Branch-and-Cut for Cardinality Optimization

by

Ankit Sikka, B.S.E.E.

A Thesis

In

INDUSTRIAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

In

INDUSTRIAL ENGINEERING

Approved

Ismael R. de-Farias JR.
Chair

Milton Smith

John Kobza

Ralph Ferguson
Dean of Graduate School

December, 2010

©2010, Ankit Sikka

ACKNOWLEDGEMENTS

My utmost gratitude goes to my advisor, Dr. Ismael de Farias, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. This work would have been impossible without his guidance. He is not only an outstanding professor with wide knowledge and extraordinary academic achievements, but also a great person. I could not have imagined having a better advisor and mentor for my master study. Besides my advisor, I would like to thank my thesis committee members: Dr. Milton Smith and Dr. John Kobza, for their encouragement, help and insightful comments.

I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

Finally, I want to give my best appreciation to my parents and friends for always being there for me, and for their understandings and supports for me in completing this thesis. Whenever thinking of them, I gained courage to continue my education, even when I was discouraged.

TABLE OF CONTENTS

ABSTRACT

A cardinality constrained linear programming problem (CCLP) is a linear programming problem with an additional constraint which restricts the number of nonnegative variables that may take on positive values. This problem arises in a large variety of applications, such as portfolio optimization, compressive sensing, metabolic engineering, and maximum feasible subsystem. In this thesis we review the branch-and-cut approach for CCLP, and we focus on the polyhedral approach to it. We first present branching strategies to solve this model through branch-and-cut. To set the stage for important results on CCLP, we give some important results on the cardinality constrained knapsack polytope (CCKP). We then determine when the trivial inequalities define facets of CCKP. Finally, we discuss the nontrivial inequalities, which can be used as cuts in a branch-and-cut scheme.

CHAPTER 1

INTRODUCTION

1.1 Problem Description

Let n be a positive integer and $N = \{1, ..., n\}$. The *cardinality constrained* linear programming problem (CCLP) is

minimize cx

subject to

$$Ax \geq b \tag{1.1}$$

$$u \geq x \geq 0 \tag{1.2}$$

$$\textit{at most } k \textit{ variables } x_j \textit{ can be nonzero} \tag{1.3}$$

$$x \in R^n, \tag{1.4}$$

where x is $n \times 1$, c is $1 \times n$, A is $m \times n$, b is $m \times 1$, u is $n \times 1$, and k is an integer between 1 and n. In CCLP the vector of variables x is to be determined, and the matrices c, A, b, and integer k are the input data. The constraints of CCLP are (1.1) - (1.4), over which the objective function cx will be optimized. We note that the variables x are continuous. We call (1.3) the *cardinality constraint*. Sometimes the number of nonzero components of a vector is called its *support* or 0-*norm*, and denoted by $\| x \|_0$. In this notation, constraint (1.3) can be written as $\| x \|_0 \leq k$.

Cardinality constraints arise in such applications as portfolio selection [12], electrical circuit design [30], gas network optimization [39], data mining [51], and transportation [1].

In this thesis we review the branch-and-cut approach for CCLP, and we focus on

the polyhedral approach to it. For various combinatorial optimization problems, it has been demonstrated that the branch-and-cut algorithm is more effective than pure branch-and-bound [31, 32, 45, 46]. This is because branch-and-cut uses inequalities that approximate the convex hull of the set of feasible solutions of the problem. Unless the problem has a special structure, it is difficult to derive such valid inequalities. Therefore, it is common practice to derive them from knapsack relaxations. This concept was pioneered by Crowder, Johnson and Padberg [46]. For CCLP, the branch-and-cut approach was pioneered by Bienstock [12] and De Farias and Nemhauser [19].

To obtain strong cuts for the branch-and-cut approach, we will study the cardinality constrained knapsack polytope (CCKP) [44] P:

$$P = conv(S), \tag{1.5}$$

where

$$S = \{x \in R^n : x_j \text{ satisfies } (1.2), (1.3), \text{ and } (1.7)\} \tag{1.6}$$

and

$$\sum_{j \in N} a_j x_j \geq b. \tag{1.7}$$

Here S is the set of feasible solutions of the cardinality constrained knapsack set. Note that the valid inequalities for P are also valid for the convex hull of the feasible set of CCLP, and therefore can be used in a branch-and-cut scheme for solving it.

1.2 Applications

As mentioned earlier, CCLP arises in a large variety of problems, such as portfolio optimization, compressive sensing, maximum feasible subsystem, and metabolic engineering. We now give more details about how (1.3) arises in them.

1.2.1 Portfolio Optimization

In 1952, Markowitz [38] proposed a model for balancing returns and risks of investment, the so called *Portfolio Optimization Model*. He suggested that in order to select a portfolio of assets, the expected return should be maximized, subject to the investor's aversion to risk. Here risk is measured as the variance of the expected return. Suppose there are N assets, and $x \in R^n$ is a vector for the weightage of each asset in a portfolio. Then the risk for this group of assets is $x^T Cx$. Matrix C is the covariance matrix of the assets returns. Now, if $r \in R^n$ is the vector of expected returns and λ is the risk-aversion parameter of the investor, then the Markowitz model can be written as

$$\begin{aligned} &\textit{maximize} && rx \text{ - } \lambda xCx \\ &\text{subject to} \\ & && \sum_{j=1}^{n} x_j = 1 && (1.8) \\ & && x \geq 0. && (1.9) \end{aligned}$$

The primary constraint of the above model is (1.8), which is the budget constraint, i.e. that the sum of all weights should be unit. This portfolio optimization problem is a convex quadratic programming problem (QP) and is a special case of convex optimization.

In a highly volatile financial market, however, the task of investment managers not

only involves balancing returns and risk to meet specified investment objectives of the investors, but also to control the transaction limits and costs. The Markowitz optimization model did not consider the constraints for transaction limits and costs associated with buying and selling assets. This was recognized by Perold [47] who gave a more realistic model, where constraints for trading size and limits were also incorporated.

The use of the cardinality constraint (1.3) arises in portfolio optimization to limit the number of transactions. Controlling the number of transactions is necessary as it is an overhead. Note that, in the present case, $u_j = 1 \ \ \forall \ j \in N$.

1.2.2 Maximum Feasible Subsystem

For a given set of inequalities $Ax \geq b$, finding a feasible subsystem with maximum number of inequalities is termed the *Maximum Feasible Subsystem Problem* (MaxFS) [15]. MaxFS has a number of practical applications in fields such as discriminant analysis [50], digital video broadcasting [49], and protein folding [40]. In discriminant analysis, the problem of finding a linear classifier maximizing the number of correct classifications can be formulated in terms of MaxFS [50]. In planning terrestrial digital video broadcasting network, the problem of determining the emission power of a given set of transmitters, in order to maximize the territory coverage, can also be modeled in terms of MaxFS [49]. If there are n transmitters and m areas in a territory, for each area i the signal quality constraint can be written as

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \tag{1.10}$$

where x_j is the emission power of the jth transmitter, a_{ij} is the strength of the signal arriving in area i from jth transmitter, and b_i is the minimum field strength required to cover area i. For additional applications, like radiation therapy [35] and data mining [28], see Amaldi et al. [3, 4, 5].

The relation between MaxFS and CCLP is that MaxFS can be modeled as a 0-*norm* optimization problem. This can be done by adding an artificial variable $y_i \in R$ to each inequality of the system. The objective now is to minimize $\| y \|_0$. Specifically, MaxFS can be written as

$$minimize \qquad \| y \|_0 \tag{1.11}$$

$$subject \;\; to \qquad Ax + y \geq b. \tag{1.12}$$

1.2.3 Compressive Sensing

Compressive sensing is a technique for recovering a signal, e.g. an image, based on the knowledge that the signal is sparse. Sparsity is an effective model for several real-life signals. For instance, the size of an image can be many mega-pixels, but when viewed in the right basis, it will have several negligible coefficients. So this image can be compressed to a much smaller size without affecting its quality. If we consider a signal $x \in R^n$ that is k-sparse, it also signifies that we need only k measurements to reconstruct it. The main objective now is to identify the samples that are required to be measured in order to adequately describe the signal. Another important objective is to study the issues regarding how accurately this approach can recover a signal.

Donho and Elad [27] have shown that recovery for sparsity k is possible up to the order of $\sqrt{m}$. Later, Candes and Tao [16] showed that the recovery up to the order of

$m/log(n/m)$ can be ensured. This was further extended by Baranuik [7], who showed that k-sparse signals can be recovered by using only $m = k + 1$ measurements.

To understand the mathematical model given by Baranuik [7], let us represent a signal by a n-dimensional vector $x \in R^n$ which is k-sparse. This means that the signal x has at most k nonzero coefficients. A signal in R^n can be represented in terms of a basis of $n \times 1$ vectors $\{\psi_i\}_{i=1}^n$. We assume this basis to be orthonormal. Now, if $\psi = [\psi_1|\psi_2|...|\psi_n]$ is a $n \times n$ basis matrix of vectors $\{\psi_i\}$ as columns, a signal x can be written as

$$x = \psi s, \tag{1.13}$$

where s is the $n \times 1$ column vector of weighting coefficients $s_i = \psi^T x$. Now we can say that the signal x, which is k-sparse, has only k of the s_i coefficients in (1.13) as nonzero and the remaining $(n - k)$ as zero. Let us consider a measurement process that computes $m < n$ inner products between x and a collection of vectors $\{\phi_j\}_{j=1}^m$ as in $y_j = \phi x$ [7]. Measurements y_j are arranged in $m \times 1$ vector y and measurements vectors $\{\phi_j\}^T$ as rows in $m \times n$ matrix ϕ. By substituting ψ from (1.13) we get

$$y = \phi x = \psi \phi s = bs, \tag{1.14}$$

where b is an $m \times n$ matrix. To count the number of nonzero entries in s, zero-norm optimization must be considered. The optimization model of Baranuik [7] can now be written as

$$minimize \qquad \| s \|_0 \tag{1.15}$$

$$subject \;\; to \qquad bs = y. \tag{1.16}$$

The above model can recover a k-sparse signal by using $k+1$ measurements.

In general, compressive sensing includes two steps. In the first step, a measurement matrix $A \in R^{m \times n}$ is chosen and measurement $b = Ax$ is taken on a k-sparse signal. In the second step, the signal $x \in R^n$ is reconstructed from $b \in R^m$.

1.2.4 Metabolic Engineering

Metabolic engineering analyzes the metabolic pathways and biochemical reactions in cells, to improve the productivity of a cellular system. This approach can help in modifying the biological pathways to produce certain products for less desirable chemical processes, thereby reducing waste production and improving efficiency.

Analyzing the metabolic pathways is related to the following two problems: the minimal network problem and minimal knockout problem [32]. The minimal network problem finds the minimum set of reactions that can sustain production under certain conditions. And the minimal knockout problem finds the minimum set of reactions that can be knockout in order to stop the production. Flux balance analysis is one of the mathematical models used for analyzing the metabolic pathways. This model can be represented as

$$Sv = 0, v \geq 0, \tag{1.17}$$

where S is the *stoichiometric matrix* of the network and v is the vector of the set of reactions. Now for finding the minimum set of reactions that can lead to the production of products P without any uptake reactions U, the optimization model can be written as [32]:

$$minimize \qquad \| v \|_0 \tag{1.18}$$

$$Sv = 0, v \geq 0, \tag{1.19}$$

$$v_{i\in P} > 0, v_{i\in U} = 0. \tag{1.20}$$

An example of its application is finding the minimal metabolic reactions in *Escherichia Coli* that can sustain the production of biomass on a glucose only medium [13].

1.3 Contents of the Remainder of the Thesis

After we introduced and motivated CCLP, we now show, in the remainder of the thesis, how it can be solved in practice through branch-and-cut. First we review in Chapter two the branch-and-cut approach. Then we discuss, in Chapter 3, previous work on formulation and the closely related issue of how to branch in the branch-and-cut scheme.

Next we enter the main part of the thesis. We give, in Chapter 4, general, but otherwise important results on CCKP. Specifically, we discuss its dimension and when it is nonempty. We also make assumptions for the remainder of the thesis.

Then, in Chapter 5, we discuss those inequalities already present in the original formulation, and for this reason typically called *trivial*. We determine when they define facets of CCKP.

In Chapter 6 we discuss those inequalities that can be used as *cuts* in a branch-and-cut scheme, and for this reason typically called *nontrivial*. We review the types of inequalities in the literature: The *critical set inequalities* of Bienstock [12] and the *lifted cover inequalities* of de Farias and Nemhauser [19].

In Chapter 7 we review the computational results of Bienstock and de Farias and Nemahuser. Finally, in chapter 6, we present conclusions and directions for further research.

CHAPTER 2

REVIEW OF THE BRANCH-AND-CUT APPROACH

2.1 Introduction

Branch-and-cut is a method of combinatorial optimization for solving integer programming problems where some or all the unknowns are restricted to integer values. It is a generalization of branch-and-bound and generates valid inequalities (cuts) that are not satisfied by all feasible points of LP relaxations. An inequality that does not satisfy the given optimal solution of LP relaxation is called a violated cut. In some nodes of the branch-and-bound tree, when the optimal solution x^* of the LP relaxation is not feasible and it finds a cut it adds it to the LP relaxation and re-solves the LP relaxation. In case no violated cut is found, the node is branched.

2.2 Polyhedral Sets

Let us now introduce some basic concepts, see by Nemhauser and Wolsey [44] for further details.

Definition Given a set $S \subseteq R^n$, a point $x \in R^n$ is a *convex combination* of points of S if there exists a finite set of points $\{x^i\}_{i=1}^t$ in S and a $\lambda \in R_+^t$ with $\sum_{i=1}^t \lambda_i = 1$ and $x = \sum_{i=1}^t \lambda_i x^i$. The *convex hull* of S, denoted $conv(S)$, is the set of all points that are convex combinations of points in S.

Definition A set of points $x^1, ..., x^k \in R^n$ is *affinely independent* if the unique solution of $\sum_{i=1}^k \alpha_i x^i = 0, \sum_{i=1}^k \alpha_i = 0$, is $\alpha_i = 0$ for $i = 1, ..., k$.

Definition A polyhedron $P \subseteq R^n$ is the set of points that satisfy a finite number of

linear inequalities: that is, $P = \{x \in R^n : Ax \leq B\}$, where (A, b) is an $m \times (n+1)$ matrix.

Definition A polyhedron $P \subseteq R^n$ is bounded if there exists an $\omega \in R_+$ such that $P \subseteq \{x \in R^n : -\omega \leq x_j \leq \omega \text{ for } j = 1, ..., n\}$. A bounded polyhedron is called a *polytope*.

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

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

Definition The inequality $\pi x \leq \pi_0$ is called a *valid inequality* for P if it is satisfied by all points in P.

Definition 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 (π, π_0) represents F. A *face* F is said to be proper if $F \neq \emptyset$ and $F \neq P$.

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

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

Theorem A full-dimensional polyhedron P has a unique minimal representation by a finite set of linear inequalities. In particular, for each facet F_i of P there is an inequality $a^i x \leq b_i$ representing F_i and $P = \{x \in R^n : a^i x \leq b_i \text{ for } i = 1, ..., t\}$.

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

2.3 The Branch-and-Cut Algorithm

The branch-and-cut algorithm is given below:

1. Perform Preprocessing
2. $P^0 \leftarrow P, L \leftarrow \{P^0\}, z^0 \leftarrow \infty, z_{lb} \leftarrow \infty, x^* \leftarrow void$
3. If $L = \varnothing$, stop. The point x^* is an optimal solution
4. Choose $P^i \in L$; $L \leftarrow L - \{P^i\}$
5. If $z^i \leq zlb$, go to 3 (prune by bound).
6. Solve the LP relaxation LP^i of P^i.
7. If LP^i is infeasible, go to 3 (prune by in-feasibility).
8. $z^i \leftarrow z(LP^i)$
9. If $z^i \leq z_{lb}$, go to 3 (prune by bound).
10. If x^* is integral, $z_{lb} \leftarrow z^i, x^* \leftarrow x^{i*}$, and go to 3 (prune by integrality).
11. Solve the separation problem for x^{i*}. If there is a cut, add it to LP^i and go to 6.
12. Divide the feasible set of P^i, and let P^{i1} and P^{i2} be the resulting problems . $L \leftarrow L \bigcup \{P^{i1}, P^{i2}\}, z^{i1} \leftarrow z^i$, and $z^{i2} \leftarrow z^i$. Go to 3 (branching).

2.4 Pre-processing

Pre-processing refers to some logical operations that are performed to improve the formulation by tightening the bounds on variables, or fixing values. This might result in reduction of size or constraint elimination. Pre-processing also helps in detecting in-feasibility. It can extremely enhance the speed of an algorithm, for example, sometimes it is not easy to recognize that certain variables can be fixed and then eliminated from the formulation. Pre-processing can also be used with probing, where a binary variable is temporarily fixed to 0 or 1 and then logical testing is applied. For example, if for $x = 0$ the IP is infeasible, then $x = 1$ is tried. In brief following operations can be performed in pre-processing:

- Testing for feasibility.
- Removing redundant constraints.
- Fixing variables.
- Tightening bounds.
- Improving inequality coefficients.
- Adding logical inequalities.

2.5 Branching

In a branch-and-cut algorithm, if a node cannot be pruned, a cut which violates the optimal solution of the LP relaxation is added. The basic approach used in branching is branch-and-conquer. Every node is represented as MIP. The first node P^0 is called the root node and represents the MIP we want to solve. The nodes representing the

children of a MIP are called children nodes, and the node of their parents are their parent node. When no enumeration is needed beyond a node, we say that the node has been pruned. Open nodes represent MIPs in the enumeration tree that have not been explored yet. They are the leaves of the tree that have not been pruned. Criteria for pruning a node are as follows:

- The linear programming relaxation (LPR) is infeasible.
- LPR has an optimal value not greater than the value of the best current MIP feasible solution known.
- LPR satisfies integrality.

The practical success of this algorithm is dependent on our ability to prune as many nodes as often, preferably even without solving their LPR. Branching should generate only two children, i.e. we should divide the MIP of a node into two subproblems. There is always an attempt to obtain children with potentially sharper LPR, for example with potentially sharper upper bounds to their MIPs. In branch-and-bound, the LP relaxation is solved and a variable with fractional solution is chosen. Now the node is chosen for evaluation and the fraction variable to branch. To keep the number of enumerations less, choosing the variable to branch on becomes very critical. The basic rule for that will be to select a fractional variable whose value is near to $\frac{1}{2}$. Otherwise it is also good to choose a variable such that the optimal value is as small as possible. In order to make the tree small, the early variable selections becomes very crucial. Computing the objective value becomes expensive if we calculate for all variables. In place of that, the rate of change of objective function can be estimated on both the up and down branches. Another approach can be Strong Branching,

where a large number of iterations are done for a few number of variables. These iterations give lower bounds that help in selecting the variable to branch on. Because memory is cheaper, if we can be smart about branching, we can allow for a tree size that, a few years back was unimaginable. Today, the best MIP software is not afraid of allowing enumeration trees to grow to millions of nodes.

2.6 Node Selection

The criteria for node selection relies on finding feasible solutions during the early stages of enumeration. Balancing the pros and cons of node selection near the top and bottom of tree is difficult. The number of nodes will be more. if we select the nodes from bottom. But if we select the number of nodes from the top, the number of active nodes will be less. This approach will also take more time to get a good feasible solution. If the branching variable is a binary variable, then one branch will have the value 0 and the other one will have 1. On the other hand, if the variable has a fractional value, and this value is between x and $x + 1$, one branch would have constraint $\leq x$ and the other $\geq x + 1$.

2.7 Cutting Planes

To find a violated cut i.e. finding a solution of the LP relaxation that does not satisfy integrality, is a separation problem. Although it is difficult to fins facet-defining inequality, it is necessary to have a violated cut that defines the convex hull of the feasible solutions. In general, heuristics are introduced to find these valid inequalities, as there is a trade off between the time taken to find a cut and the strength of the cuts. Some times these heuristics are not able to find violated inequalities.

2.8 Branch-and-cut without Auxiliary Binary Variables

Besides integrality, there are a few combinatorial constraints that are pervasive in applications. In such cases, it is a good idea to use special facilities to deal with them. In particular, specialized branching schemes for them. Some of them also provide an alternative strategy, in which one does not use auxiliary binary variables to enforce them. Rather, they enforced algorithmically, directly in the branch-and-cut algorithm, through specialized branching scheme. Laundy and Bienstock studied specialized branching schemes for cardinality constraints. Suppose that at most k of the variables $x_1, ..., x_n$, can be positive. This can be modeled as:

$$x_j \leq u_j y_j \tag{2.1}$$

$$y_1 + ... + y_n \leq k \tag{2.2}$$

where $y_j \in B, \quad I = 1, ..., n$ The Laundy-Bienstock scheme is a projection onto the continuous variable space of $0 - 1$ variable dichotomy branching of the MIP formulation. This argument is enough to show that Branch-and-bound requires finitely many nodes to complete enumeration when the problems are bounded. In the case of Laundy-Bienstock branching, however, the right child node, corresponding $x_h = 1$, may contain the optimal solution of the node being considered. Such solution may reappear as the optimal solution of a few more right children nodes. Despite of this, because at each new branching level the RHS of the branching inequality is reduced by 1, the Laundy-Bienstock branching usually performs well in practice. Sometimes, the use of a higher-dimensional formulation improves our ability to solve the problem faster. This does not seem seem to be the case with cardinality optimization. The

use of binary variables in this case appears to simply add unproductive fat to the model. Specifically it increases the:

- Size of the model.
- Rank of the matrix.

The only reason why anybody would use binary variables would be

- Lack of a formulation without auxiliary binary variables for other combinatorial constraints that use the same binary variables as cardinality.
- Belief that the arsenal for binary variables present in state-of-the-art software will make a difference.

CHAPTER 3

MODELING AND BRANCHING STRATEGY

In this chapter we discuss the issue of how constraint (1.3) can be formulated by using the language of mathematical programming. First, we present Dantzig's formulation [17]. Then, the more recent combinatorial formulation of Laundy and Bienstock, proposed independently first by R. Laundy [34] and later by D. Bienstock [12].

A closely related issue is how to branch on a branch-and-cut scheme to solve the formulation. The answer to this question is obvious for Dantzig's formulation, given than it is a $0-1$ MIP formulation. However, it is not trivial for the Laundy-Bienstock formulation. Therefore, we present the Laundy-Bienstock branching strategy to solve this model of (1.3) through branch-and-bound.

For the remainder of the thesis we assure that $u_j = 1 \ \forall j \in N$, which can always be achieved by scaling of x_j, in case x_j is bounded (which we assume from now on).

Constraint (1.3) can be modeled by introducing the auxiliary $0-1$ variables z_j, $j \in N$, and the constraints

$$x_j \leq z_j, j \in N, \tag{3.1}$$

$$\sum_{j \in N} z_j \leq k, \tag{3.2}$$

The approach to model (1.3) through (3.1) and (3.2) was introduced by Dantzig [17].

Alternatively to modeling (1.3) through (3.1) and (3.2), Laundy [34] and Bienstock [12] independently suggested managing without auxiliary $0-1$ variables and enforcing

(1.3) directly in the branch-and-bound algorithm through a special branching scheme. In this approach, constraint (1.3) is modeled using the surrogate constraint

$$\sum_j x_j \leq k. \tag{3.3}$$

Specifically, let x^* be a solution to CCLP. Let $r \in N$. Because of (1.3), either

$$x_r^* = 0 \tag{3.4}$$

or

$$\sum_{j \in N - \{r\}} x_j^* \leq k - 1 \tag{3.5}$$

Laundy [34] and Bienstock [12] proposed a branching scheme based on (3.4) and (3.5). The branching is done as follows. Suppose that at the current node of the enumeration tree, t is the number of variables that have been branched on and F is the set of variables that have not been branched on. Constraint (3.5) at the current node is

$$\sum_{j \in F} x_j \leq k - t, \tag{3.6}$$

where $0 \leq t < k$.

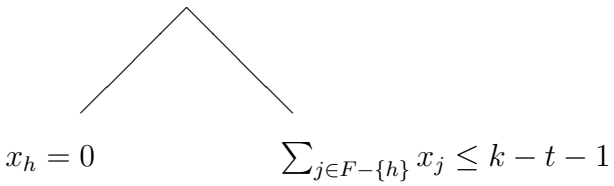
Now, if we choose any variable $h \in F$ to branch on, then (3.5) in the up branch will become

$$\sum_{j \in F - \{h\}} x_j \leq k - t - 1. \tag{3.7}$$

As for the other branch, we take

$$x_h = 0 \tag{3.8}$$

Note that in an enumeration tree for this problem, we can fathom any node that corresponds to k up branches. Therefore, this branching scheme ends within a finite number of nodes [19].



Observe here that the same node may reappear in the up branching, and that the Laundy-Bienstock [34, 12] branching is a projection onto $x - space$ of the $0 - 1$ branching. Let us take an example where we branch on variable z_j. Imposing $z_j = 0$ in constraint (3.1) implies $x_j = 0$ and imposing $z_j = 1$ in constraint (3.2) implies $\sum_{i \in F-\{j\}} x_i \leq k - t - 1$. Therefore, the LP relaxation solutions do not reoccur as the z variables are fixed at 0 or 1. On the other hand, in Laundy-Bienstock [34, 12] branching, the x variables may remain same in the up branch where $z_j = 1$.

Some potential advantages of Laundy-Bienstock [34, 12] approach are [25]:

- Numerical Accuracy.
- Solving LP relaxations is faster. Addition of auxiliary $0 - 1$ variables increases the number of variables and constraints, which adds on the time required to solve the linear relaxation.

- Less enumeration. By addition of auxiliary $0-1$ variables, the LP relaxation might have fractional basic optimal solution that satisfy the cardinality constraint (1.3). Even though it satisfies the cardinality constraint, the MIP software will continue branching, to achieve integrality of the auxiliary $0-1$ variables.

- Big-M constraints. In many optimization problems, it is not known how large the values of the variables can be. So when auxiliary $0-1$ variables are introduced, it is necessary that the continuous variables are bounded. This is done by using big-M constraints or (3.1) in the above case. Usually these constraints are not tight, but when they are, they increase the degeneracy.

De Farias [19, 21, 26] extended both MIP and Laundy-Bienstock [34, 12] approaches by studying the polyhedron of $(1.1)-(1.4)$ in the space of x variables. He obtained a number of valid inequalities for the *cardinality constrained knapsack polytope*, and demonstrated their efficiency in branch-and-cut through extensive testing (see Nemhauser and Wolsey [31] for a detailed description of the branch-and-cut algorithm).

One obvious advantage of modeling (1.3) through (3.1) and (3.2) is that the formulation resulting from it can be readily input to an academic or commercial MIP solver. This contrasts with the Laundy-Bienstock [34, 12] approach, which would have to be coded by the user in any present commercial solver available. Another arguable advantage of the MIP approach is the use of the powerful machinery available in commercial MIP solvers, such as preprocessing, primal heuristics, and cutting planes. We note that, as shown by [41, 42], it is not possible to model (1.3) through MIP unless all the variables x are bounded.

CHAPTER 4

THE CARDINALITY CONSTRAINED KNAPSACK POLYTOPE

In this chapter we give results on CCKP, that will be used to set the stage for the more important results on cuts for CCLP. First we make assumptions. Then, we discuss the dimension of CCKP. This particular result is essential in establishing the strength of the inequalities that will be presented in the next two chapters.

First, we assume without loss of generality that

$$a_1 \geq a_2 \geq \ldots \geq a_n. \tag{4.1}$$

Next, we assume, also without loss of generality, that

$$2 \leq k \leq n-1 \tag{4.2}$$

The reason this assumption does not carry loss of generality is that CCLP is trivial when $k = 1$ and constraint (1.3) is redundant when $k = n$.

Now we assume that

$$b > 0 \quad and \quad a_n > 0. \tag{4.3}$$

This assumption obviously carries loss of generality. The reason for it is that we want to focus on the model by Bienstock [12], which has been less studied than the model by De Farias and Nemhauser [19]. Once (4.3) is assumed, we can assume without loss of generality that

$$u_j = 1 \ \forall j \in N. \tag{4.4}$$

Such assumption can be made without loss of generality since x_j can be scaled in case $u_j \neq 1$ originally.

We now give a necessary and sufficient condition for CCKP to be nonempty.

Theorem:1 $P \neq 0$ if and only if

$$\sum_{j=1}^{k} a_j \geq b \tag{4.5}$$

Proof. Since we assumed (4.3) and (4.5), $\max \{\sum_{j=1}^{n} a_j x_j : x \in CCKP\} \leq \sum_{j=1}^{k} a_j$. If (4.5) does not hold, then $\sum_{j=1}^{n} a_j x_j < b \ \forall x \in R^n$ that satisfy (1.2) and (1.3). This implies that $P = 0$. So, (4.5) is a necessary condition for $(P \neq 0)$.

Suppose (4.5) holds. Since $x_1 = \cdots = x_k = 1, x_{k+1} = \cdots = x_n = 0$ satisfy (1.2), (1.3), and (1.7), it is a feasible solution, and $P \neq 0$. So, (4.5) is a sufficient condition for $P \neq 0$.

We henceforth assure that (4.5) holds, and therefore that CCKP is nonempty. □

Example 1. Let $n = 6$, $k = 3$, and (1.1) be

$$9x_1 + 6x_2 + 5x_3 + 2x_4 + 2x_5 + x_6 \geq 11 \tag{4.6}$$

Here $\sum_{j=1}^{k} a_j > 11$.

Hence, $P \neq 0$.

Now, we give a necessary and sufficient condition for P to be full-dimensional. The importance of this result resides in the fact that, in such cases, the facet-defining

inequalities for P are uniquely determined up to a positive multiplicative factor.

Theorem 2: P has dimension n (i.e. P has $n+1$ affinely independent points) if and only if the following two conditions hold:

$$\sum_{j=1}^{k} a_j > b \tag{4.7}$$

$$\sum_{j=1}^{k-1} a_j + a_i \geq b, i = k+1, \ldots, n \tag{4.8}$$

Proof. 'If': A finite set of points $\{x^1, \ldots, x^{n+1}\}$ is affinely independent if $a_1 = \ldots = a_{n+1} = 0$ is the only solution of $\sum_{j=1}^{n+1} a_j = 0$ and $\sum_{j=1}^{n+1} a_j x^j = 0$.

This can be written in matrix form as:

$$\begin{bmatrix} 1 & \cdots & 1 \\ \\ x_1^1 & \cdots & x_1^{n+1} \\ x_2^1 & \cdots & x_2^{n+1} \\ \vdots & & \vdots \\ x_n^1 & \cdots & x_n^{n+1} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{n+1} \end{bmatrix} = 0$$

To prove that $x_1, \ldots, x_{n+1} \in P$ are affinely independent points, we have to show that the rank of the above $(n+1) \times (n+1)$ matrix, call it X, is full (for example, by calculating its determinant, which cannot be zero).

Because $\sum_{j=1}^{k-1} a_j + a_i \geq b, i = k+1, \ldots, n$, the following $n-k$ points belong to P.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k-1} \\ x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Now, let $y_i \in (0,1)$ such that $\sum_{j=1, i\neq j}^{k} a_j + a_i y_i \geq b, \forall\ i = 1, \ldots, k$. These y_i exist since $\sum_{j=1}^{k} a_j > b$, and so the following $k+1$ points also belong to P.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k-1} \\ x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} y_1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ y_2 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ y_{k-1} \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ y_k \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Hence, P is full dimensional if

$$det(X) = \begin{vmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ & & & & & & & & & \\ 1 & 1 & \cdots & 1 & 1 & y_1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 & y_2 & \cdots & 1 & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & y_{k-1} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 & y_k \\ & & & & & & & & & \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{vmatrix} \neq 0.$$

Applying the cofactor formula using the last $(n-k)$ rows, the determinant is reduced to

$$det(X) = \pm \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ & & & & & \\ 1 & y_1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & y_2 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & y_{k-1} & 1 \\ 1 & 1 & 1 & \cdots & 1 & y_k \end{vmatrix}.$$

By subtracting the first row from all the other rows, we get

$$det(X) = \pm \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 0 & y_1 - 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & y_2 - 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & y_{k-1} - 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & y_k - 1 \end{vmatrix}.$$

Since this matrix is diagonal, $det(X) \neq 0$ and therefore $Dim(P) = n$.

'Only If': If P is full dimensional, then for every $i \in \{k+1, \ldots, n\}$, there exists at least one solution where $x_i > 0$.

In particular, we have

$\sum_{j=1}^{k-1} a_j + a_i \geq b, i = k+1, \ldots, n$, which is condition 2 of this proof.

As for condition 1, by assumption we have $\sum_{j=1}^{k} a_j \geq b$. So now we have to show that $\sum_{j=1}^{k} a_j \neq b$.

Suppose $\sum_{j=1}^{k-1} a_j + a_k = b$. This will lead to P not being full dimensional, as shown below.

Since $a_1 \geq \ldots \geq a_n \geq 0$, we get $b = \sum_{j=1}^{k-1} a_j + a_k \geq \sum_{j=1}^{k-1} a_j + a_i \geq b$, $i = k+1, \ldots, n$, which implies that $a_k = a_{k+1} = \ldots = a_n$ and so $\sum_{j=1}^{k-1} a_j + a_i = b$, $i = k, \ldots, n$.

This last equation tells us that all the points in S have k entries equal to 1 and the remaining $n-k$ entries equal to zero. Hence, the sum of the n coordinates of all these points is equal to k, and therefore there will not exist $n+1$ affinely independent points in S.

Now, because P is the convex hull of S, it cannot contain $(n+1)$ affinely independent

points either, which contradicts our assumption that P is full dimensional.

Hence, we must have $\sum a_{j=1}^{k} > b$, which completes the proof. □

Example 2. Let $n = 5$, $k = 2$, and (1.1) be

$$9x_1 + 6x_2 + 5x_3 + 2x_4 + x_5 \geq 9 \tag{4.9}$$

Here $\sum_{j=1}^{k} a_j > 9$ and $\sum_{j=1}^{k-1} a_j + a_i \geq 9$, $\forall \;\; i \in \{3, 4, 5\}$.

Hence, the polyhedron is full dimensional.

CHAPTER 5

TRIVIAL INEQUALITIES

In this chapter we examine the inequalities present in the original formulation, i.e. (1.2) and (1.7). Because they are present in the initial formulation, these inequalities are usually called *trivial*. However they are among the most important.

We start with the upper-bound constraints in chapter 3 and 4. We scale all variables, if needed, to assume all upper bounds are equal to 1. Then, we consider nonnegativity, and finally the knapsack inequality. One practical use of our results is to determine these inequalities when they are not facet-defining.

Theorem 3: If P is full dimensional, the inequalities $x_l \leq 1$ are facet defining if the following two conditions hold:

$$\sum_{j=1,j\neq l}^{k-1} a_j > b - a_l \tag{5.1}$$

$$\sum_{j=1,j\neq l}^{k-2} a_j + a_i \geq b - a_l, \quad \forall \quad i \in \{k+1, \ldots, n\} \tag{5.2}$$

Proof. The inequality $x_l \leq 1$ will be facet defining if setting $x_l = 1$ reduces the dimension of the polyhedron by 1. If we do that, we get:

$$a_1x_1 + \ldots + a_{l-1}x_{l-1} + a_{l+1}x_{l+1} + \ldots + a_nx_n \geq b - a_l,$$

with now at most $(k-1)$ non-zero x_j.

Hence, the problem of finding out whether or not the dimension of this new polyhedron is $(n-1)$ can be done using the result from *theorem 2* (which, by the way, still works if we have $b - a_l < 0$). □

Example 3. Let $n = 5$, $k = 3$, and (1.1) be

$$6x_1 + 5x_2 + 4x_3 + 2x_4 + x_5 \geq 12 \tag{5.3}$$

Then $x_1 \leq 1$ and $x_2 \leq 1$ are facet defining. On the other hand $x_3 \leq 1$, $x_4 \leq 1$ and $x_5 \leq 1$ are not facet defining.

Similarly, for *example 1* $x_1 \leq 1$ is facet defining but $x_2 \leq 1$, $x_3 \leq 1$, $x_4 \leq 1$ and $x_5 \leq 1$ are not facet defining.

Also the above polyhedron is full dimensional as it satisfies (4.6) and (4.7).

Theorem 4: The inequalities $x_l \geq 0$ are facet defining if (4.6) and (4.7) hold.

Proof. The answer to this question is similar (but not equal) to the above. The inequality $x_l \geq 0$ will be facet defining if setting $x_l = 0$ reduces the dimension of the polyhedron by 1. If we do that, we get:

$$a_1x_1 + \ldots + a_{l-1}x_{l-1} + a_{l+1}x_{l+1} + \ldots + a_nx_n \geq b,$$

with *still* at most k non-zero x_j.

So again, the problem of finding out whether or not the dimension of this new polyhedron is $(n-1)$ can be done using the result from *Theorem 2.* □

Example 4. Let $n = 4$, $k = 2$, and (1.1) be

$$4x_1 + 3x_2 + x_3 + x_4 \geq 4 \tag{5.4}$$

Then $x_1 \geq 0$ is not facet defining. But $x_2 \geq 0$, $x_3 \geq 0$ and $x_4 \geq 0$ are facet defining for this polyhedron.

Theorem 5: The knapsack inequality $\sum_{j=1}^{n} a_j x_j \geq b$ is facet defining if the polyhedron is full dimensional i.e. inequalities (4.6) and (4.7) hold.

Proof. If we include the equality $\sum_{j=1}^{n} a_j x_j = b$ in the description of P, the new polyhedron, call it F_1, will not have $(n+1)$ affinely independent points, so $Dim(F_1)$ will be $(n-1)$ if we can find n affinely independent points in it.

Let $\epsilon, z_i \in (0,1)$ such that $\sum_{j=1, i\neq j}^{k} a_j z_j + a_i(z_i + \epsilon) = b, \forall\ i = 1, \ldots, k$. These z_i and ϵ exist since $\sum_{j=1}^{k} a_j > b$. Also, we have that $\sum_{j=1}^{k-1} a_j + a_i \geq b, i = k+1, \ldots, n$. Thus, the following n points belong to F_1:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k-1} \\ x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} z_1 + \epsilon \\ z_1 \\ \vdots \\ z_1 \\ z_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} z_2 \\ z_2 + \epsilon \\ \vdots \\ z_2 \\ z_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} z_k \\ z_k \\ \vdots \\ z_k \\ z_k + \epsilon \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

These points are affinely independent if the rank of the following $(n+1) \times n$ matrix is n:

$$\begin{bmatrix}
1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 & 1 \\
 & & & & & & & & \\
1 & 1 & \cdots & 1 & z_1+\epsilon & z_2 & \cdots & z_{k-1} & z_k \\
1 & 1 & \cdots & 1 & z_1 & z_2+\epsilon & \cdots & z_{k-1} & z_k \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
1 & 1 & \cdots & 1 & z_1 & z_2 & \cdots & z_{k-1}+\epsilon & z_k \\
0 & 0 & \cdots & 0 & z_1 & z_2 & \cdots & z_{k-1} & z_k+\epsilon \\
 & & & & & & & & \\
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0
\end{bmatrix}.$$

Consider the submatrix obtained by eliminating the first row and apply the cofactor formula using the last $(n-k)$ rows of this submatrix. Its determinant is reduced to:

$$\pm \begin{vmatrix}
z_1+\epsilon & z_2 & \cdots & z_{k-1} & z_k \\
z_1 & z_2+\epsilon & \cdots & z_{k-1} & z_k \\
\vdots & \vdots & & \vdots & \vdots \\
z_1 & z_2 & \cdots & z_{k-1}+\epsilon & z_k \\
z_1 & z_2 & \cdots & z_{k-1} & z_k+\epsilon
\end{vmatrix} \neq 0.$$

Hence, $\sum_{j=1}^{n} a_j x_j \geq b$ is always facet defining when P is full-dimensional. □

Example 5. Let $n = 4$, $k = 2$, and (1.1) be

$$10x_1 + 7x_2 + 6x_3 + 3x_4 \geq 12 \tag{5.5}$$

Then $\sum_{j=1}^{4} a_j x_j \geq 12$ is facet defining.

CHAPTER 6

NON-TRIVIAL INEQUALITIES

6.1 Introduction

In this chapter we will discuss those inequalities that can be used as *cuts* in the branch-and-cut scheme, and for this reason typically called *nontrivial*. We will then review the two types of inequalities: The sequentially lifted cover inequalities by De Farias and Nemhauser [19] and critical set inequalities by Bienstock [12].

6.2 Cover Inequalities and Lifting

Identifying the facet defining inequalities of high dimensional polytopes is not easy. A very useful procedure used for deriving strong nontrivial inequalities is *lifting* of inequalities. In a lifting process, the polyhedron for which we want to derive facet defining inequalities is projected to a lower dimensional space by fixing few variables their upper or lower bounds. In the next step, facet defining inequalities for the projected polyhedron are obtained. And finally, facet defining inequalities for the original polyhedron are derived by introducing fixed variables in the inequality obtained for the project polyhedron. The variables can be introduced sequentially or simultaneously. In practice sequential lifting is used as simultaneous lifting is computationally expensive [6].

In general, it is easier to find facet defining inequalities in lower dimensional space. Facet derived for these low dimensional polyhedron can then be converted into a facet defining inequality for the higher dimensional polyhedron by using lifting.

But the main problem is to find, what should be the initial projected polyhedron. To address this problem, let us first consider a polytope PS. Let $C = \{j_1, \cdots, j_r\} \subset$

$m \times n$. Suppose we fix the variables $m \times n - C$ to 0. A set $C \subseteq N$ is called a cover if

$$\sum_{j \in C} a_j > b. \tag{6.1}$$

For any cover C the inequality

$$\sum_{j \in C} x_j \leq \mid C \mid -1 \tag{6.2}$$

is known as a *cover inequality*. We can also say that a cover is a subset of the variables that exceed the knapsack capacity when all are set to their upper bounds [53].

Example 6. Let

$$S = \{x \in B^7 : 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 + x_7 \leq 19\} \tag{6.3}$$

The minimal covers and the corresponding inequalities are:

$$C_1 = \{1, 2, 3\} \quad and \quad x_1 + x_2 + x_3 \leq 2 \tag{6.4}$$

$$C_2 = \{1, 2, 6\} \quad and \quad x_1 + x_2 + x_6 \leq 2 \tag{6.5}$$

$$C_3 = \{1, 5, 6\} \quad and \quad x_1 + x_5 + x_6 \leq 2 \tag{6.6}$$

$$C_1 = \{3, 4, 5, 6\} \quad and \quad x_3 + x_4 + x_5 + x_6 \leq 3 \tag{6.7}$$

Example 7. Let

$$S = \{x \in B^5 : 79x_1 + 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 178\} \tag{6.8}$$

The minimal covers and the corresponding inequalities are:

$$C_1 = \{1, 2, 3\} \quad and \quad x_1 + x_2 + x_3 \leq 2 \tag{6.9}$$

$$C_2 = \{1, 2, 4, 5\} \quad and \quad x_1 + x_2 + x_4 + x_5 \leq 3 \tag{6.10}$$

$$C_3 = \{1, 3, 4, 5\} \quad and \quad x_1 + x_3 + x_4 + x_5 \leq 3 \tag{6.11}$$

$$C_1 = \{2, 3, 4, 5\} \quad and \quad x_2 + x_3 + x_4 + x_5 \leq 3 \tag{6.12}$$

Note that $N = \{1, 2, 3, 4, 5\}$ is a cover, but corresponding cover inequality

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 4 \tag{6.13}$$

is weaker than any of the minimal cover inequalities. It is not necessary that the cover inequalities define facets. For example,

$$x_2 + x_3 + x_4 + x_5 \leq 3 \tag{6.14}$$

is valid and it is stronger than the last 3 minimal cover inequalities.

We can now derive facet defining inequalities for polytope PS by lifting cover inequalities. Let us illustrate this with an example.

Example 8. The inequality $x_2 + x_3 + x_4 + x_5 \leq 3$ defines a facet of

$$conv\{x \in B^4 : 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 178\} = \tag{6.15}$$

$conv\{x \in B^5 : 79x_1 + 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 178\} \bigcap \{x \in B^5 : x_1 = 0\}$

We now reintroduce x_1. Let the *lifting coefficient* be α. The inequality is then

$$\alpha x_1 + x_2 + x_3 + x_4 + x_5 \leq 3 \tag{6.16}$$

When $x_1 = 0$, α can have any value.

When $x_1 = 1$, α must satisfy

$$\alpha \leq 3 - x_2 - x_3 - x_4 - x_5, \tag{6.17}$$

or

$$\alpha \leq min\{3 - x_2 - x_3 - x_4 - x_5 : 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 178 - 79, x_2, ..., x_5 \in B\}, \tag{6.18}$$

which is equivalent to

$$\alpha \leq 3 - max\{x_2 + x_3 + x_4 + x_5 : 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 99, x_2, ..., x_5 \in B\} = 3 - 2 = 1. \tag{6.19}$$

So, $\alpha \leq 1$.

In particular, $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$ is valid and facet defining, since the solution of the lifting problem gives a point of S that satisfies the inequality at equality and it is linearly independent from the points for the lower dimensional inequality. Let us take another example of lifting.

Example 9. The inequality

$$x_2 + x_3 \leq 1 \tag{6.20}$$

defines a facet of

$$conv\{x \in B^5 : 53x_2 + 53x_3 \leq 99\} = \tag{6.21}$$

$conv\{x \in B^5 : 79x_1 + 53x_2 + 53x_3 + 45x_4 + 45x_5 \leq 178\}$

$\bigcap\{x \in B^5 : x_1 = 1, x_4 = x_5 = 0\}$.

We now re-introduce x_4. Let the *lifting coefficient* be α_4. The inequality is then

$$x_2 + x_3 + \alpha x_4 \leq 1, \tag{6.22}$$

and α_4 is given by

$$1 - max\{x_2 + x_3 : 53x_2 + 53x_3 \leq 54, x_1, x_2 \in B\} = 0 \tag{6.23}$$

Likewise, $\alpha_3 = 0$.

So $x_2 + x_3 \leq 1$ defines a facet of

$$conv\{x \in B^5 : x_1 = 1\} \tag{6.24}$$

We now re-introduce x_1. Let the *lifting coefficient* be α_1. The inequality is then

$$\alpha x_1 + x_2 + x_3 \leq 1 + \alpha_1, \tag{6.25}$$

and α_1 is given by

$$min\{x_2 + x_3 : 53x_2 + 53x_3 \leq 178, x_1, x_2 \in B\} - 1 = 1. \tag{6.26}$$

Thus

$$x_1 + x_2 + x_3 \leq 2 \tag{6.27}$$

is valid and facet-defining.

In general, fixing variables only at 0 does not yield all possible facets. Likewise, fixing variables only at 1 does not yield all possible facets. Also, lifting variables in a different order yields a different inequality.

6.3 Sequentially Lifted Cover Inequalities

In this section we will review about the results of De Farias and Nemhauser [19] for sequentially lifted cover inequalities. Sequential lifting and lifting of cover inequalities have been studied extensively to extend them to strong inequalities for PS. Sequentially lifted inequalities are facet defining inequalities derived by applying sequential lifting procedure to a set of inequalities that are facet defining for a polytope. These inequalities are called as cover inequalities and are defined by set of indices called as covers. By lifting these cover inequalities, De Farias-Nemhauser [19] derived facet-defining inequalities for PS which are not valid for LPS and hence, can be used in branch-and-cut scheme as cuts. As a result, De Farias-Nemhauser [19] by the use of lifting cover inequalities, derived three families of facet defining inequalities for polyhedron. De Farias and Nemhauser [19] have applied this new approach to several problems such as cardinality constrained knapsack problem.

6.4 Critical Set Inequalities

Critical set inequalities were introduced by Bienstock [12] by assuming the cardinality knapsack polytope in the space of continuous variables with $a_j > 0 \ \ \forall j \in N$ and $b > 0$ in (1.1). Let us consider a system P where

$$\sum_{j \in N} a_j x_j \geq b \tag{6.28}$$

and x satisfies (1.2) - (1.4).

We have already assumed that $u_j = 1 \ \ \forall j \in N$. A set C is called *critical* if for every $J \subset C, \mid J \mid = k, \sum_{j \in J} a_j < b$ [12]. Therefore, we must have that at most $k-1$ variables from C can be positive. We can also say that

$$\sum_{j \subset C} x_j \leq k - 1 \tag{6.29}$$

is a valid inequality for P.

Example 10. Let $n = 5$, $k = 3$, $C = \{1, 2, 3, 4\}$ and (1.1) be

$$5x_1 + 9x_2 + 11x_3 + 12x_4 + 15x_5 \geq 29 \tag{6.30}$$

If we consider a point $v = (1, \frac{1}{3}, 1, 1, 0, \frac{2}{3})$, then v satisfies (6.1) and (1.3) with equality. However, v violates (6.2).

Example 11. Let $n = 6$, $k = 3$, $C = \{1, 2, 3, 4\}$ and (1.1) be

$$8x_1 + 7x_2 + 6x_3 + 4x_4 + 12x_5 + 12x_6 \geq 22 \tag{6.31}$$

If we consider a point $v = (1, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0)$, then v satisfies (6.1) and (1.3) with equality.

However, v violates (6.2).

Bienstock in his work has also shown when (6.2) is facet defining and how to lift it, in case it is not. This approach was used to strengthen individual constraints by using valid inequalities for the $0-1$ knapsack problem. Critical set Inequalities are very important and should be further investigated.

6.5 Surrogate Cardinality Constraint

In chapter 2 we have already shown, how surrogate cardinality constraint (3.3) are used to model (1.3). In this section we will give a necessary and sufficient condition for (3.3) to be facet-defining. Kindly note that by introducing constraint (3.3), the LP relaxation of the model without auxiliary $0-1$ variables becomes as tight as the relaxation for MIP model.

Theorem 6: The inequality $\sum_{j=1}^{n} x_j \leq k$ is facet defining if the polyhedron is full dimensional i.e. inequalities (4.6) and (4.7) hold.

Proof. Like in the previous question, if we include the equality $\sum_{j=1}^{n} x_j = k$ in the description of P, the new polyhedron, call it F_2, will not have $(n+1)$ affinely independent points, so $Dim(F_2)$ will be $(n-1)$ if we can find n affinely independent points in it. Consider the following $(n+1)$ points in F_2:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k-1} \\ x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} y_1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1-y_1 \end{bmatrix}, \begin{bmatrix} 1 \\ y_2 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1-y_2 \end{bmatrix}, \cdots, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ y_k \\ 0 \\ 0 \\ \vdots \\ 1-y_k \end{bmatrix}$$

Among them, there are n affinely independent points if the rank of the following $(n+1) \times (n+1)$ matrix is n:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 \\ & & & & & & & & \\ 1 & 1 & \cdots & 1 & 1 & y_1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 & y_2 & \cdots & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & y_k \\ & & & & & & & & \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 1-y_1 & 1-y_2 & \cdots & 1-y_k \end{bmatrix}.$$

If we eliminate, for instance, the first column and the last row, the submatrix we get

is also a submatrix of X (from *Theorem 2*), and thus the rank of this matrix is n. This shows that $\sum_{j=1}^{n} x_j \leq k$ is always facet defining when P is full-dimensional. $\square$

Example 12. Let $n = 4$, $k = 2$, and (1.1) be

$$10x_1 + 7x_2 + 6x_3 + 3x_4 \geq 12 \tag{6.32}$$

Then $\sum_{j=1}^{4} x_j \leq 2$ is facet defining.

CHAPTER 7

REVIEW OF COMPUTATIONAL APPROACHES

In this chapter we will review the computational results given by De-Farias and Nemhauser [19] and Bienstock [12].

7.1 De-Farias and Nemhauser

In this section we will discuss the computational results reported by De-Farias and Nemhauser [19], that demonstrated the strength of lifted cover inequalities and the advantages of Laundy-Bienstock branching scheme over the traditional MIP approach for this class of problems.

The performance of Cardinality constrained optimization problem was tested on different instances of

- continuous formulation by using Laundy-Bienstock branching scheme in branch-and-cut algorithm and using lifted cover inequalities as cuts.
- continuous formulation, where only continuous variables are used and (1.3) is enforced algorithmically through branch-and-bound algorithm by using Laundy-Bienstock branching scheme.
- MIP formulation, where auxiliary $(0-1)$ variables are introduced, and (1.3) is modeled using (3.1) and (3.2).

Three different instances were tested for each pair $m \times n$, where m is the number of constraints and n is the number of variables. Given below is the table for cardinality k and density of the constraint matrices [19].

Table 7.1. Cardinality and Density for different Instances

$m \times n$	k	% density
20×500	150	50
$20 \times 1,000$	300	50
$20 \times 1,500$	450	50
$20 \times 2,000$	600	30
$20 \times 2,500$	750	42
$30 \times 3,000$	1,000	33
$30 \times 3,500$	1,000	28
$50 \times 4,000$	1,000	25
$50 \times 4,500$	2,000	30
$50 \times 5,000$	2,000	20
$50 \times 5,500$	2,000	18
$50 \times 6,000$	2,000	16
$50 \times 6,500$	1,000	15
$50 \times 7,000$	2,000	14
$70 \times 7,500$	2,000	13
$70 \times 8,000$	3,000	35

where densities of the constraint matrices are calculated as follows

$$100 \times \frac{\text{number of nonzero coefficients of the knapsack constraints}}{mn} \tag{7.1}$$

The cardinality remained same for instances having same m and n.

The instances were generated randomly. The density of the knapsack coefficients were generated uniformly from $1 - n$. Knapsack coefficients a_{ij} and profit coefficients a_j were generated from $5 - 20$ and $10 - 25$ respectively. RHS of knapsack constraints were given by

$$b_i = max\{0.3 \sum_{j \in N} a_{ij}, \text{greatest coefficient of the } i^{th}\text{knapsack} + 1\}, i \in M. \tag{7.2}$$

MINTO 3.0 with CPLEX 6.6 as LP solver was used for testing. For preliminary tests MINTO's node selection alternatives and pre-processing was used. It was found that MINTO is slow in comparison to CPLEX 6.6 for MIP formulation.

Specialized branching scheme proposed by Laundy-Bienstock was used.

Initial tests were performed with MINTO's node selection alternatives and pre-processing. Pre-processing, best-bound node selection, and pre-processing were the once with best performances. These options were then used in the computation for formulation. *Least index rule* where positive variable with least index is selected, was used for variable selection in the specialized branching scheme. To test the MIP formulation, default options in CPLEX were used.

Computational tests were performed using Sun Ultra 2 with UltraSPARC 200 Mhz CPUs and 256 MB memory. It was found that the difference between optimal value of CCOP and the optimAL value of LP relaxation was very less in every instance. There was a reduction of 70% in number of nodes used in branch-and-cut over branch-and-bound for continuous formulation. This suggests that branch-and-cut algorithm is more effective for continuous formulation in comparison to branch-and-bound algorithm. By using branch-and-cut algorithm for continuous formulation, an overall reduction of 97% in the average number of nodes was noticed. This indicates that addition of auxiliary $0-1$ variables does not have much advantage in case of combinatorial constraints.

By using

$$\sum_{j\in C} a_j x_j + \Delta \sum_{jinN_0} x_j + \sum_{jinN_1} a_j x_j \leq b + \sum_{jinN_1} (\alpha_j - a_j \tag{7.3}$$

as cuts in a branch-and-cut scheme for continuous formulation, the overall time reduc-

tion was 62%. There was a significant time reduction of 78% on branch-and-cut for continuous formulation over MIP. This is because of the use of CPLEX 6.6 which is much faster than MINTO 3.0. The above reduction was not only because of decrease in number of nodes, but also because of the size of MIP, which is two times the size of continuous formulation. CPLEX 6.6 generated only Gomory cuts.

7.2 Bienstock

In this section we will discuss the computational results given by Bienstock [12]. Bienstock [12] in his work presented computational experience using branch-and-cut algorithm to solve quadratic programming problems with cardinality constraints. The branch-and-cut algorithm was implemented by using the *best node* strategy to branch on. For example, at the current node, if F is the set of variables that have not been branched on, then from F, the variable furthest from the bounds is selected as the next variable. The basic optimizer had a feasibility tolerance of 10^{-10} for variables and 5×10^{-9} for constraints. If the value of the variable exceeded 10^{-8}, it was taken as positive.

The algorithm was tested for various real-life problems. All these problems were of low rank quadratic type. The data set mainly consisted of

- Quadratic Matrix
- Constraint Matrix
- Right-hand side vector
- Vector for upper bounds of the variables (u_j).

As in common practice, the values of k, upper bounds of variables u_j, and RHS of the inequalities are varied to obtain different problems. So, significant different problems can be obtained by varying k. In addition to that, some problems were solved with a condition that if the variable is positive, it must have a certain minimum value. The following data for these runs was presented in the table:

- Rank of the objective.
- Number of inequalities in the formulation.
- Number of branch-and-cut nodes.
- Number of columns.
- Number of Disjunctive inequalities or cuts.
- Minimum transaction level.

Sun Sparc 10/15 was used for all operations. Some of the problems were also solved by different heuristics other than branch-and-cut. The number of positive variables at the node was about 40, and k was $20 - 25$. If the number of positive variables were higher, surrogate constraints were not used. Due to this the amount of cutting was very unpredictable. Apart from this, the cuts were mainly saturated at a few nodes, and were used to prove optimality or prune the node. The main reason behind this was optimization of a quadratic objective function. In problems, where cutting is done, significant improvement was noticed in terms of reduction of the size of enumeration tree.

The difference between the value at the root and the value of the mixed integer program was of the order of 10^{-7}. This difference for practical purposes should be

zero. This is one of the reasons for making the problems difficult. These problems are combinatorial where the variables are fixed at their upper bounds at the node. Now the work of algorithm is to look for the right set of variables and handling the nonlinearity of the objective function. Please note that this should be done without any change in the original objective function.

For further testing of algorithm, high rank objective problems were generated, by adding a large, diagonal matrix to the quadratic. These problems didn't proved to be difficult for the code. The main reason was, no cutting took place, apart from the ones resulting from strengthening of surrogate constraint at each up branch. As a result, the algorithm was tested without disjunctive cuts.

These problems were found to be more difficult for this particular code. This is because there was not cutting apart from tightening the surrogate inequality. As a result the algorithm ran with no disjunctive cuts. The values of k used were $25, 25, 43, 25,$ and 40. The number of positive variables used were from 40 to 100. When k was 43, the side constraints were forced to a minimum transaction level of 0.01. All these variables were fixed to zero when the quadratic formulation was infeasible. All the optimal solutions were found with one heuristic while the other had no effect.

In further exercises, the value of k was changed to 43, and the constraints were enforced a minimum transaction of 0.01 on all positive variables. So now, several problems with same constraints, but different values of k were analyzed. Out of the two heuristics, the first one had no effect and the second one found the optimal solution, which was proved, by branch-and-cut running on all variables. From a practical point of view the heuristics might be good, but the results indicated that the pro-

posed heuristics were not able to prove optimality in reasonable running time. It was also illustrated that if the objective is flat, and there are a few hundred nonnegative variables at the root, the MIP might be impossible to solve to optimality. Proving optimality was difficult as no cutting took place. Whereas, finding good upper bounds was not difficult.

Bienstock [12] also described his computational experience with a symmetric-

multiprocessing, shared memory computer used for implementing the algorithm. These computers have a number of similar CPU preprocessors connected to a common system memory. These processors can execute tasks simultaneously. This technique is also know as parallel implementation. This is implemented by the use of *threads*, which are created by a program to execute in parallel, explicit computational tasks. Once created, a thread runs its tasks parallel to the main program. On completion, the thread enters a special *signaled* state. The status of a thread can be checked by the main program, by testing whether the thread holds the signaled state or not. Note that, the number of processors and number of threads created by a program are independent quantities. Furthermore, scheduling of threads to processors is completely handled by the operating system. So, the main objective of the user is to decide how to implement parallelism in an algorithm.

Bienstock [12] in his paper solved many nodes in parallel with different parallel architectures. M nodes were simultaneously solved for some integer $M \geq 1$ selected by the user. Selecting M less than the number of processors, resulted in idle periods. On the other hand, selecting M greater than the number of processors resulted in larger branch-and-cut trees. Hence, it was observed that, selecting M equal to the number of processors was the best strategy and yields best results.

ALR computers with four 100 Mhz Pentium processors were used for implementation. As a result, the speed of the branch-and-cut code with multi-threaded implementation was increased by a factor of three. With the increase in level of difficulty of problems, the 4x Pentium implementation became twice as fast. All this highlighted the potential and power of parallelism in branch-and-cut algorithm.

CHAPTER 8

CONCLUSIONS

In this thesis, we studied the inequality description of CCKP and gave facet defining inequalities that can be used in branch-and-cut algorithms. In chapter 2, we reviewed the branch-and-cut approach for cardinality constraints. In chapter 3, we presented the branching strategy for formulation of cardinality constraint (1.3) and discussed the computational advantages of branching without auxiliary $0-1$ variables. In Chapter 4, we discussed the dimension of CCKP and made assumptions. In chapter 5, we examined the trivial inequalities and determined when they are not facet-defining. In chapter 6, we discussed nontrivial inequalities and derived necessary conditions for the surrogate cardinality constraint to be facet-defining. In brief, we have shown how CCLP can be solved through branch-and-cut.

8.1 Further Research

- Derive more nontrivial inequalities for CCLP. It might also be possible to introduce new classes of facet-defining inequalities by using special structure of this problem.

- A computational study for the Bienstock knapsack structure. Bienstock studied cardinality knapsack polytope for continuous variables with $a_j > 0$ and $b > 0$ and introduced critical set inequalities. It would be interesting to further investigate these critical set inequalities.

- Study the polyhedral structure when $a_j > 0$. Some research has already been done by Bienstock and we suggest further research on how to lift critical set

inequalities when $a_j > 0$.

- Give computational results by testing the performance of CCLP for different instances. The derived valid inequalities can be used in a branch-and-cut algorithm for various real life problems. Hence it is necessary to test the performance of CCLP on different instances.

BIBLIOGRAPHY

[1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., and Reddy, M.R., "Applications of Network Optimization," *Operations Research*, 300 (1994).

[2] Amaldi, E. and Kann, V., "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, 209:237-260, (1998).

[3] Amaldi, E., "From finding maximum feasible subsystems of linear systems to feed forward neural network design," PhD thesis, Dep. of Mathematics, EPF-Lausanne, October (1994).

[4] Amaldi, E., "The maximum feasible subsystems problem and some applications," in: *Agnetis A, Di Pillo G, editors. Modelli e Algoritmi per l'ottimizzazione di sistemi complessi. Pitagora Editrice Bologna*, (2003).

[5] Amaldi, E., Pfetsch, M.E., and Trotter, L.E., Jr., "Some structural and algorithmic properties of the maximum feasible subsystem problem," In G. Cornujols, R. Burkard, and G. Woeginger, editors, Proceedings of the 10th Integer Programming and Combinatorial Optimization conference (IPCO'99), pages 45-59. Springer-Verlag,*Lecture Notes in Computer Science*, Vol. 1610.(1999).

[6] Atamturk, A., "Cover and Pack inequalities for (Mixed) Integer Programming," *Annals of Operations Research* 139, 21-38, (2005).

[7] Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M., "A simple proof of the restricted isometry property for random matrices," *Constructive Approximization.*28(3), 253-263, December 2008.

[8] Beale, E.L.M., "Integer Programming" in: K. Schittkowski (Ed.), *Computational Mathematical Program- ming*, NATO ASI Series, Vol. F15, Springer-Verlag, 124, (1985).

[9] Beale, E.L.M., Tomlin, J.A., "Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables," In: J. Lawrence (Ed.), Proceedings of the Fifth International Conference on Operations Research, Tavistock Publications, 447454, (1997).

[10] Bhaskar, K., "A multiple objective approach to capital budgeting," *Accounting and Business Research*, 9:25-46, (1979).

[11] Biegler, I.E., Grossmann, A.W. Westerberg, "Systematic Methods of Chemical Process Design," Prentice Hall,Englewood Cliffs, NJ (1997)

[12] Bienstock, D., "Computational study of a family of mixed-integer quadratic programming problems," *Math. Program.* 74, 121140 (1996).

[13] Boyd, S., "l1 -norm norm methods for convex cardinality problems," Lecture Notes for EE364b, Stanford University, (2007). Available online at www.stanford.edu/class/ee364b/

[14] Caprara, A., Kellerer, H., Pferschy, U., and Pisinger, D., "Approximation algorithms for knapsack problems with cardinality constraints," *European Journal of Operational Research*, 123:333-345, (2000).

[15] Chinneck, J.W., "Fast heuristics for the maximum feasible subsystem problem," *INFORMS J. Comput.*, 13(3):210-223, (2001).

[16] Cand's, E.J., Romberg, J., and Tao, T., "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information," *IEEE Transactions on Information Theory*, 52:489-509 (2004).

[17] Dantzig, G.B., "On the significance of solving linear programming problems with some integer variables," *Econometrica* 28, 3044 (1960).

[18] De-Farias, I.R., "A family of facets for the uncapacitated p-median polytope," *Oper. Res. Lett.* 28, 161167 (2001)

[19] De-Farias, Jr., I.R., and Nemhauser, G.L.,. "A polyhedral study of the cardinality constrained knapsack polytope," in *Integer Programming and Combinatorial Optimization, 9th IPCO conference*, volume 2337 of *Lecture Notes in Computer Science*, pages 291-303 (2002)

[20] De-Farias, Jr., I.R., Johnson, E.L., Nemhauser, G.L., "Branch-and-cut for combinatorial optimization problems without auxiliary binary variables," *Knowledge Eng. Rev.* 16, 2539 (2001).

[21] De-Farias, Jr., I.R., "A Polyhedral Approach to Combinatorial Complementarity Programming Problems," Ph.D. Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA (1995).

[22] De-Farias, Jr., I.R., Johnson, E.L., and Nemhauser, G.L., "Facets of the Complementarity Knapsack Polytope," to appear in Mathematics of Operations Research.

[23] 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*, 89, 187-203 (2000).

[24] De-Farias, Jr., I.R., and Nemhauser, G.L., "A Family of Inequalities for the Generalized Assignment Polytope," *Operations Research Letters* 29, 49-51 (2001).

[25] De-Farias, Jr., I.R., and Nemhauser, G.L., "Branch-and-Cut for Combinatorial optimization Problems without Auxiliary Binary Variables," *The Knowledge Engineering Review*, 16(1), 25-39, (2000).

[26] De-Farias, Jr., I.R., and Nemhauser, G.L., "A Polyhedral Study of the Cardinality Constrained Knapsack Problem," *Mathematical Programming* 98, 115-143 (2003).

[27] Donho, D.L., "Compressed Sensing" For most underdetermined systems of linear equations, the minimal 1 - norm near-solution approximates the sparsest near-solution. *Manuscript*, (2004)

[28] Glover, F., "Improved linear and integer programming models for discriminant analysis," *Decision Sciences*, 21:771-785, (1990).

[29] Gomory, R.E., "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematical Society*, 64, 275-278 (1958).

[30] Graf, T., Hentenryck, P.V., Pradelles-Lasserre, C., and Zimmer, L., "Simulation of hybrid circuits in constraint logic programming," *Computers and Mathematics with applications* 20(9-10), 45-56 (1990).

[31] Grotschel, M., Junger, M., and Reinelt, G., "A Cutting Plane Algorithm for the Linear Ordering Problem," *Operations Research* 32, 1195-1220 (1984).

[32] Julius A.A., Imielinski M., Pappas G.J., "Metabolic Networks Analysis using Convex Optimization," *IEEE Conference on Decision and Control, Cancun, Mexico*, Dec. 9-11, (2008).

[33] Kellerer, H., Pferschy, U., and Pisinger, D., "Knapsack problems," *Springer*,235-283, (2004).

[34] Laundy, R.S., "Some Logically Constrained Mathematical Programming Problems," Ph.D. Thesis, University of Southampton, Southampton, UK, (1983).

[35] Lee, E.K., Gallagher, R.J., and Zaider, M., "Planning implants of radionuclides for the treatment of prostate cancer: An application of mixed integer programming," *Optima*, 61:1-7, (1999).

[36] Markowitz, H.M., "Portfolio Selection," *Journal of Finance* 7(1): 77-91 (1952)

[37] Markowitz, H.M., Manne, A.S., "On the solution of discrete programming problems," *Econometrica 25*, 84110 (1957).

[38] Markowitz, H. M., "Portfolio Selection, Efficient Diversification of Investments," *Cowles Foundation, Monograh 16, Yale University Press, New Haven*, (1959).

[39] Martin, A., Moller, M., Moritz, S., "Mixed Integer Models for the Stationary Case of Gas Network Optimization," *Mathematical Programming* 105(2-3), 563-582 (2006).

[40] Meller, J., Wagner, M., and Elber, R., "Maximum feasibility guideline in the design and analysis of protein folding potentials," *J. of Computational Chemistry*, 23:111-118, (2002).

[41] Meyer, R., R., "Integer and Mixed-Integer Programming Models: General Properties," *Journal of Optimization Theory and Applications* 16, 191-206 (1975).

[42] Meyer, R., R., "Integer and Mixed-Integer Programming Models for Piecewise-Linear Functions of a Single Variable," *Discrete Mathematics* 16, 163-171 (1976).

[43] Mitchell, J.E., "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of Applied Optimization*, Oxford University Press, 65-77, (2002).

[44] Nemhauser, G.L., Wolsey, L.A., "Integer Programming and Combinatorial Optimization," *John Wiley and Sons*, (1988).

[45] Padberg, M. and Rinaldi, G., "Optimization of a 532 City Symmetric Traveling Salesman Problem by Branch and Cut," *Operations Research Letters* 6, 1-7 (1987).

[46] Padberg, M.W. "A Note on Zero-One Programming," *Oper. Res.* 23, 883837 (1975).

[47] Perold, A.F. "Large-Scale portfolio optimization," *Mgmt. Sci.* 30, 11431160 (1984).

[48] Pfetsch, M.E., "Branch-and-cut for Maximum Feasible Subsytem Problem," Working Paper, ZIB, Berlin, Germany (2005).

[49] Rossi, F., Sassano, A., Smriglio, S., "Models and algorithms for terrestrial digital broadcasting," *Annals of Operational Research*,(107):26783, (2001).

[50] Rubin PA., "Solving mixed integer classification problems by decomposition," *Annals of Operations Research*74:5164, (1997).

[51] Scholkopf, B., and Decoste, D. "Training Invariant Support Vector Machines," *Machine Learning* 46(1-3), 161-190 (2002).

[52] Teng, J.Y. and Tzeng, G.H., "A multiobjective planning approach for selecting non independent transportation investment alternatives," *Transportation Research B*, 30:291-307, (1996).

[53] Wolsey, L.A., "Facets and Strong Valid Inequalities for Integer Programs," *Operations Research* 24, 267-372 (1976).