THEORETICALLY AND COMPUTATIONALLY IMPROVING BRANCH AND BOUND THROUGH MULTIVARIATE BRANCHING WITH INTERNAL CUTTING PLANES

by

JIN HUA LEE

B.A., Johns Hopkins University, 2008

A THESIS

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2010

Approved by:

Major Professor

Dr. Todd Easton

ABSTRACT

Integer Programs (IP) are a class of discrete optimization problems that are utilized commercially to improve the function of various systems. Implementation is often aimed at reaching optimal financial objectives with constraints on resources and operation. While incredibly beneficial, IPs are $\mathcal{NP}$-complete, with many IP models being unsolvable.

Branch and bound (BB) is the primary method employed to solve IPs to optimality. BB is an exhaustive approach to enumerating all potential integer solutions for a given IP. By utilizing a hierarchical tree structure to tabulate progression of enumeration, BB can guarantee an optimal solution in finite time. However, BB can take an exponential number of iterations to solve an IP. Computationally, this can result in a tree structure that exceeds a computer's memory capacity, or a prohibitively long solution time.

This thesis introduces a modified version of BB call the Quaternary Hyperplane Branching Algorithm (QHBA). QHBA employs a quaternary branching scheme, utilizes hyperplane branching constraints, and generates internal cutting planes to increase efficiency. Implementation of these advancements theoretically improves QHBA in comparison to traditional BB. It can also be shown that QHBA guarantees an optimal solution in a finite number of iterations. A short computational study shows that QHBA results in a 26.7% decrease in solution times when compared to CPLEX, a commercially available IP solver.

Dedication

My work is dedicated my mother and father, Yuan and E. Stanley Lee, for their love and support throughout my life.

Acknowledgments

I would like to most importantly acknowledge Dr. Todd Easton. Throughout the entire research process, his help and support has been un-tiring. His technical knowledge, and most importantly patience, made this research possible. Additionally, I would like to thank Dr. Bradley Kramer and Dr. Anil Pahwa for their work and participation on my review committee. Finally, I would like to express my sincere thanks to the Faculty and Staff of the Department of Industrial and Manufacturing Systems Engineering for not only supporting me in my academic ventures, but also for showing kindness and support that can only be described as family.

Contents

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**

1.1 Motivation 6

1.2 Research Contribution 7

1.3 Thesis Outline 9

2 Background Information **10**

2.1 IP Definitions 11

2.2 Branch and Bound 11

2.2.1 Search Strategies 14

2.2.2 Branch and Bound Requirements 26

2.2.3 Hyperplane Branching 26

2.3 Cutting Planes 29

2.4 Totally Unimodular Matrices 33

3 The Quaternary Hyperplane Branching Algorithm (QHBA) 35

3.1 QHBA 35

3.2 Theory of QHBA 49

3.3 Theoretical Benefits of QHBA 57

4 Computational Results for the Quaternary Hyperplane Branching Algorithm 67

5 Conclusion 79

5.1 Future Research 80

List of Figures

1.1 Binary partition for branch and bound 5

2.1 Graph for the IP 19

2.2 First branching partition for Example 1 21

2.3 A Branch and Bound enumeration tree 22

2.4 A Branch and Bound enumeration tree using depth first right 25

2.5 Graph for the IP with cutting planes 31

3.1 Subproblems T_1^{LL} and T_1^{LG} 44

3.2 Subproblems T_1^{GL} and T_1^{GG} 46

3.3 QHBA Enumeration Tree 48

3.4 One set of children for QHBA and QBB 60

List of Tables

4.1 Data Reported for QHBA and CPLEX 76

4.2 Data Reported for QHBA and QBB 77

4.3 Data Reported for QBB and CPLEX 78

Chapter 1

Introduction

An integer program is a class of discrete optimization problems with the defining characteristic of a categorical integer restriction on the variables in the program. An integer program (IP) is canonically defined as max $c^T x$ subject to $Ax \leq b$, $x \in \mathbb{Z}^n_+$ where $A \in \mathbb{R}^{mxn}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. The feasible region is defined as $P = \{x \in \mathbb{Z}^n_+ | Ax \leq b\}$, which is the set of possible integer solutions that satisfy the constraints of the integer program. An IP is essentially a traditional linear programming problem with an integrality restriction on the decision variables.

This thesis presents a theoretically superior version of branch and bound (BB), the primary method used to solve integer programs. This new algorithm is called the Quaternary Hyperplane Branching Algorithm (QHBA) and introduces a novel approach to partitioning the solution space. QHBA enumerates more integer points per iteration, and consequently many IPs can be solved faster using QHBA when compared to traditional implementations of BB.

Integer programming problems have been used to mathematically model and manage a wide variety of systems across myriad applications. In 1989, the San Francisco Police Department implemented an IP based scheduling algorithm that significantly increased the efficiency of daily operations [29]. Prior to the implementation of the algorithm scheduling was done manually. The police department sought to achieve several objectives: 1) minimizing cost of operations, 2) maximizing citizen safety and 3) maximizing police officer morale. Implementation of this scheduling algorithm led to a 25 percent increase in the availability of officers during peak hours, a 20 percent decrease in response times, which resulted in a projected savings of approximately 11 million dollars per year.

Delta Airlines utilized another integer program based scheduling algorithm to determine airplane fleet assignments [26]. The fleet assignment problem is extremely complex due to the size of the system (at the time of the implementation there were approximately 2,500 departures to be divided among 450 airplanes). The solution to the IP model was forecasted to save the company over 300 million dollars for the following 3 years.

There have also been significant increases in IP applications for medicine. Recently, mixed integer programs (MIP) have been used to determine treatment plans for intensity-modulated radiation therapy [20]. Optimal treatment solutions were found quickly, and the solutions were superior in terms of tumor coverage and conformity while minimizing collateral radiation damage to normal tissues. Integer programs have also been used to match organ donors to potential recipients [24], and have significantly increased the efficiency of the transplant system employed in the United States. The proposed redistricting of transplant centers should result in a net increase of 17 transplants per year.

IP models have also seen extensive use in financial planning [11, 28]. One common problem consists of a set of potential investments (capital expenditures) that are constrained by a set of financial and resource limitations. Thus, the problem's goal is to choose the best subset of those potential investments [19, 25].

Integer programs have been shown to be largely beneficial to a host of problems; however, the primary limiting factor to the applicability of IPs lie in their potential computational complexity. Integer programs are $\mathcal{NP}$-complete in the strong sense [15]. As such, many IPs require an exponential amount to time to find and prove the optimal solution. The result is that often times integer programs are functionally unsolvable. One such problem is known as the Duty Scheduling Problem, which is a problem associated with public transportation [3]. Bordörfer et al. provides a 67,732 variable, 656 constraint mixed integer program from the Mixed Integer Programming Library (MIPLIB) [1] that is still unsolved even after years of researchers' attempts to solve it.

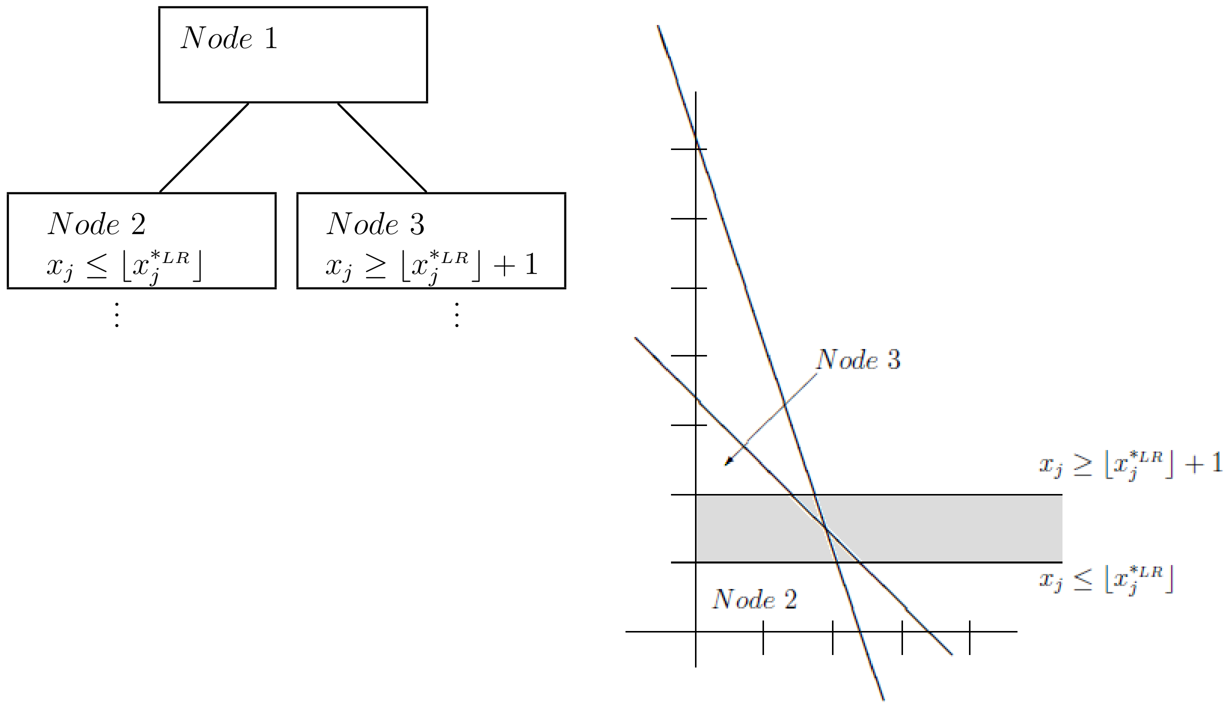
When an IP is unsolvable, practitioners are forced to compromise on various limiting factors such as constraints or variable restrictions on integrality. Relaxing the limitations for a particular problem to allow for solvability may not even result in a feasible solution.

Other concessions, such as partitioning the IP into smaller subproblems or relaxing constraints, often times leads to an inferior integer program. The solutions to these weaker integer programs are significantly less beneficial than a solution to the desired model.

The primary solution method used to solve integer programs is the Branch and Bound Algorithm (BB) [2, 7]. BB is an enumerative approach to solve an IP that utilizes a tree structure to tabulate potential solutions. In finite time BB guarantees an optimal solution to a bounded integer program, if such a solution exists, or reports that no integer solutions satisfy the constraints. Theoretically, BB requires exponentially many iterations in order to achieve this result.

A critical component of BB solves the linear relaxation (LR) of the IP. The LR is the continuous formulation of the IP where all aspects are identical to the IP except the integrality constraints on the decision variables are removed. The LR solution contains x values, x^{*LR}, and the objective value, z^{*LR}.

BB begins by solving the LR of the initial problem with x^{*LR} and z^{*LR}. If x^{*LR} is non-integer, let j be selected such that $x_j^{*LR} \notin \mathbb{Z}_+$. Two new subproblems or nodes of a BB tree are created (a generalized branching structure is displayed in Figure 1.1). One of these nodes adds on the constraint $x_j \leq \lfloor x_j^{*LR} \rfloor$, while the other node includes the constraint $x_j \geq \lfloor x_j^{*LR} \rfloor + 1$. Then one of the new nodes is selected, the LR for that subproblem is solved, and new branching constraints are generated if applicable. This process continues until all pendant nodes are fathomed. A node is fathomed if the LR



$Node\ 1$
$Node\ 2$
$x_j \leq \lfloor x_j^{*LR} \rfloor$
$Node\ 3$
$x_j \geq \lfloor x_j^{*LR} \rfloor + 1$
$Node\ 3$
$x_j \geq \lfloor x_j^{*LR} \rfloor + 1$
$x_j \leq \lfloor x_j^{*LR} \rfloor$
$Node\ 2$

Figure 1.1: Binary partition for branch and bound

is infeasible, integer, or if z^{*LR} is worse than the best known integer solution.

Observe that a single branch eliminates all of the LR space where $x_j \in (\lfloor x_j^{*LR} \rfloor, \lfloor x_j^{*LR} \rfloor + 1)$, as depicted in Figure 1.1. Due to the binary branching employed, BB may require exponentially many iterations prior to obtaining an optimal solution. This is one of the primary disadvantages to an enumerative approach to solving IPs. An exponentially large problem is not only time consuming, but often times the size of the tree can grow so large that it outstrips the memory capability of computers. For example, a 250 variable, 30 constraint IP ran until it exhausted 3 gigabytes of available storage without determining the optimal solution, as evidenced while pursuing this research.

Cutting planes are a common technique to help reduce the computational effort to solve an IP. Cutting planes are formulated so that non-integer extrema are rendered infeasible (or cut from the linear relaxation). Cutting planes are not allowed to eliminate any feasible integer solutions. Therefore if enough strong cutting planes are employed, then the solution to the linear relaxation is also the solution to the IP. Unfortunately there exists infinitely many cutting planes and there can exist exponentially many non-dominated cutting planes. While cutting planes represents an effective technique, BB is still primarily used to solve IPs with some cuts implemented prior to beginning BB.

1.1 Motivation

The branch and bound algorithm has changed very little since its inception in 1960. Improvements for the algorithm come in the form of additional branching strategies aiding in the selection of children that could expedite computation, or combinations of branching and cutting plane methods (branch and cut). Many improvements to the algorithm are problem specific, and while largely effective, such approaches cannot be successfully applied to all IPs.

The motivation for this research is derived from attempting to use cutting planes internally to solve an IP. Can cutting planes be used as branches in BB? Thus the ambitious motivation for this research is to redesign the traditional branch and bound algorithm by utilizing complex branching structures.

1.2 Research Contribution

This research's contribution is the development of a new version of BB called the Quaternary Hyperplane Branching Algorithm (QHBA). While the traditional implementation of BB focuses on a single non-integer variable at each branch, QHBA uses multivariate constraints (or cutting planes) at each branch. While potentially advantageous, expanding the scope of potential branching variables resulted in a myriad parameters that had to be accounted for in a successful implementation of this new branching scheme.

The coefficients corresponding to particular x values in a cutting plane are not restricted, thus the initial decision was to restrict potential coefficients for candidate cutting planes to $\{1, 0, -1\}$. These values were considered more desirable because they would cut the most continuous space per iteration. Additionally, these values increase the likelihood that basic feasible solutions for subproblems are integer.

One of the preliminary versions of QHBA operates very similarly to the traditional BB algorithm. Instead of using a single variable, multiple variables are selected for the branching cut. Constraints are derived by using the x values obtained from the LR solution, and instead of just using a single non-integer value, the right-hand side is calculated by inputing solution values into the complex branches. If this right-hand side value is non-integer, then the floor and ceiling values are calculated and used for the branches in those subproblem. Preliminary results indicated that it is possible to branch using cutting planes. In certain instances this algorithm terminates. However, if the initially calculated right-hand side is integral, then the algorithm fails to terminate because no rounding procedure can be implemented. Consequently the LR solution is not removed and the subsequent child node's LR is identical to its parent node's LR solution.

A potential solution to this problem is to increase the number of cutting planes being used per branch in QHBA. In doing so QHBA creates four subproblems, opposed to two for BB, licensing higher order branching (more than two branches per iteration). Ideally there is nothing prohibitive in the traditional BB algorithm that would prevent more than binary branching; however, in the traditional implementation higher order branching would be tantamount to doing multiple iterations in one step. For our implementation that was not necessarily the case. Using two branching cuts simultaneously requires a quaternary branching structure that is unique to QHBA. Most importantly, this quaternary structure enables the use of internal cutting planes (in this specific implementation Chvátal-Gomory cuts are used).

In summary the primary benefits of QHBA are as follows. QHBA provides a unique approach to branch and bound that employs quaternary branching. Theoretically this algorithm has quadratically more basic feasible integer points at each iteration than traditional branch and bound due to the use of Chvátal-Gomory cuts. Additionally, more continuous relaxation space is eliminated in one set of branches than in traditional BB. Finally, the computational benefits of QHBA have been shown to decrease the computational time by approximately 26.7% in solving bounded integer programs in comparison to traditional branch and bound on some benchmark problems.

1.3 Thesis Outline

Chapter 2 provides the background information required to understand this thesis. The traditional implementation of BB is presented. Several methods that have been regularly implemented to improve solution capability for IPs are briefly discussed. Topics including cutting planes and total unimodularity are also examined.

Chapter 3 contains the bulk of the work done for this research. The Quaternary Hyperplane Branching Algorithm is outlined in detail along with a 2-dimensional example to illustrate the advancements made. Theorems and proofs are provided to show finite convergence, and that the tree structure of QHBA is capable of iteratively partitioning the solution space.

Chapter 4 contains the computational results from the implementation of QHBA. QHBA is compared to a commercially available IP software, CPLEX, and another partitioning technique. The chapter presents the solution times for all methods, along with an interpretation of the data, which illustrates the benefits of QHBA.

Chapter 5 provides a conclusion and summarizes the work. During the implementation of this computational study, numerous strategies were developed that could be used in order to further improve solution times, and some of those future research topics are briefly discussed.

Chapter 2

Background Information

This chapter contains the information needed to understand the fundamental concepts of this research. Only a small sample of the massive amount of research done in the scope of integer programming is provided. Further information can be found in [22].

This chapter begins with an in depth look into branch and bound, and the fundamental theory behind this algorithm. More specifically the first section examines the nature of the partitions established by BB and the logic necessary for the algorithm to terminate. A detailed example is provided to further explore BB. Specific branching strategies and algorithmic improvements along with their implications are also discussed. The chapter concludes with a brief discussion of cutting planes and total unimodularity.

2.1 IP Definitions

As aforementioned, an integer program is a discrete class of optimization problems that consist of an objective function limited by a set of constraints on potential decision variables. An IP is defined as max $c^T x$ subject to $Ax \leq b$, $x \in \mathbb{Z}_+^n$ where $A \in \mathbb{R}^{mxn}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. The feasible region of an IP is defined as $P = \{x \in \mathbb{Z}_+^n | Ax \leq b\}$, which is the set of possible integer solutions that satisfy the constraints of the integer program.

BB uses the linear relaxation of the IP as a primary component. The linear relaxation (LR) is defined as max $c^T x$ subject to $Ax \leq b$, $x \in \mathbb{R}_+^n$ where $A \in \mathbb{R}^{mxn}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$ with the feasible region of the linear relaxation defined as $P^{LP} = \{x \in \mathbb{R}_+^n | Ax \leq b\}$.

Integer programs are $\mathcal{NP}$-complete [15], which has dire consequences in the effort needed to compute the optimal solution in even small integer programs. However linear programs (LP) can be solved in polynomial time [16] and most LPs are easily solved on commercially available software.

2.2 Branch and Bound

The Branch and Bound Algorithm (BB) [2, 7] is the most popular method for solving arbitrary IPs . BB guarantees an optimal solution, if it exists, and can find the solution in finite time. Therefore it has been widely used to solve IPs since its inception. BB systematically enumerates solutions using a tree structure to establish a hierarchical set of bounds and constraints for candidate solutions.

BB begins by solving the LR with x^{*LR} and z^{*LR}, this is described as the root node of the tree. Denote the node being evaluated as the parent node $T_{p'}$. If x^{*LR} is non-integer, let j be selected such that $x_j^{*LR} \notin \mathbb{Z}_+$. Two new subproblems or nodes of a BB tree are created, these are referred to as child nodes. Note that the parent-child relationship is indicative of immediate adjacency, with parent denoting the node above the current node, and child being the nodes below the current node. One of these child nodes adds the constraint $x_j \leq \lfloor x_j^{*LR} \rfloor$ to its LR formulation. This node is denoted $T_{p'}^L$. While the other node includes the constraint $x_j \geq \lfloor x_j^{*LR} \rfloor + 1$, and this node is denoted $T_{p'}^G$.

BB continues generating children nodes until all pendant nodes are fathomed. A node is fathomed if the solution to its LR is integer, if the LR is infeasible, or if z^{*LR} is worse than the best known integer solution. BB sets a bound by recording the current best integer solution z^*. If an integer solution is found and $z^{*LR} > z^*$, then this new integer bound is set as z^*. If there are no more unfathomed nodes then BB terminates and reports the optimal solution, if it exists, or reports that the problem is infeasible. Formally,

The Branch and Bound Algorithm

Initialization

Let $T = \{T_1\}$ be the starting enumeration tree and T_1 has the IP's linear relaxation.

```
        Set $z^* := -\infty$ where $z^*$ is the current best integer solution.

        Set $totalnodes := 1$.

**Main Step**

    While there exists an unfathomed pendant node.

        Let $T_p$ be any unfathomed pendant node of $T$.

        Solve the linear relaxation for $T_p$ and denote it as $z^{*T_p}$ and $x^{*T_p}$.

            If $T_p$ is infeasible, then mark $T_p$ as fathomed.

            If $x^{*T_p} \in \mathbb{Z}^n$, then mark $T_p$ as fathomed and if $z^{*T_p} > z^*$, then $z^* := z^{*T_p}$
                and $x^* := x^{*\ T_p}$.

            If $z^{T_p} \leq z^*$, then mark $T_p$ as fathomed.

        If $T_p$ is not fathomed, then Begin

                Select an $x_i^{T_p}$ such that $x_i^{*T_p} \notin \mathbb{Z}$.

                Set $\beta^{T_p} := \lfloor x_i^{*T_p} \rfloor$.

                Create the following two new nodes with $T_p$ as the parent by adding
                    the following constraints to $T_p$'s linear relaxation.

                $T_p^L = T_{totalnodes+1}$ with the following constraints appended to $T_p$'s LR.

                    $x_i \leq \beta^{T_p}$.

                $T_p^G = T_{totalnodes+2}$ with the following constraints appended to $T_p$'s LR.

                    $x_i \geq \beta^{T_p} + 1$.
            $totalnodes := totalnodes + 2$
        End
    End while
Output
    If $z^* = -\infty$, then report the problem as infeasible, else report $z^*$ and $x^*$ as the
    optimal solution.
```

The branching step in BB is indeterminate. At any iteration there are many nodes that need to be evaluated. Therefore, the next unfathomed node to be evaluated is selected arbitrarily. Selecting the next node to be evaluated can have significant implications to the efficacy of BB. Memory capacity becomes a pertinent issue in larger problems, and often times BB trees can grow large enough to exceed the memory capacity of computers. However it is possible that the overall solution time would increase when using such a strategy. Common search strategies used to explore the BB tree are the topic of the next section.

2.2.1 Search Strategies

Branching strategies were originally implemented to alleviate memory concerns in early computers, and while less of an issue with modern computing, large branching structures still occur that often times outstrip the available resources. Consequently a wide array of branching strategies have been developed in an attempt to avoid such complexities. Generally, strategies are either some form of breadth first, depth first, or best bound (also known as best child) selection.

Depth first search is an evaluation method where BB explores a child node and keeps expanding until all ancestors are fathomed. Once a fathomed node is found, then the BB algorithm backtracks and repeats the process. Depth first strategies are typically assigned a direction, right or left, denoting which direction to dive.

Depth first search is memory efficient as the evaluation tree only has to keep track of the current node being evaluated and other children that have been created as the algorithm iterates through the tree. Denote a depth d, where d demarcates the level of nodes, with the root node of depth $d = 1$. Thus, the children of the root node have $d = 2$, and the children of all nodes at $d = 2$ are marked with a depth of $d = 3$, and so on.

Using a depth first search strategy, at a depth d there are at most $d + 1$ unevaluated nodes left in the tree. Since the number of nodes in the tree is linear in order, the amount of memory necessary to store the tree is small. Additionally, depth first search strategies tend to find integer solutions quickly. Thus more nodes can be pruned from the tree.

However, there is no guarantee to the effectiveness of depth first search. It is possible to explore to a large depth and then find an integer solution that does not prune any nodes. This effect is multiplied in larger problems. There is no overt indication that a particular enumeration path or child node is the best candidate for branching, thus a depth first strategy has the potential to engage in an exponential amount of superfluous work, which has dire consequences on solution times. Thus, computation times can vary drastically from problem to problem even within the same size and class. Now that memory is not as rigid of a limiting factor, other branching strategies have been devised that typically perform better.

Breadth first strategies attempt to evenly expand the tree by exploring all children at the current depth. For example, all nodes at depth 3 are evaluated before any depth 4 nodes. Breadth first criteria also utilize a direction, such as breadth first right. Evaluating one level at a time alleviates the concerns of a depth first strategy. No 'wrong decision' can be made in regards to what direction to dive. This potentially reduces the amount of computation needed to find the optimal integer solution by not exploring all the ancestors of a useless node.

A breadth first strategy is potentially problematic because it does not seek a goal, namely to fathom a node (potentially resulting in a bound). Instead by evenly expanding the tree, the current number of unfathomed nodes (nodes that must then be stored in memory) can grow exponentially large, resulting in 2^d nodes at depth d. This can quickly result in a tree large enough to exceed the available memory on many computers.

The default approach for most implementations of BB is the best bound search, which is a mixture of breadth and depth strategies. Best bound search is a strategy that uses information provided by the objective function of evaluated nodes to determine the order of enumeration. The assumption is that a better objective value is the best candidate for enumeration. The logic follows that the optimal integer solution would be a child of a node with a good objective value. Thus the nodes are evaluated in the descending order of z^{*LR}. A near optimal (or optimal) integer solution is a better bound because it prunes a larger area of the enumeration tree, resulting in a reduced solution time. Finding a good integer solution quickly can lessen the size of the current tree, prevent excessive enumeration, and reduce the need for excessive amounts of memory to store the tree.

A more advanced search practice is to use a best bound search mixed with a depth first search condition known as random diving [27]. In this hybrid approach BB enacts the best bound search for a set number of iterations. At prescribed intervals the algorithm switches to a depth first search and dives until that path is fathomed. This diving attempts to quickly discover a superior IP solution.

There is no overt indication to the best search method and the best strategy varies from problem to problem. While one approach may be effective for a certain class of problems, there is no guarantee that it will work efficiently for all problems.

A Branch and Bound Example

The basic BB algorithm has been described along with tools fundamental to its implementation. This section uses tradition BB to solve a simple IP. Branching, fathoming, and search strategies are discussed in the context of this example.

Example 2.2.1 Consider the following problem

$$\begin{aligned} &\text{Maximize} && 5x_1 + 2x_2 \\ &\text{subject to} && 10x_1 + 2x_2 \leq 23 \\ & && 4x_1 + 2x_2 \leq 13 \\ & && x_1, x_2 \geq 0 \text{ and } x_1, x_2 \in \mathbb{Z}^n \end{aligned}$$

Figure 2.1 provides a graphical representation of this integer program with the constraints $10x_1 + 2x_2 \leq 23$ and $4x_1 + 2x_2 \leq 13$ as labeled. The points labeled on the interior of the constraints represent feasible solutions, P, to the integer program. Note that the solution to the linear relaxation is non-integer, and there are no integer extrema except $(0, 0)$ in P^{LR}.

The initialization of BB requires evaluating the root node, which is the solution to the linear relaxation. The LR solution is $z^* = \frac{44}{3}$, and $x^{*LR} = (\frac{5}{3}, \frac{19}{6})$. For this example a depth first left search strategy is used. Recall that in depth first left the preference is to select the left most unfathomed node for evaluation. Node selection continues downwards and left until a node is fathomed. Once a node is fathomed, then the algorithm selects a higher node with an unexplored branch and continues evaluating nodes depth first left.

In the root node both decision variables are fractional in the LR solution. Selection of the initial branching variable is arbitrary, therefore in this example x_1 is selected. Constraints are formulated using the rounding procedure that was previously explained. Since $x_1 = \frac{5}{3}$, the branches are $x_1 \leq \lfloor \frac{5}{3} \rfloor$ and $x_1 \geq \lfloor \frac{5}{3} \rfloor + 1$, equivalently $x_1 \leq 1$ and

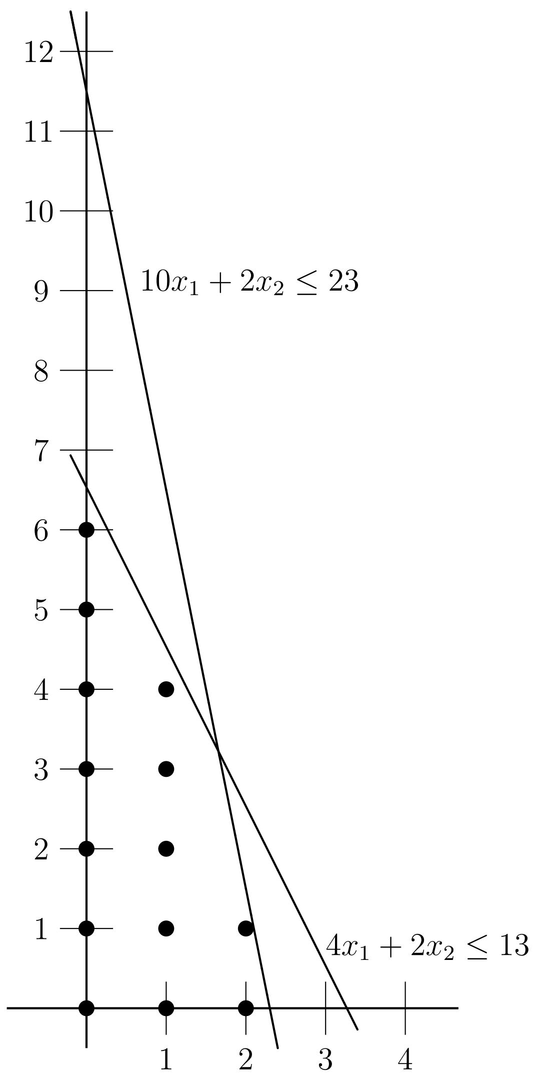


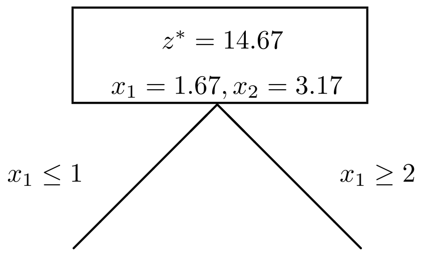
Figure 2.1: Graph for the IP

$x_1 \geq 2$.

Figure 2.2 depicts the branching of the root node. For each node there are two branches, and each branch is representative of one constraint. These constraints are referred to as branching constraints. Finding the solution to a node and creating the branching constraints is one iteration of BB. The graphical depiction in Figure 2.2 shows that the branching constraints divide the solution space into two subproblems. The shaded area indicates the continuous space eliminated by these two branching constraints. Observe that the previous LR solution is now infeasible in the two subproblems, due to the branching constraints.

For the next iteration the branch $x_1 \leq 1$ is followed due to depth first left. This subproblem can be denoted $T_1^L = T_2$ with the following constraints: $10x_1 + 2x_2 \leq 23$, $4x_1 + 2x_2 \leq 13$ and the branching constraint $x_1 \leq 1$ is included in this subproblem. The linear relaxation of node 2 is solved rendering a solution of $z^{*LR} = 14$, $x^{*LR} = (1, \frac{9}{2})$. Continuing, x_2 is the only candidate for branching because it is noninteger. Two new nodes and accompanying constraints are generated. The constraints are $x_2 \leq \lfloor \frac{9}{2} \rfloor$ and $x_2 \geq \lfloor \frac{9}{2} \rfloor + 1$, $x_2 \leq 4$ and $x_2 \geq 5$. Thus two new nodes are generated. Figure 2.3 depicts the fully explored BB enumeration tree for this problem and the remainder of the section refers back to this figure.

The next node selected is node 3 which includes the branching constraint $x_2 \leq 4$. The LR solution to node 3 is $z^{*LR} = 13$, $x^{*LR} = (1, 4)$ which is integer. Node 3 is considered fathomed. Since there is no current best integer solution (z^* is initialized to



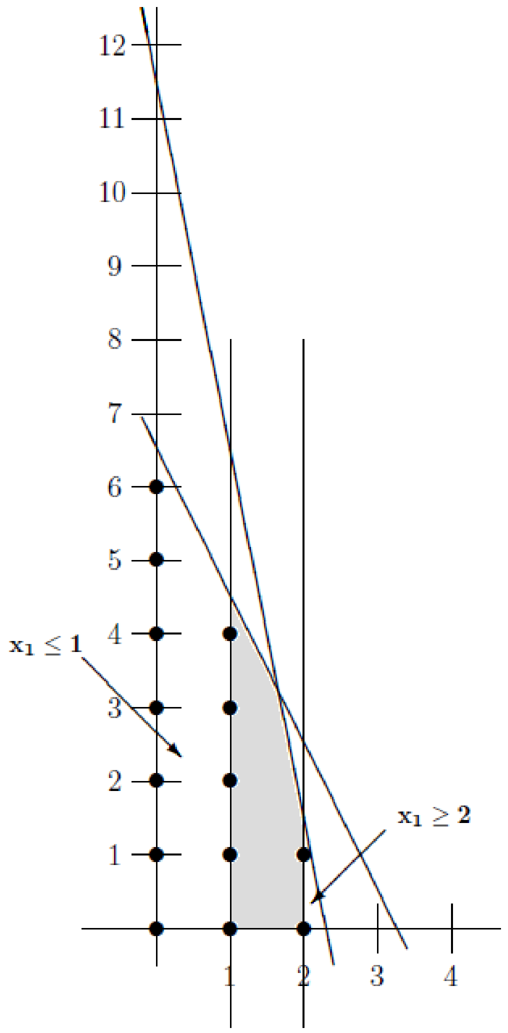


Figure 2.2: First branching partition for Example 1

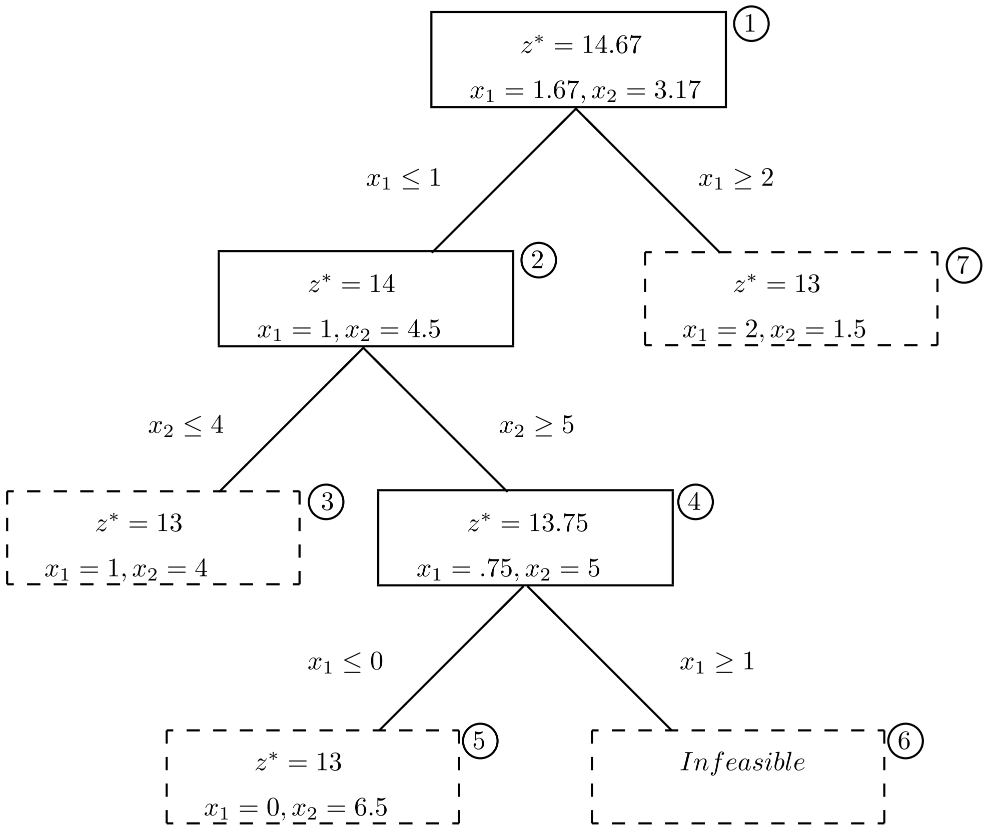


Figure 2.3: A Branch and Bound enumeration tree

$-\infty$), z^* is updated to 13.

When a node is fathomed, it denotes that the current node is not a candidate for future branching. Once fathomed, the algorithm does not continue branching along that node. Therefore, the algorithm finds another unfathomed node and continues enumeration according to the prescribed branching strategy. In Figure 2.3, the dashed line borders denote a fathomed node.

BB returns to the next unfathomed node (node 4) and continues down that path favoring depth first moving left. In node 4 the LR is found using the added constraint $x_2 \geq 5$. Next, the same constraint generation procedure is applied and the algorithm continues down the tree, as shown in Figure 2.3. In the example this path follows the labeled numbers in ascending order. Note that the constraints on the decision variables are cumulative, thus at node 6 the constraints $x_1 \geq 1$, $x_2 \geq 5$, and $x_1 \leq 1$. Furthermore, these constraints are added to linear relaxation problems of subsequent child nodes.

The best integer solution found thus far is $z^* = 13$. Consequently node 5 is fathomed because its linear relaxation objective value is less than or equal to an already existing integer solution (even though the solution is non-integer), and node 6 is fathomed because the added constraints render the LR infeasible. Finally node 7 is fathomed due to the best integer solution bound.

Note that it is possible to fathom node 4 because the coefficients of the objective function are integer, and thus integer solutions would result in integer objective values. The LR solution to node 4 is a noninteger value (13.75) greater than the current best integer solution 13, but worse than the next best integer objective value 14. Node 4 can be fathomed because there would be no integer solution that would result in a better objective value. This would reduce the total nodes necessary to solve the problem, however for this example the tree was fully explored.

Termination occurs now that there are no more unfathomed pendant nodes. Once BB terminates, the optimal integer solution is reported, which is $z^{*T} = 13$, $x^{*T} = (1, 4)$.

In this example selecting a depth first left strategy is very beneficial. The optimal integer solution is found in two iterations, and it bounds the right side of the tree rapidly. However, with other search strategies the total number of nodes evaluated can vary widely. If a depth first right strategy is adopted for Example 1, the total number of nodes evaluated would be 13, which is more than depth first left. A depth first right tree is shown in Figure 2.4. Using a best bound search method would result in an tree identical to Figure 2.3, except enumerated in a different order.

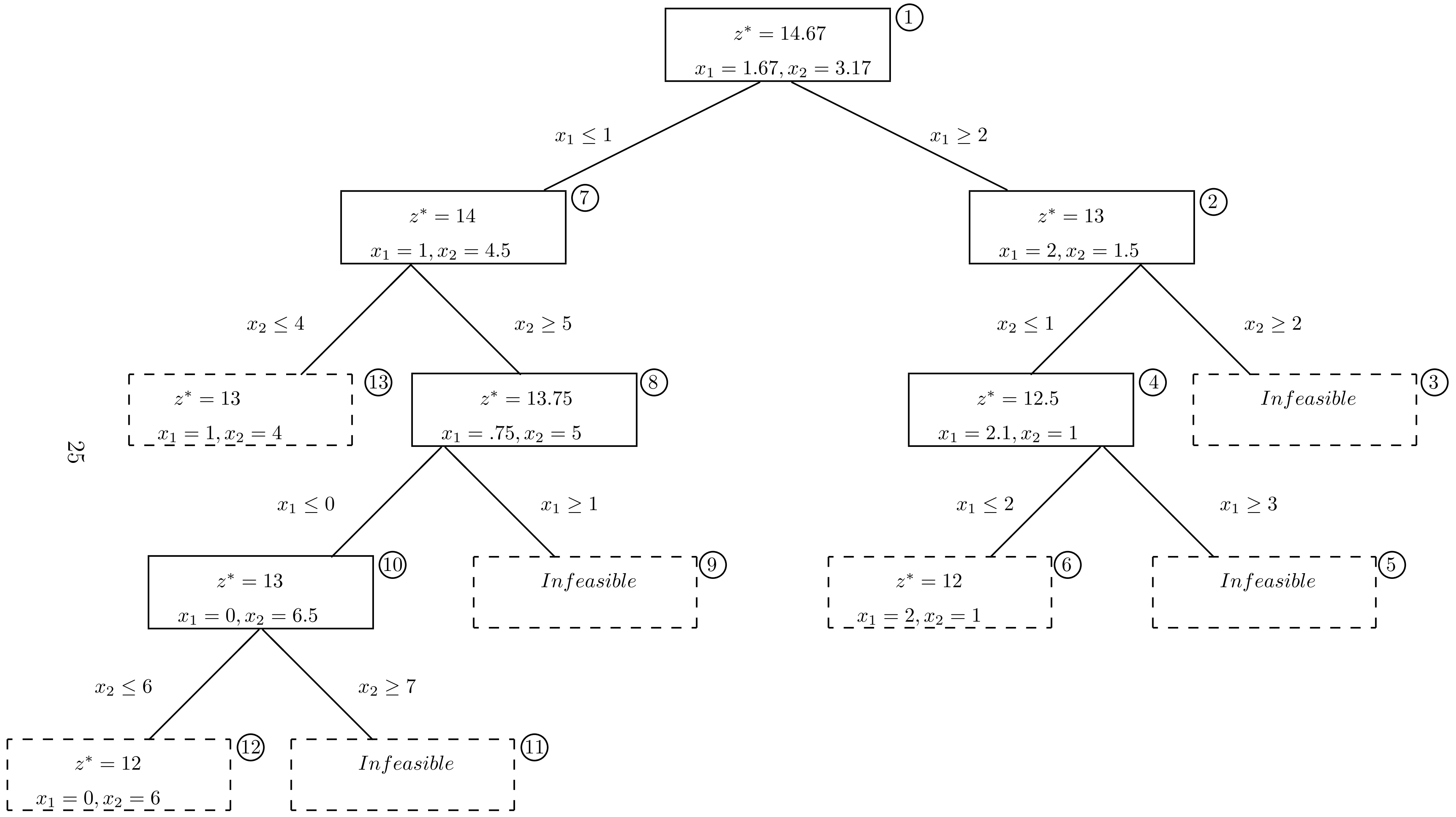


Figure 2.4: A Branch and Bound enumeration tree using depth first right

2.2.2 Branch and Bound Requirements

There exist various versions or tweaks to branch and bound. Tweaking an algorithm no longer guarantees that the algorithm optimally solves the problem. Thus, any tweak to BB must have an argument that proves that the tweak still enables BB to solve every IP. There are two main points that must be shown to validate such a tweak to BB.

First, the modification must terminate in a finite number of iterations. If not, then the algorithm may be stuck in an infinite loop and never terminate. Such an example is provided at the beginning of Chapter 3.

Second, the optimal solution must not be eliminated. This is typically shown through an argument that proves that every integer solution in the parent node is also in at least one of its children's nodes or that a better solution does not exist in a certain region. Bounding in BB is an example of this second case.

2.2.3 Hyperplane Branching

Hyperplane branching is a modification on the branching constraints used in BB. For BB, branching constraints are single variables. Hyperplane branches (or disjunctive constraints) are another implementation spanning multiple variables in an attempt to increase the efficiency of the enumeration tree. Branching constraints would follow a similar pattern to traditional BB. One child includes the constraints $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ and the other child has $\sum_{i=1}^{n} \alpha_i x_i \geq \beta + 1$ where $\alpha, \beta \in \mathbb{Z}$.

Primary research for hyperplane branching has been done by Owen and Mehrotra. They offer a technique, called disjunctions, that utilizes $\{-1, 0, 1\}$ inequalities or hyperplanes and implements them in a binary branching tree. The use of general disjunctions was derived from [21], which provides a polynomial time algorithm for solving general IPs. This paper provides much of the background ideas for the work presented here. Given an x^{*LR}, these authors create a branching constraint $(\alpha, \beta) \in \mathbb{Z}^{n+1}$. Two child nodes are created with one having the constraint $\sum_{i=1}^{n} \alpha_i x_i \leq \lfloor \sum_{i=1}^{n} \alpha_i x_i^{*LR} \rfloor$ and the other with $\sum_{i=1}^{n} \alpha_i x_i \geq \lfloor \sum_{i=1}^{n} \alpha_i x_i^{*LR} \rfloor + 1$. Theoretically, their proof of convergence is slower than BB's and some other problems with this general approach are discussed at the beginning of chapter 3.

Owen and Mehrotra provide an algorithm that is primarily concerned with the generation of 'good' disjunctions, and at any parent node use a greedy heuristic algorithm to find the best general disjunction to use that for branching. A small amount of research has been done for finding effective disjunctions [14, 17]. The primary result is traditionally a reduction in tree size, but there is not necessarily a reduction in solution time in computational studies. In contrast, this work actually decreases the solution time to many IPs.

Other researchers have used hyperplanes to help solve integer programs, but their results are much less related to the research presented here. Ryan and Foster allude to the use of multi-variate branching for use in a set partitioning problem [9]. Specifically, in duty scheduling set partitioning problems, the use of single variable branches results in uneven tree growth. Duty variables are binary, thus at any variable branch values are restricted to 0 or 1. When a variable is fixed to 0, it often times has little effect on the objective value which produces fewer branches. Restricting a variable to 1 has the opposite effect. Implementation of branching constraints would result in more even tree growth, which was computationally less demanding.

A hybrid algorithm utilizing hyperplane branching was proposed for the generalized assignment problem by Jörnsten and Värbrand [13]. They present the Complement Algorithm, which is a modified branch and cut application. The Compliment Algorithm begins by generating many valid inequalities to cut the linear relaxation solution. If no valid inequalities can be generated, then the algorithm implements a form of hyperplane branching until valid inequalities can be generated again. In the computational study, the maximum number of branches was limited to 10. Implementation of the Complement Algorithm led to an increase in the efficiency of solving test problems.

Fischetti and Lodi employ a type of hyperplane branching, that they term local branching, to assist in the solution of mixed integer programs [8]. The branching constraints were based on the binary variables included in the problem. Instead of 'hard fixing' a single binary variable to either 0 or 1, Fischetti and Lodi's branching constraints focused on the fixation of sets of these variables. In doing so they could partition the solution space into solution neighborhoods. A neighborhood that yielded a superior solution would be further searched using commercially available software. Using the branching constraints with the newly devised search heuristics yielded significant improvements in solving hard IPs.

2.3 Cutting Planes

The feasible region of a linear program is known as a polyhedron, which is the intersection of a finite number of halfspaces. A halfspace is the solution space of a single linear inequality $\{x \in \mathbb{R}^n : Ax \leq b\}$. The feasible region of a linear program is a convex polyhedron. A set S is convex if, and only if, $\lambda s_1 + (1 - \lambda)s_2 \in S$ for all $s_1, s_2 \in S$ and $\lambda \in [0, 1]$. The solution space to an integer program is a countable set of points, and consequently not convex. The polyhedron containing those points is known as the convex hull of P, denoted P^{ch}. Formally the convex hull of P is the intersection of all convex sets containing P.

A cutting plane or valid inequality typically removes non-integer space from the LR. An inequality of the form $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is a valid inequality if, and only if, $\sum_{i=1}^{n} \alpha_i x'_i \leq \beta$ is not violated for all $x' \in P$. In using cutting planes to obtain an integer solution, each application of a cutting plane seeks to invalidate the current linear relaxation solution. This is known as the separation problem.

Cutting planes aim to remove the non-integer space so that only the convex hull of P remains. Thus an integer solution can be obtained by using the simplex method. Various topics such as facet defining inequalities, lifting, disjunctive constraints, and other valid inequality generation techniques are described in [22].

Returning to Figure 2.1, the graphical depiction of Example 1 shows the impertinent continuous space related to the linear formulation that is not necessary to solve the integer program. Solving the linear relaxation results in a non-integer solution, $z^* = \frac{44}{3}$, $x_1 = \frac{5}{3}$, and $x_2 = \frac{19}{6}$. Thus consider Example 1 and include the cut $3x_1 + x_2 \leq 7$. The resulting LR solution is $z^* = \frac{27}{2}$, $x_1 = \frac{1}{2}$, and $x_2 = \frac{11}{2}$. Including the cut $2x_1 + x_2 \leq 6$, the solution to the LR becomes $z^* = 13$, $x_1 = 1$, and $x_2 = 4$, which is the optimal integer solution. Notice that neither of the inequalities eliminated a feasible integer point. This is shown graphically in Figure 2.5.

In Figure 2.5 the two new constraints cut the section of continuous space that contained the relaxation's solution from the problem. Subsequently, the first constraint removes the original LR solution, but still allows for a noninteger optimum. The addition of the second constraint cuts the new noninteger optimum from the polyhedron and actually constrains the space so that there are more integer extrema that can be accessed from the simplex method. The resulting integer optimum is achieved without the branch and bound algorithm.

While cutting planes are effective, the derivation of effective cutting planes is the primary obstacle in their implementation. There exists infinitely many cutting planes and there can exist exponentially many non-dominated cutting planes for even simple IPs. The difficulty of finding a cutting plane varies, and finding certain classes of cutting planes is $\mathcal{NP}$-hard [15].

A class of cutting planes known as Chvátal-Gomory (C-G) [5, 10] cuts are of par-

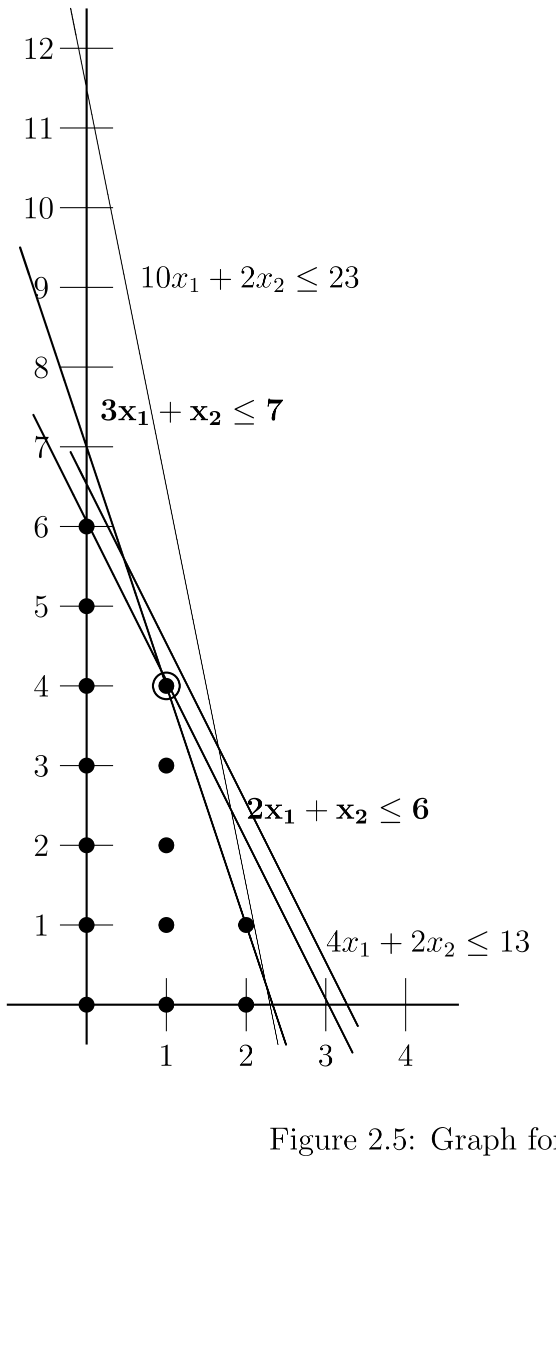


Figure 2.5: Graph for the IP with cutting planes

ticular importance to this research. C-G cuts are a useful tool that potentially allows the derivation of all valid inequalities. The general procedure is that some valid inequality is a non-negative linear combination of the constraints whose coefficients have been rounded down to the next lowest integer. The C-G procedure can be applied recursively.

The basic argument supporting the validity of a C-G cut is that a non-integer right hand side can be modified without impacting the coefficients of a constraint, given those coefficients are integer. More specifically, suppose a valid inequality of the form $\sum_{j=1}^{n} a_j x_j \leq b$ where $a_j \in \mathbb{Z}\ \forall\ j = 1, ..., n$ and $b \in \mathbb{R}$. Since a is integer, the inequality $\sum_{j=1}^{n} a_j x_j \leq \lfloor b \rfloor$ is also valid. This can be furthered applied to instances when the a coefficients are also non-integer. Suppose a valid inequality $\sum_{j=1}^{n} a_j x_j \leq b$ where $a_j \in \mathbb{R}\ \forall\ j = 1, ..., n$ and $b \in \mathbb{R}$. Then rounding down the a coefficients decreases the value of the left hand side and thus $\sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq b$ is valid. Therefore, $\sum \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor$ is also a valid inequality.

Traditionally derived C-G inequalities are positive fractional linear combinations of valid inequalities. Returning to Example 1 both cutting planes can be derived from the constraints. The inequality $2x_1 + x_2 \leq 6$ is derived by $\frac{1}{2}(4x_1 + 2x_2) \leq \frac{1}{2}(13)$, this yields the constraint $2x_1 + x_2 \leq \frac{13}{2}$. Using the C-G rounding procedure this constraint can be modified to $\lfloor 2 \rfloor x_1 + \lfloor 1 \rfloor x_2 \leq \lfloor \frac{13}{2} \rfloor$, which is equivalently $2x_1 + x_2 \leq 6$. The constraint $3x_1 + x_2 \leq 7$ can be derived in a similar fashion. The first step is $\frac{1}{2}(10x_1 + 2x_2) \leq \frac{1}{2}(23)$ which yields $\lfloor 5 \rfloor x_1 + \lfloor 1 \rfloor x_2 \leq \lfloor \frac{23}{2} \rfloor$, which is an intermediary constraint $5x_1 + x_2 \leq 11$. Utilizing the two recently derived constraints, the final cutting plane can be derived by adding $\frac{1}{3}(5x_1 + x_2) \leq \frac{1}{3}(11)$ and $\frac{2}{3}(2x_1 + x_2) \leq \frac{2}{3}(6)$, which is $3x_1 + 1_2 \leq \frac{23}{3}$. Using the C-G rounding procedure, all noninteger values can be rounded down which is expressed by $\lfloor 3 \rfloor x_1 + \lfloor 1 \rfloor x_2 \leq \lfloor \frac{23}{3} \rfloor$, resulting in the constraint $3x_1 + x_2 \leq 7$. Thus, the C-G process has created the two cutting planes used in this example.

2.4 Totally Unimodular Matrices

The final topic presented in this chapter is the concept of totally unimodular matrices (TUM). A matrix A is TUM if every square submatrix of A has a determinant of $\{1, -1, 0\}$. TUM matrices are of particular importance because if a particular matrix is TUM, then all basic feasible solutions from that matrix are integer as long as the right hand side is integer. More specifically, if A is TUM, then $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is integral for all $b \in \mathbb{Z}^m$.

In order to prove that some matrix A is not TUM, it suffices to provide a square submatrix of A where the determinant does not equal $\{1, -1, 0\}$. However, proving a matrix is TUM is significantly more difficult. Therefore there are several properties that can be used to aid in the identification of TUM matrices.

If A is TUM, then a matrix obtained by interchanging two rows or columns of A is also TUM. Additionally, if A is TUM, then (A, I) is also TUM. Alternatively, if for every $J \subset N = \{1, ..., n\}$, there exists a partition J_1, J_2, of J such that $|\sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij}| \leq 1$ for $i = 1, ..., m$ then A is TUM.

For the remainder of this thesis, presented TUM matrices are relatively easy to prove by demonstrating that all square sub-matrices have determinants that satisfy the primary definition.

Chapter 3

The Quaternary Hyperplane Branching Algorithm (QHBA)

This chapter provides an explanation of QHBA and demonstrates its implementation with a simple example. The theorems necessary to support QHBA are also presented along with theoretical evidence that shows why QHBA is an improvement to Branch and Bound.

3.1 QHBA

Briefly, QHBA is an enumeration algorithm similar to BB with several key modifications that significantly increase QHBA's efficiency when compared to traditional BB. Progression through QHBA follows a tree structure. The key differences lie in the generation of the branching constraints and in the design of the enumeration tree. The branching constraints for QHBA are multivariate, whereas BB's branching constraints are single variables. Additionally, QHBA follows a quaternary branching scheme (4 branches for each parent node). These modifications allow QHBA to better eliminate continuous relaxation space, which results in significant theoretical and computational benefits.

The primary idea behind QHBA consists of choosing branching constraints. Define (α, β) to be a branching constraint where $\alpha \in \mathbb{Z}^n$ and $\beta \in \mathbb{Z}$. In traditional branch and bound, $\alpha = e_i$. In this work, α could take any integer values, but is restricted to $\{-1, 0, 1\}^n$, which enables strong theoretical results.

Converting BB into this generalized branching constraints setting is fairly straightforward. Given the linear relaxation solution of an unfathomed node $T_{p'}$ and branching constraint (α, β), two child nodes are created. One adds on the constraint $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ and the other has the constraint $\sum_{i=1}^{n} \alpha_i x_i \geq \beta + 1$. The natural selection of β is $\lfloor \sum_{i=1}^{n} \alpha_i x^{*T_{p'}} \rfloor$.

The single biggest complication with this natural implementation is that the linear relaxation solution of a parent node may still be feasible in one of the child nodes. For instance, if the LR solution for some node T_p is $x^* = (1.5, 3.5)$ with the branching constraint $((1,1), \beta)$, then $\beta = 1 * (1.5) + 1 * (3.5) = 5$ and x^* is contained in $T_{p'}^{L}$ which includes the constraint $x_1 + x_2 \leq 5$. Changing the branching constraint to $((1,-1), \beta)$ incurs a similar problem as $\beta = 1 * (1.5) + (-1) * (3.5) = -2$ and the constraint is $x_1 - x_2 \leq -2$. Since both β values are integral, then the additional branching constraints do not eliminate x^* from future children. Consequently, the BB algorithm with these two branching constraints enters an infinite loop and will not terminate.

Two advancements are developed to overcome this serious problem and enable QBHA to be a valid solution technique to solve IPs. The first advancement involves the case when $\sum_i^n \alpha_i x^{*T_{p'}} \in \mathbb{Z}$. In such a case α is changed so that $\sum_i^n \alpha_i x^{*T_{p'}} \notin \mathbb{Z}$. Clearly the branching constraints now remove the LR solution from future children.

The second advancement involves the concept that the corner points of branching constraints are not integer, as previously shown. The new proof of finite convergence fundamentally relies on the corner points of the branching structure being integer. To guarantee integer corner points, a quaternary branching strategy is implemented. Thus, each parent node has two branching constraints (α^1, β^1) and (α^2, β^2). Since two branching constraints are being used the solution space is divided into 4 partitions, or nodes constructed with pairs of branching constraints. These 4 partitions, or children, are created as follows. Including the specific branching constraints, $\sum_{i=1}^n \alpha_i^{1T} x_i^{*T_{p'}} \leq \lfloor \beta^{1T} \rfloor$, $\sum_{i=1}^n \alpha_i^{2T} x_i^{*T_{p'}} \leq \lfloor \beta^{2T} \rfloor$, denotes $T_{p'}^{LL}$. The next partition utilizes $\sum_{i=1}^n \alpha_i^{1T} x_i^{*T_{p'}} \leq \lfloor \beta^{1T} \rfloor$, $\sum_{i=1}^n \alpha_i^{2T} x_i^{*T_{p'}} \geq \lfloor \beta^{2T} \rfloor + 1$, which is denoted by $T_{p'}^{LG}$. Another space has $\sum_{i=1}^n \alpha_i^{1T} x_i^{*T_{p'}} \geq \lfloor \beta^{1T} \rfloor + 1$, $\sum_{i=1}^n \alpha_i^{2T} x_i^{*T_{p'}} \leq \lfloor \beta^{2T} \rfloor$, which is denoted as $T_{p'}^{GL}$. The final quadrant is $\sum_{i=1}^n \alpha_i^{1T} x_i^{*T_{p'}} \geq \lfloor \beta^{1T} \rfloor + 1$, $\sum_{i=1}^n \alpha_i^{2T} x_i^{*T_{p'}} \geq \lfloor \beta^{2T} \rfloor + 1$ and is denoted as $T_{p'}^{GG}$.

Just enumerating into 4 quadrants does not eliminate any fractional corner points; however, it does enable the introduction of cutting planes based solely upon the branching constraints, which is a major advancement of this research. To each of these four nodes, the obvious Chvátal-Gomory cut (the multipliers are $(\frac{1}{2},\frac{1}{2})$) is added to each of the four subproblems. These C-G constraints are only non-redundant when the sum of the right hand side of the specific branching constraints for a subproblem are odd. This occurs in precisely 2 of the subproblems, so a total of 10 inequalities are added across the 4 nodes. In contrast, in a similar setting, BB would only have 8 such inequalities, which enables a single iteration of QHBA to eliminate more continuous relaxation space per branch.

Once the branching structure is understood, QHBA iterates through the tree using these new branching constraints when creating children. Fathoming conditions are identical to traditional BB. Thus QHBA fathoms if it finds an integer solution, if the subproblem is infeasible, or if the objective value of a subproblem is inferior to the current best integer solution. Formally, the algorithm can be described as follows.

Quaternary Hyperplane Branching Algorithm

Initialization

Let $T = \{T_1\}$ be the starting enumeration tree and T_1 contain the IP's linear relaxation.

Set $z^* := -\infty$ where z^* is the current best integer solution.

Set $totalnodes := 1$.

Main Step

```
While there exists an unfathomed pendant node.
    Let $T_p$ be any unfathomed pendant node of $T$.
    Solve the linear relaxation for $T_p$ and denote it as $z^{*T_p}$ and $x^{*T_p}$.
        If $T_p$ is infeasible, then mark $T_p$ as fathomed.
        If $x^{*T_p} \in \mathbb{Z}^n$, then mark $T_p$ as fathomed and if $z^{*T_p} > z^*$, then $z^* := z^{*T_p}$
            and $x^* := x^{*\ T_p}$.
        If $z^{*T_p} \leq z^*$, then mark $T_p$ as fathomed.
        If $T_p$ is not fathomed, begin
            Select $\alpha^{1T_p}$ and $\alpha^{2T_p} \in \{-1, 0, 1\}^n$
            For $l = 1$ and 2, begin
                If $\sum_{i=1}^{n} \alpha_i^{lT_p} x_i^{*T_p} \in \mathbb{Z}$, then select a $j \in \{1, ..., n\}$ such that $x_j^{*T_p} \notin \mathbb{Z}$,
                    and if $\alpha_j^{lT_p} = 0$, then $\alpha_j^{lT_p} := 1$, else $\alpha_j^{lT_p} := 0$.
                Set $\beta^{lT_p} := \lfloor \sum_{i=1}^{n} \alpha_i^{lT_p} x_i^{*T_p} \rfloor$.
            End for
            Create the following four new nodes with $T_p$ as the parent by adding
            the following constraints to $T_p$'s linear relaxation.
                $T_p^{LL} := T_{totalnodes+1}$ with the following constraints included in the
                subproblem.
                    $\sum_{i=1}^{n} \alpha_i^{1T_p} x_i \leq \beta^{1T_p}$,
        $\sum_{i=1}^{n} \alpha_i^{2T_p} x_i \leq \beta^{2T_p}$ and
        $\sum_{i=1}^{n} \lfloor \frac{\alpha_i^{1T_p}+\alpha_i^{2T_p}}{2} \rfloor x_i \leq \lfloor \frac{\beta^{1T_p}+\beta^{2T_p}}{2} \rfloor$.
    $T_p^{LG} := T_{totalnodes+2}$ with the following constraints included in the subproblem.
        $\sum_{i=1}^{n} \alpha_i^{1T_p} x_i \leq \beta^{1T_p}$,
        $\sum_{i=1}^{n} \alpha_i^{2T_p} x_i \geq \beta^{2T_p} + 1$ and
        $\sum_{i=1}^{n} \lfloor \frac{\alpha_i^{1T_p}-\alpha_i^{2T_p}}{2} \rfloor x_i \leq \lfloor \frac{\beta^{1T_p}-(\beta^{2T_p}+1)}{2} \rfloor$.
    $T_p^{GL} := T_{totalnodes+3}$ with the following constraints included in the subproblem.
        $\sum_{i=1}^{n} \alpha_i^{1T_p} x_i \geq \beta^{1T_p} + 1$,
        $\sum_{i=1}^{n} \alpha_i^{2T_p} x_i \leq \beta^{2T_p}$ and
        $\sum_{i=1}^{n} \lfloor \frac{-\alpha_i^{1T_p}+\alpha_i^{2T_p}}{2} \rfloor x_i \leq \lfloor \frac{-(\beta^{1T_p}+1)+\beta^{2T_p}}{2} \rfloor$.
    $T_p^{GG} := T_{totalnodes+4}$ with the following constraints included in the subproblem.
        $\sum_{i=1}^{n} \alpha_i^{1T_p} x_i \geq \beta^{1T_p} + 1$,
        $\sum_{i=1}^{n} \alpha_i^{2T_p} x_i \geq \beta^{2T_p} + 1$ and
        $\sum_{i=1}^{n} \lfloor \frac{-\alpha_i^{1T_p}-\alpha_i^{2T_p}}{2} \rfloor x_i \leq \lfloor \frac{-(\beta^{1T_p}+1)-(\beta^{2T_p}+1)}{2} \rfloor$.
    $totalnodes := totalnodes + 4$.
End if
```

End while

Output

If $z^* = -\infty$, then report the problem as infeasible, else report z^* and x^* as the optimal solution.

The introduction of internal cutting planes garners a host of benefits. The immediately available benefit is that in some instances the use of C-G cuts removes problematic noninteger solutions. Returning to the example from the opening section, suppose the LR solution for some node $T_{p'}$ is $x^* = (1.5, 3.5)$ with the branching constraints $((1, 1), \beta^1)$ and $((1, -1), \beta^2)$. Then $\beta^1 = 1 * (1.5) + 1 * (3.5) = 5$ and $\beta^1 = 1 * (1.5) - 1 * (3.5) = -2$. In the original example this is problematic because x^* is contained in T_p^{LL} for either constraint. The C-G cut derived from the two branching constraints is $((1, 0), 1)$, equivalently $x_1 \leq 1$, which clearly precludes this LR from future children. Lemma 3.2.2 formally proves this result. Now that QHBA has been established, it is beneficial to explore an example to demonstrate the unique characteristics of this method. The example from chapter 2 is explored in the context of QHBA, and the results are compared to traditional BB.

Example 3.1.1 Reconsider the following problem from Example 1

Maximize $5x_1 + 2x_2$

subject to $10x_1 + 2x_2 \leq 23$

$4x_1 + 2x_2 \leq 13$

$x_1, x_2 \geq 0$ and $x_1, x_2 \in \mathbb{Z}^n$

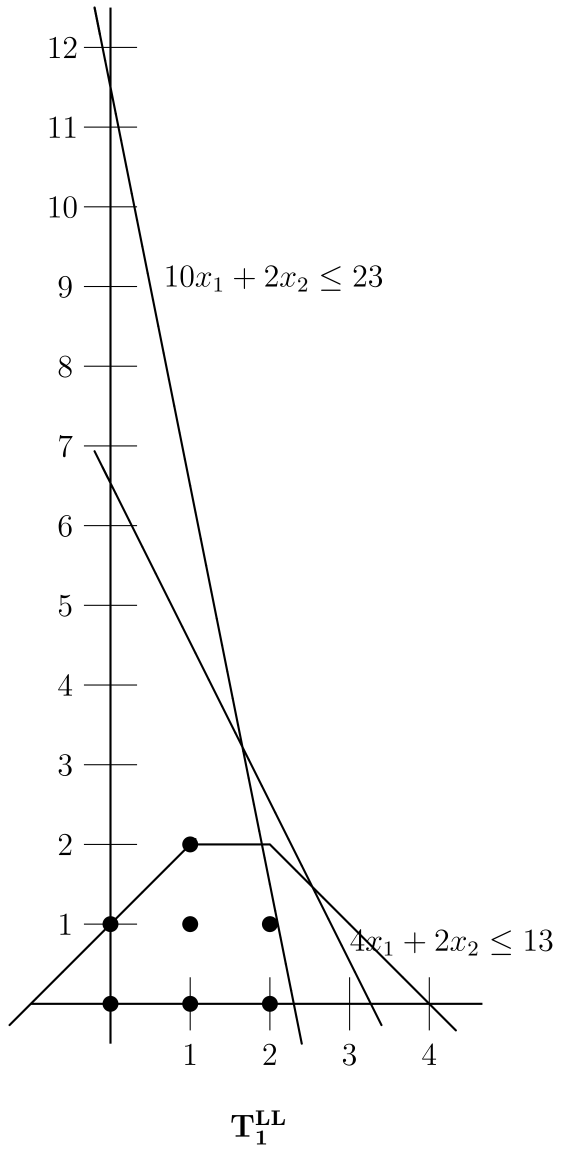
For simplicity, the same coefficients for the branching constraints are used throughout the example. The branching constraints are $1x_1 + 1x_2 \leq \beta^1$ and $-1x_1 + 1x_2 \leq \beta^2$, with β values to be determined based on the LR solution the current node. Breadth first right is used as the search strategy.

The first iteration of QHBA finds the LR solution for the root node, which is $z^* = \frac{44}{3}$ and $x^* = (\frac{5}{3}, \frac{19}{6})$ and this node is not fathomed. Now β^1 and β^2 can be calculated with $\beta^1 = \lfloor \frac{5}{3} + \frac{19}{6} \rfloor = 4$ and $\beta^2 = \lfloor -\frac{5}{3} + \frac{19}{6} \rfloor = 1$. The constraints are added to the root node's LR to create 4 children as follows:

The first child created is T_1^{GG}, due to the breadth first right strategy, is represented by T_2 with the following constraints $x_1 + x_2 \geq 5$, $-x_1 + x_2 \geq 2$ and $x_2 \geq 4$. The third constraint $x_2 \geq 4$ is generated using the Chvátal-Gomory rounding procedure. The exact derivation begins with multiplying the two '$\geq$' constraint by a -1 to transform them to a '$\leq$'. Next the constraints a multiplied by $\frac{1}{2}$ and summed resulting in $\frac{1}{2}(-x_1 + -x_2 \leq -5) + \frac{1}{2}(x_1 + -x_2 \leq -2)$. This results in the inequality $0x_1 + -x_2 \leq -\frac{7}{2}$. Using the C-G rounding procedure the constraint can be modified to $\lfloor 0 \rfloor x_1 + -\lfloor 1 \rfloor x_2 \leq \lfloor -\frac{7}{2} \rfloor$, which is equivalently $-x_2 \leq -4$ or $x_2 \geq 4$.

For the next child, $T_1^{GL} = T_3$ with the following additional constraints $x_1 + x_2 \geq 5$ and $-x_1 + x_2 \leq 1$. In this set of constraints it is also possible to use the C-G procedure to generate a third constraint. The constraint can be derived as follows, $\frac{1}{2}(-x_1 + -x_2 \leq -5) + \frac{1}{2}(-x_1 + x_2 \leq 1)$. The intermediate inequality is $-x_1 + 0x_2 \leq -2$, and with rounding $\lfloor -1 \rfloor x_1 + -\lfloor 0 \rfloor x_2 \leq \lfloor -2 \rfloor$, which is the redundant constraint $x_1 \geq 2$. Notice that the intersection of the two branching constraints occurs at $(2,3)$. Furthermore, the only point that meets $x_2 \leq 2$ at equality is $(2,3)$. Since this polyhedron has a dimension of two and $x_1 \leq 2$ has a face of dimension 0, this C-G inequality is redundant.

The next child is $T_1^{LG} = T_4$ with the added constraints $x_1 + x_2 \leq 4$ and $-x_1 + x_2 \geq 2$, and the fourth child is $T_1^{LL} = T_5$ with the following additional constraints $x_1 + x_2 \leq 4$, $-x_1 + x_2 \leq 1$ and $x_2 \leq 2$. The third constraint for T_1^{LL} is generated using similar arguments as T_1^{GG}. The Chvátal-Gomory procedure can be generalized to instances when in some branch $\beta^1 + \beta^2$ is odd. So returning to T_1^{LL}, the LR includes the extra constraints $x_1 + x_2 \leq 4$, $-x_1 + x_2 \leq 1$ and $x_2 \leq 2$. If only the first two constraints are included it would be possible to achieve a basic feasible solution from the branching structure in $\mathbb{R}^n \backslash \mathbb{Z}$. As previously mentioned, this is problematic for the termination conditions of QHBA.



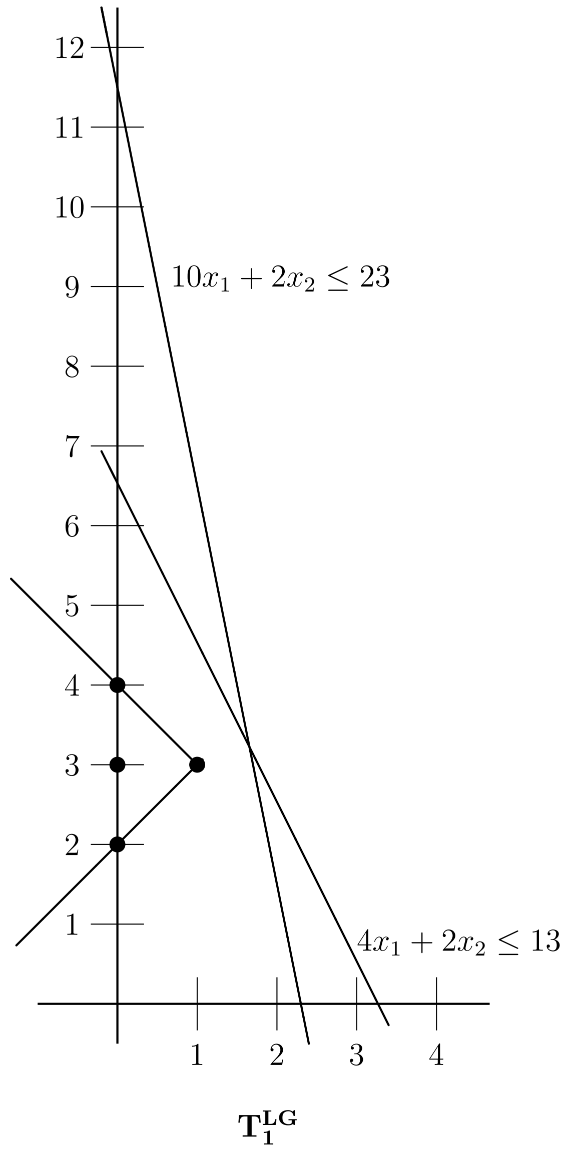


Figure 3.1: Subproblems T_1^{LL} and T_1^{LG}

As illustrated in Figures 3.2 and 3.1, these four subproblems partition the solution space without excluding any candidate integer solutions. The C-G cuts are depicted in the nodes T_1^{LL} and T_1^{GG}. Note how the C-G cut induces two integer extreme points. If these constraints were not implemented, then T_1^{LL} and T_1^{GG} would have structures similar to T_1^{LG} and T_1^{GL} and the branching constraints would not intersect at an integer point. The implementation of the C-G cuts can prevent this issue.

The search strategy is breadth first right, therefore the first node to be evaluated is T_1^{GG} denoted node 2. Node 2 has a noninteger solution with $z^{*T_1^{GG}} = 14.25$, $x^{*T_1^{GG}} = (1.25, 4)$. Node 2 is a candidate for future branching. The next node is T_1^{GL}, marked node 3, which is infeasible. Thus node 3 is fathomed. The next child to be evaluated is T_1^{LG} (node 4), which is an integer solution. Since an integer solution is found z^* is updated to 11 with $x^* = (1, 3)$ and this node is fathomed. T_1^{LL} is the next node evaluated, node 5, which is also a non-integer solution with $z^{*T_1^{LL}} = 13.5$, $x^{*T_1^{LL}} = (1.9, 2)$. Node 5 is also a candidate for future branching.

Nodes 2, and 5 are potential candidates for further exploration and are not pruned by the recorded best integer solution. For breadth first right, node 2 serves as the basis for the next four iterations. Branching constraints are generated using the new LR according to the procedure outlined earlier. The four new children are evaluated right to left as before. Node 6 is non-integer and is not fathomed with $z^{*T_2^{GG}} = 13.5$ and $x^{*T_2^{GG}} = (.5, 5.5)$. Node 7 is infeasible with the added branching constraints and is fathomed. Node 8 provides a new integer solution with $z^{*T_2^{LG}} = 13$ and $x^{*T_2^{LG}} = (1, 4)$.

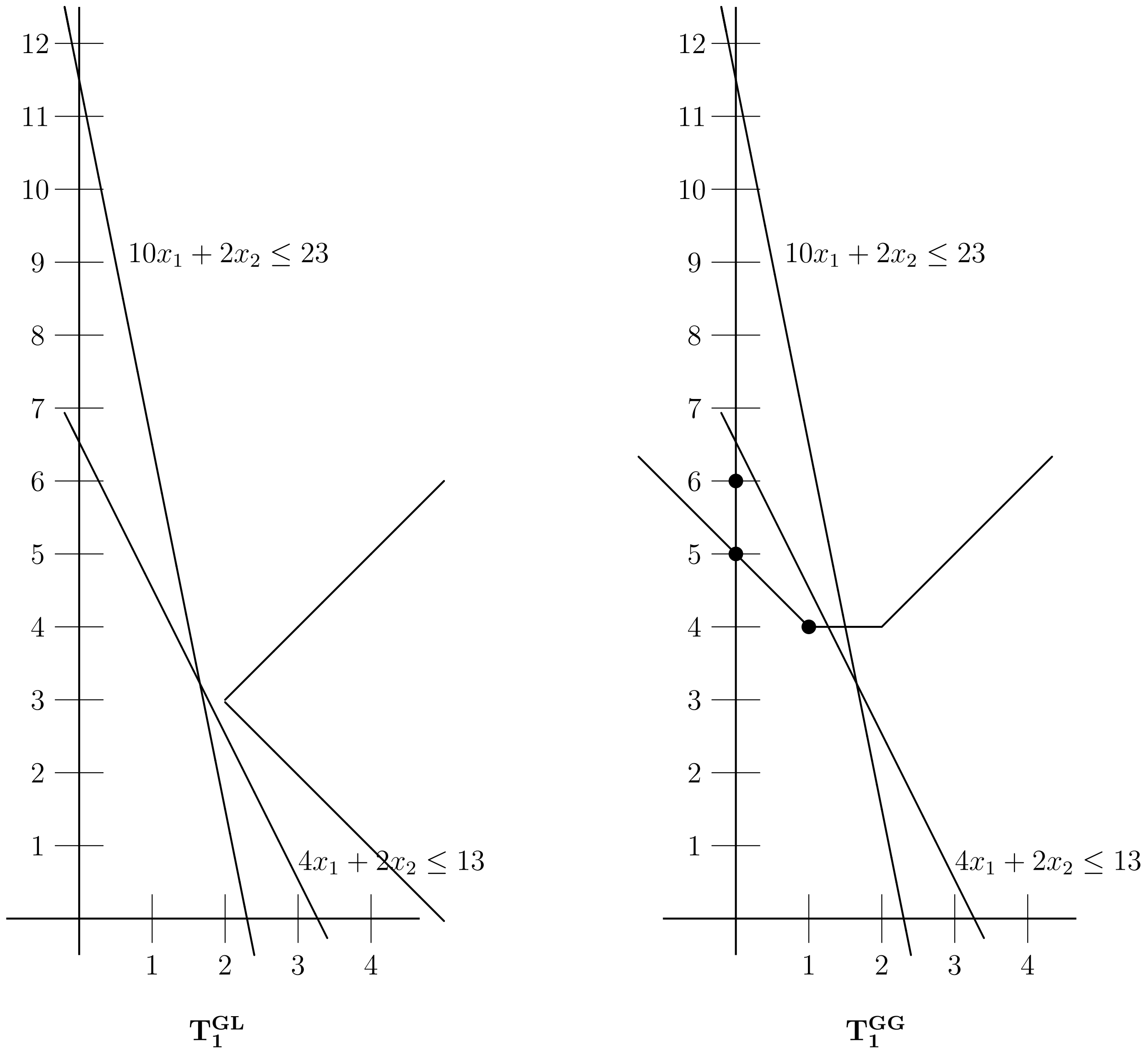


Figure 3.2: Subproblems T_1^{GL} and T_1^{GG}

Since this is a superior integer solution to the current best recorded solution, z^* is updated to 13 with $x^* = (1, 4)$ and the node is fathomed. Finally node 9 is evaluated and is infeasible and thus fathomed. Figure 3.3 shows QHBA's branching tree.

At this point the problem is functionally solved. QHBA marks nodes 5 and 6 for constraint generation, but no new nodes are evaluated because the parent nodes (nodes 5 and 6) have fractional integer solutions where the next lowest potential integer solution is z^*. This results in fathoming of nodes 5 and 6 because the objective values are non-integer values less than 14 but greater than 13. Since the objective function has integer coefficients, there are no integer solutions between 14 and 13, and fathoming can occur because there is no better integer solution.

According to QHBA as outlined, branching constraints are created for nodes 2 and 6. However, a node is only evaluated when the search strategy selects that branch to evaluate (this is where a majority of the computational complexity arises). Referring back to Figure 3.3, the breadth first right search strategy dictates that nodes 6-9 are evaluated before exploring the children of node 5, and then the children of node 6. Given that an integer solution is found at node 8, those noninteger parent nodes become fathomed. Consequently, there is no need to select any of the remaining possible branches.

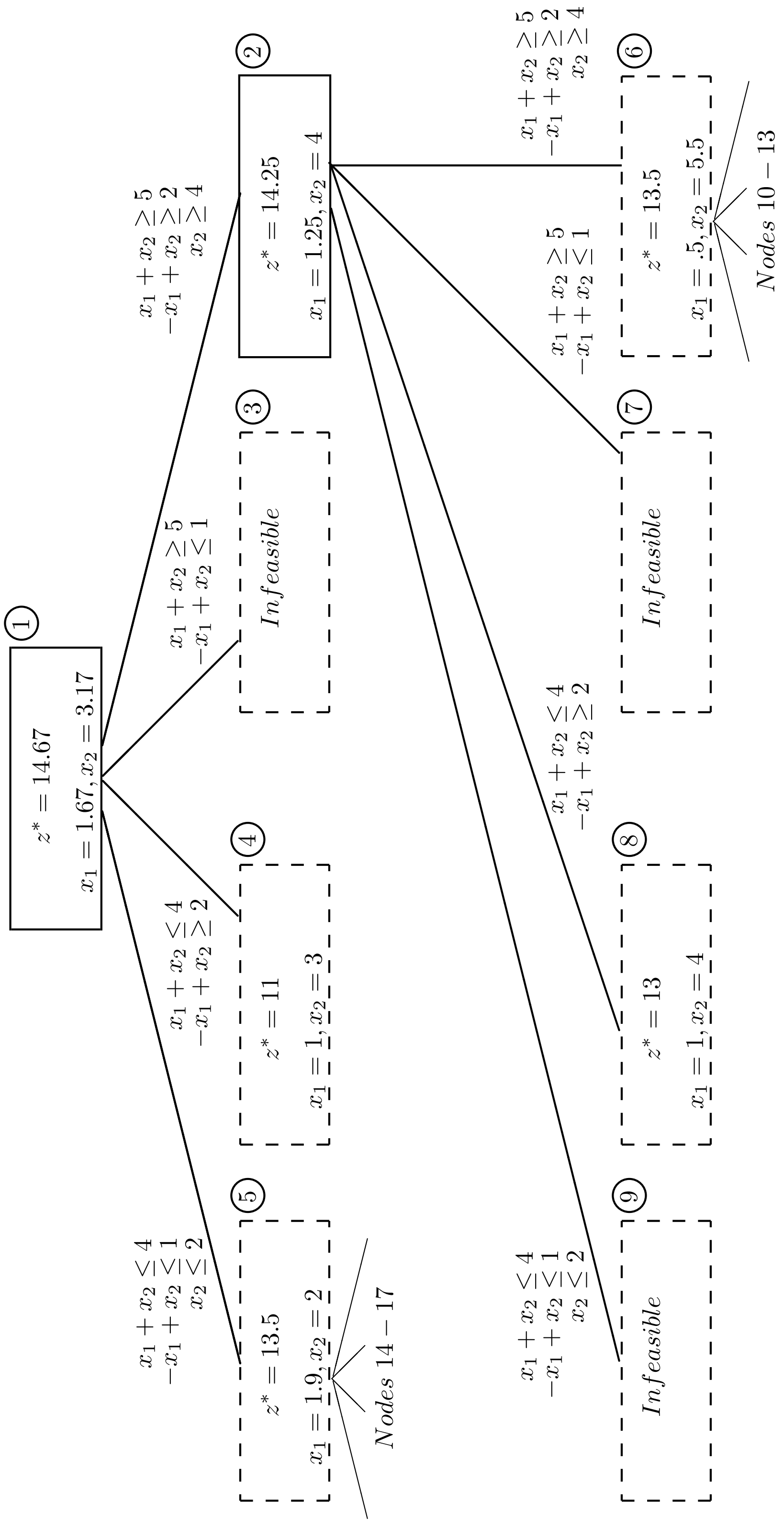

1
$z^* = 14.67$
$x_1 = 1.67, x_2 = 3.17$
2
$z^* = 14.25$
$x_1 = 1.25, x_2 = 4$
3
$Infeasible$
4
$z^* = 11$
$x_1 = 1, x_2 = 3$
5
$z^* = 13.5$
$x_1 = 1.9, x_2 = 2$
$Nodes\ 14 - 17$
6
$z^* = 13.5$
$x_1 = .5, x_2 = 5.5$
$Nodes\ 10 - 13$
7
$Infeasible$
8
$z^* = 13$
$x_1 = 1, x_2 = 4$
9
$Infeasible$
$x_1 + x_2 \geq 5$, $-x_1 + x_2 \geq 2$, $x_2 \geq 4$
$x_1 + x_2 \geq 5$, $-x_1 + x_2 \leq 1$
$x_1 + x_2 \leq 4$, $-x_1 + x_2 \geq 2$
$x_1 + x_2 \leq 4$, $-x_1 + x_2 \leq 1$, $x_2 \leq 2$
$x_1 + x_2 \geq 5$, $-x_1 + x_2 \geq 2$, $x_2 \geq 4$
$x_1 + x_2 \geq 5$, $-x_1 + x_2 \leq 1$
$x_1 + x_2 \leq 4$, $-x_1 + x_2 \geq 2$
$x_1 + x_2 \leq 4$, $-x_1 + x_2 \leq 1$, $x_2 \leq 2$

Figure 3.3: QHBA Enumeration Tree

Termination of the algorithm occurs now that all pendant nodes are fathomed. The optimal solution $z^* = 13$, $x^* = (1, 4)$ is reported. Although the total number of nodes in the tree is higher than the example in chapter 2. The computational study provides evidence that QHBA is computationally superior to BB.

3.2 Theory of QHBA

The primary result of this section is to show that QHBA guarantees an optimal solution in a finite number of steps. This section provides a lemma to show that QHBA does not exclude any potential integer solutions, followed by a proof that QHBA removes continuous space by removing the linear relaxation from a parent node when children are created. By then showing that the extrema in every subproblem are integer, a proof of finite termination is presented.

To guarantee that an optimal solution is found in any implementation of BB it must be shown that the optimal integer solution is not excluded. This is trivially shown in traditional BB, but given the unique characteristics of QHBA, this may not be the case. Simply put, it must be shown that no candidate integer solutions are eliminated at any iteration of QHBA. The quaternary branching structure of QHBA must not violate any potential integer solutions. When creating children, all integer points that were candidate solutions in the parent node must still be feasible in one of the four children. Formally,

Lemma 3.2.1. *Let $T_{p'}$ be any node in QHBA's branching tree, then $\mathbb{Z}^n \cap T_{p'} = (\mathbb{Z}^n \cap$*

$T_{p'}^{LL}) \cup (\mathbb{Z}^n \cap T_{p'}^{LG}) \cup (\mathbb{Z}^n \cap T_{p'}^{GL}) \cup (\mathbb{Z}^n \cap T_{p'}^{GG})$.

Proof: Trivially, every $x \in \mathbb{Z}^n$ satisfies either $\sum_{j=1}^n \alpha_j x_j \leq \beta$ or $\sum_{j=1}^n \alpha_j x_j \geq \beta + 1$ as long as $\alpha_j \in \mathbb{Z}^n$ and $\beta \in \mathbb{Z}$. Consequently, every $x \in \mathbb{Z}^n$ satisfies one of the following four sets of inequalities.

LL) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \leq \beta^{1T_{p'}}$ and $\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \leq \beta^{2T_{p'}}$

LG) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \leq \beta^{1T_{p'}}$ and $\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \geq \beta^{2T_{p'}} + 1$

GL) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \geq \beta^{1T_{p'}} + 1$ and $\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \leq \beta^{2T_{p'}}$

GG) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \geq \beta^{1T_{p'}} + 1$ and $\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \geq \beta^{2T_{p'}} + 1$

It is well known that Gomory-Chv́atal cuts do not eliminate any integer points. Thus, every $x \in \mathbb{Z}^n$ satisfies one of the following four sets of inequalities.

LL) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \leq \beta^{1T_{p'}}$,

$\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \leq \beta^{2T_{p'}}$ and

$\sum_{j=1}^n \lfloor \frac{{\alpha_j}^{1T_{p'}} + {\alpha_j}^{2T_{p'}}}{2} \rfloor x_j \leq \lfloor \frac{\beta^{1T_{p'}} + \beta^{2T_{p'}}}{2} \rfloor$.

LG) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \leq \beta^{1T_{p'}}$,

$\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \geq \beta^{2T_{p'}} + 1$ and

$\sum_{j=1}^n \lfloor \frac{{\alpha_j}^{1T_{p'}} - {\alpha_j}^{2T_{p'}}}{2} \rfloor x_j \leq \lfloor \frac{\beta^{1T_{p'}} - \beta^{2T_{p'}}}{2} \rfloor$.

GL) $\sum_{j=1}^n {\alpha_j}^{1T_{p'}} x_j \geq \beta^{1T_{p'}} + 1$,

$\sum_{j=1}^n {\alpha_j}^{2T_{p'}} x_j \leq \beta^{2T_{p'}}$ and

$\sum_{j=1}^n \lfloor \frac{-{\alpha_j}^{1T_{p'}} {\alpha_j}^{2T_{p'}}}{2} \rfloor x_j \leq \lfloor \frac{-\beta^{1T_{p'}} \beta^{2T_{p'}}}{2} \rfloor$.

GG) $\sum_{j=1}^{n} \alpha_j^{1T_{p'}} x_j \geq \beta^{1T_{p'}} + 1$,

$\sum_{j=1}^{n} \alpha_j^{2T_{p'}} x_j \geq \beta^{2T_{p'}} + 1$ and

$\sum_{j=1}^{n} \lfloor \frac{-\alpha_j^{1T_{p'}} - \alpha_j^{2T_{p'}}}{2} \rfloor x_j \leq \lfloor \frac{-\beta^{1T_{p'}} - \beta^{2T_{p'}}}{2} \rfloor$.

Thus $\mathbb{Z}^n \cap T_{p'} \subseteq (\mathbb{Z}^n \cap T_{p'}^{LL}) \cup (\mathbb{Z}^n \cap T_{p'}^{LG}) \cup (\mathbb{Z}^n \cap T_{p'}^{GL}) \cup (\mathbb{Z}^n \cap T_{p'}^{GG})$. Trivially, $T_{p'}^{LL}$, $T_{p'}^{LG}$, $T_{p'}^{GL}$, $T_{p'}^{GG} \subseteq T_{p'}$. Consequently, $\mathbb{Z}^n \cap T_{p'} = (\mathbb{Z}^n \cap T_{p'}^{LL}) \cup (\mathbb{Z}^n \cap T_{p'}^{LG}) \cup (\mathbb{Z}^n \cap T_{p'}^{GL}) \cup (\mathbb{Z}^n \cap T_{p'}^{GG})$.

$\square$

Lemma 3.2.1 guarantees that each integer solution satisfying a parent node is contained in one of the child nodes. However, this does not have any bearing on finite termination. This theory only guarantees that QHBA does not eliminate any feasible integer points and that the solution is optimal, if QHBA terminates.

In order to achieve finite termination we must ensure that the child nodes of a given node $T_{p'}$ excludes the LR solution. If not, then the algorithm may never end. The following theorem formally shows this exclusion.

Lemma 3.2.2. *Let $x' \in \mathbb{R}^n \setminus \mathbb{Z}^n$, then x' satisfies none of the four subproblems created by the branching step of QHBA.*

Proof: Suppose $x \in \mathbb{R}^n \setminus \mathbb{Z}^n$. Let (α^1, β^1) and (α^2, β^2) be any branching inequalities generated from QHBA. QHBA guarantees $\sum_{j=1}^{n} \alpha_j^{1T_{p'}} x'_j \notin \mathbb{Z}$ and $\sum_{j=1}^{n} \alpha_j^{2T1} x'_j \notin \mathbb{Z}$. Since β^1 and $\beta^2 \in \mathbb{Z}$, x' is trivially not contained in any of the four subproblems and the result follows.

□

Therefore, each set of children generated by QHBA removes some continuous solution space. The next result is used to provide a lower bound on how much linear relaxation space is removed by each set of children.

Lemma 3.2.3. *The extreme points of the polyhedron given by the branching constraints for LL), LG), GL) and GG) are integer.*

Proof: By definition, the extreme points of the polyhedron given by any of the constraints for LL), LG), GL) and GG) must be basic feasible solutions to these systems of equations. Multiplying through by a negative one for a greater than constraint does not change the integrality of a polytope. Thus, it suffices to only consider LL).

Case 1: If $\beta^{T_1} + \beta^{T_2}$ is even, then the Chvátal-Gomory cut is redundant and can be removed. Consider all basic solutions that could exist across the first two constraints. Assume the basic variables are x_i and x_j and their columns take the form

$$B = \begin{vmatrix} \alpha_i^{T_1} & \alpha_j^{T_1} \\ \alpha_i^{T_2} & \alpha_j^{T_2} \end{vmatrix}.$$

a) If there exists a column or row of 0s in B or if multiplying one column by -1 produces the other column, then there does not exist a basis as the selected variables are not full rank.

b) If there exists at least one 0 in any one of these four locations of B, then the basis matrix is TUM and therefore the extreme points are integer.

c) Without loss of generality, the only full rank matrix B with no zeros takes the generic form $\begin{vmatrix} 1 & 1 \\ -1 & 1 \end{vmatrix}$. The basic feasible solution can be obtained using Cramer's rule, which is a solution method for systems of equations introduced by Gabriel Cramer in 1750 in his book Introduction l'analyse des lignes courbes algbriques and is commonly taught in high school [6]. Using Cramer's rule results in $x_i = \frac{\begin{vmatrix} \beta^{T_1} & \alpha_j^{T_1} \\ \beta^{T_2} & \alpha_j^{T_2} \end{vmatrix}}{\begin{vmatrix} \alpha_i^{T_1} & \alpha_j^{T_1} \\ \alpha_i^{T_2} & \alpha_j^{T_2} \end{vmatrix}}$ and $x_j = \frac{\begin{vmatrix} \alpha_i^{T_1} & \beta^{T_1} \\ \alpha_i^{T_2} & \beta^{T_2} \end{vmatrix}}{\begin{vmatrix} \alpha_i^{T_1} & \alpha_j^{T_1} \\ \alpha_i^{T_2} & \alpha_j^{T_2} \end{vmatrix}}$. Thus, $x_i = \frac{\beta^{T_2} - \beta^{T_1}}{2}$ and $x_j = \frac{\beta^{T_1} + \beta^{T_2}}{2}$. Since $\beta^{T_1} + \beta^{T_2}$ is even, both x_i and x_j are integer.

Case 2: If $\beta^{T_1} + \beta^{T_2}$ is odd, then the Chvátal-Gomory cut is not redundant and there are 3 basic variables that must be evaluated. The three basic columns must be taken from the following matrix where the last three columns represent the slack for each constraint

$$B' = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 1 \end{vmatrix}.$$

Multiplying any column by a -1 flips the sign of a determinant. If a basis is chosen that provides a noninteger (integer) solution, then the solution would remain noninteger

(integer) under such an operation by Cramer's rule. Furthermore, a column of 0s can never be full rank and any repetitive columns can be removed, so B' can be reduced to

$$B' = \begin{vmatrix} 1 & 1 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 & 1 \end{vmatrix}.$$

The proof proceeds in cases based off of which slack variables are basic. If all the slack variables are basic, then clearly the solution is integer since the right hand side is integer. Now assume that any two slack variable are basic. In such a situation, either the basis is infeasible or it clearly has an integer solution.

Now assume that only one slack variable is in the basis. Furthermore, assume that such a basic variable is slack 1. The value of the basic variables for the second and third row can be obtained by solving with at most 2 nonzero x's.

$$\begin{vmatrix} 1 & -1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 & 1 \end{vmatrix} x = \begin{vmatrix} \beta^{T_2} \\ \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor \end{vmatrix}.$$

Clearly the right hand side is integer and this matrix is TUM. Thus the variables have an integer solution and so the basic solution is integer. Note that having slack 2 as a basic variable follows an identical argument.

Now assume that slack 3 is basic. Now the solution for the first and second basic variables can be obtained by solving with at most 2 non-zeros columns the following system of equations.

$\begin{vmatrix} 1 & 1 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 1 & 0 \end{vmatrix} x = \begin{vmatrix} \beta^{T_1} \\ \beta^{T_2} \end{vmatrix}$. Notice that this matrix is TUM except for the first two columns. Thus the solutions are integer unless the basis for this case is

$$B'' = \begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \end{vmatrix} \; x \; = \begin{vmatrix} \beta^{T_1} \\ \beta^{T_2} \\ \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor \end{vmatrix}.$$

The unique solution to this instance is $(\frac{\beta^{T_1}+\beta^{T_2}}{2}, \frac{\beta^{T_1}-\beta^{T_2}}{2}, \frac{-1}{2})$. Since $s_3 = \frac{-1}{2}$, this is not a basic feasible solution and this case follows.

Finally, assume that none of the slack variables are basic. Then the basic variables must be chosen from $B''' = \begin{vmatrix} 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \\ 1 & 0 & -1 & -1 \end{vmatrix}$.

If the first 3 columns are selected, then the solution is $(\beta^{T_1} + \beta^{T_2} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, -\beta^{T_2} + \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, \beta^{T_1} + \beta^{T_2} - 2\lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor)$. Clearly this is integer or infeasible.

If the first two columns and the fourth column are selected for the basis, then the solution is $(\beta^{T_1} + \beta^{T_2} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, \beta^{T_1} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, \beta^{T_1} + \beta^{T_2} - 2\lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor)$. Again this is clearly integer or infeasible.

If the first column and the last two column are selected for the basis, then the solution is $(\beta^{T_1} + \beta^{T_2} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, \beta^{T_1} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor, \beta^{T_2} - \lfloor \frac{\beta^{T_1}+\beta^{T_2}}{2} \rfloor)$. This is clearly integer or infeasible since $\beta^{T_1} > \beta^{T_2}$.

The last three columns cannot be chosen since they are dependent. Consequently, every basic feasible point of QBHA's branching routine is integer and the result follows.

□

Now that all extrema created by QHBA are integer, it is possible to show that QHBA finitely terminates. An advancement from this research is an alternate proof of finite termination for BB, and consequently QHBA. The standard approach to show finite termination for traditional BB rests in the calculation of a worse case scenario for the size of the resulting BB enumeration tree. By using the branching constraints $x_j \leq \lfloor x_j^{*LR} \rfloor$ and $x_j \geq \lfloor x_j^{*LR} \rfloor + 1$, the maximum number of branches necessary is two times the sum of the upper bounds for all x's. Eventually BB branches deep enough that $x_i \leq q_i$ and $x_i \geq q_i$ for all $i = 1, ..., n$, and then each x_i is integer with a value of q_i or the LR is infeasible.

This standard approach cannot be applied to QHBA. Since the branching constraints in QHBA are multi-variate and the α coefficients are not constant, any single variable is not restricted to some integer value. Instead a lower bound on the amount of continuous space removed every time a set of children is created. Therefore, if the IP is bounded, it has finite volume. Thus it can be shown that after a finite number of branches all continuous space is removed and QHBA generates an enumeration scheme. Formally,

Theorem 3.2.4. *Given any bounded integer program, QHBA solves the integer program in a finite number of steps.*

Proof: Let IP be any bounded integer program, where each x_i is bounded by 0 and u. Thus, the volume of the linear relaxation space is bounded by a hyperrectangle with volume u^n. Given any unfathomed node $T_{p'}$, the four children of $T_{p'}$ generated by QHBA eliminates a fractional point, $x^{*T_{p'}}$. Furthermore, Lemma 3.2.3 guarantees that every extreme point of the constraints used to create these four children is integer. Thus, the volume of the eliminated region is at least one unit. Consequently, QHBA requires at most u^n steps that create children. After u^n sets of child nodes are created there are no noninteger solutions remaining and every pendant node is fathomed. Furthermore, no feasible integer point is removed by lemma 3.2.1, so QHBA reports the optimal solution.

□

Given that the QHBA evaluates all integer points and that it terminates in a finite number of steps, QHBA finds an optimal solution, if one exists. However, even though these fundamental characteristics have been established, a benefit must be accrued from QHBA to show practicality in its implementation. The next section focuses on some theoretical benefits of QHBA.

3.3 Theoretical Benefits of QHBA

The implementation of QHBA leads to a host of benefits over traditional implementations of BB. A primary result is that when QHBA creates children there are more integer extreme points due to the branching structure. Recall that a total of 10 inequalities are added per set of children created. Having more integer feasible solutions at any subproblem is beneficial because this encourages a candidate integer solution to be found faster. Additionally, with the inclusion of Chvátal-Gomory cuts in certain child nodes the amount of continuous space that QHBA eliminates increases. This should result in a decrease in the total number of nodes when compared to BB.

A direct comparison between the quaternary branching QHBA and the binary traditional BB is not a fair comparison because QHBA creates four nodes and BB only generates two nodes. Therefore a quaternary branching scheme of BB, or quaternary branch and bound (QBB), should be created.

A quaternary branching structure implemented in BB with single variable constraints is identical to QHBA, except the branching constraints are of the form $x_i \leq \beta^1$, $x_i \geq \beta^1 + 1$, $x_j \leq \beta^2$ and $x_j \geq \beta^2$ where $i \neq j$ when $x_i, x_j \in \mathbb{R}^n \backslash \mathbb{Z}^n$ where $\beta^1 = \lfloor x_i \rfloor$ and $\beta^2 = \lfloor x_j \rfloor$. If such an x_j does not exist, then any $i \neq j$ will suffice. The 4 partitions, or children, are created as follows. T_p^{LL} includes the constraints $x_i \leq \beta^1$ and $x_j \leq \beta^2$. The next subproblem includes $x_i \leq \beta^1$, $x_j \geq \beta^2 + 1$ which is denoted T_p^{LG}. The subproblem T_p^{GL} includes the constraints $x_i \geq \beta^1 + 1$ and $x_j \leq \beta^2$. The final subproblem, T_p^{GG}, includes $x_i \geq \beta^1 + 1$ and $x_j \geq \beta^2 + 1$.

To illustrate single variable branching in a quaternary tree, return to the example from the beginning of the chapter articulating conditions where hyperplane branching could enter an infinite loop. The root node x^{*LR} is $(1.5, 3.5)$ so the branching constraints are $((1,0),1)$ and $((0,1),3)$. T_1^{LL} includes the constraints $x_1 \leq 1$ and $x_2 \leq 3$, T_1^{LG} includes the constraints $x_1 \leq 1$ and $x_2 \geq 4$, T_1^{GL} includes the constraints $x_1 \geq 2$ and $x_2 \leq 3$ and T_1^{GG} includes the constraints $x_1 \geq 2$ and $x_2 \geq 4$. Note that the third C-G constraint can be derived for any of the pairs, however it is always redundant.

Figure 3.4 is an illustration in two dimensions of the increased effectiveness of QHBA. The leftmost graph shows QHBA's branching structure and the rightmost graph shows

QBB's branching structure. One set of created children for QHBA has 6 integer extrema distributed over the four child nodes that are clustered around x^{*LR}. However, QBB only has 4 such integer extrema. Furthermore, QHBA eliminates an area of dimension two around x^{*LR} while QBB only eliminates an area of one. Both of these benefits are greatly expanded in higher dimensions.

After establishing a method for single variable branching constraints using a quaternary scheme, the number of integer extreme points at each set of created children can be defined. Observe that these integer points are independent to the IP, thus in certain instances they may be infeasible for a particular QBB node. Formally,

Theorem 3.3.1. *One set of siblings in QBB in $\mathbb{R}^n_+$ results in at most nine basic feasible solutions that are integer for any $n \geq 2$.*

Proof: The proof is shown by induction. Without loss of generality and through index substitution, it can be assumed that the branching variables are x_1 and x_2 in a set of siblings in standard branch and bound with a quaternary branching strategy.

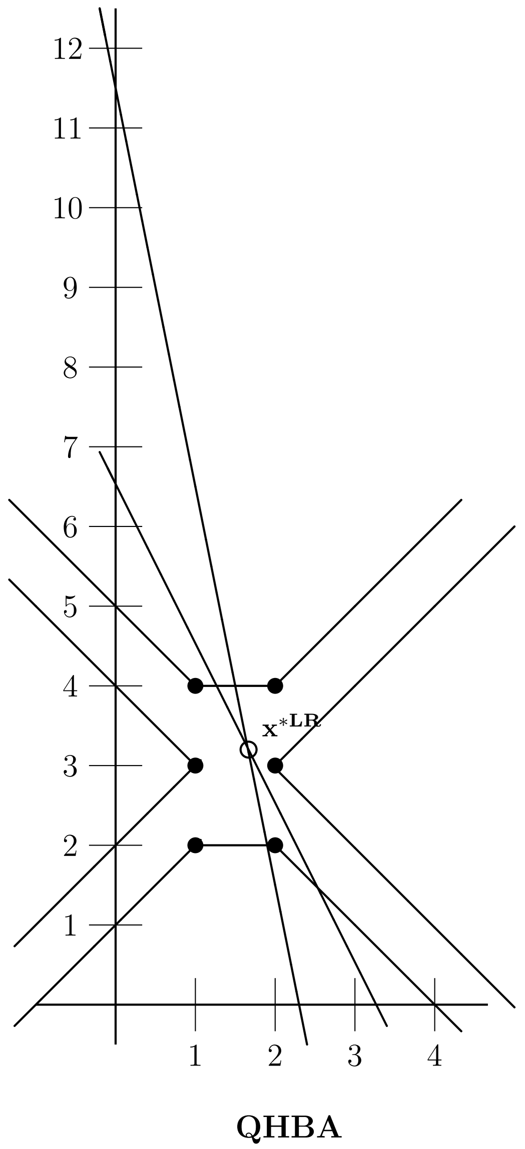
Base case: Let $n = 2$. Clearly there are at most 9 basic feasible solutions. The subproblems for the 4 child nodes are:

1) $x_1 \leq \beta^1$ $x_2 \leq \beta^2$

2) $x_1 \leq \beta^1$ $x_2 \geq \beta^2 + 1$

3) $x_1 \geq \beta^1 + 1$ $x_2 \leq \beta^2$

4) $x_1 \geq \beta^1 + 1$ $x_2 \geq \beta^2 + 1$



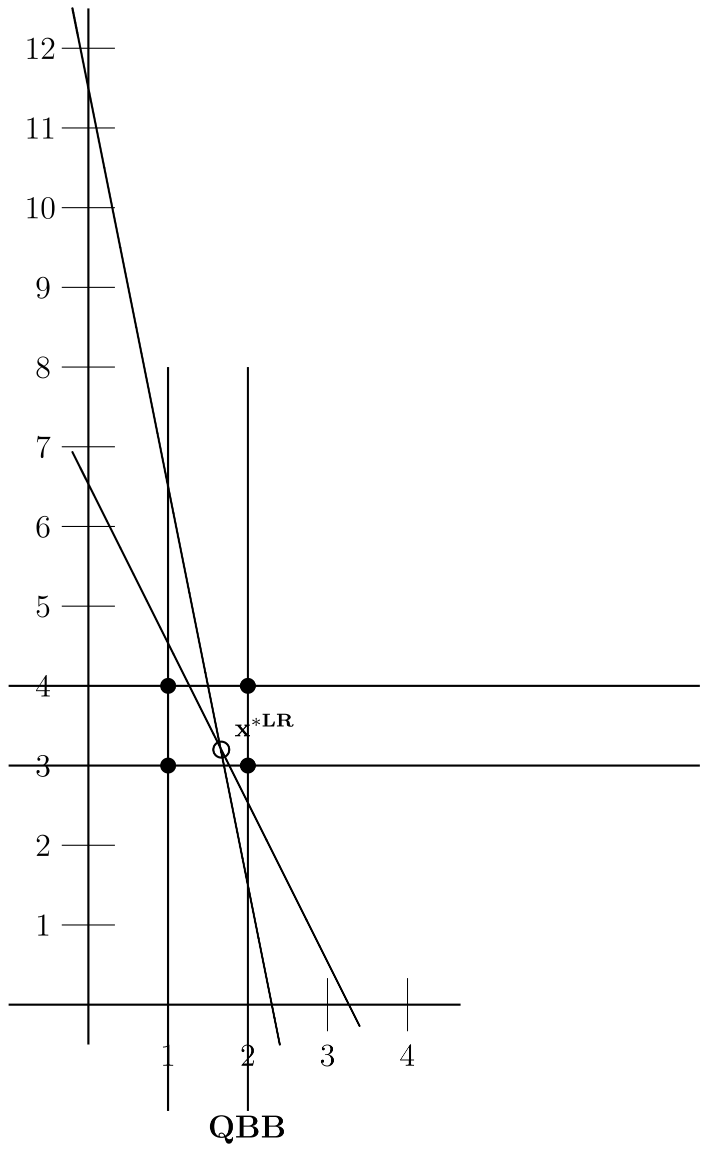


Figure 3.4: One set of children for QHBA and QBB

where β^1 and $\beta^2 \geq 0$ and integer. It is trivial to verify that the only basic feasible solutions, which are all integer, for each child subproblem are as follows:

T_{LL}: (β^1, β^2), $(0, \beta^2)$, $(\beta^1, 0)$ and $(0, 0)$

T_{LG}: $(0, \beta^2 + 1)$ and $(\beta^1, \beta^2 + 1)$

T_{GL}: $(\beta^1 + 1, 0)$ and $(\beta^1 + 1, \beta^2)$

T_{GG}: $(\beta^1 + 1, \beta^2 + 1)$.

If β^1 and β^2 are not 0, then there are at most 9 integer basic feasible solutions. If β^1 or β^2 equals zero; some points are redundant.

Induction step: Assume that standard branch and bound with a quaternary branching strategy has at most nine basic feasible solutions that are integer in $\mathbb{R}^k$. Show this is true for $\mathbb{R}^{k+1}$.

Since there are at most q basic feasible solutions in $\mathbb{R}^k$, denoted by $x^1, ..., x^q$, examine the points in $\mathbb{R}^{k+1}$ which have a 0 for x_{k+1}, $(x^1, 0), (x^2, 0), ..., (x^q, 0)$. Clearly, each of these meet the respective branching inequality at equality and only have 2 nonbasic variables. Thus, there are at least q basic feasible solution that are integer in $\mathbb{R}^{k+1}$.

Suppose that there exists a $q + 1^{th}$ basic feasible integer point in $\mathbb{R}^{k+1}$, called x^{q+1}, that is distinct from the q points in the previous paragraph. Clearly, $x^{q+1}_{k+1} = \eta > 0$ where $\eta \in \mathbb{Z}$ due to the induction step. However, the points $x^{q+1} - \eta e_{k+1}$ and $x^{q+1} + e_{k+1}$ is feasible, where e_i is the i^{th} identity point. Thus, x^{q+1} is not an extreme point and cannot be basic feasible, a contradiction. Consequently, there are exactly q basic feasible solutions and since $q \leq 9$, by induction assumption, the result follows.

□

The next theorem of this section addresses the theoretical benefits of QHBA achieved by increasing the number of integer points. An important result that follows is that by increasing the number of integer basic feasible solutions at any set of children, the amount of polyhedral space eliminated per iteration is also increased.

However in order to make these claims, the location of enumeration must be restricted. When the LR solution of a node exists on a boundary of the polytope, some integer extreme points and eliminated relaxation space is infeasible. Returning to the previous proof, if β^1 and β^2 are 0 then there are only four basic feasible solutions. This can also be seen in Figure 3.4 where some integer points established by the child nodes are not in the feasible region of the problem. This is an obstacle that exists with both QHBA and BB.

Therefore some assumptions are made so that the properties of QHBA can be more uniformly presented. For the remainder of the section it is assumed that either the boundaries are increased by one unit (functional range increases from $[0, u]$ to $[-1, u+1]$, or that QHBA is not operating near the boundaries of the hyperrectangle.

Now consider a simple branching strategy for QBHA. For ease of presentation, assume n is even. The results here can trivially be extended to n odd. Consider $\alpha_i^1 = 1$ for $i = 1, ..., n$, $\alpha_i^2 = 1$ for $i = 1, ..., \frac{n}{2}$ and $\alpha_i^2 = -1$ for $i = \frac{n}{2} + 1, ..., n$. Observe that in this structure that β^1 and $\beta^3 \geq 0$, due to the symmetric nature of this branching strategy. Initially, these types of branches motivated this research. Call such a selection for branching constraints to be the standard branches for QHBA. The following theorem provides a quadratic number of integer extreme points.

Theorem 3.3.2. *One set of siblings generated from a standard branching constraints of QHBA from an* $x^{*LR} \in interior(\{R^n : x_i \geq 1 \;\; \text{for all } i = 1, ..., n\})$ *results in at least* $\frac{3}{2}n^2$ *basic feasible solutions that are integer for any* $n \geq 2$ *and even.*

Proof: The proof follows by selecting basic feasible solutions that generate a quadratic number of basic integer points. Since these basic feasible points satisfy the constraints at equality, the sign on the constraint is irrelevant. Furthermore all four partitions are also nonempty since the linear relaxation, x^{*LR}, is in $interior(\{R^n : x_i \geq 1 \;\; \text{for all } i = 1, ..., n\})$. Thus every corner point "near" x^{*LR} is a basic feasible solution.

Assume that a child node has the right hand side of the branching constraints equal to γ^1 and γ^2, and γ^3 if applicable. Clearly, $\gamma^1 = \beta^1$ if the inequality is a "$\leq$" and a $\beta^1 + 1$ if the inequality is a "$\geq$" constraint.

Assume $\gamma^1 + \gamma^2$ is even, then there is no C-G cut. Now assume that the first and second slack variable are nonbasic. Due to the standard branching structure and the requirement to be full row rank, every basis takes the form $\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}$ with the x values obtained by solving the following equations $\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} x = \begin{vmatrix} \gamma^1 \\ \gamma^2 \end{vmatrix}$. The solution to this basis is always $\frac{\gamma^1+\gamma^2}{2}$ and $\frac{\gamma^1-\gamma^2}{2}$, which is clearly integer since $\gamma^1 + \gamma^2$ is even. There are

$\frac{n}{2}$ choices for the first column and $\frac{n}{2}$ choices for the second column. Since two children have $\gamma^1 + \gamma^2$ even, these children provide at least $2 * (\frac{n}{2})^2$ basic feasible integer extreme points.

Now consider the case when $\gamma^1 + \gamma^2$ is odd and examine all bases where only the first slack variable is basic. Due to the standard branching structure and the requirement to be full row rank, every basis takes the form of $\begin{vmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & 0 \end{vmatrix}$ with the x values obtained by solving the following equations $\begin{vmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & 0 \end{vmatrix} x = \begin{vmatrix} \gamma^1 \\ \gamma^2 \\ \gamma^3 \end{vmatrix}$. The solution to this basis is $(\gamma^3, \gamma^3 - \gamma^2, \gamma^1 - 2\gamma^3 + \gamma^2)$. These are clearly integer feasible solutions due to the requirements to be sufficiently on the interior of $\mathbb{R}^N_+$ and is shown by Lemma 3.2.3. There are $\frac{n}{2}$ possible selections for the first such column and $\frac{n}{2}$ selections for the second column. Thus there are $\frac{n^2}{4}$ such basic feasible solutions.

The case where only the second slack is basic follows almost identical to the first. The basis is $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \end{vmatrix}$ with the solution of $(\gamma^3, \gamma^1 - \gamma^3, \gamma^1 - 2\gamma^3 + \gamma^2)$. There are $\frac{n^2}{4}$ such basic feasible solutions. Since there are always two children nodes that have $\gamma^1 + \gamma^2$ odd, there are a total of n^2 such basic feasible solutions.

Combining these two results generates at least $n^2 + \frac{n^2}{2}$ basic feasible integer solutions and the result follows.

□

The use of the C-G cuts internally removes noninteger points, and increases the number of basic feasible solutions for any set of siblings. A result of increasing the number of basic feasible solutions is that the amount of continuous space eliminated is also increased. In Figure 3.4, in two dimensions the continuous space eliminated in QBB from one iteration has volume 1. In QHBA the volume increases to 2. This effect is multiplied in higher dimensions. By establishing the distance between basic feasible solutions, the total volume of removed by one set of four children can be characterized. Formally,

Theorem 3.3.3. *If n is even and QBHA utilizes standard branches, then the combination of the four children eliminates at least $2^{\frac{n}{2}}$ volume from $\mathbb{R}^n$.*

Proof: At the node T_p with standard branches, assume $\beta^1 + \beta^2$ is odd. Clearly, $T_{p'}^{LL}$ and $T_{p'}^{GG}$ have non-redundant C-G constraints. From Lemma 3.2.3, some basic feasible solutions to these problems occur when either s_1 or s_2 are basic. Consider s^1 as basic in T_{LL}, then the basic solutions of T_{LL} in such a scenario takes the form $x_i = \lfloor \frac{\beta^1+\beta^2}{2} \rfloor$ and $x_j = \lfloor \frac{\beta^1+\beta^2}{2} \rfloor - \beta^2$ for each $i = 1, ... \frac{n}{2}$ and $j = \frac{n}{2} + 1, ..., n$. Next consider s^2 as basic in T_{GG}, then the basic solutions of T_{GG} in such a scenario take the form $x_i = \lceil \frac{\beta^1+\beta^2}{2} \rceil + 1$ and $x_j = \beta_1 - \lfloor \frac{\beta^1+\beta^2}{2} \rfloor + 1$ for each $i = 1, ... \frac{n}{2}$ and $j = \frac{n}{2} + 1, ..., n$. Observe that the distance between these points in x_i is 2 for each $i = 1, ..., \frac{n}{2}$.

Now consider the case where $\beta^1+\beta^2$ is even. When $\beta^1+\beta^2$ is even, then the right hand side of the subproblems T_{LG} and T_{GL} are odd. From Lemma 3.2.3, some basic feasible solutions to these problems occur when either s_1 or s_2 are basic. Consider s^1 as basic in T_{LG}, the basic solutions for T_{LG} in such a scenario takes the form $x_i = -\beta^1 + \beta^2 + 1$ and $x_j = \beta^2 + 1$ for each $i = 1, ...\frac{n}{2}$ and $j = \frac{n}{2} + 1, ..., n$. Next consider s^2 as basic in T_{GL}, then the basic solutions of T_{GL} in such a scenario take the form $x_i = -\beta^1 + \beta^2 - 1$ and $x_j = -\beta^2 + 1$ for each $i = 1, ...\frac{n}{2}$ and $j = \frac{n}{2} + 1, ..., n$. Observe that the distance between these points in x_i is 2 for each $i = 1, ..., \frac{n}{2}$.

Consequently, the four children eliminate at least a hyperrectangle with $\frac{n}{2}$ variables that have a distance of at least 2. From Lemma 3.2.3, every other dimension of the hyperrectangle is at least 1, due to integer extreme points. Consequently, these four children eliminate at least $2^{\frac{n}{2}}$ volume from $\mathbb{R}^n$.

$\square$

Now that QHBA is known to eliminate a substantial amount of continuous relaxation space for each branch. Naively, one may attempt to provide limit to the number of branching iterations of QHBA to $O(\frac{(u)^n}{2^{\frac{n}{2}}})$, which is better than $O(u^n)$ from BB. However, it is possible that all of the $2^{\frac{n}{2}}$ eliminated hyperrectangle from Theorem 3.3.3 may not be in the feasible region of the current node. Thus, such a bound would be incorrect as all of the volume from Theorem 3.3.3 cannot be guaranteed to be removed at each branching step. The next chapter provides computational evidence to support the theoretical benefits presented here.

Chapter 4

Computational Results for the Quaternary Hyperplane Branching Algorithm

The Quaternary Hyperplane Branching Algorithm is a new approach to BB that has shown significant theoretical benefits for the solution of IPs. This chapter contains a computational study supporting these theoretical findings. The computational results show an increased efficacy in the solution times of IPs. QHBA reduces the solution time by approximately 26.7% for the class of problems used in this research.

The computational study was done on an Intel(R) Core(TM) i7-920 running at 2.67 GHz with 3 GB of RAM. Problems were solved using ILOG CPLEX 10.0 (referred to as CPLEX); a commercially available optimization software package [12].

The problems used for this study are multidimensional knapsack problems supplied from test bank [4]. There were a total of 30 problems selected. The IPs used for this study take the form of maximize $\sum_{j=1}^{n} c_j x_j$ subject to $\sum_{j=1}^{n} a_{ij} x_j \leq b_i \; \forall \; i = 1...m$, and $x_j \in [0, 100]$ and integer. The a_{ij} are integer and were drawn from a discrete uniform generator between [0, 1000]. The right hand side b_i is calculated using a tightness ratio, λ, where $b_i = \lfloor \lambda * (\sum_{j=1}^{n} a_{ij}) \rfloor \; \forall \; i = 1...m$. The objective function coefficients (c_j) are calculated by $c_j = \sum_{i=1}^{m} \frac{a_{ij}}{m} + 500 q_j \; \forall \; j = 1...n$, where q_j is a real number taken from a continuous uniform generator between [0,1]. The 30 problems were evenly divided into 3 classes, each with a different tightness ratio. The three ratios used for the problems are $\lambda = \{.25, .5, .75\}$. Each problem has 100 variables and 30 constraints. These problems are selected because solution times were not trivial but also not prohibitively high. A ceiling of 100,000 seconds was set for the maximum amount of time alloted to solve a problem.

There are three primary solution methods used in this computational study: stand alone CPLEX, QHBA, and QBB. A basic version of warm starting was implemented for all methods. Prior to running the test problems, CPLEX was run to find a value z^*, for each test problem. A lower bound of $z^* - 1$ is then implemented on all runs.

The QHBA was coded in C to implement quaternary branching with multivariate branches for a set number of nodes. In the computational study QHBA was not implemented to termination. Instead QHBA was executed for a set number of nodes (2000 nodes), then CPLEX is called and uses traditional BB to complete the problem. This modification to QHBA is necessary due to programming constraints and memory allocation concerns. The code created in the computational study could not allocate memory as efficiently as commercially available software packages. Consequently, the memory available on the test computers would be exceeded before a solution could be achieved using only QHBA. Thus QHBA was used for 200 nodes and then traditional BB was used to solve the resulting subproblems.

Any call to CPLEX is set at default with the exception that a depth first search strategy is used to conserve memory. The default setting in CPLEX would often exhaust the available memory on the test computers before solving to optimality, so depth first search was employed so that the problems would solve. In CPLEX, depth first selects the most recently created node for evaluation.

When implementing QHBA, a specific set of branching constraints is employed. Initially the standard branching pattern was used where $\alpha_i^1 = 1$ for $i = 1, ..., n$, $\alpha_i^2 = 1$ for $i = 1, ..., \frac{n}{2}$ and $\alpha_i^2 = -1$ for $i = \frac{n}{2} + 1, ..., n$. However, this often times did not have a beneficial impact on computational time and consequently was not implemented in the final study. Other branching schemes based on the standard branches were also explored with mixed results and were not used.

The first branching constraint used in this computational study for QHBA is $\alpha_i^1 = 1$ for $i = 1, ..., n$. Since all of the α coefficients are 1 for the first branching constraint, it is possible that β^1 could be integer. Therefore if $\sum_j^n \alpha_j x_j^{*T_{p'}} \in \mathbb{Z}$, then find the first x_j where $x_j \in \mathbb{R} \backslash \mathbb{Z}$ and set $\alpha_j = 0$. This insures that the linear relaxation solution is eliminated. The second branching constraint is $\alpha_i^2 = -1$ if $\pi_i \leq -300$ and noninteger, where π_i is the reduced cost of variable x_i, else $\alpha_i^2 = 0$. Branching constraints are therefore variable from subproblem to subproblem. The first constraint was held relatively constant for each subproblem (coefficients changed only when the sum was integer), and the second constraint varied from node to node according to the reduced costs.

Branching constraint selection for QBB is as follows: for some node $T_{p'}$ find some x_i and x_j such that $x_i, x_j \in \mathbb{R}$ and $i \neq j$. This was done sequentially, so x_i was the first noninteger value in $x^{*T_{p'}}$ and x_j was the second. Set $\alpha_i^1 = 1$ and $\alpha_j^2 = 1$ and all other α^1 and α^2 values to 0. So the first branching constraint is of the form $((e_i)^T, \beta^1)$, and the second is $((e_j)^T, \beta^2)$. After 2000 nodes elapses, CPLEX solves the remaining subproblems.

The immediate comparison was between CPLEX and QHBA with multivariate branching constraints. This comparison identifies any immediate benefits between traditional BB and QHBA. The next comparison is between QHBA with multivariate branching constraints and a quaternary scheme that uses single variate constraints, QBB. It is possible that any computational benefits garnered by multivariate branches are a result of effective partitioning and not of QHBA. Therefore, this comparison is made with a method that randomly partitions the solution space with single variate branchings, using a quaternary branching scheme. If benefits are accrued from random branches, then any computational benefits could be attributed to more efficient partitioning and not necessarily to the modifications introduced by QHBA. Each problem was solved using the three solution methods, and total solution time is in seconds.

The primary result is that QHBA shows a decrease in solution time when compared to both stand alone CPLEX and QBB. When comparing QHBA to CPLEX, the average solution time reduction is approximately 26.7%, shown in Table 4.1. QHBA shows an improvement over CPLEX in 23 of the 28 problems that were solved to optimality. Of the 5 problems where no benefit was garnered from QHBA, 4 of those problems were the fastest to solve. Since these problems are the easiest to solve, implementation of QHBA was the inhibiting factor. When problems are rudimentary and solved quickly by CPLEX, implementation of QHBA to a high number of nodes probably only serves to increase the computational time. Since QHBA increases the number of subproblems necessary to reach a solution (a greater number of nodes) and increase the complexity of these problems due to the added branching constraints, it is easy to see why QHBA loses on fast problems. Therefore, by inhibiting it is not necessarily that QHBA causes the impediment, but instead the elementary code of this version of QHBA increases the overall processing time.

Note that if QHBA is implemented with a node threshold of 500 (instead of 2000), then QHBA is again faster than stand alone CPLEX on problems 4, 9 and 21. For problems with a low CPLEX solution time, it is still possible to accrue a benefit using lower number of nodes. Therefore, as problems become more complex there should be more improvement from QHBA implementation. This benefit is also seen in larger problems.

Another knapsack problem from the same test bank was used in a pilot run, except this problem was 250 variables and 30 constraints (tightness constraint of .25). For the same 2000 node threshold QHBA solved the problem in 335,927 seconds, and CPLEX solved in 377,775 seconds. This is a decrease in the solution time by 41,848 seconds (approximately 11.5 hours), or 11.8%. We believe that in this instance increasing the node threshold would further reduce overall solution time.

The average preprocessing time while using QHBA was negligible to the overall solution time. The average time using the programmed QHBA code was approximately 17.5 seconds and is not reported in the table because the processing time was relatively constant across all problems. This indicates that a large benefit can be procured from very little preprocessing. The fact that 17 seconds of preprocessing can result in up to a 55% reduction in solution time indicates that QHBA at the very least can be used as a preprocessing step for commercial IP solvers.

A second comparison was done between QHBA and QBB. The data for this comparison is presented in Table 4.2. This comparison shows that QHBA decreases solution time when compared to QBB by approximately 5%. In 25 of the 28 problems QHBA produced a benefit. Problem 23 appears to be an anomaly in the overall pattern of behavior for the test methods as QBB solved 10,396 seconds faster than QHBA. When removed from the test data, the benefit of QHBA increases to approximately 14.3%. The exact reason for this anomalous reading is unknown, however QBB does reciprocally poorly for other problems. In the aforementioned test problem using 250 variables and 30 constraints, QBB took 691,674 seconds (an increase of 355,747 seconds, approximately 4 days). This result demonstrates the importance of hyperplane branches, and the effectiveness of the internal cutting planes as an improvement method when multivariable branching constraints are employed and the C-G cuts are derived.

The final comparison is between QBB and CPLEX. In this comparison QBB was beneficial in only 15 of the 28 problems, as shown in Table 4.3. However QBB was over 20% faster and thus a quaternary branching structure may be more beneficial than a binary branching structure demonstrated in traditional BB. Since QHBA shows a benefit in a majority of the problems when compared to CPLEX, QBB's ineffectiveness in approximately half of the test problems is also evidence to the effectiveness of multivariable branching constraints and internal cutting planes.

We believe that a primary reason for these computational advancements is that the use of internal C-G cuts significantly increase the number of basic feasible integer points in child subproblems. In traditional CPLEX, the number of infeasible integer points in unfathomed nodes was at maximum the number of constraints. However, in QHBA the number of integer infeasible points per node was significantly increased in QHBA. This is due to the C-G cuts. Consequently, it is possible the warm starting procedure reduces the magnitude of the benefit seen. If there were no lower integer bound set, then it is possible that in the early iterations of QHBA bounding would happen in fewer iterations. More specifically, that integer solutions would be found faster.

Observe that while QHBA showed a benefit in comparison to QBB, that benefit was not as large as when compared to CPLEX. This suggests that there is some advantage to more advanced branching structures. However, when comparing QBB to CPLEX there is not a clear benefit to implementation of a quaternary branching scheme. QHBA's superiority to QBB also further demonstrates the necessity of choosing strong branching constraints, and the efficacy of internal cutting planes.

The theoretical and computational results from this research strongly suggest that QHBA is a valuable tool for solving IPs. It must be noted that this is an elementary version of QHBA, and there is much room for potential advancements that can even further strengthen the algorithm and improve its computational depth. There are many research areas that can be further addressed in future research for hyperplane branching, and some of these research topics are further addressed in Chapter 5.

		QHBA	CPLEX		
ProblemNo.	λ	*Time*	*Time*	*Improvement*	*%Difference*
1	.25	829	1072	Yes	25.6
2	.25	7615	12445	Yes	48.2
3	.25	1241	1415	Yes	13.1
4	.25	304	216	No	-33.8
5	.25	244	87	No	-94.9
6	.25	856	947	Yes	10.1
7	.25	1362	1586	Yes	15.2
8	.25	1799	2489	Yes	32.2
9	.25	601	530	No	-12.6
10	.25	1838	2295	Yes	22.1
Total	**.25**	**16689.0**	**23082.0**	**7/10**	**32.1**
11	.50	104	20	No	-135.5
12	.50	3851	6500	Yes	51.2
13	.50	1255	1538	Yes	20.3
14	.50	585	995	Yes	51.9
15	.50	653	741	Yes	12.6
16	.50	13492	18904	Yes	33.4
17	.50	521	529	Yes	1.5
18	.50	12943	19094	Yes	38.4
19	.50	2078	3089	Yes	39.1
20	.50	1187	1790	Yes	40.5
Total	**.50**	**36669.0**	**53200.0**	**9/10**	**36.8**
21	.75	8514	7149	No	-17.4
22	.75	4425	5429	Yes	20.4
23	.75	42965	47080	Yes	9.1
24	.75	-	-	-	-
25	.75	4126	7052	Yes	52.4
26	.75	1139	1521	Yes	28.7
27	.75	7505	12113	Yes	47.0
28	.75	-	-	-	-
29	.75	2621	4632	Yes	55.5
30	.75	35353	48107	Yes	30.6
Total	**.75**	**106648.0**	**133083.0**	**7/8**	**22.1**
Cumulative Total		**160006.0**	**209365.0**	**23/28**	**26.7**

Table 4.1: Data Reported for QHBA and CPLEX

		QHBA	QBB		
ProblemNo.	λ	*Time*	*Time*	*Improvement*	*%Difference*
1	.25	829	1738	Yes	70.8
2	.25	7615	9314	Yes	20.1
3	.25	1241	1543	Yes	21.7
4	.25	304	353	Yes	14.9
5	.25	244	276	Yes	12.3
6	.25	856	1007	Yes	16.2
7	.25	1362	1451	Yes	6.3
8	.25	1799	2316	Yes	25.1
9	.25	601	670	Yes	10.9
10	.25	1838	2438	Yes	28.1
Total	**.25**	**16689.0**	**21106.0**	**10/10**	**23.4**
11	.50	104	120	Yes	14.3
12	.50	3851	4386	Yes	13.0
13	.50	1255	1593	Yes	23.7
14	.50	585	748	Yes	24.5
15	.50	653	739	Yes	12.4
16	.50	13492	16125	Yes	17.8
17	.50	521	620	Yes	17.4
18	.50	12943	15690	Yes	19.2
19	.50	2078	2261	Yes	8.4
20	.50	1187	1552	Yes	26.7
Total	**.50**	**36669.0**	**43834.0**	**10/10**	**17.8**
21	.75	8514	8473	No	-0.5
22	.75	4425	5956	Yes	29.5
23	.75	42965	32569	No	-27.5
24	.75	-	-	-	-
25	.75	4126	5781	Yes	33.4
26	.75	1139	1619	Yes	34.8
27	.75	7505	10371	Yes	32.1
28	.75	-	-	-	-
29	.75	2621	3836	Yes	37.6
30	.75	35353	34083	No	-3.7
Total	**.75**	**106648.0**	**102668.0**	**5/8**	**-3.8**
Cumulative Total		**160006.0**	**167628.0**	**25/28**	**4.7**

Table 4.2: Data Reported for QHBA and QBB

		QBB	CPLEX		
ProblemNo.	λ	*Time*	*Time*	*Improvement*	*%Difference*
1	.25	1738	1072	No	-47.4
2	.25	9314	12445	Yes	28.8
3	.25	1543	1415	No	-8.7
4	.25	353	216	No	-48.2
5	.25	276	87	No	-104.1
6	.25	1007	947	No	-6.1
7	.25	1451	1586	Yes	8.9
8	.25	2316	2489	Yes	7.2
9	.25	670	530	No	-23.3
10	.25	2438	2295	No	-6.0
Total	**.25**	**21106.0**	**23082.0**	**3/10**	**8.9**
11	.50	120	20	No	-142.9
12	.50	4386	6500	Yes	38.8
13	.50	1593	1538	No	-3.5
14	.50	748	995	Yes	28.3
15	.50	739	741	Yes	0.3
16	.50	16125	18904	Yes	15.9
17	.50	620	529	No	-15.8
18	.50	15690	19094	Yes	19.6
19	.50	2261	3089	Yes	31.0
20	.50	1552	1790	Yes	14.2
Total	**.50**	**43834.0**	**53200.0**	**7/10**	**19.3**
21	.75	8473	7149	No	-17.0
22	.75	5956	5429	No	-9.3
23	.75	32569	47080	Yes	36.4
24	.75	-	-	-	-
25	.75	5781	7052	Yes	19.8
26	.75	1619	1521	No	-6.2
27	.75	10371	12113	Yes	15.5
28	.75	-	-	-	-
29	.75	3836	4632	Yes	18.8
30	.75	34083	48107	Yes	34.1
Total	**.75**	**102688.0**	**133083.0**	**5/8**	**25.8**
Cumulative Total		**167628.0**	**209365.0**	**15/28**	**22.1**

Table 4.3: Data Reported for QBB and CPLEX

Chapter 5

Conclusion

This thesis introduced a modified version of the Branch and Bound Algorithm called the Quaternary Hyperplane Branching Algorithm. The primary advancements of QHBA are the use of multivariate branches (hyperplane branches), a quaternary branching scheme, and the creation of internal C-G cutting planes. More specifically, by using the multivariate branches in a quaternary branching tree C-G cuts can be induced from those branching constraints. QHBA has a variety of theoretical and computational benefits.

Theoretically, QHBA induces a quadratic number of integer extreme points per branch, which is far more than the nine created by classic BB in a similar setting. Additionally, one set of QHBA branches can eliminate $2^{\frac{n}{2}}$ volume of continuous relaxation area as compared to 1 for QBB. Besides these advancements, a novel proof to finite termination is presented for QHBA and BB.

Computationally, implementation of QHBA on general bounded IPs has shown a

26.7% decrease in the total solution time in certain problems over a commercial IP solver. These improvements demonstrate the significant advantages of QHBA, and suggest that QHBA would be a valuable inclusion to commercial IP solver software packages.

5.1 Future Research

The work presented in this thesis opens the door to a host of new research topics. Thus, there is abundant future work that can and should be pursued to improve or extend the results in this thesis.

For a majority of the theoretical arguments the standard branching scheme was considered. However, during the computational study it was discovered that the standard branching scheme does not necessarily result in the most efficient branches. Selection of the α values for the branching constraints is an important subproblem in the implementation of QHBA, and further work needs to be done for branching constraint selection.

Additionally, the quaternary branching structure employed was shown to increase the number of integer extrema in each subproblem. Higher orders of branching should be explored to see if this phenomena is pervasive, and if the number of integer extrema also increases with more branches per iteration. In other words, are 3 branching constraints with 8 resulting child nodes better than QHBA?

The implementation of cutting planes should also be further investigated. Cutting planes were induced in QHBA for the two branching constraints when necessary to eliminate potential non-integer points that could prevent QHBA from terminating. However, these non-integer sticking points do not only occur between the branching constraints. Often times these points occur between a branching constraint and one of the original problem constraints. Further work could be done to see if it is possible to implement more cutting planes per subproblem to decrease the volume of the continuous relaxation space.

Bibliography

[1] Achterberg, T., Koch, T. and Martin, A. (2006). "MIPLIB 2003," *Operations Research Letters*, **34** (4), 361-372.

[2] Agin, N. (1966). "Optimum seeking with branch and bound," *Management Science*, **13** (4), B176-B185.

[3] Borndörfer, R., Grötschel, M. and Löbel, A. (2003). "Duty scheduling in public transport," *Mathematics - Key Technology for the Future*, W. Jger, H. Krebs eds., 653-674.

[4] Chu, P. and Beasley, J. (1998). "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, **4**, 63-86.

[5] Chvátal, V. (1973). "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics*, **4**, 305-337.

[6] Cramer, G. (1750). *Introduction à l'analyse des lignes courbes*, Geneva.

[7] Land, A. and Doig, A. (1960). "An automatic method for solving discrete programming problems," *Econometrica*, **28**, 497-520.

[8] Fishchetti, M. and Lodi, A. (2003). "Local branching," *Mathematical Programming*, **98** (1-3), 23-47.

[9] Ryan, B. and Foster, D. (1981). "An integer programming approach to scheduling," *Computer Scheduling of Public Transport*, A. Wren eds., North-Holland Publishing Company, 269-280.

[10] Gomory, R. (1965). "On the relation between integer and non-integer solutions to linear programs,"*Proceedings of the National Academy of Science*, **53**, 260-265.

[11] Hamilton, W. and Moses, M. (1973). "An optimization model for corporate financial planning," *Operations Research*, **21** (3), 677-692.

[12] IBM ILOG CPLEX Inc. "Using CPLEX callable library," (2006). Version 10.0.

[13] Jörnsten, K.O., and Värbrand, P,. (1991). "A hybrid algorithm for the generalized assignment problem *Optimization*," **22** (2), 273-282.

[14] Karamanov, M. and Cornuéjols, G. (2009). "Branching on general disjunctions," *Mathematical Programming*", Series A.

[15] Karp, R. (1972). Reducibility among combinatorial problems, *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York **85** (103), 497-520.

[16] Khachiyan, L. (1979). "A polynomial time algorithm in linear programming," *Soviet Math. Dokl.*, **20**, 191-194.

[17] Mehrotra, and S. Li, Z. (2010). “Branching on hyperplane methods for mixed linear and convex programming using adjoint lattices,” *Journal of Global Optimization.*

[18] Padberg, M. and Rinaldi, G. (1991). “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems,” *SIAM Review*, **33** (1), 60-100.

[19] Pendharkar, P. and Rodger, J. (2006). “Information technology capital budgeting using a knapsack problem”*International Transactions in Operations Research*, **13** , 333-351.

[20] Lee, E. Fox, T. and Crocker, I. (2003). “Integer programming applied to intensity-modulated radiation therapy treatment planning,” *Annals of Operations Research*, **119** , 165-181.

[21] Lenstra, H. (1983). “Integer programming with a fixed number of variables,” *Mathematics of Operations Research*, **8** (4) , 538-548.

[22] Nemhauser, G. and Wolsey, L. (1988). *Integer and combinatorial optimization*, John Wiley and Sons, New York.

[23] Owen, J. and Mehrotra, S. (2001). “Experimental results using general disjunctions in branch-and-bound for general-integer linear programs,” *Computational Optimization and Applications*, **20** (2), 159-170.

[24] Stahl, J. Kong, N. Shechter, S. Schaefer, A. and Roberts, S. (2005). "A methodological framework for optimally reorganizing liver transplant regions," *Medical Decision Making*, **25** (35) , 35-46.

[25] Rychel, D. (1977) "Capital budgeting with mixed integer linear programming: an application," *Financial Management*, **6** (4), 11-19.

[26] Subramanian, R. Scheff, R. Quillinan, J. Wiper, D. and Marsten, R. (1994). "Coldstart: fleet assignment at delta airlines," *Interfaces*, **24** (1) , 104-120.

[27] Walker, R. J. (1960). 'An enumerative technique for a class of combinatorial problems," *Proceedings of Symposia in Applied Mathematics*, **Math 10**, 91-94.

[28] Weingartner, M. (1966). "Capital budgeting of interrelated projects: survey and synthesis," *Management Science*, **12** (7), Series A, Sciences, 485-516.

[29] Taylor, P. and Huxley, S. (1989). "A break from tradition for the San Francisco police: patrol officer scheduling using an optimization-based decision support system," *Interfaces*, **19** (1), 4-24.