapia, who has given freely of his time and advice during my stay at Rice, and to Dr. Virginia Torczon, for taking the time to listen to my presentations, even when they were not
well-prepared, and giving me advice on how to be a more effective speaker.

I would like to thank my friends for their encouragement and support, especially Jan, Shireen and Wilfred. I have special thanks to my fiancé, Rick, for his love, support and patience. He has given me great power to handle the stress and frustrations. Finally, I wish to thank my family for their love, caring and encouragement.
TO MY PARENTS
# Contents

1. **Introduction** .......................................................... 1

2. **Problem Description and Formulation** .................. 4

3. **Polyhedral Theory for the TDS Polytope** ............ 7
   - 3.1. **Introduction** ............................................... 7
   - 3.2. **Background, Definitions and Notation** ............ 8
   - 3.3. **Facets of Special Knapsack Equality Polytopes** 12
   - 3.4. **The Conflict Graph and Approximating Node Packing Polytope** 30

4. **The Branch-and-Cut IP Solver** .......................... 33
   - 4.1. **Overview** .................................................. 33
   - 4.2. **Preprocessing and Reformulation** ................. 35
   - 4.3. **The Heuristic** ............................................ 39
   - 4.4. **Continuous Reduced Cost Implications** ........... 41
   - 4.5. **Constraint Generation** ................................ 42

5. **Numerical Tests and Results** .......................... 52

6. **Summary and Future Research** ........................ 56

7. **Figure and Tables** ............................................ 58

8. **Bibliography** .................................................. 68
# List of Figure and Tables

<table>
<thead>
<tr>
<th>Figure/Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Branch-and-Cut Flowchart</td>
<td>58</td>
</tr>
<tr>
<td>Table 1</td>
<td>Problem Characteristics</td>
<td>59</td>
</tr>
<tr>
<td>Table 2</td>
<td>Numerical Results When Using Only Branch-and-Bound</td>
<td>60</td>
</tr>
<tr>
<td>Table 3</td>
<td>Numerical Results for Branch-and-Cut</td>
<td>61</td>
</tr>
<tr>
<td>Table 4</td>
<td>Branch-and-Bound vs. Branch-and-Cut</td>
<td>62</td>
</tr>
<tr>
<td>Table 5</td>
<td>Objective Values from First Call to Heuristic</td>
<td>63</td>
</tr>
<tr>
<td>Table 6</td>
<td>Numerical Results When Using Only Preprocessing and Reduced-cost Fixing</td>
<td>64</td>
</tr>
<tr>
<td>Table 7</td>
<td>Numerical Results When Using Only Preprocessing, Heuristic</td>
<td>65</td>
</tr>
<tr>
<td>Table 8</td>
<td>Times for Various Primal Pricing Strategies</td>
<td>66</td>
</tr>
<tr>
<td>Table 9</td>
<td>Times for Various Dual Pricing Strategies</td>
<td>67</td>
</tr>
</tbody>
</table>
Introduction

Structured 0/1 integer programming problems arise naturally from a variety of important problems in government and industry, including scheduling, assignment, matching, and location problems. Some of the most well-studied structured 0/1 problems are the traveling salesman problem, the set covering, set partitioning, and set packing problems, and the knapsack problem. In this work we focus on a 0/1 problem, referred to as the "truck dispatching scheduling (TDS) problem," that has characteristics of both set partitioning and knapsack problems.

The TDS problem is a vehicle routing and scheduling problem that arose from an oil company's desire to better utilize regional truck fleets and drivers. Many studies on different variations of vehicle routing and scheduling — arising from a variety of areas in industry — have appeared in the literature [4,7,8,9,11,14,15,17,18,19,26, 27]. In most of these, the emphasis has been on developing fast heuristic algorithms to find good feasible solutions [7, p. 575]. In contrast, in this work, we use a branch-and-cut algorithm to find exact solutions. The branch-and-cut approach, which integrates branching and cutting-plane techniques, has been shown to be effective in solving important "hard" combinatorial optimization problems. In particular, large-scale instances of the traveling salesman problem and the airline crew scheduling problem have been solved to proven optimality in reasonably short amounts of time [21,22,23,31,32]. In addition, Crowder, Johnson and Padberg [13] and Johnson, Kostreva and Suhl [24] obtained good computational results on
inequality constrained 0/1 problems with no special structure.

In light of these successes, we apply the branch-and-cut approach to the TDS problem. It is important to point out that this is apparently the first time that branch-and-cut has been applied to an equality constrained problem in which the entries in the constraint matrix and right-hand-side are not purely 0/1. A significant feature of this research involves exploiting the special structure of the underlying TDS polytope. Indeed, we arrive at new theoretical results in polyhedral theory that are of interest in their own right; and moreover, the application of these results is shown to be effective in solving a given set of real problem instances supplied by Texaco Trading & Transportation, Inc.

The theoretical results are presented in Chapter 2. In particular, large families of facet-defining inequalities for certain equality constrained knapsack polytopes are identified. These knapsack polytopes are associated with individual constraints of the TDS problem formulation and serve as approximations to the convex hull of 0/1 solutions. It is noteworthy that very little is known about general knapsack equality polytopes. Nevertheless, by studying the special form of the knapsack constraints in the TDS problem, we identify much of the facial structure of the associated polytope. Based on this theory, polyhedral cuts — generated explicitly as facets of individual knapsack equality polytopes — are incorporated into the branch-and-cut system.

Besides considering approximating polytopes associated with individual constraints, we also approximate the TDS polytope by a polytope formed from considering all the constraints simultaneously. In particular, in Section 2.3, we introduce the notion of conflict graph and discuss its polyhedral ramifications. Although the
concept of intersection graph has been utilized frequently in solving 0/1 integer programs involving purely 0/1 constraint matrices, an analogous approach has never been investigated when a non 0/1 constraint matrix is involved. Our results show that such an approach — which allows one to apply the polyhedral theory of node packing polytopes even when problem parameters are not all 0/1 — is effective in solving the given TDS problem instances.

Although the polyhedral theory and its implementation are an integral part of this research, the development and coding of the entire Branch-and-Cut IP Solver involves much more. Indeed, besides the cut generation phase of the algorithm — which generates polyhedral cuts based in part on the above-mentioned results on knapsack equality polytopes and the notion of conflict graph — the IP Solver also includes problem preprocessing and reformulation routines, as well as a heuristic to quickly find good integer feasible solutions. The algorithmic issues related to these three parts and their interface with the LP solver are discussed in Chapter 3.

Chapter 4 contains the numerical results on the given set of problem instances. These problem instances range in size from just over 2000 variables to over a 115,000 variables. The results indicate that there is a substantial improvement in the solution times for the large problems when branch-and-cut is used in place of plain branch-and-bound. Finally, in Chapter 5, some concluding remarks are given, and possible avenues for future research are suggested. We now turn to Chapter 1, where we describe the TDS problem and its formulation as a 0/1 integer program.
Chapter 1

Problem Description and Formulation

An oil company has dispatchers in each of 17 regions in the United States. Each dispatcher is responsible for assigning itineraries to drivers each day. Associated with each itinerary are loads, pickup points, delivery points, deadhead miles, and total hours. Itineraries are generated for each driver or group of drivers (called a driver set) by an automatic process prior to the 0/1 integer programming problem formulation. Drivers are grouped in the same driver set provided they have identical driver parameters, including home base location, hours available, starting time, breathing equipment certification, etc. The automatic process will generate a large number of possible itineraries for each driver set according to the parameters of the driver set, as well as the parameters associated with pickup points, delivery points, and Department of Transportation (DOT) regulations (e.g., open/close times for pickup and delivery points, and maximum on-duty-time allowed by DOT).

The primary objectives of the TDS problem are to minimize deadhead miles (i.e., the miles driven with an empty truck) and equalize the loaded drive-time of the drivers. By minimizing deadhead miles, it is expected that trucks and drivers will be on the road fewer hours each day, thereby lowering fuel and maintenance costs, and improving safety for the drivers. Also, since driver payroll is based on
the amount of time a driver is hauling product, by equalizing loaded drive-time, overall driver satisfaction should be increased. In order to reflect both objectives, to each generated itinerary, a weight is assigned that incorporates the associated number of deadhead miles and a penalty associated with loaded drive-time being above or below a prespecified target level. The company anticipates that even a five percent reduction in deadhead miles could result in a significant savings. In addition, obtaining optimal dispatches via solving the integer programs allows for more consistency in the dispatch creation in different locations, it lowers the training curve for dispatchers, and it enables easier substitution and relocation of dispatchers.

There are two types of constraints in the 0/1 integer programming formulation of the TDS problem. One type arises from the number of times each pickup point must be visited, and the second type arises from the number of drivers in each driver set. The parameters associated with the formulation are as follows:

\[ n = \text{number of itineraries} \]
\[ c_j = \text{weight associated with the } j\text{th itinerary} \]
\[ p = \text{number of pickup points} \]
\[ b_i = \text{number of times pickup point } i \text{ must be visited} \]
\[ a_{ij} = \text{number of times pickup point } i \text{ is visited according to itinerary } j \]
\[ q = \text{number of driver sets} \]
\[ g_k = \text{number of drivers in the } k\text{th driver set} \]
\[ n_k = \text{number of itineraries generated for the } k\text{th driver set}. \]
In matrix form, the problem can be stated as

\[ \text{minimize} \quad c^T x \]

subject to \( Ax = b \) (pickup point constraints) \( (1.1) \)

\[ Hx = g \] (driver set constraints)

\[ x \in B^n, \]

where \( c = (c_1, \ldots, c_n)^T \in R^n, \ c > 0; \ A = [a_{ij}] \) is a \( p \times n \) nonnegative integral matrix; \( b = (b_1, \ldots, b_p)^T \in Z^p, \ b > 0; \ g = (g_1, \ldots, g_q)^T \in Z^q, \ g > 0; \ B = \{0,1\} \)

and \( H \) is a \( q \times n \) matrix of the form

\[ H = \begin{bmatrix}
1^T_{n_1} \\
1^T_{n_2} \\
\vdots \\
1^T_{n_q}
\end{bmatrix}, \]

where \( 1_{n_k} \) is the vector in \( R^{n_k} \) in which every entry is 1, and \( n_1 + \cdots + n_q = n \).

Furthermore, the \( i \)th pickup point constraint \( a_i^T x = b_i, \ 1 \leq i \leq p, \) is of the special form

\[ \sum_{j \in R_{i1}} x_j + \sum_{j \in R_{i2}} 2x_j + \cdots + \sum_{j \in R_{i_m}} b_i x_j = b_i, \]

(1.2)

where each \( b_i \) is a small positive integer and \( R_{im}, \ 1 \leq m \leq b_i, \) are nonempty disjoint subsets of \( \{1, \ldots, n\}. \)
Chapter 2

Polyhedral Theory for the TDS Problem

2.0 Introduction.

Since the effectiveness of a branch-and-cut approach to an integer programming problem depends largely on the ability to generate deep cuts, and this in turn depends on an understanding of the structure of the underlying polytope of the integer program, in this chapter we study the polyhedral structure of the 0/1 integer programming formulation of the TDS problem. First, in Section 2.1, relevant background material on polyhedral theory is given. Then, in Section 2.2, we develop theoretical results related to the polyhedral structure of the special knapsack equality constraints that arise in the TDS problem. Specifically, we state several families of facet defining inequalities for polyhedra of the form

$$ S_b = \text{conv}\{x \in B^n : \sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j + \cdots + \sum_{j \in R_b} bx_j = b\}, \quad (2.0.1) $$

where $b \geq 2$ is an integer, and $R_1, R_2, \ldots, R_b$ are disjoint subsets of $N = \{1, 2, \ldots, n\}$ with $|R_1| \geq b + 1$. Moreover, complete characterizations of $S_2$ and $S_3$ as systems of linear inequalities are given. Finally, in Section 2.3, we introduce the notion of conflict graph, and discuss its implications in solving the TDS problem, as well as other 0/1 integer programming problems in which the parameters in the constraint
matrix are not purely 0/1.

2.1 Background, Definitions and Notation.

In this section we briefly list some important definitions and results related to polyhedral theory. For a more complete discussion of polyhedral theory and its relation to integer programming, the reader is referred to [29].

Definition 2.1.1. A collection of points \( y^1, \ldots, y^k \in \mathbb{R}^n \) is said to be affinely independent if the unique solution of \( \sum_{i=1}^{k} \alpha_i y^i = 0 \), with \( \sum_{i=1}^{k} \alpha_i = 0 \), is \( \alpha_i = 0 \), \( i = 1, \ldots, k \).

Clearly, if \( y^1, \ldots, y^k \in \mathbb{R}^n \) are linearly independent, they are also affinely independent. Moreover, it is straightforward to show that if \( y^1, \ldots, y^k \in \mathbb{R}^n \) are affinely independent and satisfy \( \pi^T x = \pi_0 \), where \( \pi_0 \neq 0 \), then they are also linearly independent. This fact is particularly relevant to the study of the polyhedron \( S_b \), since all of its points satisfy

\[
\sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j + \cdots + \sum_{j \in R_b} bx_j = b. \tag{2.1.1}
\]

Next, we give the definition of the dimension of a polyhedron. Although, there are several equivalent ways of defining the dimension [29, p. 86; 33, p. 12], for our purposes, the following one is most convenient.

Definition 2.1.2. A polyhedron \( P \subset \mathbb{R}^n \) is of dimension \( k \), denoted by \( \text{dim}(P) = k \), if the maximum number of affinely independent points in \( P \) is \( k + 1 \). If \( \text{dim}(P) = n \), then \( P \) is said to be full-dimensional.

An issue of fundamental importance in integer programming is to find a linear
inequality representation of a polyhedron defined as the convex hull of a finite number of points in \( \mathbb{R}^n \). The following definitions are useful in the discussion.

**Definition 2.1.3.** Let \( P \subset \mathbb{R}^n \) be a polyhedron.

(a) An inequality \( \pi^T x \leq \pi_0 \) is said to be a *valid inequality* for \( P \) if it is satisfied by every point in \( P \).

(b) If \( \pi^T x \leq \pi_0 \) is a valid inequality for \( P \), the set \( F := \{ x \in P : \pi^T x = \pi_0 \} \) is called a *face* of \( P \), and the inequality \( \pi^T x \leq \pi_0 \) is said to *define* \( F \).

(c) If \( F \) is a face of \( P \) and \( \text{dim}(F) = \text{dim}(P) - 1 \), then \( F \) is called a *facet* of \( P \). In this case, the inequality defining \( F \) is said to be *facet defining.*

It is well-known that every polyhedron \( P \subseteq \mathbb{R}^n \) can be represented as

\[
P = \{ x \in \mathbb{R}^n : A^{(1)} x = b^{(1)}, \ A^{(2)} x \leq b^{(2)} \},
\]

where \( A^{(1)} \) is an \((n - \text{dim}(P)) \times n\) matrix of full rank, and every inequality in the system \( A^{(2)} x \leq b^{(2)} \) defines a facet of \( P \). Moreover, a valid inequality \( \lambda^T x \leq \lambda_0 \) is facet-defining for \( P \) if, and only if, there exists a row \((a_i^{(2)}, b_i^{(2)})\) of \((A^{(2)}, b^{(2)})\) such that

\[
(\lambda^T, \lambda_0) = \delta(a_i^{(2)}, b_i^{(2)}) + u^T(A^{(1)}, b^{(1)})
\]

for some \( \delta > 0 \), \( u \in \mathbb{R}^{n-\text{dim}(n)} \) [29, p. 91]. In particular, if \( P \) is full-dimensional (i.e., \((A^{(1)}, b^{(1)})\) is vacuous), then each facet has a unique — to within scalar multiplication — defining inequality.

Note that the polyhedron \( S_b \) is contained in the hyperplane defined by equation (2.1.1); hence it is not full-dimensional. However, under appropriate conditions (see Lemma 2.2.2), \( \text{dim}(S_b) = n - 1 \). In this case, if \( a^T x = b \) denotes equation
(2.1.1), and if \( \pi^T x \leq \pi_0 \) defines a facet of \( S_b \), then every inequality of the form 
\[
(\delta \pi + u a)^T x \leq \delta \pi_0 + u b,
\]
where \( \delta > 0, \ u \in \mathbb{R} \), defines the same facet. This fact will be utilized in the proofs of Theorems 2.2.1 – 2.2.3, where the defining inequalities for certain families of facets of \( S_b \) are characterized.

We now give an overview of the “cutting” aspect of a branch-and-cut approach for solving an integer programming problem. For the sake of exposition, we focus on the TDS problem formulation (1.1). Obvious minor adjustments are needed for a discussion on general integer programs. We caution however, that successful applications of branch-and-cut have been limited to problems that are formulated as either pure 0/1 or mixed 0/1 integer programs.

Let \( P^{IP} \) denote the convex hull of integer solutions of problem (1.1), and let \( P^{LP} \) denote the polytope associated with the linear programming relaxation of (1.1); that is,
\[
P^{IP} = \text{conv} \{x \in B^n : Ax = b, \ Hx = g\},
\]
and
\[
P^{LP} = \{x \in \mathbb{R}^n : Ax = b, \ Hx = g, \ 0 \leq x \leq 1\}.
\]
Then \( P^{IP} \subseteq P^{LP} \). Now, let \( \bar{x} \) be a solution of the linear programming relaxation \( \text{minimize} \{c^T x : x \in P^{LP}\} \). If \( \bar{x} \) is integral, then we are done. If not, one attempts to generate one or more valid inequalities \( \pi^T \bar{x} \leq \pi_{i0} \) for \( P^{IP} \) such that \( \pi^T \bar{x} > \pi_{i0} \). One then adds these inequalities to the linear programming relaxation and obtains an updated solution \( \bar{x} \) of the linear program \( \text{minimize} \{c^T x : x \in P^{LP}, \ \pi^T x \leq \pi_{i0}\} \). Again, if \( \bar{x} \) is integral, we are done. If not, the process is repeated, saving only those valid inequalities generated from previous iterations that are active at \( \bar{x} \).
The approach just outlined is most effective when the generated valid inequalities define facets of $P^{IP}$, or at least cut deep into the current relaxation of $P^{IP}$. (The "deepness" of a cut is typically measured by the value $\pi^T \bar{x} - \pi_{i_0}$.) An effective way of generating deep cuts is to generate facet defining inequalities for various approximating polytopes of $P^{IP}$. In this work we make use of approximating knapsack equality polytopes associated with individual constraints of the TDS problem, and an approximating node packing polytope associated with a graph, called the conflict graph, obtained by considering all the constraints simultaneously.

The conflict graph, its associated node packing polytope, and implications, are discussed in Section 2.3. To begin the discussion on the knapsack equality polytopes, we let $P^i$ and $P^H$ denote the approximating polytopes

$$P^i = \text{conv}\{x \in B^n : a_i^T x = b_i\}$$

and

$$P^H = \text{conv}\{x \in B^n : Hx = g\},$$

where $a_i^T x = b_i$ is the $i$th equality in the system $Ax = b$. Then $P^{IP} \subseteq (\cap_{i=1}^p P^i) \cap P^H$. Moreover, if there is not too much overlap in the vectors $a_1, \ldots, a_p$, then it is reasonable to expect that the intersection $(\cap_{i=1}^p P^i) \cap P^H$ provides a fairly good approximation to $P^{IP}$ [13, pp. 809-810]. Since $H$ is totally unimodular and $g$ is integral, every facet of $P^H$ corresponds to an inequality of the form $x_j \geq 0$ or $x_j \leq 1$ for some $j$, $1 \leq j \leq n$. Hence, we may focus on identifying facet defining inequalities for $P^i$, $1 \leq i \leq p$. Note that since the constraint $a_i^T x = b_i$ defining $P^i$ is of the form (2.1.1), $P^i$ is a polytope of the form $S_b$, given in (2.0.1). Also,
note that the inequalities $x_j \geq 0$ and $x_j \leq 1$ determine all of the facets of $S_1$, and therefore, nothing is gained by separately considering the polytopes $P_i$ for those $i$ with $b_i = 1$. We now turn to Section 2.2, where we focus on the polytope $S_b$, $b \geq 2$.

2.2. Facets of Special Knapsack Equality Polytopes.

Consider again the polytope $S_b$:

$$S_b = \text{conv}\{x \in B^n : \sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j + \cdots + \sum_{j \in R_b} bx_j = b\}. \quad (2.2.1)$$

In this section we show that if $|R_1| \geq b + 1$, then $S_b$ has at least $\sum_{k=1}^{b-1} \binom{|R_1|}{k}$ facets — one facet corresponding to each nonempty subset of $R_1$ having cardinality less than or equal to $b - 1$. Moreover, we obtain complete characterizations of $S_2$ and $S_3$ as systems of linear inequalities. Before proceeding, it is convenient to introduce the following definitions and notation.

If $C \subseteq N := \{1, \ldots, n\}, C \neq \emptyset$, define

$$x^C := \sum_{j \in C} e_j,$$

where $e_j$ is the $j$th standard unit vector in $\mathbb{R}^n$. Thus, $x^C$ is the characteristic vector of $C$. Also, if $R \subseteq N$, $T \subseteq R$, $T \neq \emptyset$, $k \in N$, and $u \in \{0, 1\}$, define the sets

$$R(k) := \{x^C \in B^n : C \subseteq R, |C| = k\}$$

and

$$R(k, T, u) := \{x^C \in R(k) : x^C_j = u \text{ for all } j \in T\}.$$
For simplicity, if $T = \{j\}$, then write $R(k, j, u)$ instead of $R(k, \{j\}, u)$.

The first result, Lemma 2.2.1, shows that without loss of generality one may assume that $\bigcup_{i=1}^{k} R_i = N$ in the definition of $S_b$.

**Lemma 2.2.1.** Let $n_1$ and $n_2$ be positive integers, let $a \in \mathbb{R}^{n_1}$, and let $\alpha \in \mathbb{R}$.

Define the sets

$$P_1 = \text{conv}\{x \in B^{n_1} : a^T x = \alpha\}$$

and

$$P_2 = \text{conv}\{(x, y) \in B^{n_1+n_2} : a^T x = \alpha\},$$

and assume that $P_1$ is nonempty. Then the following are true:

(a) $\dim(P_2) = \dim(P_1) + n_2$;

(b) if $\pi^T x \leq \pi_0$ is a facet defining inequality for $P_1$, then it is also a facet defining inequality for $P_2$; and

(c) if $Ax \leq b$, $x \geq 0$, is a linear inequality representation of $P_1$, then $Ax \leq b$, $x \geq 0$, $0 \leq y \leq 1$, is a linear inequality representation of $P_2$. Moreover, each of the inequalities $y_j \geq 0$ and $y_j \leq 1$ defines a facet of $P_2$.

**Proof.** (a) Let $k = \dim(P_1)$, and suppose that $x^1, \ldots, x^{k+1}$ are affinely independent vectors in $P_1$. Then the vectors $(x^i, 0) \in \mathbb{R}^{n_1+n_2}$, $i = 1, \ldots, k+1$, and $(x^1, e_j) \in \mathbb{R}^{n_1+n_2}$, $j = 1, \ldots, n_2$, where $e_j$ is the $j$th standard unit basis vector in $\mathbb{R}^{n_2}$, are $k+n_2+1$ affinely independent vectors in $P_2$. Hence, $\dim(P_2) \geq k+n_2$. To show that $\dim(P_2) = k+n_2$ it suffices to show that any point $(x, y) \in P_2$ can be written as an affine combination of $(x^i, 0)$, $i = 1, \ldots, k+1$, and $(x^1, e_j)$, $j = 1, \ldots, n_2$. 

To this end, let \((x, y) \in P_2\) be given. Then \(x \in P_1\), so \(x\) can be written as an affine combination of \(x^1, \ldots, x^{k+1}\). That is, \(x = \sum_{i=1}^{k+1} \lambda_i x^i\), where \(\lambda_1, \ldots, \lambda_{k+1} \in \mathbb{R}\) satisfy \(\sum_{i=1}^{k+1} \lambda_i = 1\). We also have \(y = \sum_{j=1}^{n_2} y_j e_j\), where \(y_j\) is the \(j\)th component of \(y\). Hence,

\[
(x, y) = \sum_{i=1}^{k+1} \lambda_i (x^i, 0) + \sum_{j=1}^{n_2} y_j (x^1, e_j) - \left(\sum_{j=1}^{n_2} y_j\right) (x^1, 0). \tag{2.2.2}
\]

Since the coefficients on the right side of (2.2.2) sum to 1, \((x, y)\) has been expressed as an affine combination of \((x^i, 0), i = 1, \ldots, k+1\), and \((x^1, e_j), j = 1, \ldots, n_2\).

(b) Suppose \(\pi^T x \leq \pi_0\) is a facet defining inequality for \(P_1\). Then clearly it is a valid inequality for \(P_2\). Moreover, if \(x^1, \ldots, x^k\) are affinely independent vectors in \(P_1\) satisfying \(\pi^T x \leq \pi_0\) with equality, then \((x^i, 0), i = 1, \ldots, k,\) and \((x^1, e_j), j = 1, \ldots, n_2,\) are affinely independent vectors in \(P_2\) satisfying \(\pi^T x \leq \pi_0\) with equality. Hence, \(\pi^T x \leq \pi_0\) defines a facet of \(P_2\).

(c) It is straightforward to show that if \(P_1 = \{x \in \mathbb{R}^n : Ax \leq b, \ x \geq 0\}\), then \(P_2 = \{(x, y) \in \mathbb{R}^{n_1+n_2} : Ax \leq b, \ x \geq 0, \ 0 \leq y \leq 1\}\). To show that the trivial inequalities \(y_j \geq 0\) and \(y_j \leq 1\) define facets, let \(x^1, \ldots, x^{k+1}\) be affinely independent vectors in \(P_1\), and let \(j_0 \in \{1, \ldots, n_2\}\) be fixed. Then the \(k+n_2\) points 
\((x^i, e_{j_0}), i = 1, \ldots, k+1,\) and \((x^1, e_{j_0} + e_j), j \in \{1, \ldots, n_2\} \setminus \{j_0\}\) in \(P_2\) satisfy \(y_{j_0} \leq 1\) with equality. Hence, \(y_{j_0} \leq 1\) defines a facet of \(P_2\). Similarly, the \(k+n_2\) points 
\((x^i, 0), i = 1, \ldots, k,\) and \((x^1, e_j), j \in \{1, \ldots, n_2\} \setminus \{j_0\}\) satisfy \(y_{j_0} \geq 0\) with equality. Hence, \(y_{j_0} \geq 0\) defines a facet of \(P_2\).}

Having established Lemma 2.2.1, we assume, unless otherwise specified, that \(\cup_{i=1}^{b} R_i = N\). It is important to note that in order to determine facets of a polyhedron, we have to be able to first determine the dimension of the polyhedron.
As mentioned before, $S_b$ is not full dimensional. In Lemma 2.2.2, we show that $\dim(S_b) = n - 1$ provided $|R_1| \geq b + 1$. We also give a characterization of valid inequalities for $S_b$, and state results concerning the faces defined by the trivial inequalities.

**Lemma 2.2.2.** Let $b \geq 2$, and let $S_b$ be defined as in (2.2.1), where $\bigcup_{i=1}^b R_i = N$ and $|R_1| \geq b + 1$. Then the following are true:

(a) $\dim(S_b) = n - 1$;

(b) an inequality $\pi^T x \leq \pi_0$ is valid for $S_b$ if, and only if, for every collection $C_i \subset R_i$, $i = 1, \ldots, b$, satisfying $\sum_{i=1}^b i|C_i| = b$, we have

$$\sum_{j \in C_1} \pi_j + \sum_{j \in C_2} \pi_j + \cdots + \sum_{j \in C_b} \pi_j \leq \pi_0;$$

(c) if $j \in N \setminus R_1$, then $x_j \geq 0$ is facet defining;

(d) if $j \in R_1$ and $|R_1| \geq b + 2$, then $x_j \geq 0$ is facet defining; and

(e) if $R_b \neq \emptyset$, then $x_j \leq 1$ is not facet defining for any $j \in N$.

**Proof.** (a) Choose a subset $C$ of $R_1$ of cardinality $b+1$; say $C = \{j_1, \ldots, j_{b+1}\}$. Then the following $n$ points in $S_b$ are affinely independent:

1a) $x^C - e_j$, $j \in C$;

1b) $x^C - e_{j_1} - e_{j_2} + e_j$, $j \in R_1 \setminus C$;

(i) $x^C - \sum_{k=1}^{i+1} e_{j_k} + e_j$, $j \in R_i$, $2 \leq i \leq b - 1$;

(b) $e_j$, $j \in R_b$.

Hence, $\dim(S_b) = n - 1$.

(b) This is obvious since $x \in S_b$ if, and only if, $x = x^C$ for some set $C = \bigcup_{i=1}^b C_i$, where $C_i \subset R_i$ and $\sum_{i=1}^b i|C_i| = b$. 

(c) Let $j_0 \in N \setminus R_1$. Then the points listed in the proof of part (a), except the one indexed by $j_0$, satisfy $x_{j_0} \geq 0$ with equality. Hence, $x_{j_0} \geq 0$ defines a facet of $S_b$.

(d) Let $j_0 \in R_1$ and let $C = \{j_1, \ldots, j_{b+1}\} \subset R_1 \setminus \{j_0\}$. Then the points listed in the proof of part (a), except the one indexed by $j_0$, satisfy $x_{j_0} \geq 0$ with equality. Hence, $x_{j_0} \geq 0$ defines a facet of $S_b$.

(e) Let $j_0 \in N$ and consider the face $F = \{x \in S_b : x_{j_0} = 1\}$. If $j_0 \in R_b$, then $|F| = 1$ and so $F$ is not a facet. So suppose $j_0 \in R_k$ for some $k \leq b-1$. If $k \leq b-k$, then $x_j = 0$ for all $j \in \bigcup_{i=b-k+1}^b R_i$, and so

$$F \subset \{x \in S_b : x_{j_0} = 1 \text{ and } x_j = 0 \text{ for all } j \in \bigcup_{i=b-k+1}^b R_i\}.$$ 

But the latter set has dimension at most $(n-1) - [1 + \sum_{i=b-k+1}^b |R_i|] < n-2$; so in this case, $F$ is not a facet. On the other hand, if $k \geq b-k+1$, then $x_j = 0$ for all $j \in (\bigcup_{i=b-k+1}^b R_i) \setminus \{j_0\}$, and so

$$F \subset \{x \in S_b : x_{j_0} = 1 \text{ and } x_j = 0 \text{ for all } j \in (\bigcup_{i=b-k+1}^b R_i) \setminus \{j_0\}\}$$

$$= \text{conv}\{x \in B^n : \sum_{i=1}^{b-k} \sum_{j \in R_i} ix_j = b-k, \ x_{j_0} = 1, \ x_j = 0 \text{ for all } j \in (\bigcup_{i=b-k+1}^b R_i) \setminus \{j_0\}\}.$$ 

By part (a), this latter set has dimension $\sum_{i=1}^{b-k} |R_i| - 1 = (n - \sum_{i=b-k+1}^b |R_i|) - 1 < n-2$. Hence, $F$ is not a facet in this case either. $\square$

Two remarks concerning Lemma 2.2.2 are in order. First, the condition $|R_1| \geq b + 2$ in part (d) is sufficient for $x_j \geq 0$, $j \in R_1$, to be facet defining, but it is not necessary. For example, the inequalities $x_j \geq 0$, $j = 1, \ldots, 4$, are facet defining for
the set
\[ \text{conv}\{x \in B^6 : x_1 + x_2 + x_3 + x_4 + 2x_5 + 3x_6 = 3\}. \]

Second, the converse of part (e) is also true. In particular, if \( R_b = \emptyset \), then the inequalities \( x_j \leq 1, j \in R_1 \) are facet defining. This fact is a corollary of the following theorem which identifies a facet of \( S_b \) for each \( j \in R_1 \).

**Theorem 2.2.1.** Let \( b \geq 2 \), let \( S_b \) be defined as in (2.2.1), where \( \cup_{i=1}^b R_i = N \) and \( |R_1| \geq b + 1 \), and let \( j_0 \in R_1 \). Then the set
\[
F_{j_0} = \text{conv}\left[ \{x \in S_b \cap B^n : x_{j_0} = 1\} \cup R_b(1) \right]
\]
(2.2.3)
is a facet of \( S_b \), and a valid inequality \( \pi^T x \leq \pi_0 \) defines \( F_{j_0} \) if, and only if, it is of the form
\[
\beta x_{j_0} + \sum_{j \in R_1 \setminus \{j_0\}} \left(\frac{\alpha - \beta}{b - 1}\right) x_j + \sum_{i=2}^{b-1} \sum_{j \in R_i} \left(\frac{\alpha - \beta}{b - 1}\right) x_j + \sum_{j \in R_b} \alpha x_j \leq \alpha, \tag{2.2.4}
\]
where \( \alpha, \beta \in \mathbb{R}, \alpha < b \beta \). In particular, if \( \alpha = \beta = 1 \), the inequality
\[
x_{j_0} + \sum_{j \in R_b} x_j \leq 1 \tag{2.2.5}
\]
defines \( F_{j_0} \).

**Proof.** We first show that inequality (2.2.4) is valid for \( S_b \) and defines the facet \( F_{j_0} \) specified in (2.2.3). To this end, let \( \tilde{a}^T x \leq \alpha \) denote inequality (2.2.4), and let \( x \in S_b \cap B^n \). Then \( x = x^C \) for some set \( C = \cup_{i=1}^b C_i \), where \( C_i \subset R_i, i = 1, \ldots, b \),
and \[ \sum_{i=1}^{b} i |C_i| = b. \] For convenience, let \( p_i = |C_i| \). Then
\[
\hat{a}^T x = \hat{a}^T x^C = \beta x_{j_0} + \sum_{j \in C_1 \setminus \{j_0\}} \frac{\alpha - \beta}{b - 1} + \sum_{i=2}^{b-1} \sum_{j \in C_i} i \left( \frac{\alpha - \beta}{b - 1} \right) + \sum_{j \in C_b} \alpha
\]
\[
= \beta x_{j_0} + \left( \frac{\alpha - \beta}{b - 1} \right) (p_1 - x_{j_0}) + \sum_{i=2}^{b-1} i \left( \frac{\alpha - \beta}{b - 1} \right) p_i + \alpha p_b
\]
\[
= x_{j_0} \left( \beta - \left( \frac{\alpha - \beta}{b - 1} \right) \right) + \left( \frac{\alpha - \beta}{b - 1} \right) (b - b p_b) + \alpha p_b.
\]

If \( x_{j_0} = 1 \), then \( p_b = 0 \), and so \( \hat{a}^T x = \alpha \). If \( p_b = 1 \), then \( x_{j_0} = 0 \), and so \( \hat{a}^T x = \alpha \).

If \( x_{j_0} = 0 \) and \( p_b = 0 \), then, since \( \alpha < b \beta \), it follows that \( \hat{a}^T x < \alpha \). This establishes that \( \hat{a}^T x \leq \alpha \) is valid for \( S_b \) and defines the face \( F_{j_0} \). To show that \( F_{j_0} \) is a facet, select a subset \( D = \{j_0, j_1, \ldots, j_b\} \) of \( R_1 \) (recall \( |R_1| \geq b + 1 \)) and note that the following \( n - 1 \) points in \( F_{j_0} \) are affinely independent:

1. \( x^D - e_j, \ j \in D \setminus \{j_0\} \);
2. \( x^D - e_{j_1} - e_{j_2} + e_j, \ j \in R_1 \setminus D \);
3. \( x^D - \sum_{k=1}^{i+1} e_{j_k} + e_j, \ j \in R_i, \ 2 \leq i \leq b - 1 \); and
4. \( e_j, \ j \in R_b \).

Now suppose \( \pi^T x \leq \pi_0 \) is a valid inequality defining \( F_{j_0} \). Then, since \( F_{j_0} \) is a facet of \( S_b \), there exists \( \delta > 0, u \in \mathbb{R} \) such that
\[
(\pi^T, \pi_0) = \delta (d^T, 1) + u (a^T, b),
\]
where \( (d^T, 1) \) and \( (a^T, b) \) are the vectors of coefficients for inequality (2.2.5) and equation (2.1.1), respectively. Defining \( \alpha := \pi_0 = \delta + ub \) and \( \beta := \delta + u \), it follows that \( \pi^T x \leq \pi_0 \) is of the form (2.2.4). \( \square \)

In the next theorem, we identify a family of facets for every \( k \in \{2, 3, \ldots, b-2\} \).
In particular, given such a $k$, there are $\binom{|R_1|}{k}$ facets of $S_b$ — one facet for each subset of $R_1$ with cardinality $k$.

**Theorem 2.2.2.** Let $b \geq 4$, let $S_b$ be defined as in (2.2.1), where $\bigcup_{i=1}^{b} R_i = N$, $|R_1| \geq b + 1$ and $R_{b-1} \neq \emptyset$, let $k \in \{2, 3, \ldots, b - 2\}$, and let $T \subseteq R_1$ with $|T| = k$. Associated with $T$ is a facet $F_T$ of $S_b$; and a valid inequality $\pi^T x \leq \pi_0$ defines $F_T$ if, and only if, it is of the form

$$\sum_{j \in T} \beta x_j + \sum_{j \in R_1 \setminus T} \left( \frac{\alpha - k \beta}{b - k} \right) x_j + \sum_{i=2}^{b-k} \sum_{j \in R_i} i \left( \frac{\alpha - k \beta}{b - k} \right) x_j + \sum_{i=b-k+1}^{b-1} \sum_{j \in R_i} (\alpha - (b-i)\beta) x_j + \sum_{j \in R_b} \alpha x_j \leq \alpha, \quad (2.2.6)$$

where $\alpha, \beta \in \mathbb{R}$, $\alpha < b\beta$. In particular, if $\alpha = k$ and $\beta = 1$, the inequality

$$\sum_{j \in T} x_j + \sum_{i=b-k+1}^{b-1} \sum_{j \in R_i} (k - b + i)x_j + \sum_{j \in R_b} kx_j \leq k \quad (2.2.7)$$

defines $F_T$.

**Proof.** Let $\hat{a}^T x \leq \alpha$ denote inequality (2.2.6). We first show that this inequality is valid for $S_b$. To this end, let $x \in S_b \cap B^a$. Then $x = x^C$ for some set $C = \bigcup_{i=1}^{b} C_i$, where $C_i \subseteq R_i$, $i = 1, \ldots, b$, and $\sum_{i=1}^{b} i|C_i| = b$. For convenience, let $p_i = |C_i|$ and $r = |T \cap C_1|$. Then

$$\hat{a}^T x = \hat{a}^T x^C = \beta r + \left( \frac{\alpha - k \beta}{b - k} \right) (p_1 - r) + \sum_{i=2}^{b-k} i \left( \frac{\alpha - k \beta}{b - k} \right) p_i$$

$$+ \sum_{i=b-k+1}^{b-1} [(\alpha - (b-i)\beta)p_i] + \alpha p_b.$$

Note that if $p_b = 1$, then $r = p_1 = \cdots = p_{b-1} = 0$, and so $\hat{a}^T x^C = \alpha$. Now, if
\[ p_b = 0, \text{ then} \]
\[
\hat{a}^T x_C = \left( \frac{1}{b - k} \right) \left[ \beta r(b - k) - (\alpha - k\beta)r + (\alpha - k\beta) \sum_{i=1}^{b-k} i p_i \right] 
+ \sum_{i=b-k+1}^{b-1} (b - k)(\alpha - (b - i)\beta)p_i 
\]
\[
= \left( \frac{1}{b - k} \right) \left[ \beta (rb - ks) - (b - k)bt + (b - k)(b - s) \right] + \alpha (-r + s + (b - k)t) 
\]

where \( s = \sum_{i=1}^{b-k} i p_i \) and \( t = \sum_{i=b-k+1}^{b-1} p_i \). Thus,
\[
\hat{a}^T x_C = \left( \frac{1}{b - k} \right) \left[ b\beta ((b - k) + r - s - (b - k)t) + \alpha (-r + s + (b - k)t) \right].
\]

Since \( \alpha < b\beta \), to show that \( \hat{a}^T x_C \leq \alpha \), it suffices to show that \( z := (b - k) + r - s - (b - k)t \leq 0 \). We consider three cases. First, if \( t = 0 \), then \( s = b \) and so \( z = r - k \leq 0 \). Second, if \( t = 1 \), then \( z = r - s \leq 0 \). Finally, if \( t \geq 2 \), then \( z = (b - k)(1 - t) + r - s < 0 \). Hence, \( \hat{a}^T x \leq \alpha \) is valid for \( S_b \).

We next show that there exists \( n - 1 \) affinely independent points in \( S_b \) satisfying \( \hat{a}^T x \leq \alpha \) with equality. In fact, all points in the set
\[
R_1(b, T, 1) \bigcup \left[ \bigcup_{i=1}^{k} (T(i) + R_{b-i}(1)) \right] \bigcup \left[ \bigcup_{i=k+1}^{b-2} (R_1(i, T, 1) + R_{b-i}(1)) \right] \bigcup R_b(1)
\]
are in \( S_b \) and satisfy \( \hat{a}^T x \leq \alpha \) with equality. From this set, it is easy to enumerate \( n - 1 \) affinely independent points. For instance, take \( D \subset R_1 \) satisfying \( |D| = b + 1 \) and \( T \subset D \). Then the following \( n - 1 \) points are affinely independent:

1a) \( x^D - e_j, \ j \in D \setminus T \);

1b) \( x^D - e_{j_1} - e_{j_2} + e_j, \ j \in R_1 \setminus D \), where \( j_1 \) and \( j_2 \) are fixed points in \( D \setminus T \);

1c) \( e_{j_3} + e_j, \ j \in T \), where \( j_3 \) is a fixed point in \( R_{b-1} \);

1i) \( x^U + e_j, \ j \in R_{b-i} \), where \( U \) is a fixed subset of \( T \) with \( |U| = i, 2 \leq i \leq k \);
(i) \( x^U + e_j, \ j \in R_{b-i}, \) where \( U \) is a fixed subset of \( R_1 \) satisfying \( T \subset U \) and
\[ |U| = i, \ k + 1 \leq i \leq b - 2; \]

(b - 1) \( e_{j_4} + e_j, \ j \in R_{b-1} \setminus \{j_3\}, \) where \( j_4 \) is a fixed point in \( T; \) and

(b) \( e_j, \ j \in R_b. \)

Hence, (2.2.6) defines a facet of \( S_b. \)

Finally, suppose \( \pi^T x \leq \pi_0 \) is a valid inequality defining \( F_T. \) Then, since \( F_T \) is a facet, there exists \( \delta > 0, \ u \in \mathbb{R} \) such that

\[ (\pi^T, \pi_0) = \delta (d^T, k) + u(a^T, b), \]

where \((d^T, k)\) and \((a^T, b)\) are the vectors of coefficients for inequality (2.2.7) and equation (2.1.1), respectively. Defining \( \alpha := \pi_0 = \delta k + ub \) and \( \beta := \delta + u, \) it follows that \( \pi^T x \leq \pi_0 \) is of the form (2.2.6). \( \square \)

We remark that the condition \( R_{b-1} \neq \emptyset \) in Theorem 2.2.2 can be replaced by the weaker condition \( \cup_{i=1}^{k-1} R_{b-i} \neq \emptyset. \) To see this, suppose \( R_{b-i_0} \) is nonempty for some \( 1 \leq i_0 \leq k - 1. \) Choose \( j_3 \in R_{b-i_0} \) and choose a subset \( V \) of \( T \) with cardinality \( i_0 + 1. \) Then, in place of the points listed in (1c), use the points \( x^V + e_{j_3} - e_j, \ j \in V; \) and \( x^V + e_{j_3} - e_{j_6} - e_j, \ j \in T \setminus V, \) where \( e_{j_3} \) and \( e_{j_6} \) are fixed elements in \( V. \)

Also, in place of the points listed for \( R_{b-i_0}, \) use \( x^U + e_j, \ j \in R_{b-i_0} \setminus \{j_3\}, \) where \( U \) is an appropriately chosen subset of \( R_1 \) with cardinality \( i_0. \)

So far, in Theorems 2.2.1 and 2.2.2, we have identified a facet corresponding to every nonempty subset of \( R_1 \) having cardinality less than or equal to \( b - 2. \) Next, in Theorem 2.2.3, we show that there is a facet of \( S_b \) corresponding to every subset of \( R_1 \) having cardinality \( b - 1. \) Similar to Theorem 2.2.2, the hypothesis in Theorem
2.2.3 that $R_{b-1} \neq \emptyset$ can be relaxed to $\bigcup_{i=2}^{b-1} R_i \neq \emptyset$. However, for ease of presentation in the proof, we assume the former condition.

**Theorem 2.2.3.** Let $b \geq 2$, let $S_b$ be defined as in (2.2.1), where $\bigcup_{i=1}^{b} R_i = N$, $|R_1| \geq b + 1$, and $R_{b-1} \neq \emptyset$, and let $T \subset R_1$ with $|T| = b - 1$. Then the set

$$F_T = \text{conv} \left[ R_1(b, T, 1) \cup \bigcup_{i=1}^{b-2} (T(i) + R_{b-i}(1)) \cup R_b(1) \right]$$

(2.2.8)

is a facet of $S_b$, and a valid inequality $\pi^T x \leq \pi_0$ defines $F_T$ if, and only if, it is of the form

$$\sum_{j \in T} \beta x_j + \sum_{j \notin R_1 \setminus T} (\alpha - (b-1)\beta) x_j + \sum_{i=2}^{b} \sum_{j \in R_i} (\alpha - (b-i)\beta) x_j \leq \alpha,$$

(2.2.9)

where $\alpha, \beta \in \mathbb{R}$, $\alpha < b\beta$. In particular, if $\alpha = b-1$ and $\beta = 1$, the inequality

$$\sum_{j \in T} x_j + \sum_{i=2}^{b} \sum_{j \in R_i} (i-1) x_j \leq b - 1$$

(2.2.10)

defines $F_T$.

**Proof.** Let $\hat{a}^T x \leq \alpha$ denote the inequality (2.2.9). We first show this inequality is valid for $S_b$. Let $x = x^C \in S_b \cap B^n$, where $C = \bigcup_{i=1}^{b} C_i$, $C_i \subset R_i$, $i = 1, \ldots, b$, and $\sum_{i=1}^{b} i |C_i| = b$, and let $p_i = |C_i|$ and $r = |C_1 \cap T|$. Then

$$\hat{a}^T x^C = \beta r + (\alpha - (b-1)\beta)(p_1 - r) + \sum_{i=2}^{b} (\alpha - (b-i)\beta)p_i$$

$$= \beta r + \alpha(p_1 - r) - (b-1)(p_1 - r)\beta + \sum_{i=2}^{b} \alpha p_i - \sum_{i=2}^{b} (b-i)\beta p_i$$

$$= \beta[br - bp_1 + p_1 - \sum_{i=2}^{b} p_i + \sum_{i=2}^{b} ip_i] + \alpha[p_1 - r + \sum_{i=2}^{b} p_i]$$

$$= \beta b[r - t + 1] + \alpha[t - r],$$

(2.2.11)
where \( t = \sum_{i=1}^{b} p_i \). Notice that if \( p_1 = b \), then \( t = b \), and so, since \( r = |T \cap C_1| \leq |T| = b - 1 \), we get \( t - r = b - r \geq 1 \). Also, if \( p_1 \leq b - 1 \), then, since \( r \leq |C_1| = p_1 \), we get \( t - r \geq \sum_{i=2}^{b-1} p_i \geq 1 \). Hence, \( 1 + r - t \leq 0 \) for all \( x \in S_b \cap B^n \). Therefore, since \( \alpha < b \beta \), equation (2.2.11) implies \( \hat{\alpha}^T x^C \leq \alpha \). Hence, \( \hat{\alpha}^T x \leq \alpha \) is valid for \( S_b \).

Now let
\[
V = R_1(b, T, 1) \cup \bigcup_{i=1}^{b-2} (T(i) + R_{b-i}(1)) \cup R_b(1),
\]
and note that every point in \( V \) satisfies \( \hat{\alpha}^T x \leq \alpha \) with equality. Furthermore, note that the following \( n - 1 \) points in \( V \) are affinely independent:

1. \( x^T + e_j, \ j \in R_1 \setminus T \);
2. \( e_{j_1} + e_j, \ j \in T \), where \( j_1 \) is a fixed point in \( R_{b-1} \);
3. \( x^{D_i} + e_j, \ j \in R_{b-i} \), where \( D_i \) is a fixed subset of \( T \) with \( |D_i| = i, 2 \leq i \leq b - 2 \);
4. \( e_{j_2} + e_j, \ j \in R_{b-1} \setminus \{j_1\} \), where \( j_2 \) is a fixed point in \( T \); and
5. \( e_j, \ j \in R_b \).

This establishes that \( \hat{\alpha}^T x \leq \alpha \) defines a facet containing \( F_T = \text{conv}(V) \).

We next show that the facet is precisely \( F_T \). To this end, let \( x \in S_b \cap B^n \), let \( C, C_i, p_i, i = 1, \ldots, b \), \( t \) and \( r \) be as before, and suppose \( \hat{\alpha}^T x = \hat{\alpha}^T x^C < \alpha \). Then clearly \( x \notin R_b(1) \). Moreover, note that equation (2.2.11) implies \( t - r > 1 \). We consider two cases. First, if \( p_1 = b \), then \( x \in R_1(b) \) and \( b - r = t - r > 1 \). But since \( x \in R_1(b, T, 1) \), if, and only if, \( r = b - 1 \), it follows that \( x \notin R_1(b, T, 1) \). Hence, \( x \notin V \) when \( p_1 = b \). Second, consider the case when \( p_1 \leq b - 1 \). Then clearly \( x \notin R_1(b) \); and therefore, \( x \notin R_1(b, T, 1) \). Also, since \( t - r \geq 2 \), it follows that \( x \notin T(i) + R_{b-i}(1) \) for any \( i = 1, \ldots, b - 2 \). So, \( x \notin V \) in this case either. Hence, if
$\hat{a}^T x < \alpha$, then $x \notin V$. It follows that the facet defined by $\hat{a}^T x \leq \alpha$ is precisely $F_T$.

Finally, suppose $\pi^T x \leq \pi_0$ is a valid inequality defining $F_T$. Then, since $F_T$ is a facet, there exists $\delta > 0$, $u \in \mathbb{R}$ such that

$$(\pi^T, \pi_0) = \delta(d^T, b - 1) + u(a^T, b),$$

where $(d^T, b - 1)$ and $(a^T, b)$ are the vectors of coefficients for inequality (2.2.10) and equation (2.1.1), respectively. Defining $\alpha := \pi_0 = \delta(b - 1) + ub$ and $\beta := \delta + u$, it follows that $\pi^T x \leq \pi_0$ is of the form (2.2.9). \qed

As an immediate corollary of either Theorem 2.2.1 or Theorem 2.2.3, we obtain the following result concerning facets of $S_2$.

**Corollary 2.2.1.** If $|R_1| \geq 3$, then for every $j_0 \in R_1$, the following is a facet defining inequality for $S_2$:

$$x_{j_0} + \sum_{j \in R_2} x_j \leq 1. \quad (2.2.12)$$

Moreover, this facet can be alternatively described as

$$F_{j_0} = \text{conv}[\{e_j : j \in R_2\} \cup \{e_{j_0} + e_j : j \in R_1 \setminus \{j_0\}\}].$$

In Theorem 2.2.4 it is shown that the inequalities (2.2.12), together with the trivial inequalities $0 \leq x \leq 1$, give a complete characterization of $S_2$. First, we state and prove the following lemma.

**Lemma 2.2.3.** Let $S = \text{conv}\{x \in \mathbb{R}^n : a^T x = \alpha\}$, and let $P = \{x \in \mathbb{R}^n : a^T x = \alpha, \ Ax \leq b\}$. If $S \subseteq P \subseteq [0,1]^n$ and $P$ is integral, then $S = P$.

**Proof.** If $x$ is an extreme point of $P$, then since $P$ is integral, $x \in \mathbb{R}^n$. Thus,
since $a^T x = \alpha$, $x$ is also an extreme point of $S$. It follows that $P \subset S$; hence $P = S$.

\[ \Box \]

**Theorem 2.2.4.** Let $S_2 = \text{conv}\{x \in B^n : \sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j = 2\}$, where $R_1$ and $R_2$ are nonempty disjoint subsets of $N$ with $|R_1| \geq 3$, and let $Q = N \setminus (R_1 \cup R_2)$. Then $S_2$ has the following representation:

\[
\begin{align*}
\sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j &= 2 \quad \text{(2.2.13)} \\
x_{j_0} + \sum_{j \in R_2} x_j &\leq 1, \quad j_0 \in R_1 \quad \text{(2.2.14)} \\
x_j &\geq 0, \quad j \in N \quad \text{(2.2.15)} \\
x_j &\leq 1, \quad j \in Q. \quad \text{(2.2.16)}
\end{align*}
\]

**Proof.** By Lemma 2.2.1 we assume, without loss of generality, that $Q = \emptyset$. Let $P$ denote the polyhedron defined by (2.2.13) – (2.2.15), and note that $S_2 \subset P \subset [0,1]^n$. Hence, by Lemma 2.2.3 it suffices to show that $P$ is integral. To this end, consider the $(1 + |R_1|) \times (|R_1| + |R_2| + |R_1|)$ constraint matrix obtained from (2.2.13) – (2.2.15) by adding slack variables:

\[
\begin{bmatrix}
1^T & 2^T & 0 \\
I_{R_1} & E & I_{R_1}
\end{bmatrix},
\]

where $E$ is the $|R_1| \times |R_2|$ matrix in which every entry is 1. Now let $B$ be any $(1 + |R_1|) \times (1 + |R_1|)$ nonsingular submatrix. Then $B$ can have at most one column associated with a variable $x_k, k \in R_2$. Moreover, if $B$ has such a column, then the associated basic feasible solution is $x_k = 1$ and $x_j = 0$ for all $j \in N \setminus \{k\}$. If $B$ does not have such a column then $B$ is totally unimodular, and so the associated
solution is integral. □

We now turn our attention to \(S_3\). Corollary 2.2.2 is immediate from Theorems 2.2.1 and 2.2.3.

**Corollary 2.2.2.** If \(|R_1| \geq 4\) and \(R_2 \neq \emptyset\), then for every \(j_0 \in R_1\), and for every \(T \subset R_1\) with \(|T| = 2\), the following are facet defining inequalities of \(S_3\):

\[
\begin{align*}
x_{j_0} + \sum_{j \in R_1} x_j & \leq 1, \quad (2.2.17) \\
\sum_{j \in T} x_j + \sum_{j \in R_2} x_j + \sum_{j \in R_3} 2x_j & \leq 2. \quad (2.2.18)
\end{align*}
\]

Moreover, these facets can be alternatively described as

\[
\begin{align*}
F_{j_0} &= \text{conv} [R_1(3, j_0, 1) \cup (\{e_{j_0}\} + R_2(1)) \cup R_3(1)], \\
F_T &= \text{conv} [R_1(3, T, 1) \cup (T(1) + R_2(1)) \cup R_3(1)],
\end{align*}
\]

respectively.

The inequalities (2.2.17) and (2.2.18) do not give all the nontrivial facets of \(S_3\). In particular, the inequality

\[
\sum_{j \in R_2} x_j + \sum_{j \in R_3} x_j \leq 1 \quad (2.2.19)
\]

defines another nontrivial facet, alternatively described as \(\text{conv} [(R_1(1) + R_2(1)) \cup R_3(1)]\). For the sake of completeness, we mention that this facet is represented by every inequality of the form

\[
\sum_{j \in R_1} \beta x_j + \sum_{j \in R_2} (\alpha - \beta)x_j + \sum_{j \in R_3} \alpha x_j \leq \alpha, \quad (2.2.20)
\]
where $\alpha > 3\beta$. In Theorem 2.2.5 it is shown that the inequalities (2.2.17) – (2.2.19), together with the trivial inequalities $0 \leq x \leq 1$, give a complete representation of $S_3$.

Theorem 2.2.5. Let

$$S_3 = \text{conv}\{x \in B^n : \sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j + \sum_{j \in R_3} 3x_j = 3\},$$

where $R_1$, $R_2$ and $R_3$ are nonempty disjoint subsets of $N$ with $|R_1| \geq 4$, and let $Q = N \setminus (R_1 \cup R_2 \cup R_3)$. Then $S_3$ has the following representation:

$$\sum_{j \in R_1} x_j + \sum_{j \in R_2} 2x_j + \sum_{j \in R_3} 3x_j = 3 \quad (2.2.21)$$

$$x_{j_0} + \sum_{j \in R_1} x_j \leq 1, \quad j_0 \in R_1 \quad (2.2.22)$$

$$\sum_{j \in T} x_j + \sum_{j \in R_2} x_j + \sum_{j \in R_3} 2x_j \leq 2, \quad T \subset R_1, \ |T| = 2 \quad (2.2.23)$$

$$\sum_{j \in R_2} x_j + \sum_{j \in R_3} x_j \leq 1 \quad (2.2.24)$$

$$x_j \geq 0, \ j \in N \quad (2.2.25)$$

$$x_j \leq 1, \ j \in Q. \quad (2.2.26)$$

Proof. By Lemma 2.2.1 we assume, without loss of generality, that $Q = \emptyset$. Let $P$ denote the polyhedron defined by (2.2.21) – (2.2.25). Then $S_3 \subset P \subset [0, 1]^n$, and so, by Lemma 2.2.3, it suffices to show that $P$ is integral. To this end, let $B$ be a basis of the system of equalities obtained by adding slack variables. Then $B$ can have at most one column associated with a variable $x_k$, $k \in R_3$. Moreover, if $B$ has such a column, then the associated basic solution is $x_k = 1$ and $x_j = 0$ for
all \( j \in N \setminus \{k\} \). Now assume that \( B \) does not have a column from \( R_3 \), and note that \( B \) can have at most one column associated with a variable \( x_k, k \in R_2 \). The remainder of the proof is broken down into three cases. In case 1 it is shown that if \( x_k, k \in R_2 \), is a basic variable, then \( x_k \) is not fractional. In case 2 it is shown that if \( x_k, k \in R_2 \), is basic and \( x_k = 1 \), then the basic solution is integral. Finally, in case 3 it is shown that if the basic solution associated with \( B \) has \( x_j = 0 \) for all \( j \in R_2 \cup R_3 \), then the solution is integral. Throughout the proof, let \( x \) denote the solution corresponding to \( B \), and let

\[
J = \{ j \in R_1 : x_j > 0 \} \quad \text{and} \quad J_1 = \{ j \in J : x_j = 1 \}.
\]

**Case 1.** Suppose \( 0 < x_k < 1 \) for some \( k \in R_2 \). Then \( 0 < 2x_k < 2 \), and so from (2.2.21), it follows that \( |J| \geq 2 \). Also, constraints (2.2.23) imply that \( |J_1| \leq 1 \). Thus, from the \( 2 + (\binom{|R_1|}{2}) + |R_1| \) basic variables, the following \( 2 + |J| + (|R_1| - |J_1|) \) variables are positive: \( x_k \) and the slack for (2.2.24); \( x_j, j \in J \) and \( |R_1| - |J_1| \) slacks for (2.2.22). Hence, from the remaining slack variables (i.e., those associated with (2.2.23)) at most \( (\binom{|R_1|}{2}) - (|J| - |J_1|) \) can be positive (i.e., basic). This means at least \( |J| - |J_1| \) slacks for (2.2.23) must be zero.

Let \( s = |J| - |J_1| \), and let \( T_1, \ldots, T_s \) be distinct index sets for constraints (2.2.23) with zero slack. Clearly, \( T_i \subset J \) for all \( i = 1, \ldots, s \). Moreover, there exists \( j_0 \in J \) such that for each \( i = 1, \ldots, s \), \( T_i = \{ j_0, j \} \) for some \( j \in J \setminus \{ j_0 \} \). To see this, suppose, without loss of generality, that \( T_1 \) and \( T_2 \) are disjoint. Then \( \sum_{j \in T_1 \cup T_2} x_j + 2x_k = 4 \), which violates (2.2.21). Hence, the sets \( T_1, \ldots, T_s \) have the form \( \{j_0, j_1\}, \ldots, \{j_0, j_s\} \), respectively, where \( \{j_0, \ldots, j_s\} \subset J \). Since there are only \( |J| - 1 \) sets of this form and since at least \( |J| - |J_1| \) slacks for (2.2.23) must be
zero, it follows that $|J_1| \neq 0$. Thus, $|J_1| = 1$, and exactly $|J| - 1$ slacks for (2.2.23) are zero. Hence, the basic variables are completely determined: $x_k$ and the slack for (2.2.24); $x_j$, $j \in J$; $|R_1| - 1$ slacks for (2.2.22); and $\binom{|R_1|}{2} - (|J| - 1)$ slacks for (2.2.23). To get a contradiction, we show the square matrix determined by these variables is singular.

First, note that the $|J| - 1$ equations for (2.2.23) that have zero slack together with the fact that $|J_1| = 1$, imply that $x_{j_0} = 1$ and $x_j = 1 - x_k$ for all $j \in J \setminus \{j_0\}$. Thus, equation (2.2.21) implies $1 + (|J| - 1)(1 - x_k) + 2x_k = 3$; that is, $|J| = 3$. Say, $J = \{j_0, j_1, j_2\}$, and note that the following four rows of the square matrix determined by the basic variables are linearly dependent: the row corresponding to (2.2.21); the row corresponding to the index $j_0$ in (2.2.22); and the two rows corresponding to the index sets $\{j_0, j_1\}$ and $\{j_0, j_2\}$ in (2.2.23). This contradiction establishes that case 1 is not possible.

**Case 2.** Suppose $x_k = 1$ for some $k \in R_2$. Then (2.2.21) implies $\sum_{j \in J} x_j = 1$. It must be shown that $|J| = 1$. For the sake of contradiction, suppose $|J| \geq 2$. Then the following $1 + |J| + |R_1|$ variables are positive: $x_k$; $x_j$, $j \in J$; and the $|R_1|$ slacks for (2.2.22). Hence, at most $\binom{|R_1|}{2} - (|J| - 1)$ slacks for (2.2.23) can be positive, and consequently, at least $|J| - 1$ slacks for (2.2.23) must be zero. Let $T$ be an index set for a constraint from (2.2.23) with zero slack. Then $\sum_{j \in T} x_j + x_k = 2$. So, since $x_k = 1$, we have $\sum_{j \in T} x_j = 1$. But this implies that $T = J$. Thus, $|J| = 2$ and exactly one constraint from (2.2.23) has zero slack. Hence, the basic variables are completely determined: $x_k$; $x_j$, $j \in J$; $|R_1|$ slacks for (2.2.22); and $\binom{|R_1|}{2} - 1$ slacks for (2.2.23). But the square matrix determined by these variables is singular since the two rows corresponding to (2.2.21) and (2.2.24) and the row corresponding to
the index set \( T = J \) in (2.2.23) are linearly dependent. This contradiction implies that we must have \( |J| = 1 \).

Case 3. Suppose \( x_j = 0 \) for all \( j \in R_2 \cup R_3 \). It suffices to show \( |J_1| = 3 \).

For the sake of contradiction, suppose \( |J_1| \leq 2 \). Then \( |J| \geq 4 \) and the following \(|J| + (|R_1| - |J_1|) + 1 \) variables are positive: \( x_j, j \in J \); \(|R_1| - |J_1| \) slacks for (2.2.22); and the slack for (2.2.24). It follows that at least \(|J| - |J_1| - 1 \) slacks for (2.2.23) must be zero. But since \( |J_1| \leq 2 \), this means that at least \(|J| - 3 \) slacks for (2.2.23) must be zero. Let \( T \) be an index set for a constraint from (2.2.23) with zero slack. Then \( T \subset J_1 \), and consequently, \(|J_1| = 2 \) and \(|J| = 4 \). So, once again, the basic variables are completely determined: \( x_j, j \in J \); \(|R_1| - 2 \) slacks for (2.2.22); \( (1^{|R_1|}/2) - 1 \) slacks for (2.2.23); and the slack for (2.2.24). But the associated square submatrix is singular since the row corresponding to (2.2.21), the two rows from (2.2.22) corresponding to the two indices in \( J_1 \), and the row from (2.2.23) corresponding to the index set \( J_1 \) are linearly dependent. This contradiction implies that we must have \(|J_1| = 3 \).  

\[
\square
\]

2.3. The Conflict Graph and Approximating Node

Packing Polytope.

For convenience of notation, let \( A = \begin{pmatrix} A \\ H \end{pmatrix} \) and \( \hat{b} = \begin{pmatrix} b \\ g \end{pmatrix} \), where \( A, H, b \) and \( g \) are defined as in Chapter 1. Then the TDS polytope \( P^{IP} \) can be expressed as \( P^{IP} = \text{conv}\{x \in B^n : Ax = \hat{b}\} \). Now consider the relaxation

\[
P^{\leq} = \text{conv}\{x \in B^n : Ax \leq \hat{b}\}
\]

of \( P^{IP} \), and note that \( P^{IP} \subseteq P^{\leq} \). If \( A \) was a 0/1 matrix and \( \hat{b} = 1 \), then \( P^{\leq} \) would
just be a set packing polytope, which could be transformed into an equivalent node packing polytope. One could then generate valid inequalities for \( P^\leq \) using the techniques applicable to node packing problems [10,30,34,36]. However, for the TDS problem, the entries in \( A \) and \( \hat{b} \) are not purely 0/1.

In order to utilize some of the results associated with node packing problems, we introduce the notion of a conflict graph, which allows us to approximate \( P^\leq \) by a node packing polytope. The importance of this idea is that it provides an avenue for applying well-known results in polyhedral theory to 0/1 problems having non 0/1 data — not just for the TDS problem. (Two weeks before this writing, the author learned that E.L. Johnson and G.L. Nemhauser [25] proposed obtaining clique inequalities based on ideas similar to those below.)

**Definition 2.3.1.** The conflict graph associated with the system \( Ax \leq \hat{b}, \; x \in B^n \) is the graph \( G_C = (N, E) \), where

\[
N = \{1, 2, \ldots, n\} \quad \text{and} \quad E = \{(j, k) : j, k \in N \text{ and } a_j + a_k \nleq \hat{b}\},
\]

where \( a_j \) and \( a_k \) denote distinct columns in \( A \).

Clearly, if \((j, k) \in E\), then \( x_j + x_k \leq 1 \) is a valid inequality for \( P^{IP} \). Hence, if \( A_C \) denotes the edge-node incidence matrix of \( G_C \), then the node packing polytope

\[
P^C = \text{conv}\{x \in B^n : A_C x \leq 1\}
\]

satisfies \( P^\leq \subseteq P^C \). Consequently, \( P^{IP} \subseteq P^C \), and one can generate valid inequalities for \( P^{IP} \) by applying results from the theory of node packing polytopes. We state two particularly relevant results below.
Recall that a *clique* of a graph is a maximal complete subgraph, and a *chordless odd cycle* is a cycle with an odd number of nodes such that no two nonconsecutive nodes in the cycle are joined by an edge.

**Theorem 2.3.1.** (Padberg [30]) Let $K \subseteq N$. Then the inequality

$$\sum_{j \in K} x_j \leq 1 \quad (2.3.1)$$

defines a facet of $P^C$ if, and only if, $K$ is the node set of a clique in $G_C$.

**Theorem 2.3.2.** (Padberg [30]) Let $K \subseteq N$ be the node set of a chordless odd cycle of $G_C$. Then the inequality

$$\sum_{j \in K} x_j \leq (|K| - 1)/2 \quad (2.3.2)$$

is a valid inequality for $P^C$.

The inequality (2.3.1) is called a *clique inequality*, and the inequality (2.3.2) is called an *odd cycle inequality*. In general, an odd cycle inequality does not define a facet. However, one can "lift" this inequality to obtain a facet of $P^C$. (The notion of lifting is discussed in Section 3.4.)
Chapter 3

The Branch-and-Cut IP Solver

3.0 Overview.

The Branch-and-Cut IP Solver is designed to solve to optimality the real instances of the TDS problem. In this section we give a brief overview of the entire system. Then, in Sections 3.1 – 3.4 we describe each component of the system in more detail.

The first task of the system is to preprocess the user-supplied linear programming formulation. Here, preprocessing refers to reformulating and tightening the constraint matrix by removing redundant rows and columns, and fixing variables if possible. The preprocessing techniques are not only applied to the initial linear programming formulation, but are used throughout the entire branch-and-cut tree to reduce the size of subsequent linear programs and eliminate redundancy among rows and columns.

After initial preprocessing, we solve the linear program relaxation and obtain a lower bound for the original problem. Next, the heuristic is called in an attempt to obtain an integral upper bound. If the heuristic succeeds, we then perform reduced-cost fixing using the reduced costs from the linear programming relaxation, the lower bound, and the upper bound. If enough variables are fixed, we eliminate
columns that are fixed to zero and one, and preprocess the problem again. If the heuristic fails, we either branch or proceed to the constraint generator as described below.

Once the reduced-cost fixing is done, the linear program is solved again. If the solution is not integral, the constraint generator is called. This produces polyhedral cuts that slice off the current fractional solution and tighten the linear representation of the feasible solution set. The particular polyhedral cuts that we generate are knapsack cuts (as described in Section 2.2), clique inequalities and odd cycle inequalities. The latter two types of cuts are generated over a partial conflict graph. They are then lifted to include other nodes in the graph. It should be noted that, unlike the traditional cutting planes, these cuts are valid across the entire tree.

The cuts obtained are then appended to the original constraint matrix, and the linear program is solved again. We iterate through this process until one of the following occurs: (i) an integral solution is obtained, (ii) the linear program is infeasible, (iii) no more cuts are generated, (iv) the number of cuts generated exceeds a prespecified value, or (v) the objective value of the linear programs have not improved much. If (i) occurs at the root, then we obtain an optimal integral solution to our problem. If it occurs at one of the leaves, this node is fathomed, and the next node is chosen based on the best node strategy (i.e., we select the node with the smallest LP value among all active nodes). However, we move directly to this next node only if the integral objective value is greater than that of the current integral upper bound. Otherwise, we update the integral upper bound, return to the root node, resolve the linear program with the current cuts appended, and perform reduced cost fixing. Only after all this is completed do we move to the node in
the tree with the smallest LP value; at which point we reconstruct this node by resolving the associated linear program, again with the current cuts appended and the current variables fixed.

If (ii) occurs at the root, the original integer program is infeasible. If it occurs at one of the leaves, the node is fathomed, and we continue our search with other nodes. If (iii), (iv) or (v) occur, we perform the heuristic again. If the heuristic succeeds, we proceed as specified in the case for (i). If the heuristic fails, we select a variable to branch on and create two new nodes. A branching variable is selected only if its current value is between $1/2 - \delta$ and $1/2 + \delta$, where $\delta$ changes dynamically within the system. Condition (iv) is used in our system as one of the criteria to move on within the system since the gap between the linear program relaxation and the optimal integral solution in several of the test problems is very close (see Table 1). As a result, it is not always easy to detect the “tailing off” of the procedure as in (v). A flow chart of the entire system is given in Figure 1.

The LP solver used in the system is CPLEX [5], certain features of which are exploited when solving the successive linear programs arising in the branch-and-cut process. The choice of parameter selection in CPLEX and a comparison of results for various choices are presented and discussed in Chapter 4.

3.1 Preprocessing and Reformulation.

Preprocessing techniques fix variables permanently, remove redundant rows, and check for inconsistencies among the constraints. It has been shown that it is an effective and computationally inexpensive way to reformulate and improve
a linear program representation, that it complements the generation of polyhedral cuts, and significantly improves the computational time when used repeatedly with constraint generation and heuristics during the course of branch-and-bound [13,22,23,32]. Some of the features in our preprocessor are commonly used in the literature of branch-and-cut, including the removal of duplicate and redundant rows. In addition, we extend the row inclusion and the clique fixing techniques for set partitioning problems [23] to our integral matrix, and we develop some new techniques applicable to the TDS problem. A brief description of each of the techniques is given below.

Row Inclusion Fixing. This procedure searches for rows that are properly contained in another row. To be precise, let \( J \) and \( K \) be the sets of column indices for the nonzero entries of two distinct rows, say rows 1 and 2, respectively. If \( J \subseteq K \), if \( a_{1j} = a_{2j} \) for all \( j \in J \), and if both rows have the same right-hand-side value, then we can conclude that \( x_k = 0 \) for all \( k \in K \setminus J \) for every feasible integral solution. Row 2 can then be removed.

Row Multiple-Inclusion Fixing. This procedure searches for rows that are properly contained in a multiple of another row; i.e., an appropriate multiple of one row is an "inclusion" of the other. Again, let \( J \) and \( K \) be the sets of column indices for the nonzero entries in rows 1 and 2, respectively, and suppose \( J \subseteq K \). Thus we have the constraints

\[
\sum_{j \in J} a_{1j} x_j = b_1 \tag{3.1.1}
\]

\[
\sum_{j \in J} a_{2j} x_j + \sum_{j \in K \setminus J} a_{2j} x_j = b_2. \tag{3.1.2}
\]
Furthermore, suppose \( a_{ij} \geq 0 \) for all \( i \) and \( j \), and suppose \( a_{1j} \leq (b_1/b_2)a_{2j} \) for all \( j \in J \). Then \( x_j = 0 \) for all \( j \in K \setminus J \cup \{ j \in J : a_{1j} < (b_1/b_2)a_{2j} \} \). In this case, row 2 can be removed and row 1 can be updated.

*Removal of Duplicates and Multiples of Rows.* This procedure removes redundant rows that are duplicates or multiples of other rows.

*Creating a Clique Matrix.* Let \( G_C \) be the conflict graph (see Section 2.3) associated with the integral matrix \( A := (a_{ij}) \). Choose a row \( i \) with right-hand-side 1, and let \( M_i = \{ j \in N : a_{ij} = 1 \} \). If there exists a complete subgraph \( K \) of \( G_C \) such that \( M_i \subseteq K \), then since \( \sum_{j \in K} x_j \leq 1 \) and \( \sum_{j \in M_i} x_j = 1 \), it follows that \( x_j = 0 \) for all \( j \in K \setminus M_i \). To determine the set \( K \) of columns that forms a complete subgraph with \( M_i \), we scan each column \( j \in N \setminus M_i \) to check if it has an edge to every column in \( M_i \) in the graph \( G_C \). If so, we can fix \( x_j \) to zero.

*Subsequent Conflict-Fixing of Variables.* If some variable \( x_k \) has been fixed to one, then conflict-fixing will in turn fix variable \( x_j \) to zero if \( a^k + a^j \notin \hat{b} \).

Before discussing the last two preprocessing techniques, it is convenient to "partition" the problem according to the driver sets as follows:

\[
A = [A^{(1)} : \cdots : A^{(q)}], \quad \text{where } A^{(k)} \text{ is } p \times n_k,
\]

\[
c = [c^{(1)} : \cdots : c^{(q)}], \quad \text{where } c^{(k)} \in \mathbb{R}^{n_k}, \quad \text{and}
\]

\[
x = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(q)} \end{bmatrix}, \quad \text{where } x^{(k)} \in \{0,1\}^{n_k}.
\]
Then problem (1.1) can be written as

\[
\text{Minimize } \sum_{k=1}^{q} c^{(k)} x^{(k)}
\]

subject to \[
\sum_{k=1}^{q} A^{(k)} x^{(k)} = b
\]

\[
1_{n_k}^T x^{(k)} = g_k, \quad k = 1, \ldots, q
\]

\[
x^{(k)} \in \{0, 1\}^{n_k}, \quad k = 1, \ldots, q.
\]

Note that the individual components of \(c^{(k)}\), \(A^{(k)}\) and \(x^{(k)}\) have the following interpretations:

\[c_j^{(k)} = \text{cost associated with the } j \text{th itinerary generated for driver set } k,\]

\[a_{ij}^{(k)} = \text{number of times pickup point } i \text{ is visited according to the } j \text{th itinerary generated for driver set } k, \text{ and}\]

\[x_j^{(k)} = 1 \text{ if the } j \text{th itinerary generated for driver set } k \text{ is chosen, } 0 \text{ otherwise.}\]

Although the following technique is stated in the language of the TDS problem, it is applicable to any system of the form

\[Ax = b\]

\[Hx = g\]

\[x \in B^n,\]

where \(A\) is an integral matrix and \(H\) is made up of disjoint rows with all nonzero entries equal to 1.
Fixing variables associated with itineraries generated for the kth driver set. The relevant equations and inequalities are:

\[ 1^T_{n_k} x^{(k)} = g_k \]  \hspace{1cm} (3.1.3)

\[ A^{(k)} x^{(k)} \leq b. \]  \hspace{1cm} (3.1.4)

For each row \( A_i^{(k)} x^{(k)} \leq b_i \) of (3.1.4), find the sum \( s_{ik} \) of the \( g_k - 1 \) smallest coefficients in \( A_i^{(k)} = [a_{i1}^{(k)} \ldots a_{in_k}^{(k)}] \). Also, find the largest coefficient, say \( a_{ij_0}^{(k)} \) in row \( A_i^{(k)} \). If

\[ s_{ik} + a_{ij_0}^{(k)} > b_i, \]  \hspace{1cm} (3.1.5)

then \( x_{j_0}^{(k)} = 0 \). Furthermore, if (3.1.5) holds, one may continue this process by finding the next largest coefficient, say \( a_{ij_1}^{(k)} \) in row \( A_i^{(k)} \). Again, if

\[ s_{ik} + a_{ij_1}^{(k)} > b_i, \]

then \( x_{j_1}^{(k)} = 0 \). Etc.

Test for blatant infeasibility. Let \( \hat{s}_{ik} \) denote the sum of the \( g_k \) smallest coefficients in \( A_i^{(k)} \). If \( \hat{s}_{ik} > b_i \), then the problem is infeasible.

The preprocessor we have implemented includes all but the last two techniques mentioned. These latter two techniques will likely be useful when the constraint matrix is dense, since in that case, one would expect \( s_{ik} \) and \( \hat{s}_{ik} \) to be nonzero. However, the given instances of the TDS problem are sparse, and consequently, these values will likely be zero.

3.2 The Heuristic.

The heuristic provides a means of obtaining integer feasible solutions quickly,
and is used repeatedly within the branch-and-cut search tree. It should be pointed out that a good heuristic — one that produces good integer feasible solutions — is a crucial component in the branch-and-cut algorithm, since it provides an upper bound for reduced-cost fixing at the root, and thus allows reduction in the size of the linear program we must solve. This in turn reduces the time required to solve the linear program at each node. In addition, a good upper bound also enables us to fathom more active nodes, which is extremely important in solving large-scale integer programs as they tend to create many active nodes leading to memory explosion.

Like other heuristics [22,23,32], our heuristic is linear programming based. In particular, it works by sequentially fixing variables and solving the corresponding linear programs. We also extend the idea used for set partitioning systems in [23] to our knapsack equality system. The heuristic algorithm terminates when either an integer feasible solution is obtained, or the linear program is infeasible. If it returns an integer feasible solution with a lower objective value than the upper bound of the integer program, the upper bound is updated and we move once again to the reduced-cost fixing routine. Otherwise, we turn to cut generation.

Within the heuristic, we first solve a linear program. If a fractional solution is obtained, the heuristic first sets all nonbasic variables that are at their upper bound and basic variables with value one to one. Then, if an upper bound for the integer program exists, reduced-cost fixing is called so as to fix nonbasic variables at their corresponding lower bound. Then, using specified criteria that change dynamically within the algorithm, certain variables are fixed at one. After that, for all the rows covered by these variables, implied fixing is performed, and the
appropriate rows are removed. (A row is said to be *covered* by the variables set to one if the sum of the associated coefficients equals the right-hand-side.) For other rows containing these variables but not covered by them, we perform conflict fixing of the variables. Once we remove all these variables that are fixed to zero, we then check if there are any rows with only one column index. If so, we fix the associated variable to one and perform the same fixing as above. After all this, the LP solver is called again, and the process continues. It is easy to notice that as the heuristic continues, the linear programs get smaller in size. Also, the linear program structure gets closer to that of a set partitioning problem. Using the empirical result that the linear programming solutions to small set partitioning problems are often integral, eventually we either obtain an integral feasible solution, or infeasibility due to incorrect setting of variables.

### 3.3 Continuous Reduced Cost Implications.

In a pure zero-one linear programming minimization problem, given the continuous optimal objective value $z_{lp}$, the continuous optimal reduced cost $d_j$, and a true upper bound on the zero-one optimal objective function value $z^+$, then for all nonbasic variables $x_j$ in the continuous optimal solution, the following are true:

(a) If $x_j = 0$ in the continuous solution and $z^+ - z_{lp} < d_j$, then $x_j = 0$ in every zero-one solution.

(b) If $x_j = 1$ in the continuous solution and $z^+ - z_{lp} < -d_j$, then $x_j = 1$ in every zero-one solution.
In the branch-and-cut setting, reduced cost fixing is performed whenever an improved integral upper bound is obtained, or whenever the objective value of the linear programming relaxation at the root node improves. It is also used within the heuristic when a current integral upper bound exists.

3.4 Constraint Generation.

Constraint generation refers to the generation of cutting planes in the branch-and-bound phase. Unlike traditional fractional cuts which are local to their corresponding nodes in the search tree, the cuts generated in a branch-and-cut system are global. In particular, they are valid for $P^{IP}$, and hence are applicable to every node in the search tree. Using only global cuts reduces the amount of storage required as compared to that of traditional cuts where each node has to carry its own cut inequalities, requiring additional bookkeeping and a large amount of memory. Moreover, each global cutting plane generated improves the linear description of the polyhedron $P^{IP}$, thus allowing an improvement in the objective value at each node of the tree.

The problem of generating cutting planes is often called the separation problem, and can be formally stated as follows:

Given a fractional point $\bar{x} \in \mathbb{R}^n$ and a polyhedron $P \subseteq \mathbb{R}^n$, find a valid inequality $\pi^T \bar{x} \leq \pi_0$ for $P$ such that $\pi^T \bar{x} > \pi_0$, or prove that no such inequality exists (i.e., $\bar{x} \notin P$).

In integer programming, any fractional solution $\bar{x}$ of a linear programming relax-
ation necessarily satisfies $\bar{x} \not\in P^{IP}$, and consequently, there exist valid inequalities for $P^{IP}$ violated by $\bar{x}$. The goal of the constraint generation phase of the branch-and-cut algorithm is to efficiently generate violated valid inequalities.

As described in Section 2.3, we can approximate the underlying polyhedron $P^{IP}$ of the TDS problem by the polytopes $P^i$ associated with individual knapsack constraints, and by the node packing polytope $P^C$ associated with the conflict graph $G_C$. Any valid inequality generated for $P^i$ or $P^C$ will also be valid for $P^{IP}$. In this section, we discuss algorithmic approaches for generating valid inequalities for the polytopes $P^i$ and $P^C$ which are violated by the current fractional solution, and discuss methods for lifting these inequalities to include other variables.

Let $\bar{x}$ be the fractional solution obtained when we are at a given node in the tree, and let

\[
L = \{ j \in N : \bar{x}_j = 0 \},
\]

\[
F = \{ j \in N : 0 < \bar{x}_j < 1 \} \quad \text{and}
\]

\[
U = \{ j \in N : \bar{x}_j = 1 \}
\]

denote the index sets of variables that are at their lower bound, variables having a fractional value, and variables at their upper bound, respectively. Since we would like to identify a valid inequality, $\pi^T x \leq \pi_0$, which is violated by $\bar{x}$, we can ignore the set $L$ in the identification phase. However, unlike the set packing problem, due to the non-0/1 entries in the constraint matrix and right hand side of the TDS problem, we cannot ignore the variables indexed by $U$ when generating valid inequalities based on the conflict graph. More specifically, for the set packing problem, since every neighbor $k$ of a node in $U$ in the intersection graph necessarily satisfies $\bar{x}_k = 0$,
one cannot obtain violated valid inequalities by focusing on graph structures (e.g., cliques and odd cycles) involving nodes in $U$ [23, p. 19]. The following example shows that the same is not necessarily true when considering the conflict graph associated with a problem with non 0/1 data.

**Example 3.4.1.** Let $P^{IP} = \text{conv}\{x \in B^5 : Ax = b\}$, where

$$
A = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 2 & 2 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 2
\end{bmatrix}
$$

and

$$
b = \begin{bmatrix}
1 \\
2 \\
1 \\
1 \\
2
\end{bmatrix}
$$

Note that the conflict graph yields the valid inequality $x_2 + x_5 \leq 1$, which is violated by the fractional solution $\bar{x} = [0, 1, 0, 0, 1/2]^T$.

Let us now focus on generating valid inequalities for $P^C$. Let $G_{F \cup U} = (F \cup U, E_{F \cup U})$ denote the subgraph of $G$ induced by the node set $F \cup U$. This subgraph is computed and represented by an adjacency list. Note that the size of $G_{F \cup U}$ can be controlled by adjusting the *tolerance* used to check for fractional values of the solution vector $\bar{x}$. In most cases, the subgraph is easier to manage than $G$, since it is much smaller.

**Clique Generation.** Here, we generate violated clique inequalities utilizing the subgraph $G_{F \cup U}$. First, for each row $i$, we define an index set $M_i$ that corresponds
to the node set of a complete subgraph of $G_{F \cup U}$. In particular, define

$$M_i = \begin{cases} 
\{ j \in F \cup U : a_{ij} = 1 \}, & \text{if the right hand side of row } i \text{ equals } 1 \\
\{ j \in F \cup U : a_{ij} = b \} \cup \{ j_0 \}, & \text{if the right hand side of row } i \text{ equals } b,
\end{cases}$$

where $j_0 \in F \cup U$ and $a_{ij_0} \in \{1, 2, \ldots, b - 1\}$. After identifying this complete subgraph, we find the set $K \subseteq (F \cup U) \setminus M_i$ such that each node $k \in K$ has edges to all of the nodes in $M_i$. Then some or all of the columns in $K$ will form a clique with $M_i$. We identify exactly those columns that are violated and lift them into the remaining variables of the problem. Note that the cut we obtain is guaranteed to be valid for $P^{IP}$, but not necessarily facet defining.

**Chordless Odd Cycles.** It is well-known that if $C$ is an odd cycle without chords, then the odd cycle inequality

$$\sum_{u \in C} x_u \leq (|C| - 1)/2 \tag{3.4.1}$$

is valid for $P^C$ and defines a facet of the node packing polytope associated with the restriction of $P^C$ to the node set $C$. Once such an odd cycle inequality is obtained it can be lifted to a facet of $P^C$ by solving a sequence of lifting problems. It should be noted that by lifting the variables in different orders, it is possible to obtain distinct valid inequalities defining distinct facets of $P^C$ [30].

An effective way of finding an odd cycle $C$ is as follows [12]. First, assign to each edge $(u, v)$ of $G_{F \cup U}$ a weight $w(u, v)$. In order to maintain positive weights in our algorithm, we choose $w(u, v) = 2 - \bar{x}_u - \bar{x}_v$. Then randomly choose a node $s$ of $G_{F \cup U}$, and perform a breadth-first-search at root $s$. During the search, each
node is assigned its level (i.e., its edge distance) from \( s \), and if at some stage there is an edge joining two nodes, say \( u \) and \( v \), on the same level, then an odd cycle is obtained. Our procedure for determining if the odd cycle is chordless is modelled after that described in [23], but with some modifications. The procedure begins by determining the shortest-weight leveled-path from \( u \) to \( s \). Then, we "block" all the nodes that are adjacent to the nodes along the path from \( u \) to \( s \), excluding the vertex \( s \). We then determine a shortest-weight leveled-path from \( v \) to \( s \) using only nodes that are not blocked in the breadth-first search tree rooted at \( s \). If such a path is found, the associated odd cycle \( C \) is chordless. We then check to see if (3.4.1) is violated. If so, we lift it to include other variables in \( F \cup U \) in order to obtain an even deeper cut. Otherwise, we determine the lifting coefficients exactly for a specified number of variables to see if a violated inequality can be obtained. We then continue the process with the current level-graph until no more odd cycles are obtained. After this, another root is selected and the process is repeated. We perform this procedure on approximately 20% of the nodes.

*Facets for Individual Knapsack Equalities.* This procedure identifies facets for individual knapsack polytopes of the form \( S_b \), using the facet defining inequalities obtained in Section 2.2. In the current implementation, this procedure is specifically tailored to solve the given TDS problem instances. Since in these instances the largest coefficient in any knapsack equality constraint is 3, the current system only searches for facet defining inequalities associated with \( S_2 \) and \( S_3 \). In particular, we utilize the inequalities specified in Theorems 2.2.4 and 2.2.5 to search for valid inequalities that are violated by the current fractional solution \( \bar{x} \). Once we obtain
such a violated inequality, we lift it to include more variables in $F \cup U$. We now specify the method used to generate violated $S_2, S_3$-cuts.

First consider the facet defining inequalities (2.2.14) and (2.2.22) for $S_2$ and $S_3$, respectively. Both of these are of the form

$$x_{j_0} + \sum_{j \in R_0} x_j \leq 1,$$

where $j_0 \in R_1$. In order to search for violated inequalities of this form, for each appropriate row $i$, we determine the index set

$$M_i = \{ j \in F \cup U : a_{ij} = b_i \},$$

and calculate $\sum_{j \in M_i} \bar{x}_j$. We then scan the set $R_1 \cap (F \cup U)$ and choose the largest $\bar{x}_{j_0}$, $j_0 \in R_1 \cap (F \cup U)$. If $\bar{x}_{j_0} + \sum_{j \in M_i} \bar{x}_j > 1$, then we have identified a violated valid inequality. One may of course continue to scan $R_1$ for other values of $j_0$ which also yield violated valid inequalities.

A similar approach is used to identify violated valid inequalities of the form (2.2.23) and (2.2.24) which, for convenience, are reproduced below:

$$\sum_{j \in T} x_j + \sum_{j \in R_2} x_j + \sum_{j \in R_3} 2x_j \leq 2, \quad T \subset R_1, \quad |T| = 2$$

$$\sum_{j \in R_2} x_j + \sum_{j \in R_3} x_j \leq 1$$

This time, for each appropriate row $i$, we determine the index sets

$$M_{i2} = \{ j \in F \cup U : a_{ij} = 2 \} \quad \text{and} \quad M_{i3} = \{ j \in F \cup U : a_{ij} = 3 \}$$

and calculate the two sums

$$\sum_{j \in M_{i2}} \bar{x}_j + \sum_{j \in M_{i3}} 2\bar{x}_j \quad \text{and} \quad \sum_{j \in M_{i2}} \bar{x}_j + \sum_{j \in M_{i3}} \bar{x}_j.$$
If the second sum is greater than 1, we have identified a violated valid inequality.
For the first sum, we scan \( R_1 \cap (F \cup U) \) for the two largest values \( \bar{x}_{j_0} \) and \( \bar{x}_{j_1} \), add them to the sum, and check if the result is greater than 2.

Notice that two of the three procedures to determine violated valid inequalities are exact, namely clique generation and \( S_2, S_3 \)-facets. However, whether or not the generated inequalities yield deep cuts and/or high dimensional faces of \( P^{IP} \) depends in large part on how closely the polytopes \( P^i \) and \( P^C \) approximate \( P^{IP} \). Nevertheless, one can tighten the generated inequalities by lifting.

The Lifting Procedure. The idea of lifting was introduced by Gomory [20] in 1969, and its computational possibilities were developed in the 70's and 80's by Padberg [30], Wolsey [35], Zemel [37], and Balas and Zemel [3]. In what follows, we describe the lifting procedure, as well as some of the suggested ways of solving the integer programs that arise as subproblems in the procedure. We then conclude with a brief discussion of our tailored lifting procedure for the TDS problems.

Let \( \pi^T x \leq \pi_0 \) be a valid inequality for \( P^{IP} \), and let \( J = \{ j \in N : \pi_j \neq 0 \} \). If \( y \in \mathbb{R}^n \), let \( y_J \) denote the vector in \( \mathbb{R}^{|J|} \) with the components \( y_i, i \in N \setminus J \), removed. Similarly, let \( A_J \) denote the \((p + q) \times |J|\) matrix obtained by removing the columns indexed by \( i \in N \setminus J \) from the constraint matrix \( A \). The idea behind lifting is to extend the inequality \( \pi^T x \leq \pi_0 \). More precisely, one selects an index \( k \in N \setminus J \), and attempts to determine a lifting coefficient \( \hat{\pi}_k \) for \( x_k \) such that

(L1) the inequality

\[
\pi_J^T x_J + \hat{\pi}_k x_k \leq \pi_0
\]  

is valid for \( P^{IP} \), and
(L2) if \( \pi^T_J x_J \leq \pi_0 \) defines a face of dimension \( \ell \) of the polytope

\[
P^{IP} \cap \{ x \in \mathbb{R}^n : x_j = 0 \text{ for all } j \in N \setminus J \},
\]

then (3.4.2) defines a face of dimension \( \ell + 1 \) of the polytope

\[
P^{IP} \cap \{ x \in \mathbb{R}^n : x_j = 0 \text{ for all } j \in N \setminus (J \cup \{k\}) \}.
\]

The choice of \( \hat{\pi}_k \) can be made by first solving the **lifting problem**

\[
z_k = \text{maximize } \pi^T_J x_J \\
\text{subject to } A_J x_J \leq \hat{b} - a^k \tag{3.4.3}
\]

\[
x_j \in \{0, 1\}, \quad j \in J,
\]

where \( a^k \) denotes the \( k \)-th column of \( A \). Then \( \hat{\pi}_k = \pi_0 - z_k \). If this value is negative, we do nothing since the inequality with 0 coefficient for \( x_k \) will dominate that with a negative coefficient. If it is positive, we update \( \pi_k \leftarrow \hat{\pi}_k \) and \( J \leftarrow J \cup \{k\} \), and repeat the process. Note that the order in which the variables are lifted is important, in the sense that different lifting orders can potentially produce different valid inequalities [30].

Solving the lifting problem (3.4.3) can be expensive since it is, itself, a 0/1 integer program. Moreover, the size of the problem grows (i.e., \( |J| \) increases) as more variables are lifted, and consequently, the problems tend to become successively more difficult to solve. In our implementation, when \( |J| \leq 32 \), we use an enumeration routine to determine the lifting coefficient exactly. Of course, the enumeration process terminates early if a 0/1 vector \( x_J \) satisfying \( \pi^T_J x_J \geq \pi_0 \) is obtained, since this already implies that the lifting coefficient for \( x_k \) will be zero. Otherwise, the enumeration process continues until the optimal solution to (3.4.3) is found.
When $|J| > 32$, we use an approximation method similar to that described in [23], which is based on the following observation:

$$\max\{\pi_j^T x_j : A_j x_j \leq \hat{b} - a^k, \ x_j \in \{0, 1\} \text{ for all } j \in J\}$$
$$\leq \max\{\pi_j^T x_j : A_j x_j \leq \hat{b} - a^k, \ x_j \geq 0\}$$
$$= \min\{u^T(\hat{b} - a^k) : u^T A_j \geq \pi_j^T, \ u \geq 0\}$$
$$\leq \min\{u^T(\hat{b} - a^k) : u^T A_j \geq \pi_j^T, \ u \geq 0, \ u \text{ integral}\}.$$

Note that the last problem can be solved by the greedy algorithm. Also, if $z_k^*$ denotes the optimal objective value of this problem, then $\pi_0 - z_k \geq \pi_0 - z_k^*$, and it follows that if we take $\hat{\pi}_k = \pi_0 - z_k^*$, then (L1) is true. Although (L2) does not necessarily hold with this choice of $\hat{\pi}_k$, we nevertheless update $\pi_k \leftarrow \hat{\pi}_k$ and $J \leftarrow J \cup \{k\}$, provided $\hat{\pi}_k > 0$.

In general, the more variables that are lifted, the more accurate is the final cut. However, lifting can be computationally expensive, and as mentioned above, the larger the set $J$, the more difficult it is to solve the lifting problem (3.4.3). Hence, some sort of compromise must be made in determining the number of variables to be lifted. We experimented with lifting between 100 and 200 variables, and determined that for the TDS instances, it was best to lift only 100, due to an excessive amount of computational time needed to lift more than 100.

Finally, we mention that the special form of clique inequalities and odd cycle inequalities allow for the development of somewhat efficient methods for solving (3.4.3). First, suppose the valid inequality $\pi_j^T x_j \leq \pi_0$ arose from a clique $J$ in the graph $G_C$. Then $\pi_0 = 1$, and enumeration can be done easily by checking if there exists a column $a^j$ in $A_J$ such that $a^j \leq \hat{b} - a^k$. If so, then $z_k \geq 1$, and consequently,
\( \pi_0 - z_k \leq 0. \)

Similarly, if the inequality arose from an odd cycle, we can again apply the same trick as with the clique inequality. This time, however, we continue to check for columns less than or equal to \( \hat{b} - a^k \). If we can find \( \pi_0 \) such columns, then we can conclude that the lifted coefficient is nonpositive. Otherwise, we will obtain a positive lifted coefficient \( \pi_0 - z_k \).
Chapter 4

Numerical Tests and Results

The preprocessor, the heuristic and the constraint generator are integrated into a Branch-and-Cut IP Solver using CPLEX as the intermediate LP Solver. This IP Solver is specially tailored to solve the TDS problem to optimality. However, many of the techniques and much of the code are applicable to more general 0/1 integer programming problems. The code is written in C, and not counting the LP solver and the comment statements, has about 16,000 lines; the most code intensive parts being the constraint generator and the preprocessor.

Numerical tests on 14 real instances supplied by Texaco Trading & Transportation, Inc. were run on a SUN super SPARCstation 10 model 41 with super cache, SUN SPARCstation 2, SUN SPARCstation 1 and SUN SPARCsystem 670MP Model 41 with two processors. Table 1 provides a description of the problem instances. Cols, Rows, and Nonzeros denote, respectively, the number of columns, the number of rows, and number of non-zeros in the constraint matrix; \(Z_{LP}\) and \(Z_{IP}\) denote the optimal objective values for the linear programming relaxation and the integer program; and Gap denotes the difference between \(Z_{LP}\) and \(Z_{IP}\). Note that a value of 0 for the gap does not imply that the LP solver returned an integral feasible solution from the linear programming relaxation.
Table 2 summarizes the results for plain branch-and-bound on the problem instances. \textit{BB Nodes} and \textit{BB time} denote the number of nodes in the terminal search tree and the time required to solve to optimality, respectively.

Table 3 provides a comprehensive picture of the branch-and-cut system and the effort expended by each of its major components. \textit{Prepr. Calls} and \textit{Heur. Calls} denote the number of times the preprocessor and heuristic were called, and \textit{Prepr. Time} and \textit{Heur. Time} denote their respective times. \textit{Heur. Success} indicates the number of times the heuristic returned an integral solution, and \textit{Cuts} and \textit{BC Nodes} indicate the number of cuts generated and the number of nodes in the branch-and-cut tree. Finally, \textit{Elapsed Time} denotes the total time of the entire branch-and-cut process.

Table 4 compares the results in Tables 2 and 3. We observe that branch-and-cut outperformed branch-and-bound in all but one test problem (zb1all.1). Moreover, for those “hard” problems (e.g., zb2all, zb2all.1, zm3all, zs1new, zs2short, zs3short, zs3area) branch-and-cut substantially reduced the CPU time required to solve them to optimality. In all these cases, the time is reduced by 80\% to 99.5\%.

Table 5 records the performance of the heuristic routine when it is used as a “stand-alone” procedure. $Z_{\text{HEUR}}$ denotes the first heuristic value obtained. Comparing Tables 3 and 5 we note that for some “hard” problems (e.g., zb2all and zb2all.1) the heuristic returned optimal solutions within 30 seconds, and the effort (4576.6 and 1249.3 seconds, respectively) in the cutting phase of the branch-and-cut system was solely used to prove that optimality had been reached. In fact, in 6 out of the 14 instances, the first call to the heuristic routine returned an optimal solution to the integer program.
Table 6 reports the results on a subset of the problem instances when only the preprocessing routine is switched on. Comparing these results with those in Table 2 indicate that preprocessing pays off substantially on the more difficult problem instances, reducing the time over plain branch-and-bound by 84% in zs3short and by 77% in zs2short.

Similar to Table 6, Table 7 reports the results on a subset of problem instances when only the preprocessing routine is switched on, and the heuristic is called exactly once to obtain an initial integral upper bound. In this case, in 7 out of the 10 instances, the time over plain branch-and-bound was reduced.

As with any branch-and-cut system, the performance of the LP solver — which is repeatedly called to solve sequences of linear programs — is crucial to the overall performance of the branch-and-cut system. Therefore, we experimented with various features of the LP solver to determine which features could be exploited to speed up the overall process of the system. Tables 8 and 9 provide a summary of the CPU time required to solve the initial linear programming relaxation to optimality using various parameters in CPLEX. The system is now set to use primal reduced-cost pricing to solve the initial linear programming relaxation, and steepest-edge primal pricing for successive resolves of the linear programs within the heuristic and after reduced-cost fixing in the branch-and-cut tree. Finally, steepest-edge dual pricing is used for solving the linear programs during the resolve phase in the cutting plane routine and within the branching phase. We have tested the system using standard dual pricing, but on some of the harder instances, we ran into difficulties in solving successive linear programs, and thus all the test results presented here were run with the steepest-edge dual pricing.
In summary, the numerical tests indicate that the branch-and-cut system significantly reduces the total time to solve to optimality some hard instances of the TDS problem. Moreover, Tables 3, 6 and 7 show that each of the system's components — the preprocessor, the reduced-cost fixing routine, the heuristic, and the cut generator — contribute to the solution process. However, Table 3 gives strong evidence that it is the combination of the four components that gives the system its true strength.
Chapter 5

Summary and Future Research

In recent years several studies have shown that branch-and-cut can be a viable approach for solving certain 0/1 integer programs. In this thesis we apply the method to a class of programs arising from a truck dispatching scheduling problem and characterized by a system of equality constraints of a specific form. The branch-and-cut system that has been developed incorporates a combination of problem preprocessing, the generation of cutting planes, an LP-based heuristic, and branch-and-bound techniques.

The desire to exploit the structure of the TDS problem in the development of the constraint generation phase of the system motivated new results in polyhedral theory concerning facets of knapsack equality polytopes of the form $S_6$. It also led to the introduction of the notion of conflict graph to generate an approximating node packing polytope $P^C$ of the TDS polytope $P^{IP}$. The cutting planes generated within the system are based entirely on facet defining inequalities of the individual knapsack polytopes, and valid inequalities generated from the conflict graph for the polytope $P^C$. In particular, polyhedral cuts, generated explicitly as facets of individual knapsack equality constraints, are incorporated into the branch-and-cut system.
The numerical results show that the branch-and-cut system is an effective way of solving the TDS problem instances. Indeed, the times to reach optimality in branch-and-cut were in many instances far superior to the times required to reach optimality for ordinary branch-and-bound.

Future research plans include further investigating the polyhedral theory associated with polytopes arising from knapsack equality constraints, and developing a general algorithmic approach to generating violated cuts based on the theory. We will also investigate the effectiveness of cuts obtained by applying the polyhedral theory related to the conflict graph on general 0/1 problems with non 0/1 data.

Furthermore, since the branch-and-cut system is a very flexible code (in the sense that various switches can be adjusted to implement different algorithmic strategies) more tests will be conducted on the TDS problem as well as other 0/1 problems in an attempt to make the system more robust and efficient. Finally, we will take advantage of the fact that within the branch-and-cut system, different tasks can be performed simultaneously on different processors; hence, a parallel version of the system will be implemented.
Figure 1
Branch-and-Cut Flowchart†

†$z_p = \text{LP objective value}; z^+ = \text{objective value of best integer solution}$
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>Nonzeros</th>
<th>$Z_{LP}$</th>
<th>$Z_{IP}$</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>114138</td>
<td>2319.80</td>
<td>2319.80</td>
<td>0.00</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>103020</td>
<td>2012.28</td>
<td>2012.28</td>
<td>0.00</td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>98607</td>
<td>2630.37</td>
<td>2639.40</td>
<td>9.03</td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>89670</td>
<td>2259.50</td>
<td>2268.00</td>
<td>8.50</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>22411</td>
<td>1456.00</td>
<td>1456.00</td>
<td>0.00</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>22236</td>
<td>1444.60</td>
<td>1449.00</td>
<td>4.40</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>54216</td>
<td>1284.95</td>
<td>1285.29</td>
<td>0.34</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>18190</td>
<td>1419.69</td>
<td>1426.00</td>
<td>6.31</td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>44916</td>
<td>1305.24</td>
<td>1319.26</td>
<td>14.01</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>8403</td>
<td>1527.73</td>
<td>1539.60</td>
<td>11.87</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>666323</td>
<td>889.00</td>
<td>889.00</td>
<td>0.00</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>138322</td>
<td>904.71</td>
<td>919.21</td>
<td>14.50</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>59509</td>
<td>1048.70</td>
<td>1071.25</td>
<td>22.55</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>17154</td>
<td>1918.59</td>
<td>1924.00</td>
<td>5.41</td>
</tr>
</tbody>
</table>
Table 2

Numerical Results When Using Only
Branch-and-Bound†

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>BB Nodes</th>
<th>BB time</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>65</td>
<td>172.5</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>35</td>
<td>88.7</td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>62644</td>
<td>340690.0</td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>55460</td>
<td>333047.0</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>7</td>
<td>2.7</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>214</td>
<td>76.7</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>36</td>
<td>16.9</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>992</td>
<td>135.5</td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>43990</td>
<td>156158.0</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>14446</td>
<td>2842.7</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>12</td>
<td>171.8</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>16298</td>
<td>168292.0</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>8578</td>
<td>26874.8</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>4632</td>
<td>1357.2</td>
</tr>
</tbody>
</table>

† Run on SUN superSPARCstation 10 model 41
with super cache. Time in CPU seconds.
Table 3

Numerical Results for

Branch-and-Cut†

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>1</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>9.9</td>
<td>0*</td>
<td>0</td>
<td>11.0</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>120.3</td>
<td>0*</td>
<td>0</td>
<td>120.3</td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>2</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>26.0</td>
<td>481</td>
<td>0</td>
<td>4576.6</td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>3</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>29.9</td>
<td>406</td>
<td>0</td>
<td>1249.3</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>2</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1.7</td>
<td>0*</td>
<td>0</td>
<td>2.1</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>3</td>
<td>0.1</td>
<td>3</td>
<td>2</td>
<td>1.7</td>
<td>442</td>
<td>66</td>
<td>14.9</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>3</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>5</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>3</td>
<td>0.2</td>
<td>3</td>
<td>3</td>
<td>7.2</td>
<td>932</td>
<td>30</td>
<td>105.5</td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>1</td>
<td>0.1</td>
<td>5</td>
<td>2</td>
<td>7.1</td>
<td>5401</td>
<td>208</td>
<td>6253.4</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>3</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>7.9</td>
<td>59</td>
<td>0</td>
<td>19.6</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>1</td>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>24.2</td>
<td>0*</td>
<td>0</td>
<td>24.2</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>3</td>
<td>0.6</td>
<td>3</td>
<td>3</td>
<td>9.7</td>
<td>3029</td>
<td>147</td>
<td>5419.5</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>3</td>
<td>4.8</td>
<td>5128</td>
<td>156</td>
<td>3373.0</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>3</td>
<td>0.1</td>
<td>2</td>
<td>2</td>
<td>10.9</td>
<td>453</td>
<td>36</td>
<td>54.9</td>
</tr>
</tbody>
</table>

†Run on SUN superSPARCstation 10 model 41 with super cache.

Time in CPU seconds.

*Optimal IP Solution obtained after first call to heuristic.
Table 4

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>BB Nodes</th>
<th>BB Time</th>
<th>BC Nodes</th>
<th>BC Time</th>
<th>Cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>65</td>
<td>172.5</td>
<td>0</td>
<td>11.0</td>
<td>0</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>35</td>
<td>88.7</td>
<td>0</td>
<td>120.3</td>
<td>0</td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>62644</td>
<td>340690.0</td>
<td>0</td>
<td>4576.6</td>
<td>481</td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>55460</td>
<td>333047.0</td>
<td>0</td>
<td>1249.3</td>
<td>406</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>7</td>
<td>2.7</td>
<td>0</td>
<td>2.1</td>
<td>0</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>214</td>
<td>76.7</td>
<td>66</td>
<td>14.9</td>
<td>442</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>36</td>
<td>16.9</td>
<td>0</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>992</td>
<td>135.5</td>
<td>30</td>
<td>105.5</td>
<td>932</td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>43990</td>
<td>156158.0</td>
<td>208</td>
<td>6253.4</td>
<td>5401</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>14446</td>
<td>2842.7</td>
<td>0</td>
<td>19.6</td>
<td>59</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>12</td>
<td>171.8</td>
<td>0</td>
<td>24.2</td>
<td>0</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>16298</td>
<td>168292.0</td>
<td>147</td>
<td>5419.5</td>
<td>3029</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>8578</td>
<td>26874.8</td>
<td>156</td>
<td>3373.0</td>
<td>5128</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>4632</td>
<td>1357.2</td>
<td>36</td>
<td>54.9</td>
<td>453</td>
</tr>
</tbody>
</table>

† Run on SUN superSPARCstation 10 model 41 with super cache.

Time in CPU seconds.
Table 5

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>$Z_{LP}$</th>
<th>$Z_{IP}$</th>
<th>$Z_{HEUR}$</th>
<th>Heur.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>2319.80</td>
<td>2319.80</td>
<td>2319.80</td>
<td>9.9</td>
<td></td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2012.28</td>
<td>2012.28</td>
<td>2012.28</td>
<td>120.3</td>
<td></td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>2630.37</td>
<td>2639.40</td>
<td>2639.40</td>
<td>26.0</td>
<td></td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>2259.50</td>
<td>2268.00</td>
<td>2268.00</td>
<td>29.9</td>
<td></td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>1456.00</td>
<td>1456.00</td>
<td>1446.00</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>1444.60</td>
<td>1449.00</td>
<td>1452.00</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>1284.95</td>
<td>1285.29</td>
<td>1285.29</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>1419.69</td>
<td>1426</td>
<td>1428.70</td>
<td>4.6</td>
<td></td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>1305.24</td>
<td>1319.26</td>
<td>fail</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>1527.73</td>
<td>1539.60</td>
<td>1543.20</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>889.00</td>
<td>889.00</td>
<td>889.00</td>
<td>24.2</td>
<td></td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>904.71</td>
<td>919.21</td>
<td>922.21</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>1048.70</td>
<td>1071.25</td>
<td>1084.33</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>1918.59</td>
<td>1924.00</td>
<td>1976.50</td>
<td>4.9</td>
<td></td>
</tr>
</tbody>
</table>

†Run on SUN superSPARCstation 10 model 41 with super cache.

Time in CPU seconds.
Table 6

Numerical Results When Using Only Preprocessing and Reduced-cost Fixing†

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>Prepr. Calls</th>
<th>Prepr. Time</th>
<th>BB Nodes</th>
<th>BB time</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>1</td>
<td>0.2</td>
<td>82</td>
<td>300.7</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2</td>
<td>0.4</td>
<td>68</td>
<td>610.8</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>2</td>
<td>0.1</td>
<td>42</td>
<td>13.7</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>1</td>
<td>0.1</td>
<td>182</td>
<td>178.3</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>2</td>
<td>0.2</td>
<td>4</td>
<td>1.4</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>2</td>
<td>0.1</td>
<td>254</td>
<td>360.0</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>2</td>
<td>0.1</td>
<td>9810</td>
<td>1485.2</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>1</td>
<td>0.0</td>
<td>12</td>
<td>171.8</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>2</td>
<td>0.6</td>
<td>14648</td>
<td>37900.1</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>2</td>
<td>0.3</td>
<td>3072</td>
<td>4392.7</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>2</td>
<td>0.1</td>
<td>3572</td>
<td>632.7</td>
</tr>
</tbody>
</table>

†Run on SUN superSPARCstation 10 model 41 with super cache.

Time in CPU seconds.
## Table 7

### Numerical Results When Using Preprocessing, Heuristic Warm-start, and Reduced-cost Fizing†

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>1</td>
<td>0.2</td>
<td>9.9</td>
<td>0*</td>
<td>0.0</td>
<td>11.0</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2</td>
<td>0.1</td>
<td>120.3</td>
<td>0*</td>
<td>0.0</td>
<td>121.0</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>2</td>
<td>0.1</td>
<td>1.7</td>
<td>0*</td>
<td>0.0</td>
<td>2.1</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>2</td>
<td>0.1</td>
<td>0.7</td>
<td>168</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>3</td>
<td>0.2</td>
<td>0.2</td>
<td>2</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>3</td>
<td>0.2</td>
<td>4.6</td>
<td>106</td>
<td>14.6</td>
<td>15.0</td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>2</td>
<td>0.1</td>
<td>7.9</td>
<td>9810</td>
<td>3386.8</td>
<td>3414.8</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>1</td>
<td>0.0</td>
<td>24.2</td>
<td>0*</td>
<td>0.0</td>
<td>24.2</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>3</td>
<td>0.1</td>
<td>4.9</td>
<td>3572</td>
<td>790.1</td>
<td>794.1</td>
</tr>
</tbody>
</table>

†Run on SUN superSPARCstation 10 model 41 with super cache.

Time in CPU seconds.

*Optimal IP solution obtained after first call to heuristic.
Table 8

Time to Solve Initial LP Relaxation
for Various Primal Pricing Strategies in CPLEX†

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>$Z_{LP}$</th>
<th>Primal Parameters‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>2319.80</td>
<td>5.4  30.2  39.3  11.6</td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2012.28</td>
<td>4.4  33.1  31.6  10.4</td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>2630.37</td>
<td>4.0  31.4  45.0  14.6</td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>2259.50</td>
<td>4.2  35.7  57.5  13.9</td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>1456.00</td>
<td>0.6  2.7   3.1   1.4</td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>1444.60</td>
<td>0.6  2.8   3.5   1.3</td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>1284.95</td>
<td>1.3  8.4   10.1  4.0</td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>1419.69</td>
<td>0.4  1.3   1.9   0.6</td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>1305.24</td>
<td>2.0  7.5   10.4  3.5</td>
</tr>
<tr>
<td>zsdnew</td>
<td>2089</td>
<td>28</td>
<td>1527.73</td>
<td>0.2  0.9   1.1   0.4</td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>889.00</td>
<td>37.5 659.5 654.2 237.6</td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>904.71</td>
<td>2.9  24.4  23.0  9.4</td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>1048.70</td>
<td>1.5  5.6   7.6   2.7</td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>1918.59</td>
<td>0.5  2.4   2.5   1.0</td>
</tr>
</tbody>
</table>

† Run on SUN SPARCstation 1. Time in CPU seconds.

‡ Primal Parameters

- 1 Reduced-cost pricing
  1 Devez pricing
  2 Steepest-edge pricing
  4 Full pricing
Table 9

Time to Solve Initial LP Relaxation
for Various Dual Pricing Strategies in CPLEX†

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cols.</th>
<th>Rows</th>
<th>(Z_{LP})</th>
<th>Dual Parameters*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>zb1all</td>
<td>27768</td>
<td>47</td>
<td>2319.80</td>
<td>1088.9</td>
<td>157.2</td>
<td>137.4</td>
<td>157.3</td>
<td></td>
</tr>
<tr>
<td>zb1all.1</td>
<td>24771</td>
<td>47</td>
<td>2012.28</td>
<td>65.6</td>
<td>38.7</td>
<td>53.3</td>
<td>38.9</td>
<td></td>
</tr>
<tr>
<td>zb2all</td>
<td>24172</td>
<td>49</td>
<td>2630.37</td>
<td>61.6</td>
<td>54.3</td>
<td>69.1</td>
<td>54.3</td>
<td></td>
</tr>
<tr>
<td>zb2all.1</td>
<td>21801</td>
<td>51</td>
<td>2259.50</td>
<td>58.7</td>
<td>30.7</td>
<td>47.6</td>
<td>30.8</td>
<td></td>
</tr>
<tr>
<td>zb3all</td>
<td>5712</td>
<td>37</td>
<td>1456.00</td>
<td>6.5</td>
<td>5.8</td>
<td>8.1</td>
<td>5.8</td>
<td></td>
</tr>
<tr>
<td>zm1all</td>
<td>5301</td>
<td>26</td>
<td>1444.60</td>
<td>16.0</td>
<td>7.9</td>
<td>12.3</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>zm2all</td>
<td>11706</td>
<td>28</td>
<td>1284.95</td>
<td>20.7</td>
<td>16.2</td>
<td>21.9</td>
<td>16.3</td>
<td></td>
</tr>
<tr>
<td>zm2new</td>
<td>3636</td>
<td>29</td>
<td>1419.69</td>
<td>3.3</td>
<td>2.9</td>
<td>4.2</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>zm3all</td>
<td>10849</td>
<td>30</td>
<td>1305.24</td>
<td>14.2</td>
<td>15.7</td>
<td>19.3</td>
<td>15.7</td>
<td></td>
</tr>
<tr>
<td>zs1new</td>
<td>2089</td>
<td>28</td>
<td>1527.73</td>
<td>7.8</td>
<td>2.0</td>
<td>2.3</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>zs1short</td>
<td>115544</td>
<td>26</td>
<td>889.00</td>
<td>608.4</td>
<td>316.0</td>
<td>279.5</td>
<td>314.8</td>
<td></td>
</tr>
<tr>
<td>zs2short</td>
<td>26306</td>
<td>27</td>
<td>904.71</td>
<td>78.2</td>
<td>49.3</td>
<td>52.1</td>
<td>49.4</td>
<td></td>
</tr>
<tr>
<td>zs3short</td>
<td>11273</td>
<td>25</td>
<td>1048.70</td>
<td>16.0</td>
<td>9.1</td>
<td>9.1</td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td>zs3area</td>
<td>4030</td>
<td>34</td>
<td>1918.59</td>
<td>7.7</td>
<td>8.9</td>
<td>6.1</td>
<td>9.0</td>
<td></td>
</tr>
</tbody>
</table>

†Run on SUN SPARCstation 1. Time in CPU seconds.

‡Dual Parameters

1 Standard dual pricing
2 Steepest-edge pricing
3 Steepest-edge pricing in slack space
4 Steepest-edge pricing, unit initial norms
Bibliography


