A Polyhedral Study of Nonconvex Piecewise Linear Optimization

A Thesis
Presented to
The Academic Faculty

by

Ahmet B Keha

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial Engineering
Georgia Institute of Technology
August 2003

A Polyhedral Study of Nonconvex Piecewise Linear Optimization

Approved by:

George L. Nemhauser, Advisor

Ozlem Ergun

Shabbir Ahmed

Martin W.P. Savelsbergh

Ismael R. de Farias

Date Approved 8/15/03

Dedicated to my dear wife Özgür Sema

for her deep love and firm support

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. George Nemhauser for his guidance and support throughout this study. I also would like to thank him for providing me with the financial support I needed during my doctoral studies.

I would like to thank Dr. Ismael de Farias for his help and valuable suggestions during the writing of the papers that form the core of this dissertation.

I am thankful to the other members of the thesis advisory committee, Dr. Shabbir Ahmed, Dr. Özlem Ergun, and Dr. Martin Savelsbergh for the time they spend reading this thesis.

I am most grateful to my fellow office mates for the encouragement and support they gave me during this study. Special thanks go to Vijay Bharadwaj, Paul Brooks, Deniz Dogan, Gurdal Ertek, Brady Hunsaker, Yetkin Ileri and Dieter Vandenbussche.

I also extend my gratitude to my friends Serdar Bosca, Tahir Duzyol, Bunyamin Ozaydin, Shahzad and Nonie Chaudary and their adorable kids Aysha and Amaan for the fun we had together.

Finally, I would like to express my gratitude to my family for the love and support they give me everyday. Without their support and encouragement, this thesis would never have been completed.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

SUMMARY x

I INTRODUCTION 1

1.1 Mixed Integer Programming 3

1.2 Linear Programs 4

1.3 LP-based Branch-and-Bound 5

1.3.1 Preprocessing 7

1.3.2 Branching 7

1.3.3 Primal Heuristics 8

1.4 Branch-and-Cut Algorithms 9

1.4.1 Basic Concepts 9

1.4.2 Branch-and-cut Algorithm 11

1.4.3 Valid Inequalities From Special Structures 12

1.4.4 Separation 14

1.4.5 Lifting 14

1.5 A Branch-and-Cut Approach without Binary Variables 19

1.6 Outline of the Thesis 21

II MODELS FOR REPRESENTING PIECEWISE LINEAR COST FUNCTIONS 23

2.1 MIP Formulations 25

2.2 Two Related Relaxations and Bound Comparisons 27

III A BRANCH-AND-CUT ALGORITHM WITHOUT BINARY VARIABLES FOR NONCONVEX PIECEWISE LINEAR OPTIMIZATION 32

3.1 Simple Inequalities 35

3.2 Lifted Cover Inequalities 40

3.2.1 Lifting Approximately 42

3.3 Branch-and-Cut Algorithm 43

3.3.1 Primal Heuristic 43

3.3.2 Branching 44

3.3.3 Creating Cuts 44

3.4 Computational Experience 45

3.4.1 Test Instances 47

3.4.2 Computational Results 47

IV POLYHEDRAL STUDY OF GENERAL PIECEWISE LINEAR OPTIMIZATION 52

4.1 Lifting Cover Inequalities 53

4.2 Lifting Convexity Constraints 61

4.3 Lifting Approximately 63

4.4 Branch-and-cut Algorithm 64

4.5 Computational Experience 65

4.5.1 Test Instances 66

4.5.2 Computational Results 66

V POLYHEDRAL STUDY OF LOWER-SEMI CONTINUOUS PIECEWISE LINEAR OPTIMIZATION 70

5.1 Lifted Convexity Constraints 74

5.2 Cover Inequalities 80

5.3 Relation Between Flow Cover Cuts and Cover Cuts 83

5.4 Computational Experience 84

5.4.1 Branching 85

5.4.2 Creating Cuts 85

5.4.3 Computational Results 86

VI CONCLUSIONS AND FUTURE RESEARCH **89**

REFERENCES **91**

VITA **95**

LIST OF TABLES

1 Computations without introducing binary variables 49

2 Computations when binary variables are introduced 50

3 Comparison of branching rules with and without estimated increase . 51

4 Comparison of two algorithms for the general case 67

5 Comparison of looking for a cut at each node and at every 50 nodes . 69

6 Computations with and without our cuts 87

LIST OF FIGURES

1 A piecewise linear function 1

2 A typical LTL cost function 2

3 Finding the set H_i 38

4 Finding the sets $H_i^{k_i}$ and $\overline{H}_i^{k_i+1}$ 77

5 Finding the set $\widehat{H}_i^{k_i+1}$ 79

SUMMARY

Piecewise linear functions are widely used to approximate nonlinear functions. However, when minimizing (maximizing) a piecewise linear function (PLF), it is necessary to introduce nonlinearities in the model if the function is not convex (concave). Traditionally the nonlinearities are modelled by introducing auxiliary 0-1 variables and additional constraints that relate the continuous and 0-1 variables or by specialized branching in the scape of continuous variables. We enhance the latter approach through the use of strong inequalities valid for the convex hull of the feasible set in the space of continuous variables. In the thesis we first study the convex hull of single constraint relaxations with only positive coefficients. We then relax this assumption and extend the idea to general single constraint relaxations. We also extend the inequalities to the case where the PLF is lower semi-continuous. For each case we report computational results that demonstrate that our approach is significantly better than the traditional approaches to these problems.

CHAPTER I

INTRODUCTION

In this thesis, we study a separable *piecewise linear optimization* problem of the form

$$\text{minimize} \quad \sum_{j \in N} f_j(x_j), \tag{1.1}$$

$$\text{subject to} \quad \sum_{j \in N} a_{ij} x_j \leq b_i \quad \forall i \in \{1, \ldots, m\}, \tag{1.2}$$

$$x_j \geq 0 \quad \forall j \in N, \tag{1.3}$$

where $N = \{1, \ldots, n\}$ and the $f_j(x_j)$ are *piecewise linear functions* (PLF), see Figure 1.

PLFs are widely used to approximate nonlinear functions. Note that any arbitrary continuous function of one variable can be approximated by a piecewise linear function, with the quality of the approximation controlled by the size of the linear segments. Cost functions in many supply chain problems are piecewise linear. For example, the transportation costs in a supply chain network are frequently concave and piecewise linear, possibly with fixed costs [10]. PLFs are used to model problems

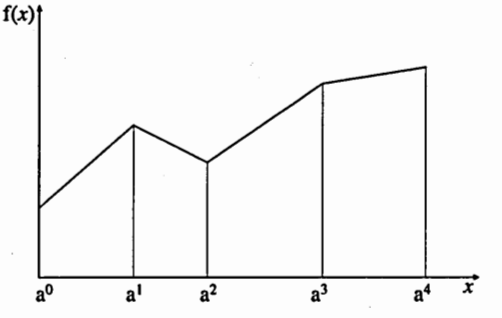

Figure 1: A piecewise linear function

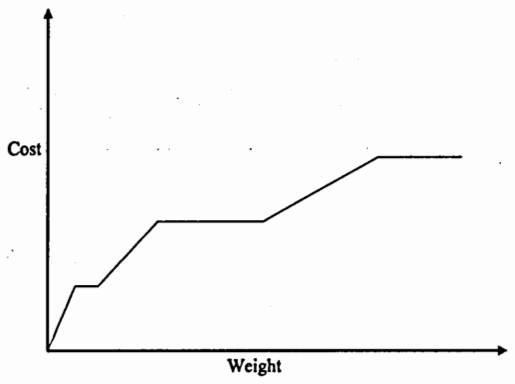

Figure 2: A typical LTL cost function

such as optimization of electronic circuits [25], transportation and production planning problems [23] , merge-in-transit (MIT) problems [10] and separable nonlinear functions.

In MIT problem, the cost functions of different modes of transportation are piecewise linear. For example, the cost function of less-than-truckload (LTL) shipments are piecewise linear, but may include flat segments with a fixed cost, see Figure 2. In production planning problems, production costs may be concave and piecewise linear due to economies of scale.

PLFs that are convex can be minimized by linear programming because the slope of the segments are increasing. But when minimizing a PLF, it is necessary to introduce nonlinearities in the model if the function is not convex. In this thesis we will study minimizing nonconvex PLFs. Several methods have been used to solve problems involving nonconvex PLFs. The most widely used ones are the *mixed integer programming* (MIP) approach and the *special ordered sets of type II* (SOS2) approach. In the MIP approach, the nonlinearities are modelled by introducing auxiliary 0-1 variables and additional constraints that relate binary and continuous variables. In the SOS2 approach, the nonlinearities are enforced by specialized branching. These approaches

will be discussed in detail in Chapter 2. In the following sections, we will introduce the concepts that we will use in the thesis and we will outline our contributions.

1.1 Mixed Integer Programming

A MIP is an optimization problem of the form

$$minimize \quad cx,$$

$$subject\ to \quad Ax = b,$$

$$l \leq x \leq u,$$

$$x_j\ integral,\ j = 1, \ldots, p.$$

The input data are the matrices $c(1xn)$, $A(m,n)$, $b(mx1)$, $l(1xn)$, and $u(1xn)$, and the n-vector x is to be determined. If $p = 0$, then the problem is a *linear program* (LP), and if $p = n$, then the problem is an *integer program* (IP). We sometimes use the notation Z for integer variables and R for continuous variables, i.e $x_j \in Z,\ j = 1, \ldots, p,\ x_j \in R,\ j = p+1, \ldots, n$.

MIP's are very important in practice and are used to model a large variety of real-life problems. Applications of MIP concern maximizing a specific criterion such as profit or productivity, or minimizing a criterion such as cost, subject to some restrictions due to scarce resources. The succesful applications of MIP include finance [6], forestry [22], steel industry [46] and supply chain management [1].

An algorithm is polynomial, if the number of "basic operations" that must be carried out is polynomial in the size of input. We say that a problem is in class P if it solvable in polynomial time. A problem is in class NP, if we can check whether a proposed solution is feasible or not in polynomial time. Clearly, P is contained in NP. A problem is $NP - hard$ means that, if this problem is polynomially solvable, then so is any problem in NP. It is believed that it is not possible to design an

algorithm that runs in polynomial time to solve any $NP-hard$ problem. It is known that solving MIP is $NP-hard$, so it is unlikely that there exists a polynomial time algorithm for MIP, see Schrijver [44].

Many methods have been proposed to solve MIP's. Recent advances in methods for solving MIP's and the ability to solve relatively large problems in a relatively small amount of time have made the use of MIP much more widespread. More information and references on MIPs can be found in the books by Schrijver [44], Nemhauser and Wolsey [38] and Wolsey [51]. The survey of Johnson, Nemhauser, and Savelsbergh [30] gives many references to recent publications.

The most common methods used for solving MIP's are LP-based branch and bound algorithms. We will next discuss components of these algorithms in detail.

1.2 Linear Programs

Linear programming algorithms are often used as a subroutine in MIP algorithms to obtain an upper bound on the value of the MIP. Any MIP can be relaxed to an LP, called the *LP relaxation*, by weakening the restriction $x_j \in Z$ to $x_j \in R$, $j = 1, \ldots, p$. Since the feasible region of the LP relaxation contains the feasible region of the MIP, an upper bound can be obtained through LP algorithms. In addition, if the optimal solution to the LP relaxation satisfies the required integrality restrictions, then that solution is also optimal to the MIP. If the LP relaxation is infeasible, then the MIP is also infeasible.

LP is well-defined in the sense that if it is feasible, and does not have unbounded optimal value, then it has an optimal solution. The first class of algorithms designed for solving LPs was *simplex algorithms*, which was proposed by Dantzig in 1947 to solve U.S. Air Force planning problems [13]. Although known simplex algorithms are not polynomial, they are very fast in practice and are part of most major commercial

LP-solvers.

The first polynomial time algorithm for LPs, proposed by Khachian [32] in 1979, is known as the *ellipsoid algorithm.* This algorithm is very poor in practice and is only used as a theoretical tool. The first class of algorithms that provide both good theoretical and good practical performance was introduced by Karmarkar [31] in 1984 and is known as *interior point algorithms.* Variants of these algorithms are also used in commercial LP-solvers.

More information and references on LPs can be found in the books by Schrijver [44] and Chvatal [7]. The paper by Dantzig [13] gives the history of linear programming.

1.3 LP-based Branch-and-Bound

Branch-and-bound is one of the first methods that was proposed to solve MIPs [30]. Branch-and-bound methods find the optimal solution to a MIP by efficiently enumerating the points in the feasible region. The basis of a branch-and-bound is the *enumeration tree.* The LP at the root node of the enumeration tree correspond to the LP relaxation of the original problem. The tree grows by a process called *branching.* First the LP relaxation at a node is solved. If the optimal solution is not feasible to the corresponding MIP then two or more child nodes may be generated by branching on an appropriately chosen fractional variable x_i. Each of the LPs at the child nodes is formed by adding constraints to the LP at the parent node. Typically branching is as follows: suppose that in a given node x_i assumes a fractional value between the integers d and $d+1$. Then two new child nodes are generated by adding the constraints $x_i \leq d$ and $x_i \geq d+1$ to the LP at the parent node. The optimal solution of the LP at the parent node does not lie among the optimal solutions of the LPs at the child nodes. If it is unnecessary to branch on a node, we say it is *fathomed.* For a minimization problem the following three situations result in a subproblem being fathomed:

1. The LP at the node is infeasible, so no feasible solution can be found at that part of the tree.

2. The LP at the node yields an integer solution. We have a feasible solution. If this optimal value has lower value than the previously found feasible solutions, then it becomes a *candidate* solution. In this case the optimal value of the node becomes the current upper bound on the optimal solution of the MIP.

3. The optimal value of the LP at the node is more than current upper bound so it may be eliminated from consideration

The first branch-and-bound algorithm for solving integer programs was introduced by Land and Doig [33] in 1960. Branch-and-bound algorithms for MIPs were proposed by Dakin [11] and Driebeck[21] in 1965. Subsequent advances in branch-and-bound algorithms made it possible to curtail the enumeration significantly by reducing the number of nodes required to prove optimality. The time required to process a node can be higher but the decrease in the overall computing time can be substantial.

At any step of the branch-and-bound tree we want to fathom the nodes to obtain a smaller enumeration tree. One case where we fathom a node is the case in which the optimal value of the LP at the node is more than the upper bound. We can apply fast heuristics to obtain lower upper bounds and increase the possibility of fathoming the node. Another approach is to add a constraint, called a *cut*, to the LP that has the property that the current optimal solution of the LP is not satisfied by the cut, but every feasible solution of the MIP is satisfied by the cut. Adding such cuts will increase the possibility of fathoming because it may result in a increase of the optimal LP value at the node. Finally LP relaxations of different formulations can be very different in terms of the quality of the bound they provide. So we may decrease the size of the tree by providing a good initial formulation. We will next discuss the ways to improve an LP-based branch-and-bound algorithm.

1.3.1 Preprocessing

Preprocessing applies simple logic to reformulate the problem and tighten the LP relaxations. Preprocessing may reduce the size of an instance by fixing variables and eliminating constraints. Sometimes preprocessing may also detect infeasibility. For example, considering one row together with lower and upper bounds may lead to dropping the row if it is redundant, declaring the problem infeasible if that row is infeasible, or tightening the bounds on the variables. Preprocessing can also be applied with probing. Probing means setting a binary variable temporarily to 0 or 1 and then applying logical testing as above. For example, if $x_j = 0$ implies that the instance is infeasible, then we can fix $x_j = 1$.

1.3.2 Branching

In a branch-and-bound algorithm an unevaluated node is chosen, the LP relaxation is solved, and a fractional variable (if there is one) is chosen to branch on. Here we make two main decisions: the first decision is choosing the active node to evaluate and the second one is choosing the fractional variable to branch on.

Choosing the variable to branch on can be critical in keeping the tree small. A simple rule is to select a variable whose fractional value is closest to 1/2. More sophisticated rules try to choose a variable such that the optimal LP value of the child nodes are as small as possible. This is important since early branching on variables that cause big changes can be critical in keeping the tree small.

It can be expensive to compute the actual objective value changes for all candidate variables. Instead, estimates of the rate of objective function change on both the down and up branches, called *pseudo costs*, are used. Another variant of this idea is *strong branching*. In strong branching a fairly large number of dual simplex iterations is carried out for a small set of candidate branching variables. From these iterations,

we obtain a lower bound on the change and this lower bound is used to choose the variable to branch on.

Node selection rules are usually based on finding good feasible solutions as early as possible. It is difficult to balance the advantages and disadvantages of selecting nodes near to the top or bottom of the tree. If we choose the nodes from bottom, then the number of active nodes will be large. If we choose the nodes from the top, then the number of active nodes will be small but it may take a long time to find a good feasible solution.

If the fractional variable we are branching is a 0-1 variable, then one branch sets that variable to 0 while the other sets it to 1. If it is a general integer variable, and the value of fractional variable x_i is between the integers d and $d+1$, then one branch constraints it to be $\leq d$ and the other to be $\geq d+1$. More complicated branching involves sets of variables. These types of branchings will be discussed later.

A recent survey of branching techniques is presented by Linderoth and Savelsbergh [35]. More information and references on branching can be found in the survey by Johnson, Nemhauser and Savelsbergh [30].

1.3.3 Primal Heuristics

Primal heuristics focus on finding better lower bounds on the optimal value of the MIP. Finding better lower bounds also means finding better feasible solutions.

If we know the structure of the problem, then any known heuristic for the problem can be used to find a feasible solution. Other ways of finding feasible solutions are enumerating integral vectors in the neighborhood of the current LP solution and using successive rounding to the current LP solution.

1.4 Branch-and-Cut Algorithms

Before presenting the algorithm we introduce some of the concepts that provide the foundation of the branch-and-cut algorithms. Most of the definitions of the next section are quoted from the book by Nemhauser and Wolsey [38]. More comprehensive discussions of polyhedral theory can be found in the books by Schrijver [44] and Nemhauser and Wolsey [38].

1.4.1 Basic Concepts

Definition 1. *A polyhedron $P \subseteq \Re^n$ is the set of points that satisfy a finite number of linear inequalities, i.e. $P = \{x \in \Re^n : Ax \leq b\}$ where (A,b) is a $m x (n+1)$ matrix. A polytope is a bounded polyhedron.*

A polyhedron is said to be rational if A and b can be chosen to be rational, i.e, it can be represented by linear inequalities with integer coefficients. In this thesis we assume that all polyhedra are rational.

Definition 2. *Given a set $S \subseteq \Re^n$, a point $y \in \Re^n$ is an affine combination of the points $\{x^1, \ldots, x^k\} \subseteq S$ if*

$$y = \sum_{i=1}^{k} \lambda_i x^i, \quad \sum_{i=1}^{k} \lambda_i = 1$$

for some $\lambda \in \Re^n$. If λ is restricted to be nonnegative then $y \in \Re^n$ is said to be an convex combination of the points $\{x^1, \ldots, x^k\} \subseteq S$.

Definition 3. *The convex hull of a set S is the uniquely smallest set of all points that are convex combinations of the points in S.*

Definition 4. *The points $x^1, \ldots, x^k$ of $\Re^n$ are said to be affinely independent if the system $\sum_{j=1}^{k} \alpha_j x^j = 0, \sum_{j=1}^{k} \alpha_j = 0$ has $\alpha_j = 0$ for $j = 1, \ldots, k$ as unique solution.*

Definition 5. *A polyhedron P is of dimension k, denoted by $dim(P) = k$, if the maximum number of affinely independent points that can be found in P is k+1.*

The dimension of a polyhedron is the smallest affine space that contains that polyhedron. A polyhedron $P \in \Re^n$ with $dim(P) = k < n$ has exactly $n - k$ linearly independent equations that are satisfied by all points in P.

Definition 6. *A polyhedron is said to be* full dimensional *if* $dim(P) = n$.

Definition 7. *An inequality* $\pi x \leq \pi_0$ *is said to be a valid inequality for a polyhedron* P, *if it is satisfied by all points in* P.

Definition 8. *If* $\pi x \leq \pi_0$ *is a valid inequality for a polyhedron* P, *and* $F = \{x \in P : \pi x = \pi_0\}$, *then* F *is called a face of* P, *and we say that* $\pi x \leq \pi_0$ *represents* F. *A face is said to be proper if* $F \neq P$ *and* $F \neq \emptyset$.

Definition 9. *A facet is face* F *of a polyhedron* P *with* $dim(F) = dim(P) - 1$.

An inequality which represents a facet is called a *facet-defining inequality.*

Theorem 1. *For each facet* F *of polyhedron* P, *one of the inequalities representing* F *is necessary in the description of* P.

Besides being necessary, the facets are sufficient for the description of P. When the polyhedron is full-dimensional, P has a unique description. This fact is stated in the next proposition.

Theorem 2. *A full-dimensional polyhedron* P *has a unique (to within scalar multiplication) minimal representation by a finite set of linear inequalities. In particular, for each facet* F_i *of* P *there is an inequality* $a^i x \leq b_i$ *(unique to within scalar multiplication) representing* F_i *and* $P = \{x \in \Re^n : a^i x \leq b_i \; for i = 1, \ldots, t\}$.

Definition 10. *A point of a polyhedron* P *is an extreme point if it can not be written as a convex combination of the other points of* P.

Theorem 3. *A polyhedron has only a finite number of extreme points.*

When we optimize a linear function over a nonempty polyhedron, there is an optimal solution that is an extreme points provided that the problem is bounded. The next theorem states that the convex hull of feasible solutions to a MIP is a polyhedron.

Theorem 4. (adapted from Meyer [37]) *Let F be set of feasible solutions to MIP. The convex hull of F is a polyhedron whose extreme points belong to F.*

Theorem 2 and 4 imply that every MIP can be converted into an equivalent linear program. Since all the extreme points of this linear program are feasible points of the MIP, it is enough to solve a linear program to find the optimal solution of the MIP. But we have two practical problems. The first one is that, for most MIP problems, although we know that the convex hull of the feasible solutions is a polyhedron, we don't know a linear inequality description of it. The second is that the number of inequalities describing the convex hull of the feasible solutions can be very large. Branch-and-cut algorithms, instead of building this very large equivalent linear program, successively improve the LP relaxation by adding inequalities that are implied by the inequalities that define the convex hull. We will discuss the algorithm after defining a powerful tool, called lifting, for finding the facets of the convex hull of feasible solutions to a MIP.

1.4.2 Branch-and-cut Algorithm

Branch-and-cut algorithms generate valid inequalities that are not satisfied by all feasible points to the LP relaxations. Such valid inequalities are called *cuts*. A cut which is not satisfied by the given optimal solution of the LP relaxation is called a *violated cut*. By adding a violated cut to the LP relaxation, we obtain a smaller LP region, while the feasible region of the MIP does not change.

Branch-and-cut is a generalization of branch-and-bound. After solving the LP relaxation, if we can not fathom the node, we try to find a violated cut. If we can

find one or more violated cuts, we add them to the LP relaxation and solve it again. If we can not find a violated cut, we branch the node.

1.4.3 Valid Inequalities From Special Structures

Since it is not easy to find strong inequalities for most problems, in practice it is common to derive valid inequalities from relaxations of the problem. If we choose a relaxation that is common to many problems, we can use the results for that relaxation for many other problems also. The first implementation of this idea for 0-1 integer problems is due to Crowder, Johnson, and Padberg [8]. Their idea was to relax the 0-1 integer program to be solved by dropping all but one of its linear constraints and then generate strong valid inequalities for the polytope defined by this single constraint. These valid inequalities associated with the single constraint are also valid for the original problem. In the following subsections, we review two classes of inequalities that have been used to generate cuts for MIP.

Knapsack Cover Inequalities

Let $B^n = \{0,1\}^n$ and K be the feasible region of a 0-1 knapsack problem, i.e.,

$$K = \{x \in B^n : \sum_{j \in N} a_j x_j \leq b\}.$$

Without loss of generality, we can assume that $a_j > 0$ and $a_j \leq b$, $\forall j \in N$. A set $C \subseteq N$ is called a *knapsack cover* if $\sum_{j \in C} a_j > b$. Since all variables in C can not be equal to 1 simultaneously, the *knapsack cover inequality*

$$\sum_{j \in C} x_j \leq |C| - 1 \tag{1.4}$$

is valid for K[2, 29, 49]. Knapsack cover inequalities are usually not facet-defining for $conv(K)$. It is possible to improve (1.4) by lifting, a procedure that will be discussed in detail later.

The computational studies presented by Gu, Nemhauser, and Savelsbergh [26]

showed that these lifted inequalities are very effective although the inequalities that we use may not be facet-defining inequalities for the underlying polyhedron.

Flow Cover Inequalities

A MIP with binary integer (BMIP) variables is the appropriate mathematical model for many practical optimization problems. This model is used, for example, for facility location problems, distribution problems, network design problems, and more generally when fixed or concave costs are required in the objective function of a linear system ([27]).

Flow cover inequalities are valid inequalities for the system

$$F = \{(x,y) \in R_+^n x B^n : \sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq d,\ x_j \leq u_j y_j,\ j \in N\}, \qquad (1.5)$$

where $N = N^+ \cup N^-$ and $n = |N|$. The system (1.5) can be viewed as a single node fixed charge network flow model. It is possible to obtain relaxations of the form F from any BMIP problem [38].

For a single-node flow model with only inflow arcs, i.e. $N^- = \phi$, a set $C^+ \subseteq N^+$ is called a *flow cover* if $\sum_{j \in C^+} u_j = d + \lambda$ with $\lambda > 0$. The inequality

$$\sum_{j \in C^+} x_j + \sum_{j \in C^{++}} (u_j - \lambda)(1 - y_j) \leq d \qquad (1.6)$$

where $C^{++} = \{j \in C^+ : u_j > \lambda\}$, is called a *flow cover inequality* and is valid for (1.5) when $N^- = \emptyset$.

For a general single-node flow model, a set $C = C^+ \cup C^-$ is called a *flow cover* if $C^+ \subseteq N^+$, $C^- \subseteq N^-$, and $\sum_{j \in C^+} u_j - \sum_{j \in C^-} u_j = d + \lambda$ with $\lambda > 0$. The inequality

$$\sum_{j \in C^+} x_j + \sum_{j \in C^{++}} (u_j - \lambda)(1 - y_j) - \sum_{j \in L^-} \lambda y_j - \sum_{j \in L^{--}} x_j \leq d + \sum_{j \in C^-} u_j, \qquad (1.7)$$

where $L^- = \{j \in N^- - C^- : u_j > \lambda\}$ and $L^{--} = N^- - (L^- \cup C^-)$, is called a *generalized flow cover inequality* and is valid for (1.5).

Flow cover inequalities for (1.5) have been derived by Padberg, Van Roy, and Wolsey [40] and Van Roy and Wolsey [42]. Gu, Nemhauser, and Savelsbergh [27] strengthened the flow cover inequalities through lifting.

1.4.4 Separation

Given a solution to the LP relaxation that does not satisfy the required integrality restrictions, the *separation problem* is to find a violated cut. Although there must be a violated cut among the linear inequalities which define the convex hull of the feasible solutions, it may be difficult to find violated facet-defining inequalities. There is usually a tradeoff between the strength of the cuts and the time it takes to find them. Therefore, in practice, heuristics are used to find the violated inequalities. Given an infeasible solution, there can be cases where these heuristics can not find any violated inequalities. Also since we don't know all of the facet-defining classes of inequalities for most systems, we may not find a violated inequality.

As an example, we describe the separation problem for the class of knapsack inequalities for K. We are given a fractional solution $x^* \in R^n_+$, and we want to find a set C with $\sum_{j\in C} a_j > b$ and $\sum_{j\in C} x^*_j > |C| - 1$. The separation problem is

$$\zeta = min\{\sum_{j\in N}(1 - x^*_j)z_j : \sum_{j\in N} a_j z_j > b, z \in B^n\},$$

where $z \in B^n$ represents the characteristic vector of the cover C that is to be determined. If $\zeta < 1$, the cover inequality (1.4) is violated by x^*. Otherwise, we conclude that there is no vioated cover inequality. Observe that the separation problem for knapsack cover inequalities is another knapsack problem. In practice, heuristics are used to solve the separation problem.

1.4.5 Lifting

Describing the facets of high dimensional polytopes can be a difficult task. An alternative is to reduce the dimension by fixing some of the variables to their upper or

lower bounds. It is usually easier to find strong valid inequalities in lower dimensional spaces. If we can find a valid inequality (facet) for this lower dimensional polytope, then it can be converted into a valid inequality (facet) of higher dimensional polytope by a tool called *lifting.*

The idea of lifting was introduced by Gomory [24]. It has been studied and generalized by Padberg [39], Wolsey [50], Zemel [52] and Balas and Zemel [3].

We first introduce sequential lifting. There are two types of sequential lifting: *uplifting* and *downlifting.* Let l_j and u_j be lower and upper bounds for the variable x_j. The procedure that extends a valid inequality for $S \cap \{x \in S : x_j = l_j\}$ to a valid inequality for S is called uplifting and the procedure that extends a valid inequality for $S \cap \{x \in S : x_j = u_j\}$ to a valid inequality for S is called downlifting. The sequential lifting procedure is given by the following theorem [15]:

Theorem 5. *Let $P \subseteq \Re^d$ be a polytope. Define $l_j = min\{x_j : x \in P\}$ and $u_j = max\{x_j : x \in P\}$, $j = 1, \ldots, d$. Let $\widetilde{x} \in P$, and suppose that*

$$\sum_{j=1}^{d-1} \alpha_j x_j \leq \beta \tag{1.8}$$

is a valid inequality for $P \cap \{x \in \Re^d : x_d = \widetilde{x_d}\}$. Define

$$\alpha_d^{max} = \begin{cases} min\{\frac{\beta - \sum_{j=1}^{d-1} \alpha_j x_j}{x_d - \widetilde{x_d}} : x \in P \text{ and } x_d > \widetilde{x_d}\} & \text{if } \widetilde{x_d} < u_d, \\ \infty & \text{if } \widetilde{x_d} = u_d, \end{cases}$$

and

$$\alpha_d^{min} = \begin{cases} max\{\frac{\beta - \sum_{j=1}^{d-1} \alpha_j x_j}{x_d - \widetilde{x_d}} : x \in P \text{ and } x_d < \widetilde{x_d}\} & \text{if } \widetilde{x_d} > l_d, \\ -\infty & \text{if } \widetilde{x_d} = l_d. \end{cases}$$

Then

$$\sum_{j=1}^{d-1} \alpha_j x_j + \alpha_d x_d \leq \beta + \alpha_d \widetilde{x_d} \tag{1.9}$$

is a valid inequality for P if and only if $\alpha_d^{min} \leq \alpha_d \leq \alpha_d^{max}$. Moreover, if (1.8) defines a face of $P \cap \{x \in \Re^d : x_d = \widetilde{x_d}\}$ of dimension t, and $\alpha_d = \alpha_d^{min} > -\infty$, or $\alpha_d = \alpha_d^{max} < \infty$, then (1.9) defines a face of P of dimension at least $t+1$.

In some cases we may want to lift some variables simultaneously. We next summarize some results of Gu, Nemhauser, and Savelsbergh [27, 28] on simultaneous lifting. Consider the set of feasible points for a BMIP given by

$$X = \{x \in R_+^{|N|} : \sum_{j \in N} a_j x_j \leq d',$$

$$\sum_{j \in C_k} w_j x_j \leq r_k, k = 0, \ldots, t,$$

$$x_j \in \{0,1\}, j \in I \subseteq N,$$

$$x_j \leq u_j, j \in N - I\}.$$

where $C_k,\ k = 0, \ldots, t$ is a partition of N; $a_j,\ j \in N$, and d' are $mx1$; and $w_j,\ j \in N$, and r_k are $m_k x 1$.

Initially, we consider the subset of X with $x_j = b_j$, i.e., x_j is fixed at one of its bounds, for $j \in N - C_0$ given by

$$X^0 = \{x \in R_+^{|C_0|} : \sum_{j \in C_0} a_j x_j \leq d,$$

$$\sum_{j \in C_0} w_j x_j \leq r_0,$$

$$x_j \in \{0,1\}, j \in I \cap C_0,$$

$$x_j \leq u_j, j \in (N - I) \cap C_0\}.$$

where $d = d' - \sum_{j \in N - C_0} a_j b_j$.

Let

$$\sum_{j \in C_0} \alpha_j x_j \leq \alpha_0 \tag{1.10}$$

be an arbitrary valid inequality for x^0. We want to construct a valid inequality for X of the form

$$\sum_{j \in C_0} \alpha_j x_j + \sum_{1 \leq k \leq t} \sum_{j \in C_k} \alpha_j (x_j - b_j) \leq \alpha_0. \tag{1.11}$$

To construct such an inequality, we start with (1.10) and lift the variables in $N - C_0$. We assume that the variables with indices in $C_1, \ldots, C_t$ are lifted sequentially in that order and that in a given set C_k they are lifted simultaneously.

The intermediate sets of feasible points X^i for $i = 1, \ldots, t$ are defined by

$$X^i = \{x \in R_+^{\sum_{0 \le k \le t} |C_k|} : \sum_{j \in C_0} a_j x_j + \sum_{1 \le k \le i} \sum_{j \in C_k} a_j (x_j - b_j) \le d,$$

$$\sum_{j \in C_k} w_j x_j \le r_k, k = 0, \ldots, i,$$

$$x_j \in \{0, 1\}, j \in I \subseteq I \cap (\cup_{k=0}^{i} C_k),$$

$$x_j \le u_j, j \in (N - I) \cap (\cup_{k=0}^{i} C_k)\}.$$

Note that if we extend X^i to X by setting $x_j = b_j$ for $j \in \cup_{i+1}^{t} C_k$, then $X^{i-1} \subseteq X^i$ for $i = 1, \ldots, t$ and $X^t = X$.

For $i = 1, \ldots, t$, the *lifting problem* associated with C_i, given a valid inequality

$$\sum_{j \in C_0} \alpha_j x_j + \sum_{1 \le k \le i} \sum_{j \in C_k} \alpha_j (x_j - b_j) \le \alpha_0 \tag{1.12}$$

for X^{i-1}, is to find α_j for $j \in C_i$ such that

$$\sum_{j \in C_0} \alpha_j x_j + \sum_{1 \le k \le i} \sum_{j \in C_k} \alpha_j (x_j - b_j) + \sum_{j \in C_i} \alpha_j (x_j - b_j) \le \alpha_0 \tag{1.13}$$

is a valid inequality for X^i.

For $i = 1, \ldots, t$, let

$$Z^i = \{z \in R^m : \exists x \in X^i : \sum_{j \in C_i} a_j (x_j - b_j) = z \text{ and}$$

$$\sum_{j \in C_0} a_j x_j + \sum_{1 \le k \le i} \sum_{j \in C_k} a_j (x_j - b_j) \le d - z\},$$

and for $z \in Z^i$ let

$$h_i(z) = max \sum_{j \in C_i} \alpha_j (x_j - b_j)$$

$$s.t. \sum_{j \in C_i} a_j (x_j - b_j) = z$$

$$\sum_{j \in C_i} w_j x_j \leq r_i$$

$$x_j \in \{0,1\},\ j \in I \cap C_i$$

$$0 \leq x_j \leq u_j,\ j \in (N-I) \cap C_i,$$

and

$$f_i(z) = min\ \alpha_0 - \sum_{j \in C_0} \alpha_j x_j - \sum_{1 \leq k \leq i} \sum_{j \in C_k} \alpha_j (x_j - b_j)$$

$$s.t. \sum_{j \in C_0} a_j x_j + \sum_{1 \leq k \leq i} \sum_{j \in C_k} a_j (x_j - b_j) \leq d - z$$

$$\sum_{j \in C_k} w_j x_j \leq r_k,\ k = 0, \ldots, i-1$$

$$x_j \in \{0,1\}, j \in I \subseteq I \cap (\cup_{k=0}^{i-1} C_k)$$

$$x_j \leq u_j, j \in (N-I) \cap (\cup_{k=0}^{i-1} C_k)\}.$$

Proposition 1. *For $i = 1, \ldots, t$, inequality (1.13) is valid for X^i for any choice of α_j for $j \in C_i$ such that $h_i(z) \leq f_i(z)$ for $z \in Z^i$.*

When α_j for $j \in C_i$ are such that $h_i(z) = f_i(z)$ has $|C_i|$ solutions $x^1, x^2, \ldots, x^{|C_i|}$ such that the components in C_i of $x^1 - b, x^2 - b, \ldots, x^{|C_i|} - b$ are linearly independent, we say that the lifting is *maximal.*

Theorem 6. *For $i = 1, \ldots, t$, if $conv(X^{i-1})$ and $conv(X^i)$ are full dimensional, (1.12) defines a facet of $conv(X^{i-1})$ and $\alpha_0 \neq 0$, then (1.13) defines a facet of $conv(X^i)$ if and only if the lifting is maximal.*

Corollary 1. *Given an arbitrary valid inequality (1.10) for X^0, we can construct a valid inequality (1.11) for X by sequentially lifting sets C_i for $i = 1, \ldots, t$. At each step i, the lifting coefficients have to be such that $h_i(z) \leq f_i(z)$ for $z \in Z^i$. If (1.10) defines a facet of $conv(X^0)$, $conv(X^i)$ is full dimensional for $i = 0, \ldots, t-1$, and at each step i the lifting is maximal, then (1.11) defines a facet of $conv(X)$.*

1.5 A Branch-and-Cut Approach without Binary Variables

Many optimization problems involve only continuous variables, but are subject to combinatorial constraints. Examples of combinatorial constraints include :

- special ordered sets of type I constraints,
- special ordered sets of type II constraints,
- cardinality constraints,
- semi-continuous constraints,

A set of variables is a *special ordered set of type I* (SOS1) when at most one variable in the set can be positive. A set of variables is *special ordered set of type II* (SOS2) when at most two variables in the set can be positive, and when two variables are positive their indices must be adjacent. A *cardinality constraint* requires no more than a specified number of variables can be positive. A variable is *semi-continuous* if the value of the variable is either 0 or greater than or equal to a positive number.

Traditionally such models have been modelled as MIPs by introducing auxiliary binary variables and additional constraints. For example, suppose that $x_1 \leq u_1, \ldots, x_l \leq u_l$. If no more than k variables among $x_1, \ldots, x_l$ are allowed to be positive, then we introduce the 0-1 variables $z_1, \ldots, z_l$, and the constraints

$$x_j \leq u_j z_j \; j \in \{1, \ldots, l\}, \; and$$

$$\sum_{j=1}^{l} z_j \leq k.$$

If x_j is semi-continuous, then we introduce the 0-1 variable z_j and the constraint

$$l_j z_j \leq x_j \leq u_j z_j.$$

Since the number of variables and constraints becomes larger and the combinatorial structure of combinatorial constraints are not used to advantage, MIP models

may not be solved satisfactorily. Beale and Tomlin [4] proposed the idea of treating *special ordered sets* (SOS) constraints without using binary variables and thereby not increasing the size of the problem. Here SOS constraints are enforced through a specialized branching scheme. Additional work on SOS1 and SOS2 can be found in Beale and Forrest [5]. Most of these combinatorial constraints suggest natural branching strategies (see de Farias, Johnson, and Nemhauser [17]).

Let $\{x_1, \ldots, x_l\}$ be SOS1. Suppose that the relaxed subproblem at the current node of the branch-and-bound tree assigns a nonzero value to x_r and x_t, where $1 \leq r < t \leq l$. Let s be an index with $r \leq s < t$. Because of SOS1, either

$$x_1 = \ldots = x_s = 0 \tag{1.14}$$

or

$$x_{s+1} = \ldots = x_l = 0. \tag{1.15}$$

(1.14) and (1.15) eliminate the optimal solution of the relaxation at the current node from both branches but no solution that satisfies SOS1. Let x_j be a semi-continuous variable. Suppose that the relaxed subproblem at the current node of the branch-and-bound tree assigns a value $0 < x_j^* < l_j$ to x_j. We can then define two branches, one requiring

$$x_j = 0 \tag{1.16}$$

and the other

$$x_j \geq l_j. \tag{1.17}$$

SOS2 is crucial to our work and it will be discussed in detail later. Recently specialized branching has been strengthened by studying the polyhedral structures of the problems. de Farias [14], de Farias, Johnson, and Nemhauser [16, 17, 18], de Farias and Nemhauser [19, 20] and Vandenbussche [47] have applied this new

approach to several problems such as the generalized assignment problem, the complementarity knapsack problem, the cardinality constrained knapsack problem, and nonconvex quadratic programming. They enforce the nonlinearities through a specialized branching scheme and the use of strong inequalities valid for the convex hull of the feasible set in the space of continuous variables. In this thesis we will study and implement this approach for piecewise linear optimization.

Solving combinatorial constraints without introducing 0-1 variables has several advantages. The most obvious advantage is that introducing 0-1 variables increase both the number of the variables and the number of the constraints. In many real-life situations it is not clear how large the variable values can be. When we introduce 0-1 variables, the continuous variables must be bounded. To overcome this situation we often have to introduce big-M constraints. Big-M constraints are usually not tight, and when they are, they increase the degeneracy of the solution of the relaxation. Introduction of 0-1 variables also tends to obscure the combinatorial structure of the problem. For example, we can have fractional solutions such that the continuous variables satisfy SOS2, while the binary variables are fractional. A general purpose MIP solver, in this case, will continue to branch unnecessarily, trying to achieve integrality.

1.6 Outline of the Thesis

In Chapter 2, we study formulations of linear programs with piecewise linear objective functions with and without additional binary variables. We show that the two formulations without additional binary variables have the same LP bounds as those of the corresponding formulations with binary variables. We also show that the two formulatons without binary variables correspond to the same polyhedron in the space of continuous variables. Therefore the formulations without binary variables are preferable for efficient computation and it is sufficient to focus on one of them.

In Chapter 3, we study the polyhedron of the one row relaxation of a separable piecewise linear optimization problem. We assume that the objective function is continuous and the coefficients of the variables are positive. We give a branch-and-cut algorithm without binary variables for this problem. We also report computational results that demonstrate that our approach is significantly better than both an SOS2 branch-and-bound algorithm and a branch-and-cut algorithm that uses a mixed integer programming formulation.

In Chapter 4, we relax the assumption that the coefficients of the variables are positive. We generalize the inequalities and the branch-and-cut algorithm of Chapter 3 for this case. We also report computational results that demonstrate the branch-and-cut algorithm without binary variables is significantly better than a branch-and-cut algorithm that uses a mixed integer programming formulation.

In Chapter 5, we relax the assumption that the objective function is continuous. We study the polyhedron when the cost function is lower semi-continuous. In this case we need to introduce binary variables to formulate the problem. Therefore we generalize the inequalities of Chapter 3 to the case where binary variables are introduced. We again report computational results that demonstrate that introducing our cuts improves the performance of the branch-and-cut algorithm.

In Chapter 6, we summarize the main contributions of this thesis and present further directions for research. In summary, we present new algorithms for a class of NP-hard problems that are of great practical importance and our results clearly show that our algorithms are substantially better than those currently used in practice.

CHAPTER II

MODELS FOR REPRESENTING PIECEWISE LINEAR COST FUNCTIONS

As mentioned in Chapter 1, when minimizing (maximizing) a *piecewise linear function* PLF, it is necessary to introduce nonlinearities in the model if the function is not convex (concave). One way to formulate these nonlinearities is by adding binary variables and new inequalities to the model, yielding a mixed-integer program (MIP), see for example [38]. Two well known MIP formulations for PLFs are the *incremental cost* [36] and the *convex combination* [12] formulations. The polyhedra of both MIPs were studied in [34, 48].

Padberg [41] compared the linear programming (LP) relaxations of the two MIP models for PLFs in the simplest case when there are no constraints. He showed that the feasible set of the LP relaxation of the incremental cost formulation is integral, i.e. the binary variables are integer in every vertex of the set. He called such formulations *locally ideal.* On the other hand, the convex combination formulation is not locally ideal, and it strictly contains the feasible set of the LP relaxation of the incremental cost formulation. Shortly after, Sherali [45] proposed a modification of the convex combination formulation that is locally ideal.

Alternatively, Beale and Tomlin [4] suggested a formulation for PLFs similar to convex combination, except that no binary variables are included in the model and the nonlinearities are enforced algorithmically, directly in the branch-and-bound algorithm, by branching on sets of variables, which they called *special ordered sets of type*

2 (SOS2). It is also possible to formulate PLFs similar to incremental cost but without binary variables and enforcing the nonlinearities directly in the branch-and-bound algorithm.

In this chapter we show that the LP relaxations of both MIP formulations produce the same bound, which is a result recently obtained independently by Croxton et al. [9]. Then we show our main result that the bound remains the same when we remove the binary variables from the two formulations. We also show that both formulations correspond to the same polyhedron in the space of the continuous variables, and that all the vertices of the feasible sets of the corresponding LP relaxations are feasible. Because the continuous formulations are considerably smaller than the MIP formulations and one can take advantage of the structure of the problem by branching on sets of variables, we believe that a continuous formulation should be the model of choice here. However, neither continuous formulation seems to have any advantage over the other.

In the remainder of the thesis we will use the term *locally ideal* in the broader sense to mean that a needed property like integrality or SOS2 is obtained for free. Thus, in this broader sense both the special ordered set formulation and the continuous formulation based on incremental cost are locally ideal.

In Section 2.1 we review the incremental cost and the convex combination formulations, and Padberg's result. In Section 2.2 we show that the bounds of both MIP formulations, as well as the bounds of the continuous formulations are the same. We also show that the two MIP formulations correspond to the same polyhedron in the space of the continuous variables, and that both continuous formulations are locally ideal.

2.1 MIP Formulations

In this section we review the incremental cost and the convex combination formulations for PLFs, and Padberg's result. Suppose we have a PLF $f(x)$ specified by the points $(a_j, f(a_j))$, $j \in \{0,\ldots,T\}$. Let $u_j = a_j - a_{j-1}$ and $g_j = f(a_j) - f(a_{j-1})$ $\forall j \in \{1,\ldots,T\}$. Then for any $a_0 \leq x \leq a_T$ we have

$$x = a_0 + \sum_{j=1}^{T} y_j \quad \text{and} \quad f(x) = f(a_0) + \sum_{j=1}^{T} \frac{g_j}{u_j} y_j, \tag{2.1}$$

where $0 \leq y_j \leq u_j$ $\forall j \in \{1,\ldots,T\}$, and

$$y_{j+1} = \cdots = y_T = 0 \text{ whenever } y_j < u_j, \quad j \in \{1,\ldots,T-1\}. \tag{2.2}$$

To enforce (2.2), one can introduce binary variables z_j, $j \in \{1,\ldots,T-1\}$, and the constraints

$$u_1 z_1 \leq y_1 \leq u_1, \quad u_j z_j \leq y_j \leq u_j z_{j-1} \ \forall j \in \{2,\ldots,T-1\}, \quad \text{and } 0 \leq y_T \leq u_T z_T \tag{2.3}$$

The MIP model given by (2.1), (2.3), and $z_j \in \{0,1\}$ $\forall j \in \{1,\ldots,T-1\}$ is the incremental cost formulation. We denote the feasible set of the LP relaxation of the incremental cost formulation by

$$IB = \{(y,z) \in R^T \times [0,1]^{T-1} : (y,z) \text{ satisfies (2.3)}\}.$$

Alternatively, we can write

$$x = \sum_{j=0}^{T} a_j \lambda_j \quad \text{and} \quad f(x) = \sum_{j=0}^{T} \lambda_j f(a_j), \tag{2.4}$$

where

$$\sum_{j=0}^{T} \lambda_j = 1, \quad \lambda_j \geq 0 \quad \forall j \in \{0, \ldots, T\}, \tag{2.5}$$

and SOS2, which states that

at most two λ_j's can be positive, and if two are positive they must be consecutive. (2.6)

To enforce (2.6), one can introduce binary variables z_j, $j \in \{0, \ldots, T-1\}$, and the constraints

$$\lambda_0 \leq z_0, \quad \lambda_j \leq z_{j-1} + z_j \; \forall j \in \{1, \ldots, T-1\}, \quad \lambda_T \leq z_{T-1}, \text{ and } \sum_{j=0}^{T} z_j = 1. \tag{2.7}$$

The MIP model given by (2.4), (2.5), (2.7), and $z_j \in \{0,1\}$ $\forall j \in \{0, \ldots, T-1\}$ is the convex combination formulation.

To compare the incremental cost and the convex combination formulations, Padberg expressed the convex combination formulation in terms of the variables of the incremental cost formulation as

$$u_j y_{j+1} \leq u_{j+1} y_j \quad \forall j \in \{1, \ldots, T-1\}, \tag{2.8}$$

$$u_1 z_1 \leq y_1 \leq u_1, \; 0 \leq y_T \leq u_T z_{T-1}, \; \frac{y_1}{u_1} - \frac{y_2}{u_2} \leq 1 - z_2, \; \frac{y_{T-1}}{u_{T-1}} - \frac{y_T}{u_T} \leq z_{T-2}, \; z_1 \geq z_2, \tag{2.9}$$

$$\frac{y_j}{u_j} - \frac{y_{j+1}}{u_{j+1}} \leq z_{j-1} - z_{j+1} \text{ and } z_j \geq z_{j+1} \; \forall j \in \{2, \ldots, T-2\}. \tag{2.10}$$

We denote the feasible set of the LP relaxation of the convex combination formulation in terms of the (y, z) variables by

$$CB = \{(y, z) \in R^T \times [0,1]^{T-1} : (y, z) \text{ satisfies } (2.8) - (2.10)\}.$$

The main result of [41] is

Theorem 7. *The incremental cost formulation is locally ideal, i.e., $z \in \{0,1\}^{T-1}$ for all extreme points of IB. The set IB is properly contained in CB. The set CB has extreme points (y,z) with $z \notin \{0,1\}^{T-1}$.* □

2.2 Two Related Relaxations and Bound Comparisons

In this section we show that the LP bounds of both MIP formulations, as well as the bounds of the continuous formulations are the same. We also show that the two MIP formulations correspond to the same polyhedron in the space of the continuous variables, and that both continuous formulations are locally ideal.

Beale and Tomlin [4] suggested formulating PLFs with the λ variables and constraints (2.5) in the model, and branching on SOS2 to enforce (2.6). For example, if $\{\lambda_0, \ldots, \lambda_T\}$ is SOS2, $(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_T)$ is the current solution with $\tilde{\lambda}_r, \tilde{\lambda}_s > 0$ for $r, s \in \{0, \ldots, T\}$, $r < s$, and $|r - s| \geq 2$, one can branch by requiring

$$\lambda_0 = \cdots = \lambda_r = 0$$

in one branch and

$$\lambda_{r+2} = \cdots = \lambda_T = 0$$

in the other branch. See [4] for effective branching strategies with SOS2. Now note that the vertices of

$$CN = \{\lambda \in \Re^{T+1} : \lambda \text{ satisfies (2.5)}\}$$

are the $(T+1)$-dimensional unit vectors, and thus satisfy (2.6). Therefore the SOS2 formulation is locally ideal.

Projecting *IB* onto the y space we obtain

$$IN = \{y \in \Re^T : y_1 \leq u_1, y_T \geq 0, \text{ and } y \text{ satisfies (2.8)}\}.$$

Thus, another way to formulate PLFs is to require $y \in IN$ and to enforce (2.2) with the following branching scheme. If $\tilde{y}$ is the current solution with $\tilde{y}_j < u_j$, and $\tilde{y}_{j+k} > 0$ for some $k \in \{1, \ldots T-j\}$, we require

$$y_j = u_j$$

in one branch, and

$$y_{j+1} = \cdots = y_T = 0$$

in the other branch. The vertices of *IN* are $(0, \ldots, 0)$, $(u_1, 0, \ldots, 0)$, $(u_1, u_2, 0, \ldots, 0)$, ..., $(u_1, \ldots, u_T)$. Thus, the vertices of *IN* satisfy (2.2), and the above continuous formulation, which we call SOSX, is locally ideal.

We now establish the relation between the bounds of the incremental cost, convex combination, SOS2, and SOSX formulations. Let f_{IB} =min$\{f(x)$: x satisfies (2.1) and $(y, z) \in IB\}$, f_{CB} =min$\{f(x)$: x satisfies (2.1) and $(y, z) \in CB\}$, f_{CN} =min$\{f(x) : x$ satisfies (2.4) and $\lambda \in CN\}$, and f_{IN} =min$\{f(x) : x$ satisfies (2.1) and $y \in IN\}$.

Theorem 8. *The SOS2 and SOSX formulations are locally ideal. Also,* $f_{IB} = f_{CB} = f_{CN} = f_{IN}$.

Proof The first statement has been proven already. Let $\tilde{\lambda} \in CN$ and $\tilde{y}_j = u_j \sum_{i=j}^{t} \tilde{\lambda}_i$, $j \in \{1, \ldots, T\}$. Since $\tilde{y} \in IN$ and

$$f(a_0) + \sum_{j=1}^{T} \frac{g_j}{u_j} \tilde{y}_j = f(a_0) + \sum_{j=1}^{T} g_j \sum_{i=j}^{T} \tilde{\lambda}_i$$

$$= f(a_0)(1 - \sum_{i=1}^{T} \tilde{\lambda}_i) + \sum_{j=1}^{T} f(a_j)\tilde{\lambda}_j = \sum_{j=0}^{T} f(a_j)\tilde{\lambda}_j,$$

thus $f_{IN} \leq f_{CN}$. Likewise, if $\hat{y} \in IN$ and

$$\hat{\lambda}_j = \frac{\hat{y}_j}{u_j} - \frac{\hat{y}_{j+1}}{u_{j+1}} \; \forall j \in \{1, \ldots, T-1\}, \hat{\lambda}_T = \frac{\hat{y}_T}{u_T} \text{ and } \hat{\lambda}_0 = 1 - \sum_{i=1}^{T} \hat{\lambda}_i,$$

then $\hat{\lambda} \in CN$ and

$$\sum_{j=0}^{T} f(a_j)\hat{\lambda}_j = f(a_0) + \sum_{j=1}^{T} \frac{g_j}{u_j} \hat{y}_j,$$

and $f_{IN} = f_{CN}$.

From Theorem 7 it follows that the LP bound obtained by the incremental cost formulation cannot be worse than the LP bound obtained by the convex combination formulation. Therefore $f_{CB} \leq f_{IB}$. Now, it is clear that $(y, z) \in CB \Rightarrow y \in IN$, and since z does not appear in the objective function, $f_{IN} \leq f_{CB}$. Finally, let $\bar{y} \in IN$ and

$$\bar{z}_j = \frac{\bar{y}_j}{u_j} \; \forall j \in \{1, \ldots, T-1\}.$$

Then, $(\bar{y}, \bar{z}) \in IB$, and since z does not appear in the objective function, $f_{IB} \leq f_{IN}$. □

Theorem 8 shows that the use of binary variables to model PLFs does not tighten the formulation. Finally, as we show next in Corollary 2, Theorem 8 implies that the SOS2 and SOSX formulations are identical in the (x, y)-space as well as in the (x, λ)-space. This implies that there does not seem to be an advantage of one continuous formulation over the other, i.e. SOS2 and SOSX seem to be equally effective. Specifically, let

$$CH = conv(\{(x,\lambda) \in R^{T+2} : (x,\lambda) \text{ satisfies } (2.4) - (2.6)\})$$

denote the convex hull of the SOS2 formulation, and

$$IH = conv(\{(x,y) \in R^{T+1} : (x,y) \text{ satisfies } (2.1) \text{ and } (2.2)\})$$

the convex hull of the SOSX formulation. Let

$$CH' = \{(x,y) : \text{ for some } (x,\lambda) \in CH,\ y_j = u_j \sum_{i=j}^{T} \lambda_i\ \forall j \in \{1,\ldots,T\}\},$$

which is the set CH in the (x,y)-space, and

$$IH' = \{(x,\lambda) : \text{ for some } (x,y) \in IH,\ \lambda_j = \frac{y_j}{u_j} - \frac{y_{j+1}}{u_{j+1}}\ \forall j \in \{1,\ldots,T-1\},$$

$$\lambda_T = \frac{y_T}{u_T}, \text{ and } \lambda_0 = 1 - \sum_{j=1}^{T} \lambda_j\},$$

which is the set IH in the (x,λ) space.

Corollary 2. *$CH' = IH$ and $IH' = CH$.*

Proof Since all the vertices of IN satisfy (2.2), it follows that

$$IH = \{(x,y) \in \Re^{T+1} : (x,y) \text{ satisfies } (2.1) \text{ and } y \in IN\}.$$

Now, let $(\tilde{x},\tilde{y}) \in CH'$. We have $\tilde{y}_1 \leq u_1$, $\tilde{y}_T \geq 0$, and $\tilde{y}$ satisfies (2.8). So, $\tilde{y} \in IN$, and $CH' \subseteq IH$.

Let $(\hat{x},\hat{y})$ be a vertex of IH, where $\hat{y} = (u_1,\ldots,u_j,0,\ldots,0)$ for some $j \in \{1,\ldots,T\}$ and let $\hat{\lambda}$ be given by

$$\hat{\lambda}_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

The point $(\hat{x},\hat{\lambda}) \in CH$, and the corresponding point of CH' is $(x,\hat{y})$. Thus, $IH \subseteq CH'$. $IH' = CH$ can be shown similarly. □

Since the formulations without binary variables give the same LP bound as those with binary variables, are locally ideal, and are more compact, they should be better models for PLFs than the MIP models. Moreover, when additional constraints are present, the structure of the models without binary variables can be dealt with explicitly in the space of the continuous variables through branching and cuts. Theorem 8 and Corollary 2 seem to indicate that there should be no difference between the convex combination and the incremental cost formulations in the space of the continuous variables.

What is currently missing however, is the ability to tighten the LP relaxation of the model without binary variables through the addition of cuts and thus obtain a branch-and-cut algorithm in the space of the continuous variables. This gap is filled by the results of the following chapters where we first do a polyhedral study of the convex hull of solutions and then present a branch-and-cut algorithm.

CHAPTER III

A BRANCH-AND-CUT ALGORITHM WITHOUT BINARY VARIABLES FOR NONCONVEX PIECEWISE LINEAR OPTIMIZATION

In Chapter 1 we introduced nonconvex piecewise linear optimization problem, what we call *NPLOP*, given by

$$\textit{minimize} \quad \sum_{j \in N} f_j(x_j), \tag{3.1}$$

$$\textit{subject to} \quad \sum_{j \in N} a_{ij} x_j \leq b_i \quad \forall i \in \{1, \ldots, m\}, \tag{3.2}$$

$$x_j \geq 0 \quad \forall j \in N, \tag{3.3}$$

where $N = \{1, \ldots, n\}$ and the $f_j(x_j)$ are piecewise linear functions. In this chapter we give a branch-and-cut algorithm for NPLOPs with the restriction that $a_{ij} \geq 0 \ \forall i, j$ and $f(x_j)$ is continuous $\forall j \in N$. First we will show that NPLOP is NP-hard by reducing the NP-hard subset sum problem to NPLOP.

Theorem 9. *NPLOP is NP-hard.*

Proof We reduce the subset sum problem to NPLOP.

Subset Sum

Instance: Two sets of positive integers $N = \{1, \ldots, n\}$ and $M = \{a_1, \ldots, a_n\}$ and a positive integer b.

Question: Does N have a subset S such that $\sum_{j \in S} a_j = b$?

The solution to Subset Sum is "yes" if and only if the optimal value to

$$Min \quad \sum_{j \in N} f_j(x_j)$$

$$s.t. \quad \sum_{j \in N} a_j x_j \geq b$$

$$0 \leq x_j \leq 1 \quad \forall j \in N$$

where

$$f_j(x_j) = \begin{cases} 2a_j x_j, & \text{if } x_j \leq \frac{1}{2}, \\ a_j, & \text{otherwise} \end{cases}$$

is b. Observe that $\sum_{j \in N} f_j(x_j) \geq \sum_{j \in N} a_j x_j$ and the inequality is satisfied at equality if and only if $x_j \in \{0,1\}$ $\forall j \in N$. Thus, by solving NPLOP we can solve Subset Sum. □

By using a $SOS2$ formulation, we can replace (3.1) - (3.3) with

$$minimize \quad \sum_{j \in N} \sum_{k=0}^{T} f_j(a_j^k) \lambda_j^k, \tag{3.4}$$

$$subject\ to \quad \sum_{j \in N} \sum_{k=0}^{T} \gamma_{ij}^k \lambda_j^k \leq b_i \quad \forall i \in \{1, \ldots, m\}, \tag{3.5}$$

$$\sum_{k=0}^{T} \lambda_j^k = 1 \quad \forall j \in N, \tag{3.6}$$

$$\lambda_j^k \geq 0 \quad \forall j \in N,\ \forall k \in \{0, \ldots, T\}, \tag{3.7}$$

$$\{\lambda_j^k : k \in \{0, \ldots, T\}\}\ is\ an\ SOS2\ set\ \forall j \in N, \tag{3.8}$$

where $\gamma_{ij}^k = a_{ij} a_j^k$.

In this chapter we give a branch-and-cut algorithm for solving the model defined by (3.4) - (3.8). We derive the cuts by considering a knapsack relaxation that includes one inequality from (3.5), which we denote generically as

$$\sum_{j \in N} \sum_{k=0}^{T} a_j^k \lambda_j^k \leq b, \tag{3.9}$$

and all of (3.6) - (3.8). We assume that $b \geq a_j^T \ \forall j \in N$ (otherwise $\lambda_j^T < 1$), $a_j^k \geq 0 \ \forall j \in N, \ k \in \{1,\ldots,T\}$ and $a_j^0 = 0 \ \forall j \in N$. It is convenient to work with a full-dimensional polytope because it has a unique inequality (to within scalar multiplication) for each facet of P. Therefore we eliminate $\lambda_j^0 \ \forall j \in N$ and replace (3.6) with

$$\sum_{k=1}^{T} \lambda_j^k \leq 1 \quad \forall j \in N. \tag{3.10}$$

Then to accommodate (3.10) we must require $\sum_{k=1}^{T} \lambda_j^k = 1$ if $\lambda_j^k > 0$ for some $k > 1$. Specifically, λ_j needs to satisfy

$$if\ \lambda_j^k > 0\ for\ k > 1\ then\ \lambda_j^{k-1} + \lambda_j^k + \lambda_j^{k+1} = 1\ for\ k < T,\ and\ \lambda_j^{k-1} + \lambda_j^k = 1\ for\ k = T,$$

in addition to *SOS2*. We call this new set *SOS2'*. Let $K = \{1, 2, \ldots, T\}$. Finally, the polyhedron that we study to obtain cuts is the convex hull of the system given by

$$\sum_{j \in N} \sum_{k \in K} a_j^k \lambda_j^k \leq b, \tag{3.11}$$

$$\sum_{k \in K} \lambda_j^k \leq 1 \ \forall j \in N, \tag{3.12}$$

$$\lambda_j^k \geq 0 \ \forall j \in N, k \in K, \tag{3.13}$$

$$\{\lambda_j^k : k \in K\}\ is\ a\ SOS2'\ set\ \forall j \in N. \tag{3.14}$$

Let $P = conv\{\lambda : \lambda\ satisfies\ (3.11), \ldots, (3.14)\}$. In Section 3.1 we give trivial facets for P and show how we can lift the convexity constraints. In Section 3.2 we give a family of cover inequalities that are facet-defining in a lower dimensional space and then we improve these inequalities by sequential lifting. In Section 3.3 we present a branch-and-cut algorithm for linear programs with piecewise linear cost functions that uses the cuts developed in Sections 3.1 and 3.2 and SOS2 branching. In Section 3.4 we present computational results for solving transportation problems

with piecewise linear costs. These results clearly demonstrate that our approach is significantly better than both an SOS2 branch-and-bound algorithm (i.e. without binary variables and without cuts) and a branch-and-cut algorithm that uses a mixed integer programming formulation.

3.1 Simple Inequalities

In this section we give some simple families of facet-defining inequalities for P. The proofs of propositions 2-5 are easy and are omitted.

Proposition 2. *P is full-dimensional.*

Proposition 3. *All of the inequalities of (3.13) are facet-defining for P.*

Proposition 4. *(3.11) is a facet-defining inequality for P if and only if*

$$a_j^1 + \sum_{i \in N-\{j\}} a_i^T \geq b \quad \forall j \in N.$$

Proposition 5. $\forall j \in N$, *(3.12) is facet-defining for*

$$P_j' = \{\lambda \in P : \lambda_i^k = 0 \quad \forall i \in N - \{j\}, k \in K\}. \tag{3.15}$$

An inequality of (3.12) is facet-defining for P for $j \in N$ if and only if

$$a_j^1 + a_i^{T-1} < b \quad \forall i \in N - \{j\}. \tag{3.16}$$

We next lift the convexity constraints with respect to the variables that are fixed to zero in P_j'. Let $I = \{i \in N-\{j\} : b-a_j^1 \leq a_i^T\}$ and $k_i = min\ \{k \in K : b-a_j^1 \leq a_i^k\}$ $\forall i \in I$. For any $i \in N - \{j\} - I$ the lifting coefficients of λ_i^k, $k \in K$, are zero since $\lambda_i^k > 0$ does not imply $\lambda_j^1 < 1$.

For any $i \in I$, we lift in the order

$$\lambda_i^1, \lambda_i^2, \ldots, \lambda_i^{k_i-2}, (\lambda_i^{k_i-1}, \lambda_i^{k_i}), \lambda_i^{k_i+1}, \ldots, \lambda_i^T,$$

where $(\lambda_i^{k_i-1}, \lambda_i^{k_i})$ states that these two variables are lifted simultaneously. Note that $a_i^{k_i-1} < b - a_j^1 \leq a_i^{k_i}$. The lifting coefficients of λ_i^k, $k < k_i - 1$, are zero since $\lambda_i^k > 0$ does not imply $\lambda_j^1 < 1$.

We next simultaneously lift the variables $\lambda_i^{k_i-1}$ and $\lambda_i^{k_i}$. Let $\alpha_i^{k_i-1}$ and $\alpha_i^{k_i}$ be the lifting coefficients of $\lambda_i^{k_i-1}$ and $\lambda_i^{k_i}$ respectively. For $z \in [0, b]$ let

$$h_i(z) = max \quad \alpha_i^{k_i-1}\lambda_i^{k_i-1} + \alpha_i^{k_i}\lambda_i^{k_i}$$

$$s.t. \quad a_i^{k_i-1}\lambda_i^{k_i-1} + a_i^{k_i}\lambda_i^{k_i} = z$$

$$\lambda_i^{k_i-1} + \lambda_i^{k_i} = 1$$

$$\lambda_i^{k_i-1} \geq 0, \ \lambda_i^{k_i} \geq 0,$$

and

$$f_i(z) = min \quad 1 - \sum_{k \in K} \lambda_j^k$$

$$s.t. \quad \sum_{k \in K} a_j^k \lambda_j^k \leq b - z$$

$$\sum_{k \in K} \lambda_j^k \leq 1$$

$$\lambda_j^k \geq 0 \ \forall k \in K$$

$$\{\lambda_j^k : k \in K\} \ is \ an \ SOS2' \ set.$$

Proposition 6.

$$f_i(z) = \begin{cases} 0, & if \ z \leq b - a_j^1, \\ 1 - \frac{b-z}{a_j^1}, & if \ b - a_j^1 \leq z \leq b \ . \end{cases}$$

Proof Since $\sum_{k \in K} \lambda_j^k \leq 1$, we have $f_i(z) \geq 0$. When $z \leq b - a_j^1$, $\lambda_j^1 = 1$ is feasible and we have $f_i(z) = 0$. If $z > b - a_j^1$, then the maximum value λ_j^1 can take is $\frac{b-z}{a_j^1}$ and we obtain the result. □

Gu, Nemhauser, and Savelsbergh [28] states that we obtain lifting coefficients that define facet inducing inequalities if and only if the pair $(\alpha_i^{k_i-1}, \alpha_i^{k_i})$ is such that $h_i(z) = f_i(z)$ has two solutions $\widehat{\lambda}$ and $\overline{\lambda}$ are such that $(\widehat{\lambda}_i^{k_i-1}, \widehat{\lambda}_i^{k_i})$ and $(\overline{\lambda}_i^{k_i-1}, \overline{\lambda}_i^{k_i})$ are

linearly independent. Since we have a closed form expression for the function f_i, we can compute the set H_i of lifting coefficients $(\alpha_i^{k_i-1}, \alpha_i^{k_i})$ that define facet inducing inequalities. The points $0, b - a_j^1$, and b define the lower envelope of the function $f_i(z)$. Furthermore, the pairs of adjacent points on the lower envelope, i.e., $(0, b-a_j^1)$ and $(b - a_j^1, b)$, define the sets of two values z for which $h_i(z) = f_i(z)$ and for which the associated solutions $\widehat{\lambda}$ and $\overline{\lambda}$ are such that $(\widehat{\lambda}_i^{k_i-1}, \widehat{\lambda}_i^{k_i})$ and $(\overline{\lambda}_i^{k_i-1}, \overline{\lambda}_i^{k_i})$ are linearly independent. We get the set H_i by computing the slopes and the intercepts of the lines defining the lower envelope.

Observe that $h_i(z)$ is defined for $z \in [a_i^{k_i-1}, a_i^{k_i}]$ and we have $h_i(z) = \alpha_i^{k_i-1} + (\alpha_i^{k_i} - \alpha_i^{k_i-1})\lambda_i^{k_i}$ and $z = a_i^{k_i-1} + (a_i^{k_i} - a_i^{k_i-1})\lambda_i^{k_i}$ when $z \in [a_i^{k_i-1}, a_i^{k_i}]$. The slope and intercept of the line defined by the $(z, f_i(z))$ points $(0,0)$ and $(b - a_j^1, 0)$ are 0 and 0 respectively and gives the lifting coefficient pair $(0,0)$. The slope and intercept of the line defined by the points $(b - a_j^1, 0)$ and $(b, 1)$ are $\frac{1}{a_j^1}$ and $\frac{a_j^1 - b}{a_j^1}$ respectively. If $b - a_j^1 < a_i^{k_i}$, from the line

$$h_i(z) = \frac{a_j^1 - b}{a_j^1} + \frac{1}{a_j^1} z = \frac{a_i^{k_i-1} + a_j^1 - b}{a_j^1} + \frac{a_i^{k_i} - a_i^{k_i-1}}{a_j^1} \lambda_i^{k_i}$$

we obtain the lifting coefficient pair $(\frac{a_i^{k_i-1}+a_j^1-b}{a_j^1}, \frac{a_i^{k_i}+a_j^1-b}{a_j^1})$. As a result we obtain

$$H_i = \begin{cases} \{(0,0), (\frac{a_i^{k_i-1}+a_j^1-b}{a_j^1}, \frac{a_i^{k_i}+a_j^1-b}{a_j^1})\}, & \text{if } b - a_j^1 < a_i^{k_i}, \\ \{(0,0)\}, & \text{if } b - a_j^1 = a_i^{k_i}. \end{cases} \tag{3.17}$$

Finding the set H_i is illustrated in Figure 3. The line $h_i^1(z)$ corresponds to lifting coefficients (0,0) and the line $h_i^2(z)$ corresponds to lifting coefficients

$$(\frac{a_i^{k_i-1} + a_j^1 - b}{a_j^1}, \frac{a_i^{k_i} + a_j^1 - b}{a_j^1}).$$

The lifting coefficient of λ_i^l, $l > k_i$ is the greatest value of α_i^l for which

$$\alpha_i^{l-1}\lambda_i^{l-1} + \alpha_i^l \lambda_i^l + \lambda_j^1 \leq 1$$

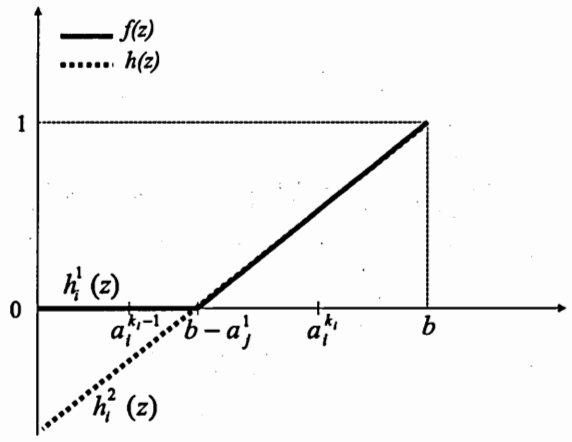

Figure 3: Finding the set H_i

$\forall \lambda \in \{P : \lambda_i^k = 0 \ k > l, \ \lambda_v^k = 0 \quad \forall v \in I - \{i\}, k \in K\}$. When $\lambda_i^l = 0$ there is no restriction on the value of α_i^l. So suppose $\lambda_i^{k_i} > 0$. The greatest value of α_i^l is obtained when $\lambda_i^l = 1$ and λ_j^1 is at its greatest possible value when $\lambda_i^l = 1$. So from

$$\alpha_i^l + \frac{b - a_i^l}{a_j^1} \leq 1,$$

we obtain

$$\alpha_i^l = 1 - \frac{(b - a_i^l)}{a_j^1}.$$

If $a_i^{k_i-1} + a_{i'}^{k_{i'}-1} > b \ \forall \ i, i' \in I, \ i \neq i'$, then the lifting coefficients of each variable can be found independently since when $\lambda_i^{k_i} > 0$, we have $\lambda_{i'}^{k_i'} = 0$.

We next give a proposition that summarizes the above discussion. The proof also follows from this discussion because the lifting is maximum.

Proposition 7. *For $j \in N$, let $I = \{i \in N - \{j\} : b - a_j^1 \leq a_i^T\}$, $k_i = min\ \{k \in K : b - a_j^1 \leq a_i^k\}\ \forall i \in I$, and V be a subset of I such that if $|V| > 1$, then $a_v^{k_v-1} + a_{v'}^{k_{v'}-1} > b \ \forall v, v' \in V, \ v \neq v'$. Then*

$$\sum_{k \in K} \lambda_j^k + \sum_{v \in V} \sum_{k=k_v-1}^{T} \alpha_v^k \lambda_v^k \leq 1, \tag{3.18}$$

with

$$(\alpha_v^{k_v-1}, \alpha_v^{k_v}) \in H_v \; \forall v \in V, \quad \alpha_v^k = 1 - \frac{(b - a_v^k)}{a_j^1} \;\; k > k_v \; \forall v \in V,$$

where the H_v are given by (3.17), is a valid inequality. (3.18) is a facet-defining inequality for P if $V = I$.

Example 1 Let $|N| = 4$, $T = 3$, and (3.11) be

$$2\lambda_1^1 + 6\lambda_1^2 + 8\lambda_1^3 + 3\lambda_2^1 + 7\lambda_2^2 + 10\lambda_2^3 + 4\lambda_3^1 + 8\lambda_3^2 + 10\lambda_3^3 + 5\lambda_4^1 + 7\lambda_4^2 + 9\lambda_4^3 \leq 10. \quad (3.19)$$

Let $P = conv\{\lambda : \lambda \; satisfies \; (3.12), (3.13), (3.14), \; and \; (3.19)\}$. When $j = 1$, we have $I = V = \{2, 3, 4\}$ where $k_2 = k_4 = 3$ and $k_3 = 2$. Since

$$H_2 = \{(0,0), (-\frac{1}{2}, 1)\},$$

$$H_3 = \{(0,0)\},$$

and

$$H_4 = \{(0,0), (-\frac{1}{2}, \frac{1}{2})\}$$

the inequalities

$$\lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_3^3 \leq 1,$$

$$\lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_3^3 - \frac{1}{2}\lambda_2^2 + \lambda_2^3 \leq 1,$$

$$\lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_3^3 - \frac{1}{2}\lambda_4^2 + \frac{1}{2}\lambda_4^3 \leq 1,$$

$$\lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_3^3 - \frac{1}{2}\lambda_2^2 + \lambda_2^3 - \frac{1}{2}\lambda_4^2 + \frac{1}{2}\lambda_4^3 \leq 1$$

are facet-defining for P. Note that each of these inequalities uses one element from H_i, for $i = 2, 3, 4$. □

3.2 Lifted Cover Inequalities

In this section we study a family of cover inequalities that are easily shown to be facet-defining when some variables are fixed at zero. We also show how these inequalities can be improved by lifting.

Definition 11. *Let $2 \leq k_j \leq T \; \forall j \in N$ and $C \subseteq N$ be a set such that*

$$\sum_{j \in C} a_j^{k_j} > b, \text{ and } \sum_{j \in C - \{i\}} a_j^{k_j} \leq b, \; \forall i \in C.$$

The set C is called a cover.

Proposition 8. *Let C be a cover. The cover inequality*

$$\sum_{j \in C} (\alpha_j \lambda_j^{k_j - 1} + \lambda_j^{k_j}) \leq |C| - 1 \tag{3.20}$$

is valid where

$$\alpha_j = \begin{cases} 0, & \text{if } N_j \leq a_j^{k_j - 1}, \\ \frac{N_j - a_j^{k_j - 1}}{b - \sum_{i \in C} a_i^{k_i}}, & \text{otherwise,} \end{cases} \tag{3.21}$$

and $N_j = b - \sum_{i \in C - \{j\}} a_i^{k_i}$. (3.20) is facet-defining for

$$P' = conv\{\lambda \in P : \lambda_j^k = 0 \; k > k_j, \; k \in K, \; \forall j \in C, \lambda_j^k = 0 \; \forall j \in N - C, \; k \in K\},$$

if and only if

$$a_j^{k_j - 2} < N_j \quad \forall \, j \in C. \tag{3.22}$$

Proof For any $j \in C$, let $\lambda_i^{k_i} = 1, \; \forall i \in C - \{j\}$. If $N_j > a_j^{k_j - 1}$, the greatest value of $\lambda_j^{k_j}$ is $\frac{N_j - a_j^{k_j - 1}}{a_j^{k_j} - a_j^{k_j - 1}}$, otherwise it is zero. Since for both cases $\alpha_j \lambda_j^{k_j - 1} + \lambda_j^{k_j} = 0$, (3.20) is valid.

We now show that (3.20) is facet-defining for P' when (3.22) is satisfied by giving $\sum_{j \in C} k_j$ linearly independent points. For each $j \in C$, let $v_j^0 \in P'$ be a point such that $\lambda_i^{k_i} = 1, \; \forall i \in C - \{j\}$ and $\lambda_j^k = 0 \; k \leq k_j$, $\{v_j^l\} \in P'$ be points such that $\lambda_i^{k_i} = 1, \; \forall i \in C - \{j\}$ and $\lambda_j^l = 1$, for $l = 1, \ldots, k_j - 2$, and let $v_j^{k_j - 1} \in P'$ be a point

such that $\lambda_i^{k_i} = 1,\ \forall i \in C - \{j\}$ and $\lambda_j^{k_j-1} = 1$ if $N_j \leq a_j^{k_j-1}$, and $\lambda_j^{k_j} = \frac{N_j - a_j^{k_j-1}}{a_j^{k_j} - a_j^{k_j-1}}$ and $\lambda_j^{k_j-1} = 1 - \lambda_j^{k_j}$ otherwise. Note that if (3.22) is not satisfied for a $j \in C$, then $v_j^{k_j-1} \notin P'$.

Suppose that the points $v_j^l,\ l < k_j,\ \forall j \in C$ lie on the plane $\sum_{j\in C}\sum_{k=1}^{k_j} \alpha_j^k \lambda_j^k = |C| - 1$. The points $v_j^0, \forall j \in C$ imply

$$\alpha_j^{k_j} = 1,\ \forall j \in C. \tag{3.23}$$

From the points v_j^l, $1 \leq l \leq k_j - 2\ \forall j \in C$ and (3.23) we obtain

$$\alpha_j^k = 0\ k < k_j - 1,\ \forall j \in C, \tag{3.24}$$

From the points $v_j^{k_j-1}\ \forall j \in C$ and (3.23) we obtain

$$\alpha_j = \begin{cases} 0, & \text{if } N_j \leq a_j^{k_j-1}, \\ \frac{N_j - a_j^{k_j-1}}{a_j^{k_j} - a_j^{k_j-1}}, & \text{otherwise.} \end{cases}$$

Since we obtain the inequality uniquely by using these points, (3.20) is facet-defining for P'.

Suppose that (3.22) is not satisfied, so that there exists a $j \in C$ such that $b - \sum_{i\in C-\{j\}} a_i^{k_i} \leq a_j^{k_j-2}$. We claim that $\lambda_j^{k_j-1} = 0$ when (3.20) is satisfied at equality. Suppose that $\lambda_j^{k_j-1} > 0$ is feasible when (3.20) is satisfied at equality. Since $\sum_{i\in C-\{j\}} a_i^{k_i} \geq b - a_j^{k_j-2}$, we can not have $\lambda_i^{k_i} = 1,\ \forall i \in C - \{j\}$. Therefore we must have $\lambda_j^{k_j} > 0$ and there should exist an extreme point such that both $\lambda_j^{k_j-1}$ and $\lambda_j^{k_j}$ are positive and (3.20) is satisfied at equality. In this extreme point, all other variables will be at either zero or one. This can only be satisfied when $\lambda_j^{k_j} = 1$. Therefore we have $\lambda_j^{k_j-1} = 0$ and (3.20) can not be facet-defining when (3.22) is not satisfied. □

Example 1 (Continued) For (3.19), $C = \{1, 2\}$ with $k_1 = k_2 = 2$ is a cover and therefore the cover inequality

$$-\frac{1}{3}\lambda_1^1 + \lambda_1^2 - \frac{1}{3}\lambda_2^1 + \lambda_2^2 \leq 1 \tag{3.25}$$

is facet-defining for $P' = \{\lambda \in P : \lambda_1^3 = \lambda_2^3 = 0,\ \lambda_j^k = 0\ j \in \{3,4\}, k \in \{1,2,3\}\}$. □

3.2.1 Lifting Approximately

When $k_j < T$, for a $j \in C$, we need to lift the variables that are fixed to zero to obtain strong inequalities for P. In this section we give lower bounds on the lifting coefficients.

Suppose that for $j \notin V$

$$\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k + \sum_{k=1}^{l-1} \alpha_j^k \lambda_j^k \leq \gamma,$$

is a valid inequality when λ_j^k $k \geq l$ are fixed at zero. The lifting coefficient of λ_j^l is the greatest value of α_v^l for which

$$\alpha_j^l \lambda_j^l + \sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k + \alpha_j^{l-1} \lambda_j^{l-1} \leq \gamma.$$

After replacing λ_j^{l-1} with $1 - \lambda_j^l$, we obtain

$$\alpha_j^l \leq \frac{\gamma - \alpha_j^{l-1} - \sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k}{\lambda_j^l} + \alpha_j^{l-1}$$

When $\lambda_j^l = 0$ there is no restriction on the value of the lifting coefficient. So suppose $\lambda_j^l > 0$. Since when $\lambda_j^l > 0$, we consume more of the capacity of the constraint than when λ_j^{l-1} takes the same value, we have $\gamma > \alpha_j^{l-1} + \sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k$. As a result we obtain $\alpha_j^l \geq \alpha_j^{l-1}$.

An immediate consequence of this result is

Corollary 3. *Let C be a cover. The lifted cover inequality*

$$\sum_{j \in C} (\alpha_j \lambda_j^{k_j - 1} + \sum_{k=k_j}^{T} \lambda_j^k) \leq |C| - 1 \tag{3.26}$$

is valid where the α_j are given by (3.21).

Example 1 (Continued) The approximate lifting of (3.25) by Corollary 3 yields the valid inequality

$$-\frac{1}{3}\lambda_1^1 + \lambda_1^2 + \lambda_1^3 - \frac{1}{3}\lambda_2^1 + \lambda_2^2 + \lambda_2^3 \leq 1.$$

3.3 Branch-and-Cut Algorithm

In this section we present a branch-and-cut algorithm for solving linear programs with piecewise linear cost functions. Branch-and-cut is a generalization of branch-and-bound. After solving the LP relaxation, if we can not fathom a node, we try to find a violated cut. If we can find one or more violated cuts, we add them to the LP relaxation and solve it again. If we can not find a violated cut, we branch. In addition, we use a primal heuristic to increase the possibility of fathoming nodes by bounding.

3.3.1 Primal Heuristic

We give a procedure that modifies an LP solution that does not satisfy SOS2 to one that does. Assume that we have a solution λ such that for at least one $j \in N$ the set $\{\lambda_j^k : k \in K\}$ does not satisfy SOS2. Suppose $x_j = \sum_{k \in K} a_j^k \lambda_j^k$ and $a_j^{k-1} < x_j \leq a_j^k$. We create a new vector λ' given by

$$\lambda_j'^{k-1} = \frac{a_j^k - x_j}{a_j^k - a_j^{k-1}}, \tag{3.27}$$

$$\lambda_j'^{k} = \frac{x_j - a_j^{k-1}}{a_j^k - a_j^{k-1}}, \tag{3.28}$$

$$\lambda_j'^{l} = 0, \quad l = 1, \ldots, k-2, k+1, \ldots, T. \tag{3.29}$$

This new vector satisfies the SOS2 constraint for j. Since we haven't changed x_j, the solution is feasible. By applying the above procedure to all $j \in N$ not satisfying SOS2 and assigning $\lambda_j^k = \lambda_j'^k \; \forall k \in K$ for all $j \in N$ satisfying SOS2, we obtain a feasible vector λ'.

3.3.2 Branching

We apply SOS2 branching. We first determine the variable j to branch on and then for a given j we determine the variable λ_j^k to branch on.

Assume that we have a solution λ such that for at least one $j \in N$, the set $\{\lambda_j^k : k \in K\}$ does not satisfy SOS2. Let $c_j^k = f(x_j^k)$. Currently the cost of x_j is $C_j = \sum_{k \in K} c_j^k \lambda_j^k$ which is lower than the true cost which is $C_j' = \sum_{k \in K} c_j^k \lambda_j'^k$ when $\lambda_j'^k$ are given by (3.27) - (3.29). The difference between C_j and C_j' is the estimated increase in the optimal cost value after we branch on j. By choosing the j that gives the maximum $C_j' - C_j$, we hope to obtain a large increase in the LP bound.

The next step is determining the variable to branch on. Suppose that x_j is the value of variable j and we have $a_j^k < x_j \leq a_j^{k+1}$. Along one branch, we assign

$$\lambda_j^l = 0, \quad for\, l = 1, \ldots, k-1,$$

and in the other we assign

$$\lambda_j^l = 0, \quad for\, l = k+1, \ldots, T.$$

3.3.3 Creating Cuts

If λ_j violates SOS2, then we have a k_j such that $\lambda_j^{k_j} > 0$ and $\lambda_j^{k_j-1} = \lambda_j^{k_j+1} = \ldots = \lambda_j^T = 0$. We may be able to derive a violated cut from constraint i if $a_{ij} > 0$.

For each $v \in N - \{j\}$ that satisfies $\sum_{k \in K} \lambda_v^k = 1$, we try to create a lifted convexity cut. If $a_{iv}^1 + a_{ij}^{k_j-1} \geq b$, then

$$\sum_{k \in K} \lambda_v^k + \sum_{k=k_j}^{T} \alpha_j^k \lambda_j^k \leq 1 \tag{3.30}$$

with

$$\alpha_j^k = 1 - \frac{b - a_{ij}^k}{a_{iv}^1}$$

cuts off the infeasible point. If $a_{ij}^{k_j-1} < b - a_{iv}^1 < a_{ij}^{k_j}$, then

$$\sum_{k \in K} \lambda_v^k + \alpha_j^{k_j-1}\lambda_j^{k_j-1} + \alpha_j^{k_j}\lambda_j^{k_j} \leq 1 \tag{3.31}$$

with

$$\alpha_j^{k_j} = 1 - \frac{b - a_{ij}^{k_j}}{a_{iv}^1} \text{ and } \alpha_j^{k_j-1} = \frac{a_{iv}^1 + a_{ij}^{k_j-1} - b}{a_{iv}^1}$$

cuts off the infeasible point.

To obtain a cover inequality (3.26) for each $v \in N - \{j\}$, we find C by choosing the largest $k_v \in K$ that satisfies $\sum_{k=k_v}^{T} \lambda_v^k = 1$. Let $C = \{v \in N : k_v > 0\}$. Note that $j \in C$. If $\sum_{v \in C} a_{iv}^{k_v} > b$, then we create the cover cut

$$\sum_{v \in C} (\alpha_v^{k_v-1}\lambda_v^{k_v-1} + \sum_{k=k_v}^{T} \lambda_v^k) \leq |C| - 1, \tag{3.32}$$

where the $\alpha_v^{k_v-1}$are defined by (3.21). Since $\lambda_v^{k_v-1} = 0 \ \forall v \in C$, $\lambda_j^{k_j} > 0$, and $\sum_{k=k_v}^{T} \lambda_v^k = 1 \ \forall v \in C - \{j\}$, (3.32) cuts off the infeasible point.

3.4 Computational Experience

Our optimization software is MINTO 3.0 [43] with CPLEX 7.5 as LP solver. We tested our algorithm against standard approaches on instances of transportation problems with concave piecewise linear cost functions. Specifically we compared

- a convex combination formulation with binary variables and MINTO's default setting.
- a convex combination formulation with binary variables, MINTO's default setting and our cuts.

- a convex combination formulation without binary variables with our cuts and $SOS2$ branching.
- a convex combination formulation without binary variables with $SOS2$ branching but without cuts.

We consider the transportation problem

$$Min \quad \sum_{i\in I}\sum_{j\in J} f_{ij}(x_{ij}), \tag{3.33}$$

$$\sum_{j\in J} x_{ij} = s_i, \quad \forall i \in I, \tag{3.34}$$

$$\sum_{i\in I} x_{ij} = d_j, \quad \forall j \in J, \tag{3.35}$$

$$x_{ij} \geq 0, \quad \forall i \in I,\ j \in J, \tag{3.36}$$

where the index sets of supply and demand points are $I = \{1, .., m\}$ and $J = \{1, .., n\}$ respectively. We assume that $f_{ij}(x_{ij})$ is a piecewise linear function and the transportation problem is balanced, i.e $\sum_{i\in I} s_i = \sum_{j\in J} d_j$.

Let the index set of partitions be $K' = \{0, \ldots, T\}$. We can reformulate (3.33)-(3.36) as

$$Min \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K'} c_{ij}^k \lambda_{ij}^k, \tag{3.37}$$

$$\sum_{j\in J}\sum_{k\in K'} a_{ij}^k \lambda_{ij}^k = s_i, \qquad \forall i \in I, \tag{3.38}$$

$$\sum_{i\in I}\sum_{k\in K'} a_{ij}^k \lambda_{ij}^k = d_j, \qquad \forall j \in J, \tag{3.39}$$

$$\sum_{k \in K'} \lambda_{ij}^k = 1, \qquad \forall i \in I,\, j \in J, \tag{3.40}$$

$\lambda_{ij}^k \geq 0,\ \forall i \in I,\ j \in J,\ k \in K',\ and\ \{\lambda_{ij}^k : k \in K\}\ satisfy\ SOS\ type\ II\ \forall (i,j) \in IxJ.$ (3.41)

We assume that $a_{ij}^0 = c_{ij}^0 = 0, \forall (i,j) \in IxJ$, $a_{ij}^k \geq 0, \forall (i,j) \in IxJ\ k \in K'$ and $a_{ij}^T \leq Min\{s_i, d_j\}$. We use single constraint relaxations of the transportation problem to find valid inequalities.

3.4.1 Test Instances

We tested 20 instances. The instances are randomly generated as follows. Let T be the number of partitions in the piecewise linear function. We used T=4 and 5. The demands are integers uniformly distributed between T+1 and T+20. The supplies are integers uniformly distributed between T+1 and $T + (20 * \frac{|J|}{|I|})$. $s_{|I|}$ and $d_{|J|}$ are modified to obtain a balanced problem. For any variable (i,j), $a_{ij}^T = min\{s_i, d_j\}$. To find the other a_{ij}^k's, the interval $[0, a_{ij}^T]$ is divided into T partitions randomly. The costs are concave so that for any (i,j) the slopes of the partitions decrease as k increase. c_{ij}^{T-l} is randomly chosen from the $[5(l-1), 5l]$.

3.4.2 Computational Results

We used a Sun Ultra Sparc 60 workstation with dual UltraSparc II 450 MHz CPUs and 4MB cache to perform computational tests. The results are summarized in Tables 1 and 2. The instances are named in axbxc.d format where a,b,c and d correspond to the number of supply points, number of demand points, number of partitions, and seed number respectively. For each instance we give the number of nodes required to solve it and the CPU time in seconds. The instances are terminated after 10,000

CPU seconds. * indicates that the problem has been terminated without proving optimality. For these instances we give the number of nodes that has been evaluated and the optimality gap at the time of termination. For the algorithms that use our cuts, we also give the number of cuts created. We also give total number of nodes and total CPU time in each table. We excluded the instances that has been terminated without proving optimality when calculating the totals. Table 1 gives results for two algorithms that use the convex combination formulation without binary variables. The algorithms are a branch-and-bound algorithm with $SOS2$ branching and the branch-and-cut algorithm given in Section 3.3. Table 2 gives results for the convex combination formulation with binary variables with and without our cuts. For both cases we also use MINTO's default setting (i.e. MINTO's default branching, cuts, primal heuristics, etc.).

The results show that the branch-and-cut algorithm given in Section 3.3 is significantly better than both an SOS2 branch and bound algorithm and a branch-and-cut algorithm that uses a mixed integer programming formulation. The branch-and-cut algorithm given in Section 3.3 reduced the total number of nodes by 99% and the total CPU time by 90 % compared to the branch-and-bound algorithm with $SOS2$ branching. Table 2 shows that our cuts reduced the computational time and the number of nodes processed significantly even if we use a mixed integer formulation. With the addition of our cuts, the total number of nodes was reduced by 89 %, and the total CPU time was reduced by 79 %. Finally in comparing the results with our cuts from tables 1 and 2, we have that the formulation without binary variables reduced the number of nodes by 24 % and the time by 48 %. All of these percentages are conservative since only the instances that terminated are considered.

The branch-and-cut algorithm given in Section 3.3 determines the variable to branch on by finding the estimated increase for each $j \in N$ such that λ_j does not satisfy $SOS2$. We tested 5 instances to see what happens if we choose the smallest

Table 1: Computations without introducing binary variables

	Branch-and-bound			Branch-and-cut					% Red in	
Problem	Nodes	Time	Gap %	Nodes	Time	(3.30)	(3.31)	(3.32)	Nodes	Time
12x18x4.1	166712	*	1.00	16287	6125	182	577	1244		
12x18x4.2	7919	74		259	42	68	172	390	97	43
12x18x4.3	20909	269		961	239	106	258	673	95	11
12x18x4.4	52479	825		1385	476	182	334	816	97	42
12x18x4.5	168734	6942		999	230	101	307	676	99	97
12x18x5.1	159909	*	1.16	3475	1821	473	297	1618		
12x18x5.2	174211	*	0.70	3125	1463	380	305	1369		
12x18x5.3	69831	1920		697	342	318	256	1093	99	82
12x18x5.4	192550	*	0.10	1549	1072	514	316	1470		
12x18x5.5	170175	*	0.79	4425	3187	487	403	1852		
15x15x4.1	168204	*	0.68	2967	947	239	407	890		
15x15x4.2	198791	*	0.08	3067	1028	245	534	920		
15x15x4.3	168438	*	1.82	17657	3676	383	785	1150		
15x15x4.4	15717	158		373	65	96	214	432	98	59
15x15x4.5	27371	340		1001	205	162	326	602	96	40
15x15x5.1	175589	9224		667	244	226	202	950	99	97
15x15x5.2	182864	*	0.53	4961	4100	516	525	2043		
15x15x5.3	110891	3321		897	384	287	271	928	99	88
15x15x5.4	24487	372		509	221	239	247	917	98	40
15x15x5.5	162935	*	0.85	3795	3159	626	426	1823		
Total	673927	23446		7748	2450				99	90

Table 2: Computations when binary variables are introduced

	Without our cuts			With our cuts						% Red in	
Problem	Nodes	Time	Gap %	Nodes	Time	Gap	(3.30)	(3.31)	(3.32)	Nodes	Time
12x18x4.1	49116	*	0.23	21592	*	0.21	155	434	1055		
12x18x4.2	2033	272		444	143		42	155	415	78	48
12x18x4.3	5703	923		1006	470		99	226	653	82	49
12x18x4.4	9361	1906		2547	1137		130	242	726	73	40
12x18x4.5	11581	1803		1351	350		56	170	464	88	81
12x18x5.1	37742	*	0.73	6203	3010		334	218	1211		
12x18x5.2	37384	*	0.47	4152	2956		358	302	1340		
12x18x5.3	27881	6897		1913	1277		258	235	1092	93	82
12x18x5.4	38103	*	0.05	1721	921		278	241	989		
12x18x5.5	35118	*	1.26	14359	9585		406	366	1674		
15x15x4.1	29885	7933		4805	2228		149	272	698	84	72
15x15x4.2	35447	9187		3445	1220		109	304	675	90	87
15x15x4.3	38166	*	1.19	10203	6474		173	430	949		
15x15x4.4	2071	297		441	131		69	171	373	79	56
15x15x4.5	7513	1344		1103	423		114	220	579	85	69
15x15x5.1	30407	*	0.28	3723	3402		444	385	1467		
15x15x5.2	31300	*	0.74	4023	3249		283	325	1309		
15x15x5.3	25683	7596		1299	830		181	204	808	95	89
15x15x5.4	9475	2164		512	293		118	136	657	95	86
15x15x5.5	31730	*	0.96	5608	5540		465	374	1612		
Total	166633	40323		18866	8502					89	79

Table 3: Comparison of branching rules with and without estimated increase

	Without		With	
Problem	Nodes	Time	Nodes	Time
15x15x4.1	*	9685	2967	947
15x15x4.2	32015	4013	3067	1028
15x15x4.3	*	9677	17657	3676
15x15x4.4	12259	1138	373	65
15x15x4.5	4353	407	1001	205

$j \in N$ such that λ_j does not satisfy $SOS2$ to branch on. The results are summarized in Table 3. The instances are terminated after 50,000 nodes. $*$ indicates that the problem has been terminated without proving optimality. For these instances we give CPU time at the time of termination. The results show that looking for the variables that gives the maximum expected increase significantly decreases the number of nodes and the CPU time.

We had two main assumptions in this chapter: PLF is continuous and all coefficients are positive. We can relax the constraints with negative coefficients by dropping the variables with negative coefficients and increasing the right hand side of the constraint with the summation of upper bounds of the variables with negative coefficients multiplied by the absolute value of their coefficients. The relaxation is a constraint with all positive coefficients and the results of this chapter can be used to create cuts. However our intuition is that these cuts will not be strong inequalities. Therefore, further polyhedral study of the single constraint relaxations with both positive and negative coefficients is needed. This polyhedral investigation is the subject of Chapter 4. We will discuss the case where PLF is lower semi-continuous in Chapter 5.

CHAPTER IV

POLYHEDRAL STUDY OF GENERAL PIECEWISE LINEAR OPTIMIZATION

In this chapter we generalize the results of Chapter 3 to a problem where there is no restriction on the signs of the coefficients. Suppose for each x_j the cost function, $f(x_j)$, is piecewise linear, continuous, and specified by the points $(a_j^k, f(a_j^k))$ for $k = 0, .., T$. Let $K = \{1, \ldots, T\}$. In this chapter we give a branch-and-cut algorithm for solving the model defined by (3.4) - (3.8) where the $a_{ij} \geq 0 \ \forall i, j$ restriction of Chapter 3 is relaxed. As in Chapter 3, we use single constraint relaxations to derive cuts. The polyhedron that we study to obtain cuts is the convex hull of the system

$$\sum_{j \in N^+} \sum_{k \in K} a_j^k \lambda_j^k - \sum_{j \in N^-} \sum_{k \in K} a_j^k \lambda_j^k \leq b \tag{4.1}$$

$$\sum_{k \in K} \lambda_j^k \leq 1 \quad \forall j \in N \tag{4.2}$$

$$\lambda_j^k \geq 0 \quad \forall j \in N, k \in K \tag{4.3}$$

$$\{\lambda_j^k : k \in K\} \textit{ is a SOS2' set } \forall j \in N \tag{4.4}$$

where $N = N^+ \cup N^-$. We assume that $b + \sum_{j \in N^-} a_j^T > a_j^T \ \forall j \in N^+$, $a_j^k \geq 0 \ \forall j \in N$, $k \in K$, and $a_j^0 = 0 \ \forall j \in N$. Let $PG = conv\{\lambda : \lambda \textit{ satisfies } (4.1), \ldots, (4.4)\}$. We will find valid inequalities for PG by lifting.

Suppose for each $j \in M \subseteq N^-$ we fix $\lambda_j^{k_j} = 1$, $k_j \in K$ and $\lambda_j^k = 0$, $k \in K - \{k_j\}$. We also fix $\lambda_j^k = 0$, $\forall j \in N^- - M$, $k \in K$. Let $k_j = 0$, $\forall j \in N^- - M$. Then (4.1) can be written as

$$\sum_{j \in N^+} \sum_{k \in K} a_j^k \lambda_i^k \leq b + \sum_{j \in N^-} a_j^{k_j}. \tag{4.5}$$

Note that (4.5) is a constraint with all positive constraints. Let $PG(\beta)$ be the convex hull of

$$\sum_{j\in N^+}\sum_{k\in K} a_j^k\lambda_j^k \le \beta, \tag{4.6}$$

$$\sum_{k\in K}\lambda_j^k \le 1 \quad \forall j \in N^+, \tag{4.7}$$

$$\lambda_j^k \ge 0 \quad \forall j \in N^+, k \in K, \tag{4.8}$$

$$\{\lambda_j^k : k \in K\} \textit{ is a } SOS2' \textit{ set } \forall j \in N^+. \tag{4.9}$$

To find a valid inequality for PG, we first find valid inequalities for $PG(b+\sum_{j\in N^-} a_j^{k_j})$ by using the results of Chapter 3. We then find valid inequalities for PG by lifting the variables that are fixed.

In Section 4.1 we derive valid inequalities for PG by lifting the cover inequalities of Chapter 3. We also give a generalized cover inequality which is valid for PG. In Section 4.2 we derive valid inequalities for PG by lifting the convexity constraints of Chapter 3. In Section 4.3 we show how we can approximately lift a valid inequality for $PG(b+\sum_{j\in N^-} a_j^{k_j})$ to obtain a valid inequality for PG. In Section 4.4 we will present a branch-and-cut algorithm that uses the generalized cover inequalities and $SOS2$ branching. In Section 4.5 computational results are presented for solving network flow problems with piecewise linear costs. These results demonstrate that our approach is significantly better than a branch-and-cut approach that uses a mixed integer formulation.

4.1 Lifting Cover Inequalities

Let $C \in N^+$ be a cover for $PG(b+\sum_{j\in N^-} a_j^{k_j})$ defined by (11) and

$$\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) \le |C| - 1 \tag{4.10}$$

be the corresponding cover inequality given by (3.20). To obtain a valid inequality for PG we lift the variables λ_j^k $\forall j \in N^-$, $k \in K$.

We start with lifting the variables λ_j^k, $k \leq k_j \; \forall j \in N^-$. Since $\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) \leq |C| - 1$ when $\lambda_j^k = 0$, $k > k_j \; \forall j \in N^-$, zero is always a feasible lifting coefficient value for λ_j^k $k \leq k_j \; \forall j \in N^-$. Therefore we use zero as lifting coefficients for the variables λ_j^k, $k \leq k_j \; \forall j \in N^-$. Let $M = Min\{\sum_{j\in C-\{l\}} a_j^{k_j} : l \in C\}$. If $M \leq b + \sum_{i\in N^- - \{j\}} a_i^{k_i}$, then the maximal lifting coefficients for λ_j^k $k \leq k_j \; \forall j \in N^-$ are zero.

Let $\lambda = \sum_{j\in C} a_j^{k_j} - \sum_{j\in N^-} a_j^{k_j} - b$ and $g(\beta) = Max\{\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) : \lambda \in PG(\beta)\}$ for $\beta \in [b + \sum_{j\in N^-} a_j^{k_j}, b + \sum_{j\in N^-} a_j^T]$.

Proposition 9. *If* $\exists j \in C$ *s.t.* $\alpha_j^{k_j-1} < 0$, *then*

$$g(\beta) = \begin{cases} |C| - 1 + \frac{\beta - b - \sum_{j\in N^-} a_j^{k_j}}{\lambda}, & \text{if } b + \sum_{j\in N^-} a_j^{k_j} \leq \beta \leq \sum_{j\in C} a_j^{k_j}, \\ |C|, & \text{if } \beta \geq \sum_{j\in C} a_j^{k_j}. \end{cases}$$

Proof When $\beta = b + \sum_{j\in N^-} a_j^{k_j}$ we have $\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) = |C| - 1$. Suppose we increase β by $l \leq \lambda$. $\lambda_j^{k_j}$ will increase by $\frac{l}{a_j^{k_j} - a_j^{k_j-1}} = \frac{l}{u_j^{k_j}}$.

For a $j \in C$ with $\alpha_j^{k_j-1} < 0$, when $\lambda_j^{k_j} > 0$ and $\lambda_j^{k_j+1} = 0$ we have

$$\alpha_j^{k_j-1}(1 - \lambda_j^{k_j}) + \lambda_j^{k_j} = \alpha_j^{k_j-1} + (1 - \alpha_j^{k_j-1})\lambda_j^{k_j}.$$

Thus the increase in $(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j})$ when we increase $\lambda_j^{k_j}$ by $\frac{l}{u_j^{k_j}}$ will be

$$(1 - \alpha_j^{k_j-1})\frac{l}{a_j^{k_j} - a_j^{k_j-1}} = \frac{l}{\lambda}.$$

If $\alpha_j^{k_j-1} = 0$, then $\lambda \geq u_j^{k_j}$. Thus $\lambda_j^{k_j}$ will increase by zero if $0 \leq l \leq \lambda - u_j^{k_j}$, and $\frac{l - u_j^{k_j}}{u_j^{k_j}}$ if $\lambda - u_j^{k_j} \leq l \leq \lambda$.

If for at least one $j \in C$ we have $\alpha_j^{k_j-1} < 0$, then the maximum increase in $\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j})$ will be $\frac{l}{\lambda}$ when we increase the capacity by l and we obtain the result. □

If for at least one $i \in C$ we have $\alpha_i^{k_i-1} < 0$, then we can use the above proposition to find the lifting coefficients. We lift for an $i \in N^-$ in increasing order of k, $k \in K$. Suppose we are lifting λ_i^k. We want to find a lifting coefficient α_i^k that satisfies

$$\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) + \alpha_i^{k-1}\lambda_i^{k-1} + \alpha_i^k\lambda_i^k \leq |C| - 1$$

$$\Rightarrow \frac{|C| - 1 - \sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k$$

We have three cases to investigate:

Case 1 If $a_i^k - a_i^{k_j} \leq \lambda$, then the increase in $\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j})$ will be $\frac{a_i^k - a_i^{k_i}}{\lambda}$ when $\lambda_i^k = 1$. Thus we find the lifting coefficient α_i^k as $-\frac{a_i^k - a_i^{k_j}}{\lambda}$.

Case 2 If $a_i^{k-1} - a_i^{k_i} < \lambda$ and $a_i^k - a_i^{k_i} \geq \lambda$ then we can have $\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) = |C|$ when

$$0 < \lambda_i^k = \Lambda^* = \frac{\sum_{j \in C} a_j^{k_j} - b - \sum_{j \in N^- - \{i\}} a_j^{k_j} - a_i^{k-1}}{a_i^k - a_i^{k-1}} \leq 1.$$

Thus we find the lifting coefficient as

$$\alpha_i^k = \frac{-1 - \alpha_i^{k-1}}{\Lambda^*} + \alpha_i^{k-1}.$$

Case 3 If $a_i^{k-1} - a_i^{k_j} \geq \lambda$, then $\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) = |C|$, for any value of λ_i^k. Thus we obtain the lifting coefficient of λ_i^k as -1.

The above coefficients are the actual coefficients if $i \in N^-$ is the first index that is being lifted. Suppose i is not the first index being lifted and $L \subseteq N^-$ is the set of indices that have been lifted. To find the actual lifting coefficients we need to find the maximum value

$$\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) + \sum_{j \in L}\sum_{k=k_j}^{T} \alpha_j^k\lambda_j^k \tag{4.11}$$

can take when λ_i^k is at a given value. If λ_j^k is positive for some $j \in L$, then the increase in $\sum_{j \in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j})$ is equal to $-\sum_{j \in L}\sum_{k=k_j}^{T} \alpha_j^k\lambda_j^k$. So we have an

optimal solution when we maximize (4.11) such that $\lambda_j^k = 0\ \forall j \in L$. Thus we can replace (4.11) with

$$\sum_{j \in C} (\alpha_j^{k_j - 1} \lambda_j^{k_j - 1} + \lambda_j^{k_j}).$$

Therefore the lifting coefficients that we have found in cases 1,2 and 3 are the actual coefficients for $\lambda_i^k\ \forall i \in N^-$.

Example 4: Let $|N| = 4$, $T = 3$, and (4.1) be

$$2\lambda_1^1 + 6\lambda_1^2 + 8\lambda_1^3 + 5\lambda_2^1 + 9\lambda_2^2 + 20\lambda_2^3 - 4\lambda_3^1 - 6\lambda_3^2 - 8\lambda_3^3 - 2\lambda_4^1 - 5\lambda_4^2 - 8\lambda_4^3 \leq 10. \quad (4.12)$$

Let $PG = conv\{\lambda : \lambda\ satisfies\ (4.2), (4.3), (4.4),\ and\ (4.12)\}$. Suppose that $k_3 = 1$ and $k_4 = 0$. For $PG(14)$ the cover inequality

$$-3\lambda_1^1 + \lambda_1^2 - 3\lambda_2^1 + \lambda_2^2 \leq 1$$

is valid where $\lambda = 1$. Zero is a feasible value for the lifting coefficient, α_3^1, of λ_3^1, therefore let $\alpha_3^1 = 0$. We next find the lifting coefficient, α_3^2, of λ_3^2. We have the Case 2 and $\Lambda^* = \frac{1}{2}$. Therefore we obtain $\alpha_3^2 = -2$. Since for any value λ_3^3 we can have $\lambda_1^2 = \lambda_2^2 = 1$, the lifting coefficient, α_3^3, of λ_3^3 is -1. We can lift $\lambda_4^k\ k \in K$ similarly and obtain the inequality

$$-3\lambda_1^1 + \lambda_1^2 - 3\lambda_2^1 + \lambda_2^2 - 2\lambda_3^2 - \lambda_3^3 - 2\lambda_4^1 - \lambda_4^2 - \lambda_4^3 \leq 1.$$

□

Suppose that in the cover cut $\alpha_j^{k_j - 1} = 0\ \forall j \in C$. For this case, the lifting coefficients we found above may not work. Let $\widehat{u} = Max_{j \in C}(a_j^{k_j} - a_j^{k_j - 1})$ and $\overline{b} = b + \sum_{j \in N^-} a_j^{k_j}$.

Proposition 10. *If* $\alpha_j^{k_j - 1} = 0\ \forall j \in C$, *then*

$$g(\beta) = \begin{cases} |C| - 1, & if\ \overline{b} \leq \beta \leq \overline{b} + \lambda - \widehat{u}, \\ |C| - 1 + \frac{\beta - \overline{b} - \lambda + \widehat{u}}{\widehat{u}}, & if\ \overline{b} + \lambda - \widehat{u} \leq \beta \leq \sum_{j \in C} a_j^{k_j}, \\ |C|, & if\ \beta \geq \sum_{j \in C} a_j^{k_j}. \end{cases}$$

Proof When $\beta = \bar{b}$ we have $\sum_{j\in C}(\alpha_j^{k_j-1}\lambda_j^{k_j-1} + \lambda_j^{k_j}) = |C| - 1$. Suppose we increase β by l. Since $\alpha_j^{k_j-1} = 0$, $\forall j \in C$, we have $\lambda \geq u_j^{k_j}$, $\forall j \in C$. Thus $\lambda_j^{k_j}$ will increase by zero if $0 \leq l \leq \lambda - u_j^{k_j}$, and $\frac{l-u_j^{k_j}}{u_j^{k_j}}$ if $\lambda - u_j^{k_j} \leq l \leq \lambda$. We obtain the result from the above discussion. □

We use Proposition 10 to find the lifting coefficients. We lift for an $i \in N^-$ in increasing order of k, $k > k_i$. Suppose we are lifting λ_i^k. We want to find a lifting coefficient α_i^k that satisfies

$$\sum_{j\in C} \lambda_j^{k_j} + \alpha_i^{k-1}\lambda_i^{k-1} + \alpha_i^k\lambda_i^k \leq |C| - 1$$

$$\Rightarrow \frac{|C| - 1 - \sum_{j\in C}\lambda_j^{k_j} - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k$$

$$\Rightarrow \frac{|C| - 1 - g(b + \sum_{j\in N^- - \{i\}} a_j^{k_j} + a_i^{k_i-1} + u_i^{k_i}\lambda_i^{k_i}) - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k$$

We have three cases to investigate:

Case 1 If $a_i^k - a_i^{k_i} \leq \lambda$, then $\sum_{j\in C}\lambda_j^{k_j} = g(b + \sum_{j\in N^- - \{i\}} a_j^{k_j} + a_i^{k_i})$ when $\lambda_i^k = 1$. Thus we find the lifting coefficient α_i^k as $|C| - 1 - g(b + \sum_{j\in N^- - \{j\}} a_j^{k_j} + a_i^{k_i})$

Case 2 If $a_i^{k-1} - a_i^{k_i} < \lambda$ and $a_i^k - a_i^{k_i} \geq \lambda$ then we can have $\sum_{j\in C}\lambda_j^{k_j} = |C|$ when

$$0 < \lambda_i^k = \Lambda^* = \frac{\sum_{j\in C} a_j^{k_j} - b - \sum_{j\in N^- - \{i\}} a_j^{k_j} - a_i^{k-1}}{a_i^k - a_i^{k-1}} \leq 1.$$

Thus we find the lifting coefficient as

$$\alpha_i^k = \frac{-1 - \alpha_i^{k-1}}{\Lambda^*} + \alpha_i^{k-1}.$$

Case 3 If $a_i^{k-1} - a_i^{k_j} \geq \lambda$, then $\sum_{j\in C}\lambda_j^{k_j} = |C|$, for any value of λ_i^k. Thus we obtain the lifting coefficient of λ_i^k as -1.

The above coefficients are the actual coefficients if $i \in N^-$ is the first index that is being lifted. Suppose i is not the first index being lifted and $L \subseteq N^-$ is the set of

indices that have been lifted. To find the actual lifting coefficients we need to find the maximum value

$$\sum_{j\in C}\lambda_j^{k_j}+\sum_{j\in L}\sum_{k=k_j}^{T}\alpha_j^k\lambda_j^k \tag{4.13}$$

can take for a given value of λ_i^k. Thus for a given value of λ_i^k we need to solve the problem

$$Max \quad \sum_{j\in C}\lambda_j^{k_j}+\sum_{j\in L}\sum_{k=k_j}^{T}\alpha_j^k\lambda_j^k \tag{4.14}$$

$$s.t. \quad \lambda\in PG \tag{4.15}$$

$$\sum_{j\in C}\sum_{k\in K}a_j^k\lambda_j^k-\sum_{j\in L}\sum_{k\in K}a_j^k\lambda_j^k-\sum_{j\in N^- -L}\sum_{k=1}^{k_j}a_j^k\lambda_j^k\le b+a_i^{k-1}+u_i^k\lambda_i^k=\beta \tag{4.16}$$

We define two new problems as follows:

$$Max \quad \sum_{j\in C}\lambda_j^{k_j} \tag{4.17}$$

$$s.t. \quad \lambda\in PG \tag{4.18}$$

$$\sum_{j\in C}\sum_{k\in K}a_j^k\lambda_j^k\le\beta+\sum_{j\in N^- -L}a_j^{k_j}+\gamma \tag{4.19}$$

and

$$Min \quad \sum_{j\in L}\sum_{k=k_j}^{T}|\alpha_j^k|\lambda_j^k \tag{4.20}$$

$$s.t. \quad \lambda\in PG \tag{4.21}$$

$$\sum_{j\in L}\sum_{k\in K}a_j^k\lambda_j^k=\gamma \tag{4.22}$$

The optimal value of (4.17)-(4.19) can be found by using $g(\beta)$ given before. If we can solve (4.20)-(4.22) for each possible value of γ, then we can find the optimal value of (4.14)-(4.16), and we can find the maximal lifting coefficients by using this optimal value. We will next give a Proposition that gives an inequality that can be obtained by approximatley lifting cover inequalities for $PG(b+\sum_{j\in N^-}a_j^{k_j})$.

Definition 12. *Let* $C^+ \subseteq N^+$, $C^- \subseteq N^-$, $2 \leq k_j \leq T\ \forall j \in C^+$,$1 \leq k_j \leq T\ \forall j \in C^-$, $k_j = 0\ \forall j \in N^- - C^-$ *and* $C = C^+ \cup C^-$ *be a minimal set such that*

$$\sum_{j \in C^+} a_j^{k_j} - \sum_{j \in C^-} a_j^{k_j} = b + \lambda$$

where $\lambda > 0$. *The set* C *is called a generalized cover.*

Proposition 11. *Let* C *be a generalized cover. The generalized cover inequality*

$$\sum_{j \in C^+} (\alpha_j \lambda_j^{k_j - 1} + \lambda_j^{k_j}) - \sum_{j \in N^-} (\alpha_j \lambda_j^{k_j + 1} + \sum_{k=k_j+2}^{T} \lambda_j^k) \leq |C^+| - 1 \tag{4.23}$$

is valid for PG *where*

$$\alpha_j = \begin{cases} Min\{0, \frac{\lambda - u_j^{k_j}}{\lambda}\}, & \text{if } j \in C^+, \\ Max\{1, \frac{u_j^{k_j+1}}{\lambda}\}, & \text{if } j \in N^-. \end{cases} \tag{4.24}$$

Proof Note that $\sum_{j \in C^+}(\alpha_j \lambda_j^{k_j - 1} + \lambda_j^{k_j}) \leq |C^+| - 1$ is a valid cover inequality for $PG(b + \sum_{j \in N^-} a_j^{k_j})$. We need to lift λ_j^k $j \in N^-$ to obtain a valid inequality for PG.

We have shown that zero is a feasible lifting coefficient for λ_j^k $k \leq k_j\ \forall j \in N^-$. We next lift $\lambda_i^{k_i+1}$ for $i \in N^-$. The lifting coefficient $\alpha_i^{k_i+1}$ of $\lambda_i^{k_i+1}$ satisfies

$$\frac{|C| - 1 - \sum_{j \in C}(\alpha_j^{k_j - 1} \lambda_j^{k_j - 1} + \alpha_j^{k_j} \lambda_j^{k_j})}{\lambda_i^{k_i+1}} \geq \alpha_i^{k_i+1}. \tag{4.25}$$

Let $S = |C| - 1 - \sum_{j \in C}(\alpha_j^{k_j - 1} \lambda_j^{k_j - 1} + \alpha_j^{k_j} \lambda_j^{k_j})$. Suppose that $u_i^{k_i+1} \leq \lambda$. If $\exists j \in C$ *s.t.* $\alpha_j^{k_j - 1} < 0$, then when $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}$ we have $S = -\frac{l}{\lambda}$. Therefore we obtain

$$-\frac{u_i^{k_i+1}}{\lambda} \geq \alpha_i^{k_i+1},$$

and -1 is a feasible lifting coefficient for $\lambda_i^{k_i+1}$.

If $\alpha_j^{k_j - 1} = 0\ \forall j \in C$, then when $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}$ we have

$$S = Min\{0, -\frac{l - (\lambda - u_i^{k_i+1})}{u_i^{k_i+1}}\}.$$

Therefore we obtain

$$-\frac{l - \lambda + u_i^{k_i+1}}{l} \geq \alpha_i^{k_i+1}$$

and -1 is a feasible lifting coefficient for $\lambda_i^{k_i+1}$.

Suppose that $u_i^{k_i+1} > \lambda$. If $\exists j \in C$ *s.t.* $\alpha_j^{k_j-1} < 0$, then when $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}\, l \leq \lambda$ we have $S = -\frac{l}{\lambda}$. Therefore we obtain

$$-\frac{u_i^{k_i+1}}{\lambda} \geq \alpha_i^{k_i+1}. \tag{4.26}$$

When $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}\, l > \lambda$, we have $S = -1$ and

$$-\frac{u_i^{k_i+1}}{l} \geq \alpha_i^{k_i+1}. \tag{4.27}$$

From (4.26) and (4.27) we obtain

$$-\frac{u_i^{k_i+1}}{\lambda} \geq \alpha_i^{k_i+1}.$$

If $\alpha_j^{k_j-1} = 0\ \forall j \in C$, then when $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}\, l \leq \lambda$ we have

$$S = Min\{0, -\frac{l - (\lambda - u_i^{k_i+1})}{u_i^{k_i+1}}\}.$$

Therefore we obtain

$$-\frac{l - \lambda + u_i^{k_i+1}}{l} \geq \alpha_i^{k_i+1} \Rightarrow -\frac{u_i^{k_i+1}}{\lambda} \geq \alpha_i^{k_i+1}. \tag{4.28}$$

When $\lambda_i^{k_i+1} = \frac{l}{u_i^{k_i+1}}\, l > \lambda$, we have $S = -1$ and

$$-\frac{u_i^{k_i+1}}{l} \geq \alpha_i^{k_i+1}. \tag{4.29}$$

From (4.28) and (4.29) we obtain

$$-\frac{u_i^{k_i+1}}{\lambda} \geq \alpha_i^{k_i+1}.$$

The lifting coefficient, $\alpha_i^{k_i+2}$, of $\lambda_i^{k_i+2}$ satisfies

$$\frac{S - \alpha_i^{k_i+1}}{\lambda_i^{k_i+2}} + \alpha_i^{k_i+2} \geq \alpha_i^{k_i+2}.$$

Since $\alpha_i^{k_i+1} \leq -1$ and $S \geq -1$, we have $S - \alpha_i^{k_i+1} > 0$. Therefore $\frac{S-\alpha_i^{k_i+1}}{\lambda_i^{k_i+2}}$ is minimized when $\lambda_i^{k_i+2} = 1$ and we obtain

$$S \geq \alpha_i^{k_i+2}$$

and -1 is a feasible lifting coefficient for $\lambda_i^{k_i+2}$. The lifting coefficients for λ_i^k $k > k_i+2$ can be found similarly.

Since $S < S - \sum_{k=k_i+1}^{T} \alpha_i^k \lambda_i^k$, the above lifting coefficients will be valid for $\lambda_{i'}^k, i' \in N - \{i\}$. □

The following Corrollary is an immediate result of the above proof.

Corollary 4. *(4.23) is a facet-defining inequality for* $PG \cap \{\lambda_j^k = 0\ k > k_j\ j \in C,\ \lambda_j^k = 0\ k \in K,\ j \in N^+ - C\}$ *if* $u_j^{k_j+1} \geq \lambda\ \forall j \in N^-$ *and* $M \leq b + \sum_{i \in C-\{l\}} a_j^{k_j}$.

Example 4 (Continued): Let $C^+ = \{1,2\}$ and $C^- = \{3\}$ with $k_1 = k_2 = 2$ and $k_3 = 1$. The generalized cover inequality

$$-3\lambda_1^1 + \lambda_1^2 - 3\lambda_2^1 + \lambda_2^2 - 2\lambda_3^2 - \lambda_3^3 - 2\lambda_4^1 - \lambda_4^2 - \lambda_4^3 \leq 1$$

is valid for PG. This inequality is facet-defining for $PG \cap \{\lambda_1^3 = \lambda_2^3 = 0\}$. □

4.2 Lifting Convexity Constraints

Suppose that the improved convexity constraint

$$\sum_{k \in K} \lambda_j^k + \sum_{v \in V} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k \leq 1 \tag{4.30}$$

is a facet-defining inequality for $PG(b + \sum_{i \in N^-} a_i^{k_i})$. We need to lift λ_i^k $i \in N^-$ $k \in K$ to find a valid inequality for PG. If

$$a_j^1 \leq b + \sum_{i \in N^- - \{i'\}} a_i^T \ \forall i' \in N^-. \tag{4.31}$$

is satisfied, then (4.30) can be satisfied at equality when $\lambda_i^{k_i} = 0$ for any $i \in N^-$. Therefore downlifting coefficients of $\lambda_i^{k_i}$ $\forall i \in N^-$ are zero. Since for any value of

λ_i^k $i \in N^-$ $k < k_i$, (4.30) can be satisfied at equality their lifting coefficients will all be zero. If (4.31) is not satisfied, then zero is a feasible lifting coefficient for λ_i^k $i \in N^-$, $k \leq k_i$, but not necessarily the maximal lifting coefficient. Therefore we use zero as lifting coefficients for λ_i^k $i \in N^-$, $k \leq k_i$.

We will next show how we can lift λ_i^k $i \in N^-$ $k > k_i$. For any $i \in N^-$, we lift in increasing order of k, $k \in K$. Let $L \subseteq N^-$ be the set of indices that have already been lifted. It is easy to see that the lifting coefficients will be negative. For λ_i^k, we want to find the lifting coefficient α_i^k that satisfies

$$\sum_{k \in K} \lambda_j^k + \sum_{v \in V} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k + \sum_{v \in L} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k + \alpha_i^{k-1} \lambda_i^{k-1} + \alpha_i^k \lambda_i^k \leq 1$$

$\forall \lambda \in \{PG : \lambda_j^k = 0\ j \in N^- - L - \{i\}\ k > k_j\ , \lambda_i^l = 0\ l > k\}$. So α_i^k will satisfy

$$\frac{1 - \sum_{k \in K} \lambda_j^k - \sum_{v \in V} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k - \sum_{v \in L} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k.$$

To find the lifting coefficients, we need to solve the optimization problem

$$max \quad \sum_{k \in K} \lambda_j^k + \sum_{v \in V} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k + \sum_{v \in L} \sum_{k=k_v}^{T} \alpha_v^k \lambda_v^k \tag{4.32}$$

$$subject\ to \quad \sum_{j \in N^+} \sum_{k \in K} a_j^k \lambda_j^k - \sum_{j \in L} \sum_{k \in K} a_j^k \lambda_j^k \leq b + a_i^{k-1} + (a_i^k - a_i^{k-1}) \lambda_i^k \tag{4.33}$$

$$\sum_{k \in K} \lambda_j^k \leq 1\ \forall j \in N^+ \cup L \tag{4.34}$$

$$lambda_j^k \geq 0\ \forall j \in N^+ \cup L, k \in K \tag{4.35}$$

$$\{\lambda_j^k : k \in K\}\ is\ a\ SOS2\ set\ \forall j \in N^+ \cup L \tag{4.36}$$

for any value of λ_i^k. An upper bound to (4.32)-(4.36) can be found by relaxing the $SOS2$ restriction. Another upper bound is $1 + \sum_{v \in V} \alpha_v^T$.

When the data is integer, it can be shown that the values λ_i^k can take in an extreme point are given by

$$\{0, \frac{1}{u_i^k}, \ldots, \frac{u_i^k - 1}{u_i^k}, 1\}$$

where $u_i^k = a_i^k - a_i^{k-1}$. We can solve the above optimization problem for each possible value of λ_i^k and choose the lifting coefficient accordingly.

4.3 Lifting Approximately

Suppose that for $PG(b + \sum_{i \in N^-} a_i^{k_i})$ we have the valid inequality

$$\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k \leq \gamma$$

where $V \subseteq N^+$. We need to lift the variables λ_i^k $i \in N^-$, $k \in K$ to find a valid inequality for PG. Let U be an upper bound on the value of $\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k$. An simple upper bound is

$$\sum_{v \in V} Max_{k \in K}\{\alpha_v^k\}.$$

We will next show how we can lift the variables approximately. We assume that $b + \sum_{i \in N^- - \{i'\}} a_i^{k_i} \geq 0 \quad \forall i' \in N^-$.

Proposition 12. *Zero is a feasible lifting coefficient for* λ_i^k $i \in N^-$, $k \leq k_i$.

Proof When $\lambda_i^{k_i} < 1$ $i \in N^-$, we have $\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k \leq \gamma$. Therefore the downlifting coefficient will not be positive and zero is an upper bound for it. When $\lambda_i^k > 0$ $i \in N^-$, $k < k_i$, we have $\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k \leq \gamma$, therefore the lifting coefficient will be nonnegative and zero is a lower bound for it. □

By using the above proposition, we assign zero as lifting coefficients for λ_i^k $i \in N^-$, $k \leq k_i$. For each $i \in N^-$ we will lift in increasing order of k, $k \in K$ and $k > k_i$. Let $L \subseteq N^-$ be set of indices that have already been lifted and $i \in N^-$ be the index of the variable set that is being lifted. Suppose that we are lifting λ_i^k. Since $\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k \geq \gamma$, when we increase the capacity of the constraint, we have $\alpha_i^k \leq 0$. We want to find the lifting coefficient α_i^k that satisfies

$$\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k + \sum_{j \in L} \sum_{k=k_j+1}^{T} \alpha_j^k \lambda_j^k + \alpha_i^{k-1} \lambda_i^{k-1} + \alpha_i^k \lambda_i^k \leq \gamma$$

$\forall \lambda \in \{PG : \lambda_j^k = 0 \; \forall j \in (N^- - L - \{i\}) \; k > k_j, \; \lambda_i^l = 0 \; l > k\}$. Since

$$\frac{\gamma - \sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k - \sum_{j \in L} \sum_{k=k_j+1}^{T} \alpha_j^k \lambda_j^k - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k$$

we can find a bound on the lifting coefficient by replacing

$$\sum_{v \in V} \sum_{k \in K} \alpha_v^k \lambda_v^k + \sum_{j \in L} \sum_{k=k_j+1}^{T} \alpha_j^k \lambda_j^k$$

with U. Thus

$$\frac{\gamma - U - \alpha_i^{k-1}}{\lambda_i^k} + \alpha_i^{k-1} \geq \alpha_i^k.$$

When the data is integer the values λ_i^k can take are

$$\{0, \frac{1}{u_i^k}, \ldots, \frac{u_i^k - 1}{u_i^k}, 1\}.$$

We can find an approximate value to the lifting coefficients by finding a bound for each possible value of λ_i^k.

4.4 Branch-and-cut Algorithm

The branch-and-cut algorithm that we use is similar to the one that we have proposed in Chapter 3. We use the same branching and the same primal heuristic. The main difference is in the way that we create the cuts.

We use Proposition 11 to obtain cuts. Since the separation algorithm is time consuming we looked for a cut at every 50 nodes. Suppose that λ_j does not satisfy $SOS2$ and we have $\lambda_j^{k_j} > 0$ and $\lambda_j^{k_j-1} = \lambda_j^{k_j+1} = \ldots = \lambda_j^T = 0$. If $a_{ij} > 0$, we will try to create a cut from constraint i.

For each $v \in N^-$, we find C^- by choosing the smallest $k_v \in K$ such that $\sum_{k=0}^{k_v} \lambda_v^k = 1$. For each $v \in N^+ - \{j\}$ we find C^+ by choosing the largest $k_v \in K$ that satisfies $\sum_{k=k_v}^{T} \lambda_v^k = 1$. Let $C^- = \{v \in N^- : k_v > 0\}$, $C^+ = \{v \in N^+ : k_v > 0\}$, and

$C = C^+ \cup C^-$. Note that $j \in C^+$. If $\sum_{v \in C^+} a_{iv}^{k_v} - \sum_{v \in C^-} a_{iv}^{k_v} > b$, then we can create the cover cut

$$\sum_{v \in C^+} (\alpha_v \lambda_v^{k_v - 1} + \lambda_v^{k_v}) - \sum_{v \in N^-} (\alpha_v \lambda_v^{k_v + 1} + \sum_{k=k_v+2}^{T} \lambda_v^k) \leq |C^+| - 1 \tag{4.37}$$

where the α_j are defined by (4.24). Since $\lambda_v^{k_v - 1} = 0 \; \forall v \in C^+$, $\lambda_j^{k_j} > 0$, $\sum_{k=k_v}^{T} \lambda_v^k = 1$ $\forall v \in C - \{j\}$, and $\sum_{k=k_v}^{T} \lambda_v^k = 0$, (4.37) cuts off the infeasible point.

4.5 Computational Experience

We use MINTO [43] with CPLEX 7.5 as LP solver for our computations. We tested our branch-and-cut algorithm against MINTO's default branch-and-cut algorithm on instances of network flow problems with concave piecewise linear cost functions. Specifically we compare:

- the convex combination formulation with binary variables and MINTO's default setting.
- the convex combination formulation without binary variables with our cuts and SOS2 branching.

We consider the network problem

$$Min \quad \sum_{i \in N} \sum_{j \in N} f_{ij}(x_{ij}), \tag{4.38}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = b_i \quad \forall i \in N, \tag{4.39}$$

$$x_{ij} \geq 0 \quad \forall i, j \in N \tag{4.40}$$

where $N = \{1, \ldots, n\}$ and $f_{ij}(x_{ij})$ is a piecewise linear function. We assume that the network problem is balanced, i.e. $\sum_{i \in N} b_i = 0$.

Let the index set of the partitions be $K' = \{0, \ldots, T\}$. We can reformulate (4.38)-(4.40) as

$$Min \quad \sum_{i\in N}\sum_{j\in N}\sum_{k\in K'} b_{ij}^k \lambda_{ij}^k, \tag{4.41}$$

$$\sum_{j\in N}\sum_{k\in K'} \lambda_{ij}^k - \sum_{j\in N}\sum_{k\in K'} \lambda_{ji}^k = b_i \quad \forall i \in N, \tag{4.42}$$

$$\sum_{k\in K'} \lambda_{ij}^k = 1, \qquad \forall i,j \in N, \tag{4.43}$$

$$\lambda_{ij}^k \geq 0, \quad \forall i,j \in N,\ k \in K',\ and\ \{\lambda_{ij}^k : k \in K\}\ satisfy\ SOS\ type\ II\ \forall i,j \in N. \tag{4.44}$$

We assume that $a_{ij}^0 = 0, \forall i,j \in N$ and $a_{ij}^T \leq b_i + \sum_{v\in N} a_{vi}^T\ \forall i,j \in N$. We use single constraint relaxations of the network problem to find valid inequalities.

4.5.1 Test Instances

We tested 15 instances. The instances are randomly generated as follows. Let T be the number of partitions in the piecewise linear function. We first assign each node as a transshipment, a supply or a demand point with probability equal to $\frac{1}{3}$. The demands and supplies are integers uniformly distributed between 1 and $10 * T$. $b_{|N|}$ is modified to obtain a balanced problem, i.e $b_{|N|} = -\sum_{j=1}^{N-1} b_j$. Let $u_{ij}^k = a_{ij}^k - a_{ij}^{k-1}$ be integers uniformly distributed between 1 and 10. The costs are found such that the slopes of the partitions decrease as k increase. Let d_{ij}^k be uniformly distributed between 0 and 1 and $b_{ij}^0 = 0$. The cost function satisfies $b_{ij}^k = b_{ij}^{k-1} + d_{ij}^k$.

4.5.2 Computational Results

We used an Ultra Sparc 60 workstation with dual UltraSparc II 450 MHz CPUs and 4MB cache to perform computational tests. The results are summarized in Table 4. The table gives results for our branch-and-cut algorithm and MINTO's default setting (i.e. MINTO's default branching, cuts, primal heuristics, etc.). The problems are named in axb.c format where a,b, and c correspond to the number of nodes, number of partitions and seed number respectively. For each problem we give the number of nodes required to solve the problem and CPU time in seconds. The problems

Table 4: Comparison of two algorithms for the general case

	MINTO default			Our algorithm				% Red in	
Problem	Nodes	Time	Gap %	Nodes	Time	Gap %	(4.37)	Nodes	Time
20x4.1	8029	385		2073	43		182	74	89
20x4.2	2339	154		1297	17		117	45	89
20x4.3	7793	476		1913	35		164	75	93
20x4.4	1045	66		313	5		94	70	92
20x4.5	12293	908		4295	209		374	65	77
20x5.1	7179	507		1475	51		208	79	90
20x5.2	765	48		259	4		71	66	92
20x5.3	20969	1604		6731	261		357	68	84
20x5.4	89756	*	0.77	10211	1022		609		
20x5.5	65485	*	0.93	42070	*	0.56	1260		
30x4.1	35473	8038		11133	1123		483	69	86
30x4.2	53083	8977		26415	2203		476	50	75
30x4.3	51644	*	0.17	21841	2113		546		
30x4.4	33370	*	4.5	34352	*	3.03	935		
30x4.5	56202	*	0.61	22869	2451		503		
Total	148968	21162		55904	3952			62	81

are terminated after 10,000 CPU seconds. * indicates that the problem has been terminated without proving optimality. For these problems we give the number of nodes that has been evaluated and the optimality gap at the time of termination. For the algorithms that use our cuts, we also give the number of cuts created. We also give total number of nodes and total CPU time in each table. We excluded the problems that has been terminated without proving optimality when calculating the totals. The results show that our branch-and-cut algorithm is significantly better than a branch-and-cut algorithm that uses a mixed integer programming formulation. Our algorithm reduced the total number of nodes by 62% and the total CPU time by 81 % compared to the MINTO's default setting.

In the branch-and-cut algorithm we looked for a cut at every 50 nodes. We also tested the performance of our branch-and-cut algorithm when we look for a cut at each node. The results are summarized in Table 5. Looking for a cut at every 50 nodes reduced the CPU time by 71 % and increased the number of nodes by 16% compared to looking for a cut at each node. Since it is significantly faster, we pereferred looking for a cut at every 50 nodes in our computational experiments.

Table 5: Comparison of looking for a cut at each node and at every 50 nodes

	At each node				At every 50 nodes				% Red in	
Problem	Nodes	Time	Gap %	(4.37)	Nodes	Time	Gap %	(4.37)	Nodes	Time
20x4.1	1713	110		388	2073	43		182	-21	61
20x4.2	893	60		491	1297	17		117	-45	71
20x4.3	1291	161		678	1913	35		164	-48	78
20x4.4	173	12		260	313	5		94	-80	56
20x4.5	4091	736		993	4295	209		374	-5	72
20x5.1	2163	401		866	1475	51		208	32	87
20x5.2	145	16		352	259	4		71	-79	76
20x5.3	4837	1095		1181	6731	261		357	-39	76
20x5.4	9709	4396		1871	10211	1022		609	-5	77
20x5.5	12559	*	0.83	2357	42070	*	0.56	1260		
30x4.1	10453	4285		1423	11133	1123		483	-7	74
30x4.2	24129	8180		1496	26415	2203		476	-9	73
30x4.3	15363	5580		1433	21841	2113		546	-42	62
30x4.4	11024	*	3.7	1851	34352	*	3.03	935		
30x4.5	20539	7899		1301	22869	2451		503	-11	69
Total	95469	32929			110825	9537			-16	71

CHAPTER V

POLYHEDRAL STUDY OF LOWER-SEMI CONTINUOUS PIECEWISE LINEAR OPTIMIZATION

In Chapter 3 we assumed that the cost function is continuous. In this chapter we will generalize the results of Chapter 3 to the case where the cost function is lower semi-continuous. The SOS2 formulation that we used in Chapter 3 can not be used when the function is discontinuous. We need to introduce binary variables to define the jumps in the function. Therefore, in this chapter we will study the polyhedron associated with a mixed integer formulation. There are two main MIP formulations that can be used to formulate lower semi-continuous piecewise linear functions. These are the *incremental cost* and the *modified convex combination* formulations.

The incremental cost formulation for this case is similar to the continuous case. Suppose we have a plf $f(x_j)$ specified by the points $(a_j^k, f(a_j^k))$, $k \in \{0,\ldots,T\}$. Let $K = \{1,\ldots,T\}$, $u_j^k = a_j^k - a_j^{k-1}$ $k \in K$, c_j^k $k \in K$ be the slope of the k^{th} segment and b_j^k, $k \in K$ be the the jump in the cost function at the breakpoint between segment $k-1$ and k. By using incremental cost formulation, we can write x_j and $f(x_j)$ as

$$f(x_j) = \sum_{k \in K} (c_j^k y_j^k + b_j^k z_j^k) \tag{5.1}$$

$$x_j = \sum_{k \in K} y_j^k \tag{5.2}$$

$$u_j^1 z_j^1 \leq y_j^1 \leq u_j^1 \quad u_j^k z_j^k \leq y_j^k \leq u_j^k z_j^{k-1}\ k = 2,\ldots,T-1 \quad 0 \leq y_j^T \leq z_j^{T-1} \tag{5.3}$$

$$z_j^k \in \{0,1\}\ k \in \{1,\ldots,T-1\} \tag{5.4}$$

We can not use the convex combination formulation defined in Chapter 3 here. We need to modify it. Let $d_j^k = f(a_j^{k-1}) + b_j^{k-1}$. Then we can write x_j and $f(x_j)$ as

$$f(x_j) = \sum_{k \in K} (d_j^k \mu_j^k + f(a_j^k)\lambda_j^k) \tag{5.5}$$

$$x_j = \sum_{k \in K} (a_j^{k-1} \mu_j^k + a_j^k \lambda_j^k) \tag{5.6}$$

$$\mu_j^k + \lambda_j^k = z_j^k \quad \forall k \in K \tag{5.7}$$

$$\sum_{k \in K} z_j^k \leq 1 \tag{5.8}$$

$$\mu_j^k \geq 0, \lambda_j^k \geq 0 \quad \forall k \in K \tag{5.9}$$

$$z_j^k \in \{0,1\} \quad \forall k \in K \tag{5.10}$$

by using the convex combination formulation. In this chapter, without loss of generality, we assume that $a_j^0 = 0$ and $a_j^T < b$, $\forall j \in N$.

Croxton, Gendron and Magnanti [9] showed that both formulations have the same LP-bound. Padberg [41] showed that the feasible set of the LP relaxation of the incremental cost formulation is integral, i.e. the binary variables are integer in every vertex of the set. He called such formulations *locally ideal.* Sherali [45] proposed the above modification of the convex combination formulation that is locally ideal.

In this chapter we will use the incremental cost formulation. We make the variable change $x_j^k = \frac{y_j^k}{u_j^k}$ $\forall j, k$. Observe that $0 \leq x_j^k \leq 1$ $\forall j, k$. Let $N = \{1, \ldots, n\}$. The polyhedron that we will use to create cuts is

$$\sum_{j \in N} \sum_{k \in K} u_j^k x_j^k \leq b, \tag{5.11}$$

$$z_j^1 \leq x_j^1 \leq 1, \quad z_j^k \leq x_j^k \leq z_j^{k-1} \; k \in \{2, \ldots, T\}, \quad 0 \leq x_j^T \leq z_j^{T-1} \quad \forall j \in N, \tag{5.12}$$

$$z_j^k \in \{0,1\} \; \forall j \in N, \; k \in K. \tag{5.13}$$

Let $S = conv\{(x,z) : (x,z)\ satisfies\ (5.11) - (5.13)\}$. The binary variables are inserted to enforce the restriction

$$x_j^{k+1} = \cdots = x_j^T = 0\ whenever\ x_j^k < 1,\ k \in \{1,\ldots,T-1\}. \tag{5.14}$$

Let $PX = conv\{x : x\ satisfies\ (5.11)\ and\ (5.14)\}$.

In Chapter 2 we have shown that the incremental cost and convex combination formulations are equivalent in the space of continuous variables. Thus any inequality that we have created in Chapter 3 can be converted into an inequality in the incremental cost formulation. Since these inequalities are valid in the continuous variable space, they can also be used for the MIP formulation. The variable change that we need to make to convert an inequality in convex combination formulation into an inequality in incremental cost formulation is

$$\lambda_j^k = x_j^k - x_j^{k+1}\ for\ 1 \le k \le T-1, \quad \lambda_j^T = x_j^T. \tag{5.15}$$

We will next modify some propositions from Chapter 3 using variable change (5.15). Since the inequalities in the propositions are valid for P, they are also valid for PX and S.

From Proposition 4 of Chapter 3, we obtain

Proposition 13. *(5.11) is a facet-defining inequality of PX if and only if*

$$a_j^1 + \sum_{i \in N-\{j\}} a_i^T \ge b \quad \forall j \in N.$$

From Propositions 5 and 7 of Chapter 3, we obtain

Proposition 14. $\forall j \in N$,

$$x_j^1 \le 1 \tag{5.16}$$

is a facet-defining inequality for

$$PX_j = \{x \in PX : x_i^k = 0 \quad \forall i \in N - \{j\}, k \in K\}. \tag{5.17}$$

(5.16) is a facet-defining inequality for PX for $j \in N$ if and only if

$$a_j^1 + a_i^{T-1} < b \quad \forall i \in N - \{j\}. \tag{5.18}$$

Proposition 15. *For $j \in N$, let $I = \{i \in N - \{j\} : b - a_j^1 \leq a_i^T\}$, $k_i = min\ \{k \in K : b - a_j^1 \leq a_i^k\}$ $\forall i \in I$, and V be a subset of I such that if $|V| > 1$, then $a_v^{k_v-1} + a_{v'}^{k_{v'}-1} > b$ $\forall v, v' \in V$, $v \neq v'$. Then*

$$x_j^1 + \sum_{v \in V} \sum_{k=k_v-1}^{T} \alpha_v^k x_v^k \leq 1, \tag{5.19}$$

with

$$(\alpha_v^{k_v-1}, \alpha_v^{k_v}) \in H_v,$$

$$\alpha_v^{k_v+1} = \begin{cases} 1 - \frac{b - a_v^{k_v+1}}{a_j^1}, & \text{if } \alpha_v^{k_v} = 0, \\ \frac{u_v^{k_v+1}}{a_j^1}, & \text{otherwise,} \end{cases}$$

$$\alpha_v^k = \frac{u_v^k}{a_j^1} \quad k > k_v + 1,$$

where

$$H_v = \begin{cases} \{(0,0)\}, & \text{if } b - a_j^1 = a_v^{k_v}, \\ \{(0,0), (\frac{a_v^{k_v-1} + a_j^1 - b}{a_j^1}, \frac{u_v^{k_v}}{a_j^1})\}, & \text{if } b - a_j^1 < a_v^{k_v}. \end{cases}$$

$\forall v \in V$, is a valid inequality. (5.19) is a facet-defining inequality for PX if $V = I$.

From Proposition 8 of Chapter 3, we obtain

Proposition 16. *Let C be a cover defined in (11). The cover inequality*

$$\sum_{j \in C} (\alpha_j x_j^{k_j-1} + (1 - \alpha_j) x_j^{k_j}) \leq |C| - 1 \tag{5.20}$$

is valid where

$$\alpha_j = \begin{cases} 0, & \text{if } b - \sum_{i \in C - \{j\}} a_i^{k_i} \leq a_j^{k_j-1}, \\ \frac{b - \sum_{i \in C-\{j\}} a_i^{k_i} - a_j^{k_j-1}}{b - \sum_{i \in C-\{j\}} a_i^{k_i} - a_j^{k_j}}, & \text{otherwise.} \end{cases}$$

(5.20) is facet-defining for the subspace

$$PX' = conv\{x \in PX : x_j^k = 0\ k > k_j,\ k \in K,\ \forall j \in C,\ x_j^k = 0\ \forall k \in K, j \in N - C,\},$$

if and only if

$$a_j^{k_j-2} < b - \sum_{i \in C-\{j\}} a_i^{k_i}\ \forall\ j \in C. \tag{5.21}$$

5.1 Lifted Convexity Constraints

Proposition 17. *$\forall j \in N$, (5.16) is a facet-defining inequality for*

$$S_j = \{(x,z) \in S : x_i^k = 0 \quad \forall i \in N - \{j\}, k \in K,$$

$$z_i^k = 0 \quad \forall i \in N - \{j\}, k \in \{1,\ldots,T-1\}\} \tag{5.22}$$

Proof Let

$$v_j = [x_j^1, z_j^1, x_j^2, \ldots, z_j^{k_j-1}, x_j^{k_j}] \tag{5.23}$$

$\forall j \in C$, and v_j^l be v_j vector where first l elements are equal to 1 and the rest is zero. e.g.

$$v_j^4 = [1,1,1,1,0,\ldots,0].$$

The points

$$[v_1^0, \ldots, v_{j-1}^0, v_j^l, v_{j+1}^0, \ldots, v_{|N|}^0]$$

for $l = 1,\ldots,2T-1$ are in S_j, satisfy (5.16) at equality and are linearly independent. Therefore (5.16) is a facet-defining inequality for S_j. □

We next lift (5.16). We lift the variables that are fixed at zero in (5.22). Observe that x_j^k can be positive only if $z_j^{k-1} = 1$. Therefore we define classes $C_i^1 = \{x_i^1\}$ and $C_i^k = \{z_i^{k-1}, x_i^k\}$ $k > 1$. For each $i \in N - \{j\}$ we lift the classes sequentially in the order of

$$C_i^1 = \{x_i^1\}, C_i^2 = \{z_i^1, x_i^2\}, C_i^3 = \{z_i^2, x_i^3\}, \ldots, C_i^T = \{z_i^{T-1}, x_i^T\}.$$

In each class we will do simultaneous lifting. For a pair (z_i^{k-1}, x_i^k) we want to find a pair $(\beta_i^{k-1}, \alpha_i^k)$ that satisfies

$$x_j^1 + \sum_{l=1}^{k-2} \beta_i^l z_i^l + \sum_{l=1}^{k-1} \alpha_i^l x_i^l + \beta_i^{k-1} z_i^{k-1} + \alpha_i^k x_i^k \leq 1 \tag{5.24}$$

For $z \in [0, b]$, let

$$h_i^k(z) = max \quad \beta_i^{k-1} z_i^{k-1} + \alpha_i^k x_i^k$$

$$s.t. \quad a_i^{k-1} z_i^{k-1} + u_i^k x_i^k = z$$

$$0 \leq x_i^k \leq z_i^{k-1}$$

$$z_i^{k-1} \in \{0, 1\},$$

and

$$f_i^k(z) = min \quad 1 - x_j^1 - \sum_{l=1}^{k-2} \beta_i^l z_i^l - \sum_{l=1}^{k-1} \alpha_i^l x_i^l$$

$$s.t. \quad u_j^1 x_j^1 \leq b - z$$

$$0 \leq x_j^1 \leq 1.$$

Let $k_i = min\,\{k \in K : b - a_j^1 \leq a_i^k\}$ for any $i \in N - \{j\}$. For $k < k_i$, since $x_i^k = 1$ does not imply $x_j^1 < 1$, we obtain $(0, 0)$ as lifting coefficients of (z_i^{k-1}, x_i^k). Thus

$$f_i^{k_i}(z) = min \quad 1 - x_j^1,$$

$$s.t. \quad a_j^1 x_j^1 \leq b - z,$$

$$0 \leq x_j^1 \leq 1.$$

The closed form expression of $f_i^{k_i}(z)$ is easy to find and can be written as

$$f_i^{k_i}(z) = \begin{cases} 0, & \text{if } 0 \leq z \leq b - a_j^1, \\ \frac{a_j^1 - b + z}{a_j^1}, & \text{if } b - a_j^1 \leq z \leq b. \end{cases}$$

Since we have a closed form expression for the function $f_i^{k_i}$, we can compute sets $H_i^{k_i}$ of lifting coefficients $(\beta_i^{k_i-1}, \alpha_i^{k_i})$. We obtain lifting coefficients that define facet inducing inequalities if and only if the pairs $(\beta_i^{k_i-1}, \alpha_i^{k_i})$ are such that $f_i^{k_i}(z) = h_i^{k_i}(z)$

has two solutions $(\widehat{z},\widehat{x})$ and $(\overline{z},\overline{x})$ such that $(\widehat{z}_i^{k_i-1},\widehat{x}_i^{k_i})$ and $(\overline{z}_i^{k_i-1},\overline{x}_i^{k_i})$ are linearly independent. The points 0 and $b-a_j^1$ define the lower envelope of the function $f_i^{k_i}$. Furthermore, the pairs of adjacent points on the lower envelope, i.e, $(0,b-a_j^1)$ and $(b-a_j^1,b)$, define the sets of two values z for which $f_i^{k_i}(z)=h_i^{k_i}(z)$ and for which the associated solutions have linearly independent i^{th} components. We get the set $H_i^{k_i}$ by computing the slopes and intercepts of the lines defining the lower envelope.

Observe that $h_i^{k_i}(z)$ is only defined when $z\in[a_i^{k_i-1},a_i^{k_i}]$ and in this region we have $z=a_i^{k_i-1}+u_i^{k_i}x_i^{k_i}$ and $\beta_i^{k_i-1}z_i^{k_i-1}+\alpha_i^{k_i}x_i^{k_i}=\beta_i^{k_i-1}+\alpha_i^{k_i}x_i^{k_i}$. From the points $(0,0)$ and $(b-a_j^1,0)$ we obtain the lifting coefficients as $(0,0)$. The slope and intercept of the line defined by the points $(b-a_j^1,0)$ and $(b,1)$ are $\frac{1}{a_j^1}$ and $\frac{a_j^1-b}{a_j^1}$ respectively. If $b-a_j^1<a_i^{k_i}$ from the line

$$h_i^{k_i}(z)=\frac{a_j^1-b}{a_j^1}+\frac{1}{a_j^1}z=\frac{a_i^{k_i-1}-b+a_j^1}{a_j^1}+\frac{u_i^{k_i}}{a_j^1}x_i^{k_i}$$

we obtain the lifting coefficient pair $(\frac{a_j^1+a_i^{k_i-1}-b}{a_j^1},\frac{u_i^{k_i}}{a_j^1})$. Thus

$$H_i^{k_i}=\begin{cases}\{(0,0),(\frac{a_j^1+a_i^{k_i-1}-b}{a_j^1},\frac{u_i^{k_i}}{a_j^1})\}, & \text{if } b-a_j^1<a_i^{k_i},\\ \{(0,0)\}, & \text{if } b-a_j^1=a_i^{k_i}.\end{cases}\tag{5.25}$$

Finding the set $H_i^{k_i}$ is illustrated in Figure 4. The line $\overline{h}_i^{k_i}(z)$ corresponds to lifting coefficients (0,0) and the line $\widehat{h}_i^{k_i}(z)$ corresponds to lifting coefficients $(\frac{a_j^1+a_i^{k_i-1}-b}{a_j^1},\frac{u_i^{k_i}}{a_j^1})$.

Suppose that we have chosen the pair $(0,0)$ as lifting coefficients for $(z_i^{k_i-1},x_i^{k_i})$. Then we have

$$f_i^{k_i+1}(z)=min\quad 1-x_j^1,$$

$$s.t.\quad u_j^1x_j^1\le b-z,$$

$$0\le x_j^1\le 1.$$

The closed form expression for $f_i^{k_i+1}(z)$ is same as before. Finding $H_i^{k_i+1}$ is similar and we obtain

$$\overline{H}_i^{k_i+1}=\{(\frac{a_j^1+a_i^{k_i}-b}{a_j^1},\frac{u_i^{k_i}}{a_j^1})\}\tag{5.26}$$

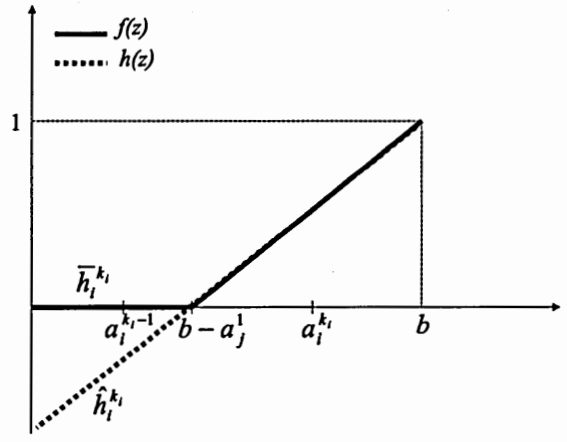

Figure 4: Finding the sets $H_i^{k_i}$ and $\overline{H}_i^{k_i+1}$

Finding the set $\overline{H}_i^{k_i+1}$ is illustrated in Figure 4. The line $\hat{h}_i^{k_i}(z)$ corresponds to lifting coefficients. Note that $\overline{h}_i^{k_i}$ is not defined for $z \in [a_i^{k_i}, a_i^{k_i+1}]$.

Suppose we have chosen the pair $(\frac{a_j^1+a_i^{k_i-1}-b}{a_j^1}, \frac{u_i^{k_i}}{a_j^1})$ as lifting coefficients for $(z_i^{k_i-1}, x_i^{k_i})$. We have

$$f_i^{k_i+1}(z) = min \quad 1 - x_j^1 - \frac{a_j^1 + a_i^{k_i-1} - b}{a_j^1} z_i^{k_i-1} - \frac{u_i^{k_i}}{a_j^1} x_i^{k_i},$$

$$s.t. \quad u_j^1 x_j^1 \leq b - z,$$

$$0 \leq x_j^1 \leq 1.$$

If $z \leq a_i^{k_i}$, then $f_i^{k_i+1}(z)$ should be zero because the pair $(\beta_i^{k_i-1}, \alpha_i^{k_i})$ are the maximal lifting coefficients. When $z \in [a_i^{k_i}, a_i^{k_i+1}]$, we have

$$1 - x_j^1 - \frac{a_j^1 + a_i^{k_i-1} - b}{a_j^1} z_i^{k_i-1} - \frac{u_i^{k_i}}{a_j^1} x_i^{k_i} = 1 - x_j^1 - \frac{a_j^1 + a_i^{k_i} - b}{a_j^1}.$$

The closed form expression of $f_i^{k_i+1}(z)$ follows from above observations and we have

$$f_i^{k_i+1}(z) = \begin{cases} 0, & \text{if } 0 \leq z \leq a_i^{k_i}, \\ \frac{z-a_i^{k_i}}{a_j^1}, & \text{if } a_i^{k_i} \leq z \leq b. \end{cases}$$

The points 0 and $a_i^{k_i}$ define the lower envelope of the function $f_i^{k_i+1}$. Furthermore, the pairs of adjacent points on the lower envelope, i.e, $(0, a_i^{k_i+1})$ and $(a_i^{k_i+1}, b)$, define the sets of two values z for which $f_i^{k_i}(z) = h_i^{k_i}(z)$ and for which the associated solutions have linearly independent i^{th} components. We get the set $H_i^{k_i+1}$ by computing the slopes and intercepts of the lines defining the lower envelope.

Observe that $h_i^{k_i}(z)$ is only defined when $z \in [a_i^{k_i}, a_i^{k_i+1}]$ and we have $z = a_i^{k_i} + u_i^{k_i+1} x_i^{k_i+1}$ and $\beta_i^{k_i} z_i^{k_i} + \alpha_i^{k_i+1} x_i^{k_i+1} = \beta_i^{k_i} + \alpha_i^{k_i+1} x_i^{k_i+1}$. We can not use the points $(0,0)$ and $(a_i^{k_i}, 0)$ because $h_i^{k_i}(z)$ is only defined when $z \in [a_i^{k_i}, a_i^{k_i+1}]$. The slope and intercept of the line defined by the points $(a_i^{k_i}, 0)$ and $(b, \frac{b-a_i^{k_i}}{a_j^1})$ are $\frac{1}{a_j^1}$ and $-\frac{a_i^{k_i}}{a_j^1}$ respectively. From the line

$$h_i^{k_i+1} = -\frac{a_i^{k_i}}{a_j^1}\frac{1}{a_j^1} z = \frac{u_i^{k_i}}{a_j^1} x_i^{k_i}$$

we obtain the pair $(0, \frac{u_i^{k_i+1}}{a_j^1})$. Thus

$$\widehat{H}_i^{k_i+1} = \{(0, \frac{u_i^{k_i+1}}{a_j^1})\}. \tag{5.27}$$

Finding the set $\widehat{H}_i^{k_i+1}$ is illustrated in Figure 5. The line $h_i^{k_i+1}(z)$ corresponds to lifting coefficients. From (5.26) and (5.27) we obtain

$$H_i^{k_i+1} = \begin{cases} \overline{H}_i^{k_i+1}, & \text{if } \alpha_i^{k_i} = 0, \\ \widehat{H}_i^{k_i+1}, & \text{if } \alpha_i^{k_i} > 0. \end{cases} \tag{5.28}$$

Finding H_i^k $k > k_i + 1$ is similar and we have

$$H_i^k = \{(0, \frac{u_i^k}{a_j^1})\} \tag{5.29}$$

$k > k_i + 1$.

We next give a proposition that summarizes the above discussion. The proof also follows from this discussion.

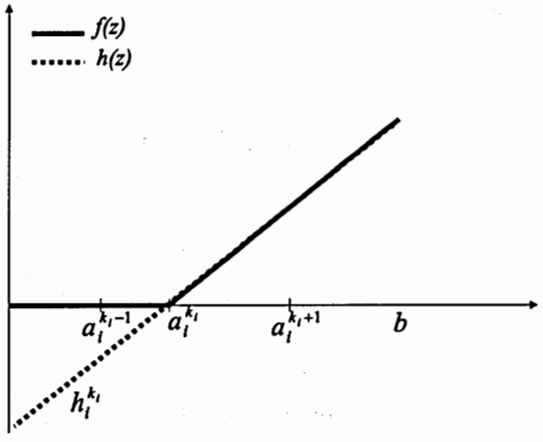

Figure 5: Finding the set $\widehat{H}_i^{k_i+1}$

Proposition 18. *For an $j \in N$ that does not satisfy (5.18), let $I = \{i \in N - \{j\} : a_i^{T-1} \geq b - a_j^1\}$, $k_i = min\ \{k \in K : b - a_j^1 \leq a_i^k\}$, and V be a subset of I such that if $|V| > 1$, then $a_v^{k_v-1} + a_{v'}^{k_{v'}} > b\ \forall v, v' \in V,\ v \neq v'$. Then*

$$x_j^1 + \sum_{v \in V}(\sum_{k=k_v}^{k_v+1}(\beta_v^{k-1} z_v^{k-1} + \alpha_v^k x_v^k) + \sum_{k=k_v+2}^{T} \frac{u_v^k}{a_j^1} x_v^k) \leq 1 \tag{5.30}$$

with

$$(\beta_v^{k_v-1}, \alpha_v^{k_v}) \in H_v^{k_v},$$

$$(\beta_v^{k_v}, \alpha_v^{k_v+1}) \in H_v^{k_v+1}$$

$\forall v \in V$, where $H_v^{k_v}$ and $H_v^{k_v+1}$ are defined by (5.25) and (5.28) respectively. (5.30) is a facet-defining inequality for S if $V = I$.

Example 2 Let $|N| = 3$, $T = 3$, and (5.11) is

$$2x_1^1 + 4x_1^2 + 2x_1^3 + 3x_2^1 + 4x_2^2 + 3x_2^3 + 4x_3^1 + 4x_3^2 + 2x_3^3 \leq 10. \tag{5.31}$$

Let $SE = conv\{(x, z) : (x, z)\ satisfy\ (5.12), (5.13),\ and\ (5.31)\}$. When $j = 1$, we have $I = V = \{2, 3\}$ where $k_2 = 3$ and $k_3 = 2$. Then

$$x_1^1 + x_3^3 \leq 1$$

and

$$x_1^1 - \frac{1}{2}z_2^2 + \frac{3}{2}x_2^3 + x_3^3 \leq 1$$

are facet-defining inequalities for SE.

5.2 Cover Inequalities

Definition 13. *Let $2 \leq k_j \leq T\ \forall j \in N$ and $C \subseteq N$ be a set such that*

$$\sum_{j \in C} a_j^{k_j} - b = \lambda > 0,\ \sum_{j \in C - \{i\}} a_j^{k_j} \leq b,\ \forall i \in C,\ \text{and}\ \exists j \in C\ s.t.\ u_j^{k_j} > \lambda. \tag{5.32}$$

The set C is called a cover.

Proposition 19. *Let C be a cover. Then*

$$\sum_{j \in C}((1 - \frac{u_j^{k_j}}{\lambda})z_j^{k_j - 1} + \frac{u_j^{k_j}}{\lambda}x_j^{k_j}) \leq |C| - 1 \tag{5.33}$$

is a valid inequality. (5.33) is facet-defining for the subspace

$$S' = conv\{(x,z) \in S : x_j^k = 0\ k > k_j,\ k \in K,\ \forall j \in C,\ x_j^k = 0\ \forall k \in K, j \in N - C,$$

$$z_j^k = 0\ k \geq k_j,\ k \in K - \{T\},\ \forall j \in C,\ z_j^k = 0\ \forall k \in K - \{T\}, j \in N - C\},$$

if and only if

$$a_j^{k_j - 2} < b - \sum_{i \in C - \{j\}} a_i^{k_i}\ \forall\ j \in C. \tag{5.34}$$

Proof Since the right hand side is $|C| - 1$, for any $j \in C$ we need to consider the points where $x_i^{k_i} > 0\ \forall i \in C - \{j\}$ and $x_j^{k_j} \geq 0$. It suffices to consider the extreme points. For any $j \in C$, let $x_i^{k_i} = 1\ \forall i \in C - \{j\}$. If $u_j^{k_j} \geq \lambda$, the greatest value of $x_j^{k_j}$ is $\frac{b - \sum_{i \in C - \{j\}} a_i^{k_i} - a_j^{k_j - 1}}{u_j^{k_j}}$ and $(1 - \alpha_j)z_j^{k_j - 1} + \alpha_j x_j^{k_j} = 0$ and inequality is valid. If $u_j^{k_j} < \lambda$, then the greatest value of $x_j^{k_j}$ and $z_j^{k_j - 1}$ are zero and the inequality is valid. The last case we need to consider is the case where $x_j^{k_j - 1} = z_j^{k_j - 1} = 1$ and $x_j^{k_j} = 0$. We claim that when $x_j^{k_j - 1} = z_j^{k_j - 1} = 1$, the maximum value $\sum_{j \in C}((1 - \alpha_j)z_j^{k_j - 1} + \alpha_j x_j^{k_j})$ can take is less than or equal to $|C| - 2 + \frac{u_j^{k_j}}{\lambda}$. Let

$$g_j^{k_j}(\beta) = Max\quad (1 - \alpha_j)z_j^{k_j - 1} + \alpha_j x_j^{k_j},$$

$$s.t. \quad a_j^{k_j-1} z_j^{k_j-1} + u_j^{k_j} x_j^{k_j} \leq \beta.$$

The closed form expression for $g_j^{k_j}$ is

$$g_j^{k_j}(\beta) = \begin{cases} 0, & \text{if } \beta < a_j^{k_j}, \\ \frac{\lambda - u_j^{k_j}}{\lambda} + \frac{\beta - a_j^{k_j-1}}{\lambda}, & \text{if } a_j^{k_j-1} \leq \beta \leq a_j^{k_j}, \\ 1, & \text{if } \beta > a_j^{k_j}. \end{cases}$$

When the right hand side of the constraint is greater than or equal to $\sum_{i \in C-\{j\}} a_i^{k_i}$, we have $\sum_{j \in C}((1-\alpha_j) z_j^{k_j-1} + \alpha_j x_j^{k_j}) = |C| - 1$. Since decreasing β by $l < u_i^{k_i}$ will decrease $g_i^{k_i}$ by $\frac{l}{\lambda}$ $\forall i \in C$ and there exists at least one $i \in C - \{j\}$ such that $\lambda > u_i^{k_i}$, fixing $x_j^{k_j-1} = z_j^{k_j-1}$ at 1 will decrease the sum $\sum_{j \in C}((1-\alpha_j) z_j^{k_j-1} + \alpha_j x_j^{k_j})$ by $\frac{\lambda - u_j^{k_j}}{\lambda}$ and we prove the claim and validity of the inequality.

For each $j \in C$, we give $2k_j - 1$ points. Let v_j be a vector defined by (5.23) $\forall j \in C$, and v_j^l be v_j vector where first l elements are equal to 1 and the rest is zero. For each $j \in C$ we define $2k_j - 3$ points as

$$[v_1^{2k_1-1}, v_2^{2k_2-1}, \ldots, v_{j-1}^{2(k_{j-1})-1}, v_j^l, v_{j+1}^{2(k_{j+1})-1}, \ldots, v_{|C|}^{2(k_{|C|})-1}] \tag{5.35}$$

for $l = 0, 1, \ldots, 2k_j - 4$.

If $u_j^{k_j} \geq \lambda$ we define two more points as

$$[v_1^{2k_1-1}, v_2^{2k_2-1}, \ldots, v_{j-1}^{2(k_{j-1})-1}, v_j^{2k_j-3}, v_{j+1}^{2(k_{j+1})-1}, \ldots, v_{|C|}^{2(k_{|C|})-1}] \tag{5.36}$$

and

$$[v_1^{2k_1-1}, v_2^{2k_2-1}, \ldots, v_{j-1}^{2(k_{j-1})-1}, \overline{v}_j^{2k_j-1}, v_{j+1}^{2(k_{j+1})-1}, \ldots, v_{|C|}^{2(k_{|C|})-1}] \tag{5.37}$$

where

$$\overline{v}_j^{2k_j-1} = [1, \ldots, 1, \frac{b - \sum_{i \in C-\{j\}} a_i^{k_i} - a_j^{k_j-1}}{u_j^{k_j}}].$$

If $u_j^{k_j} < \lambda$ we define two more points as

$$[v_1^{2k_1-1}, v_2^{2k_2-1}, \ldots, v_{j-1}^{2(k_{j-1})-1}, \overline{v}_j^{2k_j-3}, v_{j+1}^{2(k_{j+1})-1}, \ldots, v_{|C|}^{2(k_{|C|})-1}] \tag{5.38}$$

and

$$[\widetilde{v}_1^{2k_1-1}, v_2^{2k_2-1}, \ldots, v_{j-1}^{2(k_{j-1})-1}, v_j^{2k_j-2}, v_{j+1}^{2(k_{j+1})-1}, \ldots, v_{|C|}^{2(k_{|C|})-1}] \tag{5.39}$$

where

$$\overline{v}_j^{2k_j-3} = [1, \ldots, 1, \frac{b - \sum_{i \in C-\{j\}} - a_j^{k_j-2}}{u_j^{k_j-1}}, 0, 0]$$

and

$$\widetilde{v}_1^{2k_1-1} = [1, \ldots, 1, \frac{b - \sum_{i \in C-\{1,j\}} - a_1^{k_1-1} - a_j^{k_j-1}}{u_1^{k_1}}].$$

Suppose that these points lie on the plane $\sum_{j \in C} \sum_{k \leq k_j} \alpha_j^k x_j^k + \sum_{j \in C} \sum_{k < k_j} \beta_j^k z_j^k = |C| - 1$. By using the points given in (5.35) for $l = 0$ we obtain

$$\sum_{k \leq k_j} \alpha_j^k + \sum_{k < k_j} \beta_j^k = 1 \; \forall j \in C. \tag{5.40}$$

By using the points given in (5.35) for $1 \leq l \leq 2k_j - 4$ we obtain

$$\alpha_j^k = \beta_j^k = 0 \; k \leq k_j - 2 \; \forall j \in C. \tag{5.41}$$

By using the points (5.36) and (5.38) for a $j \in C$ such that $u_j^{k_j} \geq \lambda$ and $u_j^{k_j} < \lambda$ respectively, we obtain $\alpha_j^{k_j-1} = 0 \; \forall j \in C$. Therefore we have

$$\alpha_j^{k_j} + \beta_j^{k_j-1} = 1 \; \forall j \in C. \tag{5.42}$$

For a j such that $u_j^{k_j} \geq \lambda$, by using the points given by (5.37) and (5.42) we uniquely obtain

$$\alpha_j^{k_j} = \frac{u_j^{k_j}}{\lambda} \; and \; \beta_j^{k_j-1} = 1 - \frac{u_j^{k_j}}{\lambda}.$$

For a j such that $u_j^{k_j} < \lambda$, by using the points given by (5.39), (5.42), and

$$\beta_j^{k_j-1} + \alpha_1^{k_1} \frac{b - \sum_{i \in C-\{1,j\}} - a_1^{k_1-1} - a_j^{k_j-1}}{u_1^{k_1}} + \beta_1^{k_1} = 1$$

we uniquely obtain

$$\alpha_j^{k_j} = \frac{u_j^{k_j}}{\lambda} \; and \; \beta_j^{k_j-1} = 1 - \frac{u_j^{k_j}}{\lambda}.$$

We obtain the inequality uniquely by using these points, therefore (5.33) is a facet-defining inequality for S'. □

Example 2 (Continued) Let $C = \{1,2\}$ where $k_1 = k_2 = 2$. Then the cover inequality

$$-\frac{1}{3}z_1^1 + \frac{4}{3}x_1^2 - \frac{1}{3}z_2^1 + \frac{4}{3}z_2^2 \leq 1$$

is valid for SE.

Example 3 Let $|N| = 2$, $T = 3$, and (5.11) be

$$4x_1^1 + 2x_1^2 + 2x_1^3 + 3x_2^1 + 4x_2^2 + 3x_2^3 \leq 10. \tag{5.43}$$

Let $SE = conv\{(x,z) : (x,z)\ satisfy\ (5.12), (5.13),\ and\ (5.43)\}$. Let $C = \{1,2\}$ where $k_1 = k_2 = 3$. The cover inequality

$$\frac{1}{3}z_1^1 + \frac{2}{3}x_1^2 - \frac{1}{3}z_2^1 + \frac{4}{3}x_2^2 \leq 1$$

is valid for SE.

5.3 Relation Between Flow Cover Cuts and Cover Cuts

Suppose that in a cover C we have

$$u_j^{k_j} \geq \lambda \; \forall j \in C, \tag{5.44}$$

and we set

$$x_j^k = 1 \; \forall j \in C \; k < k_j. \tag{5.45}$$

Let $y_j^k = u_j^k x_j^k \; \forall j,k$. When (5.45) is satisfied we can write a relaxation of (5.11)-(5.13) as

$$\sum_{j \in C} y_j^{k_j} \leq b - \sum_{j \in C} a_j^{k_j - 1}, \tag{5.46}$$

$$y_j^{k_j} \leq u_j^{k_j} z_j^{k_j - 1} \; \forall j \in C, \tag{5.47}$$

$$z_j^{k_j - 1} \in \{0,1\} \; \forall j \in C. \tag{5.48}$$

From (5.32) and (5.45), we have

$$\sum_{j\in C} u_j^{k_j} > b - \sum_{j\in C} a_j^{k_j-1}, \text{ and } \sum_{j\in C-\{i\}} u_j^{k_j} \leq b - \sum_{j\in C} a_j^{k_j-1}, \ \forall i \in C. \tag{5.49}$$

We can create the flow cover inequality

$$\sum_{j\in C}(y_j^{k_j} + (u_j^{k_j} - \lambda)(1 - z_j^{k_j-1})) \leq b - \sum_{j\in C} a_j^{k_j-1}. \tag{5.50}$$

We can rewrite (5.50) as

$$\sum_{j\in C} u_j^{k_j} x_j^{k_j} - \sum_{j\in C}(u_j^{k_j} - \lambda) z_j^{k_j-1} \leq b - \sum_{j\in C} a_j^{k_j-1} - \sum_{j\in C} u_j^{k_j-1} + |C|\lambda \Rightarrow$$

$$\sum_{j\in C} u_j^{k_j} x_j^{k_j} - \sum_{j\in C}(u_j^{k_j} - \lambda) z_j^{k_j-1} \leq \lambda(|C| - 1).$$

Therefore the flow cover inequality for (5.46)-(5.48) is a cover inequality when (5.44) is satisfied. This result is not true in general. If (5.44) is not satisfied, we need to lift the flow cover inequality to obtain a valid inequality for S.

5.4 Computational Experience

We used MINTO 3.0 [43] with CPLEX 7.5 as LP solver. We tested the performance of our cuts on instances of transportation problems with lower semi-continuous piecewise linear cost functions. Specifically we compared

- the incremental cost formulation with binary variables and MINTO's default setting.
- the incremental cost formulation with binary variables, MINTO's default setting and our cuts.

We use single constraint relaxations of the transportation problem to find valid inequalities as in Chapter 3.

5.4.1 Branching

Suppose that we have z_j^k fractional. Traditional branching enforces $z_j^k = 0$ in the first branch and $z_j^k = 1$ in the other. When $z_j^k = 0$ we have $x_j^{k+1} = \ldots = x_j^T = 0$ and when $z_j^k = 1$ we have $x_j^1 = \ldots = x_j^k = 1$. Thus the traditional branching and $SOSX$ branching are equivalent and we will use traditional branchng in our computational experiments.

5.4.2 Creating Cuts

Cuts are created similar to the cuts in Chapter 3. Suppose that for j we have $x_j^{k_j} > 0$, $x_j^{k_j-1} < 1$, and $x_j^{k_j+1} = \ldots = x_j^T = 0$. Clearly this x_j does not satisfy $SOSX$. We may be able to derive a cut from constraint i if $a_{ij} > 0$. Suppose that we are looking for a cut in constraint i.

For each $v \in N - \{j\}$ that satisfies $x_v^1 = 1$, we try to create a lifted convexity cut. If $a_{iv}^1 + a_{ij}^{k_j-1} \geq b$, then

$$x_v^1 + \sum_{k=k_j}^{T} \alpha_j^k x_j^k \leq 1 \tag{5.51}$$

with

$$\alpha_j^{k_j} = 1 - \frac{b - a_{ij}^{k_j}}{a_{iv}^1}$$

and

$$\alpha_j^k = \frac{a_{ij}^k - a_{ij}^{k-1}}{a_{iv}^1} \quad k > k_j$$

cuts off the infeasible point. If $a_{ij}^{k_j-1} < b - a_{iv}^1 < a_{ij}^{k_j}$, then

$$x_v^1 + \alpha_j^{k_j-1} x_j^{k_j-1} + \alpha_j^{k_j} x_j^{k_j} \leq 1 \tag{5.52}$$

with

$$\alpha_j^{k_j} = \frac{a_{ij}^{k_j} - a_{ij}^{k_j-1}}{a_{iv}^1} \text{ and } \alpha_j^{k_j-1} = \frac{a_{iv}^1 + a_{ij}^{k_j-1} - b}{a_{iv}^1}$$

cuts off the infeasible point.

To obtain a cover inequality for each $v \in N - \{j\}$, we find the cover set C by choosing the largest $k_v \in K$ that satisfies $x_v^{k_v} = 1$. If $x_v^1 < 1$, we say $k_v = 0$. Let $C = \{v \in J : k_v > 0\}$. Note that $j \in C$. If $\sum_{v \in C} a_{iv}^{k_v} > b$, then we create the cover cut

$$\sum_{v \in C}((1 - \alpha_v^{k_v})x_v^{k_v - 1} + \alpha_v^{k_v} x_v^k) \leq |C| - 1, \tag{5.53}$$

where

$$\alpha_v^{k_v} = \frac{a_{ij}^{k_j} - a_{ij}^{k_j - 1}}{\sum_{v \in C} a_{iv}^{k_v} - b}.$$

Since $x_v^{k_v} = 1 \; \forall v \in C$ and $x_j^{k_j} > 0$, (5.53) cuts off the infeasible point.

5.4.3 Computational Results

We tested 20 instances. We used the same instances that we have used in Chapter 3, and we added a jump, an integer uniformly distributed between 0 and 10, after each segment to the cost function.

We used a Sun Ultra Sparc 60 workstation with dual UltraSparc II 450 MHz CPUs and 4MB cache to perform computational tests. The results are summarized in Table 6 which compares the incremental cost formulation with and without our cuts. For both cases we also use MINTO's default setting (i.e. MINTO's default branching, cuts, primal heuristics, etc.). The problems are named in axbxc.d format where a,b,c and d correspond to the number of supply points, number of demand points, number of partitions, and seed number respectively. For each problem we give the number of nodes required to solve the problem and CPU time in seconds. The problems are terminated after 10,000 CPU seconds. $*$ indicates that the problem has been terminated without proving optimality. For these problems we give the number of nodes that has been evaluated and the optimality gap at the time of termination. For the algorithm that use our cuts, we also give the number of cuts created. We also give the total number of nodes and the total CPU time. The total does not

Table 6: Computations with and without our cuts

	Without our cuts			With our cuts					% Red in	
Problem	Nodes	Time	Gap %	Nodes	Time	(5.51)	(5.52)	(5.53)	Nodes	Time
12x18x4.1	13201	2352		2179	689	39	182	415	83	70
12x18x4.2	6378	931		640	188	30	122	280	90	80
12x18x4.3	8435	1589		2141	520	51	130	349	75	67
12x18x4.4	5057	960		1449	463	47	159	325	71	52
12x18x4.5	5879	704		565	117	20	112	200	90	83
12x18x5.1	36572	*	0.30	3089	1680	157	150	765		
12x18x5.2	26872	*	0.29	4127	2826	198	182	797		
12x18x5.3	23122	*	0.06	679	595	104	125	474		
12x18x5.4	17785	5080		889	573	92	83	386	95	89
12x18x5.5	24548	*	0.83	4055	3052	149	146	747		
15x15x4.1	13801	2373		1230	315	51	129	260	91	87
15x15x4.2	5783	879		677	179	42	139	262	88	80
15x15x4.3	42902	*	0.25	5384	2126	80	262	429		
15x15x4.4	1313	140		241	57	30	78	173	82	59
15x15x4.5	1687	267		313	90	37	98	229	81	66
15x15x5.1	24101	7996		1620	1072	139	178	663	93	87
15x15x5.2	10665	4665		768	490	87	139	507	93	89
15x15x5.3	18321	6614		657	426	69	121	378	96	94
15x15x5.4	5797	2176		291	225	94	108	451	95	90
15x15x5.5	20801	*	1.05	4513	5475	187	205	782		
Total	138203	36725		13660	5403				90	85

include the problems that has been terminated without proving optimality. The results show that our cuts reduces the computational time and the number of nodes processed significantly. Specifically our cuts reduced the total number of nodes and the total CPU time by 90% and 85% respectively in those problems that could be solved without the cuts in 10,000 seconds. If more time was allowed the results would be even better.

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we studied the polyhedral structure of piecewise linear optimization problems and derived strong valid inequalities that can be used in branch-and-cut algorithms. In Chapter 2, we studied the formulations of linear programs with piecewise linear objective functions. We showed that the two formulations without additional binary variables have the same LP bound as those of the corresponding formulations with binary variables. We also showed that the two formulations without binary variables correspond to the same polyhedron in the space of continuous variables. Therefore, there appeared to be computational adventages that could be obtained from the formulations without binary variables. Chapter 3 presented a polyhedral study of the one row relaxation of a separable piecewise linear optimization problem. We assumed that the objective function is continuous and the coefficients of the variables are positive. We derived several classes of valid inequalities for this problem and presented a branch-and-cut algorithm without binary variables that uses these inequalities and SOS2 branching. Our computational results demonstrate that our approach is significantly better than the standard approaches to piecewise linear optimization. In Chapters 4 and 5 we relaxed the assumptions of Chapter 3. In Chapter 4, we relaxed the assumption that the coefficients of the variables are positive. We derived valid inequalities for this case by lifting the inequalities of Chapter 3. In Chapter 5, we relaxed the assumption that the objective function is continuous. In this case we need to introduce binary variables to formulate the function properly. We generalized the inequalities of Chapter 3 to the case where binary variables are introduced. In both chapters we reported computational results that demonstrate

that our valid inequalities improve the performance of the branch-and-cut algorithm.

In summary, we presented new algorithms for a class of NP-hard problems that are of great practical importance. Our computational results clearly showed that our algorithms are substantially better than those currently used in practice.

Most of the inequalities that we derived are facet-defining for a lower dimensional polytope. In the thesis, we usually lifted these inequalities approximately to obtain valid inequalities. Finding maximal lifting coefficients is a future research area.

We derived the inequalities of Chapters 4 and 5, using the inequalities of Chapter 3. It may be possible to create new classes of valid inequalities by using the special structure of these problems. We did not study the case where both assumptions of Chapter 3 are relaxed. Hence further investigation of inequalities for this problem is necessary. The valid inequalities for this problem can be used in a branch-and-cut algorithm for many real life problems, e.g. merge-in-transit problems [10].

In this thesis, we studied general nonconvex piecewise linear functions. It may also be interesting to study the special cases of nonconvex piecewise linear functions. Step functions and concave piecewise linear functions are examples of these special cases. For these cases, we can use the structure of the cost function in addition to the polyhedral structure, to obtain an algorithm. Alternative formulations may also be studied for these cases.

REFERENCES

[1] ARNTZEN, B., BROWN, G., HARRISON, T., and TRAFTON, L., "Global supply chain management at digital equipment corporation," *Interfaces*, vol. 25, pp. 69 – 93, 1995.

[2] BALAS, E., "Facets of the knapsack polytope," *Mathematical Programming*, vol. 8, pp. 146 – 164, 1975.

[3] BALAS, E. and ZEMEL, E., "Facets of knapsack polytope from minimal covers," *SIAM Journal on Applied Mathematics*, vol. 34, pp. 119–148, 1978.

[4] BEALE, E. L. M. and TOMLIN, J. A., "Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables," pp. 447–454, 1970.

[5] BEALE, E. M. L. and FORREST, J. J. H., "Global optimization using special ordered sets," *Mathematical Programming*, vol. 10, pp. 52–69, 1976.

[6] BERTSIMAS, D., DARNELL, C., and SOUCY, R., "Portfolio construction through mixed integer programming at grantham, mayo, van otterloo and company," *Interfaces*, vol. 29, pp. 49 – 66, 1995.

[7] CHVATAL, V., *Linear Programming.* W H Freeman & Co., 1983.

[8] CROWDER, H., JOHNSON, E. L., and PADBERG, M., "Solving large scale zero-one integer programming problems," *Operations Research*, vol. 31, pp. 803–834, 1983.

[9] CROXTON, K. L., GENDRON, B., and MAGNANTI, T. L., "A comparison of mixed-integer programming models for non-convex piecewise linear cost minimization problems," Tech. Rep. PublicationCRT-2000-31, Centre de Recherche sur les Transports, Universite de Montreal, 2000.

[10] CROXTON, K. L., GENDRON, B., and MAGNANTI, T. L., "Models and methods for merge-in-transit operations," *Transportation Science*, vol. 37, pp. 1 – 22, 2003.

[11] DAKIN, R. J., "A tree search algorithm for mixed integer programming problems," *Computer Journal*, vol. 8, pp. 250–255, 1965.

[12] DANTZIG, G. B., "On the significance of solving linear programming problems with some integer variables," *Econometrica*, vol. 28, pp. 30–44, 1960.

[13] DANTZIG, G. B., "Linear programming," *Operations Research*, vol. 50, pp. 42 – 47, 2002.

[14] DE FARIAS, I. R., *A polyhedral approach to complementarity programming problems.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1995.

[15] DE FARIAS JR., I. R., *A Polyhedral Approach to Combinatorial Problems.* PhD thesis, Sachool of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, 1995.

[16] DE FARIAS JR., I. R., JOHNSON, E. L., and NEMHAUSER, G. L., "A generalized assignment problem with special ordered sets: a polyhedral approach," *Mathematical Programming*, vol. 89, pp. 187–203, 2000.

[17] DE FARIAS JR., I. R., JOHNSON, E. L., and NEMHAUSER, G. L., "Branch-and-cut for combinatorial optimization problems without auxiliary binary variables," *The Knowledge Engineering Review*, vol. 16, pp. 25–39, 2001.

[18] DE FARIAS JR., I. R., JOHNSON, E. L., and NEMHAUSER, G. L., "Facets of the complementarity knapsack polytope," *Mathematics of Operations Research*, vol. 27, pp. 210–226, 2002.

[19] DE FARIAS JR., I. R. and NEMHAUSER, G. L., "A family of inequalities for the generalized assignment polytope," *Operations Research Letters*, vol. 29, pp. 49–55, 2001.

[20] DE FARIAS JR., I. R. and NEMHAUSER, G. L., "A polyhedral study of the cardinality constrained knapsack problem," Tech. Rep. 2002-02, CORE Discussion Paper, CORE, Universite Catholique de Louvain, 2002.

[21] DRIEBECK, N. J., "An algorithm for the solution of mixed integer programming problems," *Management Science*, vol. 12, pp. 576–587, 1965.

[22] EPSTEIN, R., MORALES, R., SERON, J., and WEINTRAUB, A., "Use of or systems in the chilean forest industries," *Interfaces*, vol. 29, pp. 7 – 29, 1999.

[23] FOURER, R., GAY, D. M., and KERNIGHAN, B. W., *AMPL- A modelling language for mathematical programming.* The Scientific Press, 1993.

[24] GOMORY, R. E., "Some polyhedra related to combinatorial problems," *Linear Algebra and Its Apllications*, vol. 2, pp. 451–558, 1969.

[25] GRAF, T., HENTENRYCK, P. V., PRADELLES-LASSERRE, C., and ZIMMER, L., "Simulation of hybrid circuits in constraint logic programming," *Computers Mathematical Applications*, vol. 20, pp. 45 – 56, 1990.

[26] GU, Z., NEMHAUSER, G. L., and SAVELSBERGH, M. W. P., "Lifted cover inequalities for 0-1 integer programs: Computation," *INFORMS Journal on Computing*, vol. 10, pp. 427–437, 1998.

[27] GU, Z., NEMHAUSER, G. L., and SAVELSBERGH, M. W. P., "Lifted flow cover inequalities for mixed 0-1 integer programs," *Mathematical Programming*, vol. 85, pp. 439–467, 1999.

[28] GU, Z., NEMHAUSER, G. L., and SAVELSBERGH, M. W. P., "Sequence independent lifting in mixed integer programming," *Journal of Combinatorial Optimization*, vol. 4, pp. 109–129, 2000.

[29] HAMMER, P. L., JOHNSON, E. L., and PELED, U. N., "Facets of regular 0-1 polytope," *Mathematical Programming*, vol. 8, pp. 179 – 206, 1975.

[30] JOHNSON, E. L., NEMHAUSER, G. L., and SAVELSBERGH, M. W. P., "Progress in linear programming based algorithms for integer programming: an exposition," Tech. Rep. 97-02, Technical Report, Georgia Institute of Technology, 1999.

[31] KARMARKAR, N., "A new polynomial time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 375–395, 1984.

[32] KHACHIAN, L. G., "A polynomial algorithm in linear programming," *Soviet Mathematics Doklady*, vol. 220, pp. 191–194, 1979.

[33] LAND, A. H. and DOIG, A. G., "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[34] LEE, J. and WILSON, D., "Polyhedral methods for piecewise linear functions: the lambda method," *Discrete Applied Mathematics*, vol. 108, pp. 269–285, 2001.

[35] LINDEROTH, J. and SAVELSBERGH, M. W. P., "Search strategies for mixed integer programming," *INFORMS Journal on Computing*, vol. 11, pp. 173 – 187, 1999.

[36] MARKOWITZ, H. M. and MANNE, A. S., "On the solution of discrete programming problems," *Econometrica*, vol. 25, pp. 84–110, 1957.

[37] MEYER, R. R., "On the existence of optimal solutions to integer and mixed integer programming problems," *Mathematical Programming*, vol. 7, pp. 223–235, 1974.

[38] NEMHAUSER, G. L. and WOLSEY, L. A., *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[39] PADBERG, M., "On the facial structure of set packing polyhedra," *Mathematical Programming*, vol. 5, pp. 199–215, 1973.

[40] PADBERG, M., ROY, T. J. V., and WOLSEY, L. A., "Valid linear inequalities for fixed charge problems," *Operations Research*, vol. 32, pp. 842–861, 1984.

[41] PADBERG, M. W., "Approximating separable nonlinear functions via mixed zero-one programs," *Operations Research Letters*, vol. 27, pp. 1–5, 2000.

[42] ROY, T. J. V. and WOLSEY, L. A., "Valid inequalities for mixed 0-1 programs," *Discrete Applied Mathematics*, vol. 14, pp. 199–213, 1986.

[43] SAVELSBERGH, M. W. P., NEMHAUSER, G. L., and SIGISMONDI, G. C., "Minto, a mixed-integer optimizer," *operations Research Letters*, vol. 15, pp. 47–58, 1994.

[44] SCHRIJVER, A., *Theory of Linear and Integer Programming.* John Wiley and Sciences, 1987.

[45] SHERALI, H. D., "On mized-integer zero-one representations for separable lower-semicontinuous piecewise linear functions," *Operations Research Letters*, vol. 28, pp. 155–160, 2001.

[46] SINHA, G. P., CHANDRASHEKARAN, B. S., MITTER, N., DUTTA, G., SINGH, S. B., ROY, P. N., and CHOUDHURY, A. R., "Strategic and operational management with optimization at tata steel," *Interfaces*, vol. 25, pp. 6 – 19, 1995.

[47] VANDENBUSSCHE, D., *Polyhedral approaches to solving nonconvex quadratic programs.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2003.

[48] WILSON, D., *Polyhedral methods for piecewise linear functions.* PhD thesis, University of Kentucky, Kentucky,, 1998.

[49] WOLSEY, L. A., "Faces for linear inequality in 0-1 variables," *Mathematical Programming*, vol. 8, pp. 165 – 178, 1975.

[50] WOLSEY, L. A., "Facets and strong inequalities for integer programs," *Operations Research*, vol. 24, pp. 367–372, 1976.

[51] WOLSEY, L. A., *Integer Programming.* Wiley, 1998.

[52] ZEMEL, E., "Lifting the facets of zero-one polytopes," *Mathematical Programming*, vol. 15, pp. 268–277, 1978.

VITA

Ahmet Burak Keha was born in Ankara, Turkey on November 2, 1976. He graduated from high school in 1994, majoring in science and math. He studied at the Middle East Technical University in Ankara between 1994 and 1999, where he received B.S. and M.S. degrees in Industrial Engineering. He started his doctoral studies in the School of Industrial and Systems Engineering at the Georgia Institute of Technology in August 1999. His doctoral research concentrated on the development of strong valid inequalities for piecewise linear optimization problems. After receiving a Ph.D. in Industrial Engineering from the Georgia Institute of Technology, he joined the Industrial Engineering faculty at the Arizona State University in August 2003.