LOWER BOUNDS FOR INTEGER PROGRAMMING PROBLEMS

A Thesis
Presented to
The Academic Faculty

by

Yaxian Li

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
August 2013

LOWER BOUNDS FOR INTEGER PROGRAMMING PROBLEMS

Approved by:

George L. Nemhauser, Ozlem Ergun,
Committee Chair
School of Industrial and Systems
Engineering
Georgia Institute of Technology

George L. Nemhauser, Ozlem Ergun,
Advisor
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Martin Savelsbergh
School of Mathematical and Physical
Sciences
University of Newcastle

Ellis Johnson
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Maria-Florina Balcan
School of Computer Science
Georgia Institute of Technology

Date Approved: May 9th 2013

To my beloved grandfathers.

ACKNOWLEDGEMENTS

I would like to give my deepest gratitude to my advisors, Prof. George L. Nemhauser and Prof. Ozlem Ergun. This work could never be done without their generous support, invaluable guidance, and greatest patience with me. I am honored to work with both of them and have learned a great deal from them. I truly thank them for having belief in me which made me have more faith in myself.

I would also like to thank Prof. Savelsbergh for his guidance in my early stage of research. Working with him was truly an enjoyable and inspiring experience. It is an honor to have Prof. Johnson on my committee, who has given me advice not limited to this thesis. I also want to thank Prof. Balcan for her encouragement and for exposing me a new direction in my research.

I want to thank my friends and colleagues at ISYE, Huizhu, Qie, Linkan, Pengyi, Luyi, Kale, Feng, Juan Pablo, Fatma, who have made my days here happy and fulfilled.

I cannot think of finishing the work without the encouragement and support from my parents. Their unconditional love has always been the biggest support for me. Finally, my most sincere thanks go to my husband, Nan, who has always been there for me. His encouragement, suggestions and love has made me who I am today, thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Solving real world problems with mixed integer programming (MIP) involves efforts in modeling and efficient algorithms. To solve a minimization MIP problem, a lower bound is needed in a branch-and-bound algorithm to evaluate the quality of a feasible solution and to improve the efficiency of the algorithm. This thesis develops a new MIP model and studies algorithms for obtaining lower bounds for MIP.

The first part of the thesis is dedicated to a new production planning model with pricing decisions. To increase profit, a company can use pricing to influence its demand to increase revenue, decrease cost, or both. We present a model that uses pricing discounts to increase production and delivery flexibility, which helps to decrease costs. Although the revenue can be hurt by introducing pricing discounts, the total profit can be increased by properly choosing the discounts and production and delivery decisions. We further explore the idea with variations of the model and present the advantages of using flexibility to increase profit.

The second part of the thesis focuses on solving integer programming(IP) problems by improving lower bounds. Specifically, we consider obtaining lower bounds for the multi-dimensional knapsack problem (MKP). Because MKP lacks special structures, it allows us to consider general methods for obtaining lower bounds for IP, which includes various relaxation algorithms. A problem relaxation is achieved by either enlarging the feasible region, or decreasing the value of the objective function on the feasible region. In addition, dual algorithms can also be used to obtain lower bounds, which work directly on solving the dual problems.

We first present some characteristics of the value function of MKP and extend some properties from the knapsack problem to MKP. The properties of MKP allow some large scale problems to be reduced to smaller ones. In addition, the quality of corner relaxation bounds of MKP is considered. We explore conditions under which the corner relaxation is tight for MKP, such that relaxing some of the constraints does not affect the quality of the

lower bounds. To evaluate the overall tightness of the corner relaxation, we also show the worst-case gap of the corner relaxation for MKP.

To identify parameters that contribute the most to the hardness of MKP and further evaluate the quality of lower bounds obtained from various algorithms, we analyze the characteristics that impact the hardness of MKP with a series of computational tests and establish a testbed of instances for computational experiments in the thesis.

Next, we examine the lower bounds obtained from various relaxation algorithms computationally. We study methods of choosing constraints for relaxations that produce high-quality lower bounds. We use information obtained from linear relaxations to choose constraints to relax. However, for many hard instances, choosing the right constraints can be challenging, due to the inaccuracy of the LP information. We thus develop a dual heuristic algorithm that explores various constraints to be used in relaxations in the Branch-and-Bound algorithm. The algorithm uses lower bounds obtained from surrogate relaxations to improve the LP bounds, where the relaxed constraints may vary for different nodes. We also examine adaptively controlling the parameters of the algorithm to improve the performance.

Finally, the thesis presents two problem-specific algorithms to obtain lower bounds for MKP: A subadditive lifting method is developed to construct subadditive dual solutions, which always provide valid lower bounds. In addition, since MKP can be reformulated as a shortest path problem, we present a shortest path algorithm that uses estimated distances by solving relaxations problems. The recursive structure of the graph is used to accelerate the algorithm. Computational results of the shortest path algorithm are given on the testbed instances.

CHAPTER I

INTRODUCTION

The study of mixed integer programming (MIP) involves developing models to describe economic and industrial problems and providing algorithms to solve them. Since the 1950s, MIP has become an important modeling tool to solve real-life problems. At the same time, extensive efforts, including developing many general-purpose strategies and problem-specific algorithms, have been devoted to efficiently solving MIP problems. This thesis addresses two topics in modeling and algorithm design by MIPs. We first develop a new variation of the classical lot-sizing model, in which optimal decisions on production and delivery can be made to maximize profit. We then investigate methods to solve MIP problems efficiently by improving lower bounds. Specifically, we focus on solving the multi-dimensional knapsack problems (MKP) with a large number of constraints, as they represent a class of hard MIP problems and have important applications.

The first part of the thesis is dedicated to a new lot-sizing model with pricing decisions and production and delivery flexibility. A lot-sizing model decides when and how much to produce and deliver in order to satisfy a set of demands while achieving an optimal profit. The classical uncapacitated single-item single-level lot-sizing model, introduced by Wagner and Whitin [62], can be stated as:

$$\begin{aligned}
\max \ & \sum_{i=1}^{n}(p_i D_i - e_i z_i - h_i s_i - g_i x_i) \\
\text{s.t.} \quad & x_1 = s_1 + D_1 \\
& x_i + s_{i-1} = s_i + D_i, \ i = 2, ..., n \\
& x_i \leq D_{in} z_i, \ i = 1, ..., n \\
& s_n = 0 \\
& x, s \in \mathbb{R}^n_+, z \in \{0,1\}^n.
\end{aligned}$$

For finite time periods $\{1, ..., n\}$ with demand D_i in each period, let x_i, s_i and z_i be decision

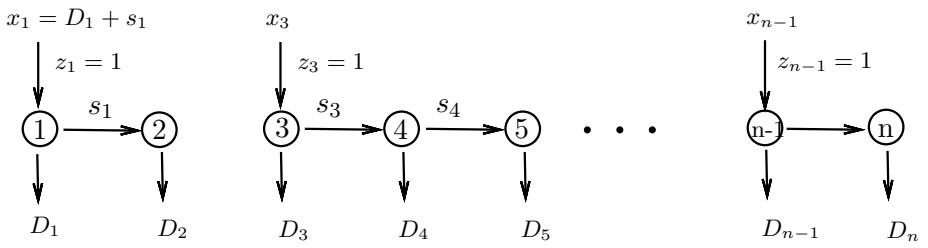


Figure 1: Lot-sizing model

variables on production amount, inventory level, and whether or not production takes place at time i respectively. With unit production price p_i, fixed operation cost e_i, unit inventory holding cost h_i and unit production cost g_i for each period i, the objective is to maximize the total profit while requiring the production to satisfy the flow conservation, see Figure 1.

Numerous variations of the classical lot-sizing model have been developed and widely used to capture different production planning and supply chain problems. For a comprehensive study of these models, we refer to Pochet and Wolsey [56]. While the profit of a company can be boosted by either increasing revenue or decreasing cost, previous literature mainly focuses on adjusting prices to increase revenue. In contrast, we investigate adjusting prices to reduce cost. Cost reduction can be realized by using production and delivery flexibility in forms of production time windows as discussed in Dauzere-Peres et al. [15], or demand time windows as in Lee et al. [48]. In particular, we present a new variation of the uncapacitated lot-sizing model, and consider offering price discounts in return for production and delivery flexibility. We show that even though the resulting optimization problem has a nonlinear objective function, it can still be solved in polynomial time. As a special case, we analyze the setting where there is a tradeoff between the decrease in production and inventory costs and decrease in revenue due to the loss from price discounts. In addition, we consider several generalizations of the proposed model to further extend the practical use of production and delivery flexibility. Finally, we perform a computational study to analyze the benefits of offering price discounts in return for production and delivery flexibility in various settings. The empirical results show that by properly choosing

the price discounts, cost can be significantly reduced and profit can be increased by using production and delivery flexibility.

The second part of the thesis studies improving lower bounds to efficiently solve linear MIP problems, which can be stated as:

$$\begin{aligned} z_{MIP} = \min\ & cx + dy \\ \text{s.t. } & Ax + By \geq b \\ & x \in \mathbb{Z}^n_+, y \in \mathbb{R}^k_+, \end{aligned} \tag{1}$$

where x is an $n-$dimensional vector of integer decision variables and y is a $k-$dimensional vector of continuous decision variables. A MIP instance is specified by data (c, d, A, B, b), where $c \in \mathbb{R}^n$, $d \in \mathbb{R}^k$ are the objective coefficients defined by vectors of real numbers, $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times k}$ are constraint matrices, and $b \in \mathbb{R}^m$ is the right-hand side of the constraints. Denote S to be the set of solutions to problem (1), which is called the *feasible region*. A solution to (1) is called a *feasible solution*, and a feasible solution that minimizes the objective function is called an *optimal solution*. If $S = \emptyset$, we call problem (1) infeasible, otherwise it is feasible. A valid *upper bound* and *lower bound* for problem (1) are values $\bar{z}$ and $\underline{z}$ that $\underline{z} \leq z_{MIP} \leq \bar{z}$. A *relative gap* is defined by $|\bar{z} - \underline{z}|/(|\bar{z}| + \epsilon)$, which is the relative difference between upper and lower bounds and $\epsilon > 0$ is a small real number. The linear programming (LP) relaxation of problem (1) is obtained by relaxing the integrality constraints on integer variables x.

MIP is NP-hard. General methods to solve a MIP problem include branch-and-bound and cutting plane algorithms. Branch-and-bound, introduced by Land and Doig [47], is a divide-and-conquer type algorithm, which builds a tree with each node corresponding to a subregion of S. The goal is to divide the feasible region into smaller pieces that can be relatively easy to solve. At each node of the tree, an LP relaxation is solved with three possible outcomes: Either the node is pruned if the LP relaxation is infeasible or the objective value is larger than the current upper bound, implying that no feasible or optimal solution can be found in the region; or the node is "solved" if the LP relaxation has an integral solution with a smaller objective value than the current upper bound, then the upper bound can be decreased and an optimal solution in the region is found; or the LP

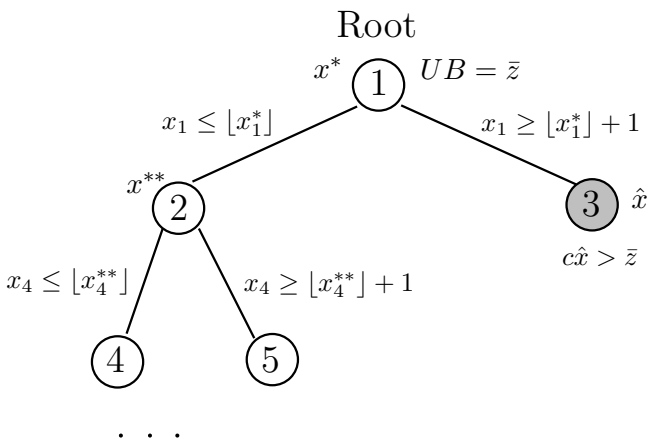


Figure 2: Branch and bound algorithm

relaxation has a fractional solution, which means that at least one of the integer variables has a fractional value, thus two new nodes will be created by adding linear constraints. The linear constraints can be bound constraints for an integer variable with fractional value, and this is called branching on the variable, see Figure 2. When several variables have fractional values, we use variable selection strategies to choose variables to branch on. Complex variable selection rules include strong branching by Applegate, Bixby, Chvatal and Cook [2] and Linderoth and Savelsbergh [49], and pseudo-cost branching by Benichou et al. [7]. The strategies that determine the order of nodes to be processed are called node selection strategies. These include best-bound first and depth first. An efficient branch-and-bound algorithm requires effective strategies for variable and node selections. A survey of different strategies for branch-and-bound algorithms is given in Lodi [51].

Based on a different approach, cutting plane algorithms iteratively add valid inequalities and solve LP relaxations. An inequality $fx + hy \leq g$ is called a *valid inequality* for S if it is satisfied by all feasible solutions. Suppose (x^*, y^*) is a fractional optimal solution to the LP relaxation of (1), and $fx + hy \leq g$ is a valid inequality. If $fx^* + hy^* > g$, we call the valid inequality a *cutting plane* that separates (x^*, y^*) from S, see Figure 3. The essential idea here is to add linear inequalities, which are cutting planes that are generated by solving separation problems, to better approximate the convex hull of S. The performance of cutting plane algorithms depends on the strength of the inequalities they add at each iteration, and

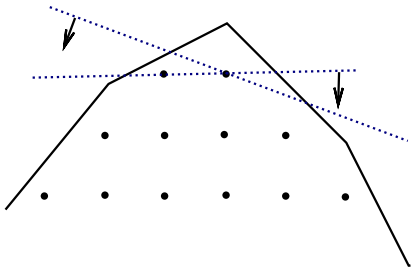


Figure 3: Cutting plane algorithm

there is a large literature on generating different types of cuts on general and specific MIP problems. The best-known cuts include Chvatal-Gomory cuts [13], Gomory fractional cuts [31], Gomory mixed integer cuts [30], disjunctive cuts by Balas [4], and lift-and-project cuts by Balas, Ceria and Cornuéjols [5].

The combination of branch-and-bound and cutting plane algorithms leads to branch-and-cut algorithms. In such algorithms, the tree structure from the branch-and-bound algorithm is preserved. In particular, at some nodes, after the LP relaxation is solved, cutting planes are generated and added to the problem. The LP relaxation problem is then resolved. This procedure may be repeated until no additional cutting planes can be found. By adding cutting planes, the LP relaxations are tightened, better lower bounds can be obtained, and the number of nodes thus can be reduced by using the pruning mechanism. Branch-and-cut algorithms now serve as the foundation of state-of-the-art MIP solvers, and have the ability to solve a variety of difficult MIP problems efficiently. For more information on this topic, see Nemhauser and Wolsey [54]. From now on, we restrict our discussion within the framework of branch-and-cut algorithms.

A MIP problem is solved if there is solution for which the relative gap is zero. Therefore, efficient strategies for obtaining high-quality upper and lower bounds are important in improving the overall performance of the algorithms. In order to obtain a good upper bound, many primal heuristics for obtaining "good" feasible solutions have been developed. Some use neighborhood search, also referred to as local search, to iteratively improve the current incumbent solution. Various neighborhood search techniques are discussed in Aarts and Lenstra [1]. There are also heuristics using diving techniques, which involve iterative

rounding and variable fixing to construct feasible solutions from LP relaxation solutions. Among the most successful primal heuristics for general MIP's are the feasibility pump developed by Fischetti, Glover and Lodi [18], local branching by Fischetti and Lodi [19], and RINS by Dana, Rothberg and Pape [14]. These heuristics have proven to be very effective and widely applied within state-of-the-art MIP solvers.

On the other hand, the only method that is used within MIP solvers to obtain lower bounds is based on solving LP relaxations possibly tightened by the adding of cutting planes. In general, lower bounds for MIP problems are obtained by duality or relaxations. The advantage of using duality is that any feasible solution to a dual problem provides a valid lower bound for z_{MIP}, however, only an optimal solution to a relaxation problem guarantees a valid lower bound.

A weak dual of problem (1) is an maximization problem defined as:

$$z_D = \max\{z_D(u) : u \in S_D\},$$

where $z_D(u) \leq cx + dy$, for all $u \in S_D$ and $(x, y) \in S$. By definition, z_D is a lower bound of z_{MIP}. A strong dual of problem (1) is a weak dual with $z_D = z_{MIP}$ if problem (1) is feasible and z_{MIP} is bounded. Therefore, tight lower bound can be obtained by solving a strong dual problem. The most studied class of strong dual problems are subadditive dual problems, which were introduced by Gomory [32] and further studied by Johnson [39]. Specifically, a function $f(\cdot) : C \to \mathbb{R}$ is *subadditive* if it satisfies

$$f(x + y) \leq f(x) + f(y), \text{ for } x, y, x + y \in C.$$

A subadditive dual problem of problem (1) is defined as

$$\begin{aligned} \max\ & F(b) \\ \text{s.t. } & F(A_j) \leq c_j, j = 1, ..., n \\ & F(B_i) \leq d_i, i = 1, ..., k \\ & F(0) = 0 \\ & F : \mathbb{R}^m_+ \to \mathbb{R} \text{ is nondecreasing and subadditive,} \end{aligned} \tag{2}$$

where A_j, $j = 1, ..., n$ and B_i, $i = 1, ..., k$ are columns of the constraint matrix. A function $F : \mathbb{R}^m_+ \to \mathbb{R}$ that is feasible to problem (2) is called a subadditive dual function. Constructing subadditive dual functions has received some attention, however, due to the lack of framework for implementation, subadditive duality is not often used to obtain lower bounds in practical computation. We refer to Guzelsoy [33] for a detailed review of topics on MIP dual problems.

A problem relaxation is achieved by either enlarging the feasible region, or decreasing the value of the objective function on the feasible region or both. The most well-studied and widely-used relaxation is the LP relaxation, mostly because the formulation only involves linear constraints, and can be solved efficiently. There are also many other ways to relax the inequality and equality constraints in a MIP problem. Partition the index set of the constraints in problem (1) into two sets S_1 and S_2. Suppose the constraints in S_1 can be easily handled, so they will remain in the relaxation problems, and are called active constraints. The constraints in S_2 are complicating constraints that will be relaxed, and are called inactive constraints. Let A_{S_i} and B_{S_i} be the submatrices of constraint matrices A and B, respectively, with rows in set S_i; b_{S_i} be the subvector of b with components in set S_i, for $i = 1, 2$.

A simple constraint relaxation algorithm drops a set of constraints, and solves the following relaxation problem:

$$\begin{aligned} &\min\ cx + dy \\ &\text{s.t. } A_{S_1}x + B_{S_1}y \geq b_{S_1} \\ &x \in \mathbb{Z}^n_+, y \in \mathbb{R}^k_+. \end{aligned}$$

Lagrangian relaxation algorithms, introduced by Held and Karp [35, 36] and generalized by Shapiro [59], relaxes constraints by dualizing them. Lagrangian relaxation can be formulated as:

$$\begin{aligned} z_{LD}(\lambda) = \min\ & cx + dy + \lambda(b_{S_2} - A_{S_2}x - B_{S_2}y) \\ \text{s.t. } & A_{S_1}x + B_{S_1}y \geq b_{S_1} \\ & x \in \mathbb{Z}^n_+, y \in \mathbb{R}^k_+, \end{aligned} \tag{3}$$

where $\lambda \in \mathbb{R}_+^{|S_2|}$ are called *lagrangian multipliers*. Since any feasible solution for problem (1) is also feasible for problem (3), and corresponds to a smaller objective value, $z_{LD}(\lambda)$ is a valid lower bound of z_{MIP} for any $\lambda \in \mathbb{R}_+^{|S_2|}$. Therefore, by taking the best such lower bound over all $\lambda \in \mathbb{R}_+^{|S_2|}$, $z_{LR} = \max_{\lambda \in \mathbb{R}_+^{|S_2|}} z_{LD}(\lambda)$ is a valid lower bound of z_{MIP}. The goal of Lagrangian relaxation algorithms is to iteratively solve (3) and search for λ such that z_{LR} can be obtained. We refer to Fisher [20] for a detailed survey on Lagrangian relaxation algorithms.

Surrogate relaxation algorithms, introduced by Glover [28], replaces a set of constraints by an aggregated constraint. A generic formulation can be stated as

$$\begin{aligned} z_{SD}(\lambda) = \min\ & cx + dy \\ \text{s.t. } & A_{S_1}x + B_{S_1}y \geq b_{S_1} \\ & \lambda(A_{S_2}x + B_{S_2}y) \geq \lambda b_{S_2} \\ & x \in \mathbb{Z}_+^n, y \in \mathbb{R}_+^k, \end{aligned} \tag{4}$$

where $\lambda \in \mathbb{R}_+^{|S_2|}$ are called *surrogate multipliers*. Again, since any feasible solution for problem (1) is also feasible for problem (4), $z_{SD}(\lambda)$ is a valid lower bound of z_{MIP} for any $\lambda \in \mathbb{R}_+^{|S_2|}$. Therefore, surrogate relaxation algorithms iteratively solve (4) and search for the $\lambda \in \mathbb{R}_+^{|S_2|}$ that maximizes $z_{SD}(\lambda)$. We refer to Glover [28] for a study on surrogate relaxation algorithms.

In addition to the general inequality and equality constraints, relaxations of non-negativity constraints on integer variables can also be considered. Group relaxation, introduced by Gomory [32], relaxes the non-negativity constraints on basic variables in the optimal LP relaxation solution. For simplicity, we demonstrate the group relaxation on an *integer programming* (IP) problem

$$\begin{aligned} \min\ & cx \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{5}$$

where decision variables are all integer. Given an optimal basis B for the LP relaxation of problem (5), let N be the index set of nonbasic variables. The group relaxation is obtained

by relaxing the non-negativity constraints of all basic variables, which can be stated as

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } Ax = b \\ &\quad x_N \geq 0, x \in \mathbb{Z}^n. \end{aligned}$$

Related to group relaxation, corner relaxation, sometimes referred to as strict group relaxation, is obtained by relaxing the the non-negativity constraints on all variables that have positive LP relaxation values. Let x^* be the optimal LP relaxation solution, then the corner relaxation can be stated as

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } Ax = b \\ &\quad x_I \geq 0, x \in \mathbb{Z}^n, \end{aligned}$$

where $I = \{i : x_i^* = 0\}$. Therefore, if x^* is non-degenerate, corner relaxation is the same as group relaxation. Group and corner relaxations can be used to generate cuts for MIP problems. Richard and Dey [57] conducted a detailed survey on the group-theoretic approach of MIP.

Although both adding cutting planes and developing relaxation algorithms have been used to improve lower bounds computationally, the methods are based on different philosophies. Adding cutting planes tightens the problem formulation, but in doing so increases the size of the problem and can make the LP relaxations more difficult to solve. In contrast, relaxation algorithms relax the complicating constraints with the goal of rapidly solving one or a sequence of relaxed problems which are computationally easier to handle. For example, Lagrangian relaxation algorithms have been proved to be very effective in obtaining lower bounds for large network flow and assignment problems with side constraints, since the combinatorial properties of the active constraints can be used to rapidly solve the Lagriangian relaxations. However, for many problems that do not have nice constraint structure, the study of relaxation algorithms is deficient with open questions and computational possibilities. Therefore, in this thesis, we investigate obtaining lower bounds by

using relaxation algorithms on MIPs with a large number of constraints. In particular, we focus on multi-dimensional knapsack problems (MKP) with unbounded integer variables.

The classical packing problem is defined as

$$\begin{aligned} \max \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \ i = 1, ..., m \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{6}$$

where $c_j \in \mathbb{Z}_+$ is the value of item j, $a_{ij} \in \mathbb{Z}_+$ is the amount of ingredient i in item j, and $b_i \in \mathbb{Z}_+$ is the total amount of ingredient i available, $i = 1, ..., m$ and $j = 1, ..., n$. The goal of the packing problem is to maximize the total value of the items using the ingredients in the pack.

On the other hand, the classical covering problem is defined as

$$\begin{aligned} \min \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ i = 1, ..., m \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{7}$$

where $c_j \in \mathbb{Z}_+$ is the unit cost of item j, $a_{ij} \in \mathbb{Z}_+$ is the amount of ingredient i in item j, and $b_i \in \mathbb{Z}_+$ is the minimum requirement on ingredient i in the pack, $i = 1, ..., m$ and $j = 1, ..., n$. The goal of the covering problem is to minimize the total cost of all items while satisfying all requirements on the ingredients.

The packing problem with $m = 1$ is called the knapsack problem (KP), and the packing problem is referred to as the multi-dimensional knapsack problem. Since any solution satisfying $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ must also satisfy $x_j \leq b_i / a_{ij}$ if $a_{ij} > 0$, the constraints in the packing problem imply an upper bound u for x. Thus replacing x with $u - y$ converts a packing problem to a covering problem, where y is an $n-$dimensional vector of integer variables. Since the packing problem and the covering problem are similar in various ways, and can be solved with similar approaches, for simplicity of notation and consistency of the context, in this thesis, we only consider the multi-dimensional knapsack problem formulated

as:

$$F(b) = \min \sum_{j=1}^{n} c_j x_j$$
$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ i = 1, ..., m$$
$$x \in \mathbb{Z}_+^n,$$

where $F(b)$ is called a *value function* of the right-hand side b.

MKP generalizes the knapsack problem with multiple knapsack constraints. It often arises in resource allocation, capital budgeting and cutting stock problems. It is NP-hard and can be solved in pseudo-polynomial time with a dynamic programming algorithm for a fixed number of constraints. Many studies have been devoted to improving the efficiency of the dynamic programming algorithm for solving KP, including the influential work by Gilmore and Gomory [27]. However, it is quite impractical to use the dynamic programming algorithm to solve MKP with a large number of constraints and large right-hand side, as the size of the state space increases exponentially with the number of constraints. For MKP with binary variables, many general and special-purpose algorithms have been developed. Fréville and Hanafi [21] surveyed theoretical and computational results concerning such problems. However, very little attention has been given to solving MKP with unbounded integer variables. We especially notice that for many MKP instances with a large number of constraints, it remains difficult to close the relative gaps even with the state-of-the-art MIP solvers. For a comprehensive discussion of topics related to the knapsack problem, see Martello and Toth [52] and more recently Kellerer [42].

A main obstacle for efficiently solving MKP is the lack of special structure. From the perspective of lower bounds, since the constraints are presented in a symmetric way, it is difficult to determine effective relaxations in order to obtain good lower bounds. For MKP with hundreds or thousands of constraints, adding many cutting planes may cause computational difficulties with branch-and-cut algorithms. These characteristics of large size MKP call for the development of general-purpose lower-bound-improving strategies, which is the focus of the second part of this thesis. In particular, we extend the idea of relaxing constraints in exchange for a much smaller problem size for obtaining lower bounds.

The remainder of the thesis is structured as follows:

In Chapter 3, we study characteristics of the value function of MKP, and extend some properties that are established in Gilmore and Gomory [27] from KP to MKP. In addition, for some special cases, we present conditions under which the corner relaxation is tight and we provide worst-case analysis on the bounds of the corner relaxation for MKP.

In Chapter 4, in order to identify parameters that contribute the most to the hardness of MKP, we conduct computational tests by varying parameters and analyzing characteristics of the instances. In addition, we establish a testbed of instances that are used in the subsequent computational experiments.

In Chapter 5, we study relaxation algorithms for MKP to obtain lower bounds. We first study various relaxations for MKP by conducting a series of computational tests and analyzing the results to answer the following questions: How should we choose active constraints and solve the relaxations efficiently? Which relaxation algorithm gives the best lower bounds for MKP? Furthermore, inspired by the success of many primal heuristics, we design a dual heuristic algorithm which incorporates relaxation algorithms within the branch-and-bound algorithm. The dual algorithm replaces LP relaxation bounds with surrogate relaxation bounds to improve lower bounds. We also conduct experiments on adaptively controlling the dual algorithm as many successful heuristics do.

In Chapter 6, we take an alternative approach to obtain lower bounds and study solving the subadditive dual problem of MKP. Based on a subadditive lifting method developed by Johnson [38] to solve KP, we present a new implementation of the method on MKP as an exact algorithm that iteratively improves lower bounds. Because MKP can be reformulated as a shortest path problem, we also present an efficient shortest path algorithm that uses estimated distance labels and solves relaxations to determine node labels. We use the characteristics of MKP to accelerate the algorithm. Finally, we conduct computational tests to evaluate the lower bounds produced by the algorithm.

In summary, this thesis deals with two components of MIP: modeling and computational improvement. Most importantly, the thesis provides an in-depth investigation in various algorithmic ideas for improving lower bounds for MIP. By combining relaxation algorithms

with a branch-and-bound algorithm, we introduce local search into dual algorithms for the first time. In particular, we hope our research demonstrates that there is still a great potential in exploiting dual heuristic algorithms in solving MIP, and brings more attention to this topic.

CHAPTER II

PRICING FOR PRODUCTION AND DELIVERY FLEXIBILITY

2.1 Introduction

A company can boost its profits by increasing its revenue, decreasing its cost, or both. Adjusting prices to influence demand so as to increase revenue has become common practice and a large body of literature exists on models and techniques to do so. Adjusting prices to influence demand so as to decrease cost, however, appears to still be in its infancy. This is the focus of our research. We investigate the value of adjusting prices to increase delivery flexibility and thus to decrease cost. We do so in the context of a fundamental production planning model, namely the single-item, single-level, uncapacitated lot-sizing problem. Increasing profits by reducing costs can also be found, to some extent, in the ideas of dynamic lead time quotation, which has received considerable attention in recent years, see Celik and Maglaras [11], Ata and Olson [3], and Savasaneril et al.[58].

In the single-item, single-level, uncapacitated lot-sizing problem, we have a set of discrete time periods and the goal is to plan the production, i.e., determine the lot size in each period, in order to satisfy demand and minimize production and inventory holding costs. The production cost of a lot consists of two components, a fixed setup cost that is independent of the size of the lot and a marginal cost incurred for each unit produced in the lot. Inventory holding costs are incurred for each unit that is held over from one period to the next. Demand in a period can be satisfied by production or from inventory. The challenge is to find a proper balance between setup costs and inventory holding costs. For an extensive and comprehensive treatment of lot-sizing problems, and more generally production planning problems, see Pochet and Wolsey [56].

To investigate the potential of using pricing to boost profits by reducing cost, we consider a variant of the single-item, single-level, uncapacitated lot-sizing problem, in which the producer offers a price discount in return for which the producer receives the flexibility to

satisfy demand in a later period. We assume that offering a price discount has no effect on the total demand in a period, but that a fraction of the demand in the period can be satisfied later, i.e., only a fraction of the customers placing an order in that period are willing to allow delivery to take place later in return for a price discount. The goal is to find a discount factor and production plan that maximizes profit by offsetting the loss in revenue due to discounts with a (greater) reduction in production and inventory holding costs.

Because we need to simultaneously decide on a discount factor and a production plan, the resulting model has a quadratic objective function and binary variables. Nevertheless, we show that the problem can still be solved in polynomial time. Furthermore, our computational study demonstrates that substantial profit increases can be obtained.

After Wagner and Whitin [62] introduced the uncapacitated lot-sizing problem, many variants have been proposed and studied. We briefly mention the ones that are most closely related to the variant that we study in this paper. Lot-sizing problems with demand windows were introduced by Lee et al.[48]. A demand window defines a grace period during which demand can be delivered or satisfied without penalty, i.e., no inventory holding or backlogging cost will be charged during the grace period. Wolsey [64] developed an extended formulation for this variant and presented an $O(T^2)$ dynamic programming algorithm for its solution. Demand windows introduce more flexibility for the producer and therefore result in lower cost. Lot-sizing problems with production windows were introduced by Dauzere-Peres et al.[15]. A production window defines a period during which demand must be produced. Production windows take away flexibility from the producer and therefore result in higher costs. Lot-sizing problems with production windows were further studied by Brahimi[8, 9]. Lot-sizing problems with pricing decisions were first discussed in Thomas [60], who considered a setting in which demand is a deterministic function of the price and a different price can be set in each period. Thomas showed that that problem can still be solved in polynomial time. Kunreuther and Schrage[46] considered the situation in which a single price must be set for all periods and proposed a heuristic for solving the problem. Van den Heuvel and Wagelmans[61] have shown that this variant too can be solved in

polynomial time.

The remainder of the paper is organized as follows. In Section 2, we formally introduce the uncapacitated lot-sizing problem with pricing for production and delivery flexibility. In Section 3, we present a solution approach. In Section 4, we give some quantitative results. In Section 5, we discuss several natural extensions. In Section 6, we present the results of a computational study.

2.2 The uncapacitated lot-sizing problem with pricing for delivery flexibility

Let $T = \{1, 2, ..., n\}$ be the set of time periods, D_i the total demand in period i, e_i the setup cost in period i, g_i the unit production cost in period i, h_i the unit inventory holding cost in period i, p_i the unit price in period i, and $D_{st} = \sum_{i=s}^{t} D_i$ the total demand in periods $s, s+1, ..., t$. Let x_i be the amount produced in period i, s_i the inventory at the end of period i, and z_i a binary indicator of whether production takes place in period i or not. The integer programming formulation for the uncapacitated lot-sizing problem (ULS) is:

$$\max \sum_{i=1}^{n} (p_i D_i - e_i z_i - h_i s_i - g_i x_i)$$

subject to

$$\begin{aligned} x_1 &= s_1 + D_1 \\ x_i + s_{i-1} &= s_i + D_i, \ i = 2, ..., n \\ x_i &\leq D_{in} z_i, \ i = 1, ..., n \\ s_n &= 0 \\ x, s &\geq 0, z \in \{0, 1\}^n. \end{aligned}$$

Next, suppose that we offer a price discount α with $0 \leq \alpha \leq 1$ for the flexibility to deliver a fraction of the demand D_i either in period i itself or in any of the periods $\{i+1, ..., i+\Delta\}$ where $\Delta \geq 1$. Specifically, we assume that by doing so a fraction α of demand D_i becomes flexible. Thus, the result of offering a discount α is a flexible demand $d_i^f = \alpha D_i$ in period i with associated price $p_i^f = (1 - \alpha) p_i$. Let $d_i = D_i - d_i^f$ denote the demand in period i that has to be delivered in period i and for which the full price p_i is collected. We will

sometimes refer to d_i as the inflexible demand and use $d_{st} = \sum_{i=s}^{t} d_i$ to denote the total inflexible demand in periods $s, s+1, ..., t$. The challenge is to determine the price discount α that maximizes profit by properly trading off the reduction in production and holding costs with the reduction in revenue due to discounts. Note that by allowing flexible demand we are trading off reduced revenue against reduced cost, i.e. it may be better to collect less revenue to have the opportunity to save setup and holding costs and thus increase profit. Note too that the fraction of demand that becomes flexible has the same characteristics as the demand in the ULS with demand time windows, as the demand can be satisfied in a time window rather than a single period. However, in the setting we consider, a price has to be paid for that flexibility, in the form of a discount, and only a fraction of the total demand will have that flexibility.

Let r_i denote the total accumulated flexible demand of periods $\{1, ..., i\}$ that will be satisfied in periods $\{i+1, ..., i+\Delta\}$. Thus, when $r_i > r_{i-1}$, then some of the flexible demand of period i will be satisfied in a later period, and when $r_i < r_{i-1}$, then some of the flexible demand from periods prior to period i will be delivered in period i. Note that it is only advantageous to deliver flexible demand of period i in period $k \in \{i+1, ..., i+\Delta\}$ if production takes place in period k. The integer programming formulation for the uncapacitated lot-sizing problem with pricing for production and delivery flexibility (ULS-PDF) is:

$$\max \sum_{i=1}^{n} (p_i d_i + p_i^f d_i^f - e_i z_i - h_i s_i - g_i x_i)$$

$$
\begin{aligned}
\text{subject to} \qquad x_1 &= s_1 - r_1 + D_1 && (8)\\
x_i + s_{i-1} - r_{i-1} &= s_i - r_i + D_i,\ i = 2, ..., n && (9)\\
x_i &\leq D_{1n} z_i,\ i = 1, ..., n && (10)\\
r_i &\leq \sum_{j=\max\{1, i-\Delta+1\}}^{i} d_j^f,\ i = 1, ..., n && (11)\\
r_i - r_{i-1} &\leq d_i^f,\ i = 2, ..., n && (12)\\
\alpha &\leq 1 && (13)\\
x, s, r, \alpha &\geq 0 && (14)\\
s_n &= 0 && (15)\\
r_n &= 0 && (16)\\
z &\in \{0,1\}^n. && (17)
\end{aligned}
$$

Observe that for each feasible setup plan z, the model is a network flow problem. Note also that the variables r_i are similar to backlogging variables in the ULS with backlogging. However, they do not have any cost associated with them since they represent flexible demand that is delivered in a later period and they are bounded from above. Equations (8) and (9) represent flow conservation and inequality (10) forces a setup when production takes place. Inequalities (11)-(12) ensure that any demand delivered late is flexible demand. It is easy to see that there exists an optimal solution in which the flexible demand d_i^f of period i is satisfied in a single period in which production takes place.

2.2.1 Solving the problem for a given discount factor α

For a given discount factor α, ULS-PDF can be solved in polynomial time by dynamic programming. Before presenting the dynamic program, we summarize some relevant structural properties of an optimal solution to ULS-PDF.

Theorem 1 *There exists an optimal solution z to ULS-PDF with the following structure. Let $I^z = \{i_1, ..., i_k\}$ be the set of periods in which a setup takes place. For all periods $i \in [i_s, i_{s+1} - 1]$ for any $s \in \{1, ..., k-1\}$, the inflexible demand d_i is produced in period i_s and delivered in period i and the flexible demand d_i^f is either produced in period i_s and*

delivered in period i or produced in one of the periods in $\{i+1,...,i+\Delta\} \cap I^z$ and delivered in that period depending on the production and holding costs. Furthermore, if d_i^f is delivered in period $t > i$, then flexible demand $d_{i'}^f$ in period $i < i' \leq t$ will be delivered either in period t or in period $t' > i + \Delta$. Finally, for all periods $i \in \{i_k, ..., n\}$, the demand D_i is produced in period i_k and delivered in period i.

Proof Consider an optimal solution z^* to ULS-PDF. For any period i, the inflexible demand d_i of period i will be produced in the latest period $s \leq i$ in which a setup occurs, because otherwise either a solution with lower holding costs exists or the setup in period s was unnecessary. The flexible demand d_i^f of period i on the other hand, may either be produced in period s or in a period in $\{i+1, ..., i+\Delta\} \cap I^{z^*}$. If d_i^f is produced no later than period i, then it can be produced in the same period as d_i, because otherwise a solution with lower holding costs can be obtained by changing the production period for either d_i or d_i^f. If d_i^f is produced, and thus delivered, in a period in $\{i+1, ..., i+\Delta\}$, then the unit production cost in that period will be the smallest among $I^{z^*} \cap \{i+1, ..., i+\Delta\}$. Furthermore, if d_i^f is produced in period $t > i$, then for any period i' with $i < i' \leq t$, the flexible demand $d_{i'}^f$ will be produced in period t or in a period in $\{i+\Delta+1, ..., i'+\Delta\} \cap I^{z^*}$, because producing in any period in $\{i', ..., i+\Delta\}$ other than period t will be more expensive that producing in period t. □

The structural properties of optimal solutions allow us to solve ULS-PDF for a given discount factor α with the following forward dynamic programming algorithm. Since α is fixed, the revenue is constant and the objective is to minimize total cost. Let $H(t-i, t)$ for $t = 1, ..., n$ and $i = 0, 1, ..., \min(t, \Delta)$ be the minimum cost of delivering inflexible demand of periods 1 to t and flexible demand of periods 1 to $t - i$ with setups only occurring in one or more of the periods in $\{1, ..., t\}$. Furthermore, let $H^+(t-i, t)$ be the minimum cost of delivering inflexible demand of periods 1 to t and flexible demand of periods 1 to $t - i$ with one setup occurring in period t and possibly other setups occurring in periods in $\{1, ..., t-1\}$. For convenience, let $c(k, j) = g_k + \sum_{k \leq l \leq j-1} h_l$ for any $k < j$.

For $t = 1, ..., n$ and $i = 0, 1, ..., \min(t, \Delta)$:

$$H^+(t-i,t) = e_t + g_t d_t + \min_{\max(0,i-1)\le k\le\Delta} \{g_t \sum_{j=t-k}^{t-i} d_j^f + H(t-k-1, t-1)\}, \qquad (18)$$

$$H(t-i,t) = \min_{1\le k\le t} \{H^+(\min(t-i,k),k) + \sum_{j=k+1}^{t} c(k,j)d_j + \sum_{j=k+1}^{t-i} c(k,j)d_j^f\}. \qquad (19)$$

To see (18), observe that since a setup takes place in period t, the inflexible demand of period t is produced and delivered in period t (Theorem1). Next, observe that flexible demand in one or more of the periods $t-\Delta, ..., t-i$ can be produced and delivered in period t. However, if this happens, then it will happen for a set of consecutive periods $t-k, ..., t-i$ ($i \le k \le \Delta$). Finally, observe that inflexible demand in periods before t is produced at or before period $t-1$. To see (19), we are enumerating over the time when the last setup at or before period t can take place.

The dynamic program is initialized by setting $H^+(0,0) = H(0,0) = 0$ and $H(n,n)$ gives the value of an optimal solution. The dynamic program yields an $O(n^3)$ algorithm for ULS-PDF with a given discount factor α. For each t, it takes $O(\Delta t)$ time to compute $H^+(t,t)$ and $O(\Delta t^2)$ time to calculate $H(t,t)$. Therefore, computing $H(n,n)$ takes $O(\sum_{t=1}^{n} t^2) = O(n^3)$.

2.2.2 Analysis of the objective function

Because we need to simultaneously decide on a discount factor and a production plan, the resulting model has a nonlinear objective function. In this section, we analyze the structure of the objective function. More specifically, we will analyze the profit functions $P^z(\alpha)$, the profit as a function of α for a given feasible setup plan z, and $P(\alpha) = \max_z P^z(\alpha)$, the profit as a function of α. Note that for any feasible setup plan z, there always exists a solution satisfying the structure property described in Theorem 1. Therefore we only consider solutions with such structure property.

Theorem 2

1. *For a given feasible setup plan z, the profit function $P^z(\alpha)$ is a concave quadratic function of α.*

2. The profit function $P(\alpha)$ is a piecewise concave quadratic function.

Proof For a given feasible setup plan z and discount factor α, let $Q_i^z(\alpha)$ be the profit associated with inflexible demand d_i. Since inflexible demand can only be produced at or before its delivery, we have

$$Q_i^z(\alpha) = (p_i - c_i^z)d_i = (p_i - c_i^z)(1-\alpha)D_i$$

where $c_i^z = g_{i_t} + h_{i_t} + ... + h_{i-1}$ gives the per unit cost of d_i, and i_t is the latest period with a setup at or before period i, i_{t+1} is the first period with a setup after period i.

Similarly, for a given feasible setup plan z and discount factor α, let $R_i^z(\alpha)$ be the profit associated with flexible demand d_i^f. Recall that flexible demand d_i^f can be satisfied either in period i or in any of the periods $\{i+1, ..., i+\Delta\}$. Therefore

$$R_i^z(\alpha) = (p_i^f - C_i^z)d_i^f = D_i(p_i - p_i\alpha - C_i^z)\alpha$$

where C_i^z is the per unit cost of d_i^f, i.e.,

$$C_i^z = \begin{cases} c_i^z & \text{if } i_t \le i < i_{t+1} - \Delta \text{ for some } i_t \in I^z \\ \min\{c_i^z, g_k : k \in I^z,\ i < k \le i+\Delta\} & \text{otherwise,} \end{cases}$$

Therefore, for a given feasible setup plan z and discount factor α the total profit $P^z(\alpha)$ is

$$\begin{aligned} P^z(\alpha) &= \sum_i (R_i^z(\alpha) + Q_i^z(\alpha) - e_i z_i) \\ &= -(\sum_i D_i p_i)\alpha^2 + \sum_i D_i(c_i^z - C_i^z)\alpha + \sum_i [D_i(p_i - c_i^z) - e_i z_i], \end{aligned} \tag{20}$$

a quadratic function of α with a negative coefficient on the quadratic term, and thus concave. This proves the first part of the proposition.

We can write $P(\alpha)$ as $revenue(\alpha) - cost(\alpha)$, where $revenue(\alpha) = \sum_i (p_i d_i + p_i^f d_i^f) = \sum_i p_i D_i - \sum_i p_i D_i \alpha^2$ represents the revenue for a given discount factor α and $cost(\alpha)$ represents the cost of an optimal production plan for a given discount factor α, i.e., the optimal value of $\min \sum_{i=1}^n (e_i z_i + h_i s_i + g_i x_i)$ subject to constraints (8)-(17).

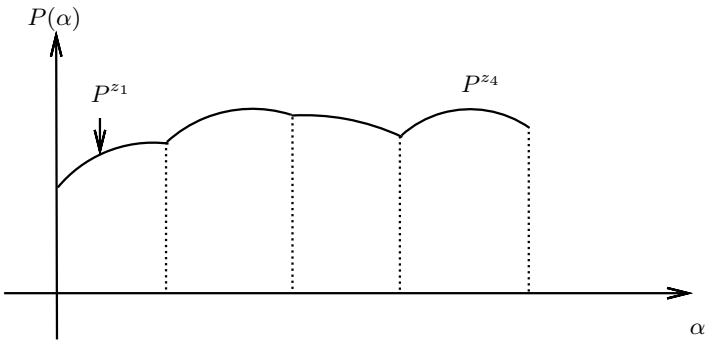


Figure 4: Piecewise Quadratic Objective Function

Since $revenue(\alpha)$ is a concave quadratic function, to prove the second part of the proposition it is sufficient to show that $cost(\alpha)$ is piecewise linear. This is the case, because $cost(\alpha) = \min_z cost^z(\alpha)$, where

$$cost^z(\alpha) = \sum_i (e_i z_i + d_i c_i^z + d_i^f C_i^z) = \sum_i (e_i z_i + D_i c_i^z) - \sum_i D_i (c_i^z - C_i^z)\alpha.$$

Thus, $P(\alpha)$ is a piecewise concave quadratic function of α, with each piece corresponding to a feasible setup plan z; see Figure 4. Note that $cost(\alpha)$ is also a concave function and this property will be used in our polynomial time algorithm for problem ULS-PDF. □

2.3 Solution approach

Van den Heuvel and Wagelmans [61] presented a polynomial-time algorithm for the variant of ULS in which demand is a function of price and a price needs to be set for each period. Their algorithm can be extended to the ULS-PDF showing that ULS-PDF too can be solved in polynomial time.

A feasible setup plan z is said to be *dominating* if there exists some α in the feasible region $[0, 1]$, such that z is an optimal setup plan. The algorithm is based on enumerating the dominating setup plans. Since for each feasible setup plan z, the profit $P^z = \max_\alpha P^z(\alpha)$ can be found in polynomial time, since $P^z(\alpha)$ is a quadratic function of α (Section 2.2.2), the algorithm is polynomial if the number of dominating feasible setup plans z is polynomial and they can be generated in polynomial time.

Note that each dominating feasible setup plan z corresponds to one piece of the piecewise

quadratic function $P(\alpha)$, and thus also to one piece of the piecewise linear function $cost(\alpha)$ (Section 2.2.2). Therefore, it suffices to show that cost(α) has a polynomial number of breakpoints, as this implies that the linear pieces can be enumerated in polynomial time.

We show that there is a polynomial number of breakpoints with arguments similar to those presented in Van den Heuvel and Wagelmans [61]. Since there are exponential number of feasible setup plans, the number of breakpoints for $cost(\alpha)$ is in general exponential. However, it can be reduced to a polynomial number using the following theorem.

Lemma 3 *(Van den Heuvel and Wagelmans 2006) Given $2n$ ($n \geq 2$) linear functions $l_i(\alpha) = a_i - b_i\alpha$ with the property that $a_{2k-1} - a_{2k} = a$ and $b_{2k-1} - b_{2k} = b$ for $k = 1, ..., n$, then the function $l(\alpha) = \min_{i=1,...,2n} l_i(\alpha)$ has at most n breakpoints. (That is, at least $n-1$ of the possible $2n-1$ breakpoints of $l(\alpha)$ have been "lost".)*

Theorem 4 *$cost(\alpha)$ has a polynomial number of breakpoints.*

Proof Define $l^z(\alpha) = cost^z(\alpha) = \sum_i (e_i z_i + D_i c_i^z) - \sum_i D_i(c_i^z - C_i^z)\alpha = a^z - b^z\alpha$ so that $cost(\alpha) = \min_z l^z(\alpha)$. We will identify a sufficiently large number of pairs of linear functions $l^z(\alpha)$ with the desired properties that shows that the function $cost(\alpha) = \min_z l^z(\alpha)$ has a polynomial number of break points.

Consider pairs of feasible setup plans of the form

$z \qquad 1 \bullet \cdots \bullet \; 1\, 0 \cdots 0$

and

$z' \qquad 1 \bullet \cdots \bullet \; 1 \underbrace{\circ \cdots \circ}_{m}$

where a 1 indicates a setup, a 0 indicates no setup, a $\bullet$ indicates that the "setup decision" is the same in z and z', and a sequence of $\circ$ indicates a set of periods in which exactly one setup occurs. Let the number of periods in z' with $\circ$ be m and let the one setup occur in period k.

We can divide these pairs into two classes. For each class, it will be easy to see that the linear functions $l_z(\alpha)$ and $l_{z'}(\alpha)$ satisfy the pair-properties described in Lemma 3.

Class 1: $k \geq n - m + \Delta$: Consider the pair

$$\begin{array}{lcccccccccccc}
 & \overset{1}{\downarrow} & \overset{2}{\downarrow} & \cdots & & \overset{n-m}{\downarrow} & & \cdots & & \overset{n}{\downarrow} & & & \\
z & 1 & \bullet & \cdots & \bullet & 1 & 0 & \cdots & 0 & \cdots & & 0 & \\
z' & 1 & \bullet & \cdots & \bullet & 1 & \underbrace{0 \cdots 1}_{\geq \Delta} & \cdots & & & & 0 & \\
 & & & & & & \underbrace{\qquad\qquad}_{m} & & & & & &
\end{array}$$

Note that there are 2^{n-m-2} such pairs. Since $a_z - a_{z'}$ and $b_z - b_{z'}$ only depend on k and m which are fixed, the assumption of Lemma 3 is satisfied. The number of breakpoints lost for each combination of k and m is $2^{n-m-2} - 1$. Therefore the total number of breakpoints lost is

$$\sum_{m=\Delta}^{n-3} (m - \Delta + 1)[2^{n-m-2} - 1] = 2^{n-\Delta} - P_1(n),$$

where $P_1(n) = \frac{(n-\Delta)^2+(n-\Delta)+2}{2}$ is a polynomial function of n.

Class 2: $k < n - m + \Delta$: Consider the pair

$$\begin{array}{lcccccccccccccc}
 & \overset{1}{\downarrow} & \overset{2}{\downarrow} & \cdots & \overset{i-j+2}{\downarrow} & \cdots & \overset{i+1}{\downarrow} & \cdots & \overset{n-m}{\downarrow} & \cdots & \overset{n}{\downarrow} & & & \\
z & 1 & \bullet & \cdots & \bullet & 1 & 0 \cdots 0 & * \cdots * & 1 & 0 \cdots 0 \cdots & & 0 & & \\
z' & 1 & \bullet & \cdots & \bullet & \underbrace{1\ 0 \cdots 0}_{j} & \underbrace{* \cdots *}_{\Delta} & \underbrace{1\ 0 \cdots}_{} & 1 \cdots & & & 0 & & \\
 & & & & \underbrace{\qquad\qquad\qquad}_{i} & & & & & \underbrace{\qquad\qquad}_{m} & & & &
\end{array}$$

where $*$ also indicates an identical setup decision in z and z'. Since $a_z - a_{z'}$ and $b_z - b_{z'}$ only depend on k, m, and j, the number of breakpoints lost is at least $2^{i-j} - 1$, where $i + 2 + \Delta = k$. There are $2^{n-m-i-2}$ choices for $*$, therefore the total number of breakpoints lost is at least

$$(\sum_{m=1}^{\Delta-1} \sum_{k=n-m+1}^{n} + \sum_{m=\Delta}^{n-3} \sum_{k=n-m+1}^{n-m+\Delta-1}) 2^{n-m-i-2} \sum_{j=1}^{i-1} (2^{i-j} - 1) = 2^{n-1} - 2^{n-\Delta} - P_2(n),$$

where $P_2(n) = 2^{\Delta-1}[(n-\Delta)^2 + (n-\Delta) - \Delta^2 + 7\Delta - 12] - (n-\Delta)^2 + (n-\Delta) + 8$ is a polynomial function of n.

Therefore $cost(\alpha)$ has at most $2^{n-1} - (2^{n-\Delta} - P_1(n)) - (2^{n-1} - 2^{n-\Delta} - P_2(n)) = P_1(n) + P_2(n)$ breakpoints, the number is polynomial in n. □

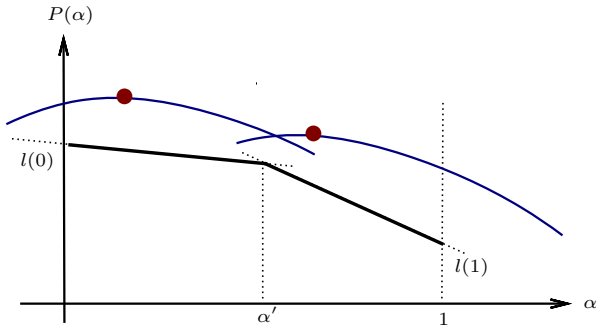


Figure 5: Recursive Partitioning

The procedure to enumerate dominating feasible setup plans z is as follows (see also Kunreuther and Schrage[46]). For each $\alpha \in [0,1]$, the corresponding optimal setup plan $z(\alpha)$ can be found in polynomial time by solving problem ULS-PDF with fixed α (Section 2.2.1). Let $l(\alpha)$ denote the line $\sum_i (e_i z(\alpha)_i + D_i c_i^{z(\alpha)}) - \sum_i D_i (c_i^{z(\alpha)} - C_i^{z(\alpha)})\alpha$. Note that $l(\alpha)$ coincides with $cost(\alpha)$ at α. Because $cost(\alpha)$ is concave, $l(0)$ and $l(1)$ intersect at a point $\alpha' \in [0,1]$; see Figure 2. We can easily find $z(\alpha')$ and $l(\alpha')$ (Section 2.2.2). By recursively applying this procedure in the two new intervals $[0,\alpha']$ and $[\alpha',1]$, we will enumerate all dominating feasible setup plans.

2.4 A special case

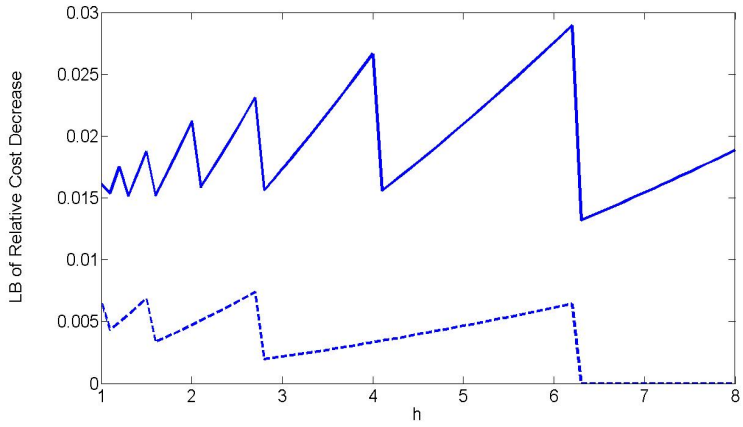
In this section, we consider a special case in which all of the parameters are identical for all periods in the planning horizon and we derive a lower bound on the relative decrease in controllable cost with flexible demand. Let $\delta_c^{rel} = \frac{c^{ULS} - c^{ULS-PDF}}{c^{ULS}}$.

Theorem 5 *For instances of ULS-PDF with $D_i = D, p_i = p, h_i = h, e_i = e$ for $i = 1, ..., n$, $\epsilon > 0$ and $n > 2\lceil \hat{k} \rceil / \epsilon$*

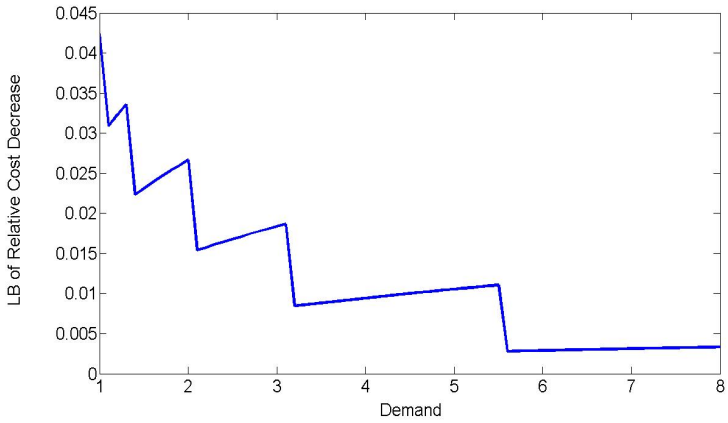
$$\delta_c^{rel} \geq \frac{p}{h} \frac{(\min\{\frac{h}{4p} f(\lfloor \hat{k} \rfloor), 1\})^2}{\hat{k}} - \epsilon,$$

where $\hat{k} = \sqrt{2e/Dh}$, $f(k) = k - 1$ if $k \leq \Delta$ and $f(k) = 2\Delta - \frac{\Delta(\Delta+1)}{k}$ if $k > \Delta$.

To better appreciate and understand the lower bound, we examine the function $g = \frac{p}{h}(\min\{\frac{h}{4p} f(\lfloor \hat{k} \rfloor), 1\})^2 / \hat{k}$ where $\hat{k} = \sqrt{2e/Dh}$. Figure 6(a) shows g for an instance with $\Delta = 8, e = 100, p = 30$, and h varying from 1 to 8. The solid curve shows the function for $D = 2$ and the dotted curve shows the function for $D = 8$. We see that the function values



(a) Varying h



(b) Varying D

Figure 6: Lower Bound for Relative Controllable Cost Decrease

increase in a jagged way with the holding costs h rather than monotonically. In Figure 6(b), we vary demand D from 1 to 8 with the holding costs fixed at $h = 4$. Similarly, we see that the function values decrease in a jagged way with D. The jumps down occur for values of D where $\lfloor\sqrt{2e/Dh}\rfloor$ decreases by 1, i.e., where the number of setups changes.

Proof First consider the controllable costs for ULS. Since all parameters are identical for all periods, for each production interval of length k the controllable costs a_k, i.e, the setup and holding costs, are

$$a_k = e + D(1 + 2 + ... + (k-1))h = e + Dhk(k-1)/2. \tag{21}$$

Let x_k be the number of production intervals of length k in a feasible production plan,

which implies that $\sum_{k=1}^{n} kx_k = n$. Finding an optimal production plan is equivalent to solving the integer program

$$\begin{aligned} &\min \sum_{k=1}^{n} a_k x_k \\ &\sum_{k=1}^{n} k x_k \geq n \\ &x_k \geq 0, integer. \end{aligned}$$

We first show that for any $\epsilon > 0$, there exists a planning horizon n, such that there exists a periodic production plan with cost that is within ϵ percent of the cost of an optimal production plan. Let $k^* = arg\min_{k=1...n}\{a_k/k\}$. We have that $x = (0, ..., n/k^*, ..., 0)$ is the optimal solution to the linear programming relaxation, and that $\bar{x} = (0, ..., \lceil n/k^* \rceil, ...0)$ is a feasible integer solution, representing a periodic production plan with periodicity k^*. Let c and $\bar{c}$ be the costs associated with x and $\bar{x}$ respectively, and let c^{ULS} be the optimal cost. We have $c \leq c^{ULS} \leq \bar{c}$ and $\bar{c} - c = a_{k^*}\lceil \frac{n}{k^*} \rceil - a_{k^*}\frac{n}{k^*} \leq a_{k^*}$, which implies $c \leq c^{ULS} \leq c + a_{k^*}$. Since $c = a_{k^*}n/k^*$, we have that for any $\epsilon > 0$ and $n > 2k^*/\epsilon$,

$$\frac{a_{k^*}}{c} \leq \epsilon/2. \tag{22}$$

Thus

$$c \leq c^{ULS} \leq c(1 + \epsilon/2). \tag{23}$$

Next, let $c(k) = n(\frac{e}{k} + \frac{Dh}{2}(k-1))$. Since the function $c(k)$ with $k > 0$ is unimodal with minimum at $\hat{k} = \sqrt{2e/Dh}$ where $e/\hat{k} = Dh\hat{k}/2$, it follows that the optimal periodicity k^* can only be $\lfloor \hat{k} \rfloor$ or $\lceil \hat{k} \rceil$. (This shows that, as expected, large setup costs lead to fewer setups and large demand and holding costs lead to more setups.) Then by (21) and $c = a_{k^*}n/k^*$,

$$c - c(\hat{k}) \leq c(\lceil \hat{k} \rceil) - c(\hat{k}) \leq \frac{nDh}{2}(\lceil \hat{k} \rceil - \hat{k}) \leq nDh/2.$$

Therefore, since $c(\hat{k}) = nDh(\hat{k} - 1/2)$, we have

$$c \leq c(\hat{k}) + nDh/2 = nDh\hat{k}. \tag{24}$$

Next, we consider the controllable costs in the presence of flexible demand, i.e., $c^{ULS-PDF}$, comprised of discount costs, setup costs and holding costs. For a given discount factor α, the inflexible demand d is $(1-\alpha)D$ and the flexible demand d^f is αD. An upper bound on $c^{ULS-PDF}$ is obtained by considering a periodic production plan with periodicity k^*.

- If $k^* \leq \Delta$, we do not incur holding costs for flexible demand because all flexible demand can be satisfied in the next setup period. The controllable cost is at most

$$\begin{aligned} & \min_\alpha\{nDp\alpha^2 + \lceil\frac{n}{k^*}\rceil(e + dh\frac{k^*(k^*-1)}{2})\} \\ \leq\ & \min_\alpha\{nDp\alpha^2 + (\frac{n}{k^*}+1)(e + dh\frac{k^*(k^*-1)}{2})\}, \end{aligned}$$

  where $Dp\alpha^2$ represents the loss in revenue due to discounts.

  By (21) we have $a_{k^*} \geq e + dh\frac{k^*(k^*-1)}{2}$ and therefore, using $d = (1-\alpha)D$, the controllable costs are bounded by

$$\begin{aligned} & \min_\alpha\{nDp\alpha^2 + \frac{n}{k^*}(e + dh\frac{k^*(k^*-1)}{2}) + a_{k^*}\} \\ =\ & n\min_\alpha\{Dp\alpha^2 - \frac{Dh}{2}(k^*-1)\alpha + \frac{e}{k^*} + \frac{Dh}{2}(k^*-1)\} + a_{k^*}. \end{aligned}$$

- For $k^* > \Delta$, we incur holding costs for some of the flexible demand. The controllable costs are at most

$$\begin{aligned} & \min_\alpha\{nDp\alpha^2 + \lceil\frac{n}{k^*}\rceil(e + dh\frac{k^*(k^*-1)}{2} + d^f h\frac{(k^*-\Delta)(k^*-\Delta-1)}{2})\} \\ \leq\ & \min_\alpha\{nDp\alpha^2 + (\frac{n}{k^*}+1)(e + dh\frac{k^*(k^*-1)}{2} + d^f h\frac{(k^*-\Delta)(k^*-\Delta-1)}{2})\}. \end{aligned}$$

  Since $d + d^f = D$ and by (21) $a_{k^*} \geq e + dh\frac{k^*(k^*-1)}{2} + d^f h\frac{(k^*-\Delta)(k^*-\Delta-1)}{2}$, we have that the controllable costs are at most

$$\begin{aligned} & \min_\alpha\{nDp\alpha^2 + \frac{n}{k^*}(e + dh\frac{k^*(k^*-1)}{2} + d^f h\frac{(k^*-\Delta)(k^*-\Delta-1)}{2}) + a_{k^*}\} \\ =\ & \min_\alpha\{nDp\alpha^2 + \frac{n}{k^*}(e + Dh\frac{k^*(k^*-1)}{2} - Dh\alpha\frac{\Delta(k^*-\Delta+k^*-1)}{2})\} + a_{k^*} \\ =\ & n\min_\alpha\{Dp\alpha^2 - Dh\frac{\Delta(2k^*-\Delta-1)}{2k^*}\alpha + \frac{e}{k^*} + \frac{Dh}{2}(k^*-1)\} + a_{k^*}. \end{aligned}$$

Combining the two cases, we obtain

$$\begin{aligned} c^{ULS-PDF} &\leq n \min_{\alpha}\{Dp\alpha^2 - \frac{Dh}{2}f(k^*)\alpha + \frac{e}{k^*} + \frac{Dh}{2}(k^*-1)\} + a_{k^*} \\ &= n \min_{\alpha}\{Dp\alpha^2 - \frac{Dh}{2}f(k^*)\alpha\} + c + a_{k^*}, \end{aligned} \tag{25}$$

where $f(\cdot)$ is nondecreasing and defined by

$$f(k) = \begin{cases} k-1, & \text{if } k \leq \Delta \\ 2\Delta - \frac{\Delta(\Delta+1)}{k}, & \text{if } k > \Delta. \end{cases}$$

By (23), the relative decrease in controllable costs satisfies

$$\begin{aligned} \frac{c^{ULS} - c^{ULS-PDF}}{c^{ULS}} &\geq \frac{c - c^{ULS-PDF}}{c(1+\epsilon/2)} \\ &= \frac{(1+\epsilon/2)(c - c^{ULS-PDF}) - \epsilon/2(c - c^{ULS-PDF})}{c(1+\epsilon/2)} \\ &\geq \frac{c - c^{ULS-PDF}}{c} - \frac{\epsilon}{2}\frac{c - c^{ULS-PDF}}{c} \\ &\geq \frac{c - c^{ULS-PDF}}{c} - \frac{\epsilon}{2}. \end{aligned} \tag{26}$$

By (24) and (25), we get

$$\begin{aligned} \frac{c - c^{ULS-PDF}}{c} &\geq \frac{-n \min_{\alpha}\{Dp\alpha^2 - \frac{Dh}{2}f(k^*)\alpha\} - a_{k^*}}{c} \\ &\geq \frac{-n \min_{\alpha}\{Dp\alpha^2 - \frac{Dh}{2}f(k^*)\alpha\}}{nDh\hat{k}} - \frac{a_{k^*}}{c} \\ &= \frac{-\min_{\alpha}\{p\alpha^2 - \frac{h}{2}f(k^*)\alpha\}}{h\hat{k}} - \frac{a_{k^*}}{c}. \end{aligned} \tag{27}$$

From (22), we obtain

$$\frac{c - c^{ULS-PDF}}{c} > \frac{-\min_{\alpha}\{p\alpha^2 - \frac{h}{2}f(k^*)\alpha\}}{h\hat{k}} - \epsilon/2. \tag{28}$$

Using properties of quadratic functions and the fact that $f(\cdot)$ is nondecreasing, we find

$$\begin{aligned}
\frac{-\min_\alpha\{p\alpha^2 - \frac{h}{2}f(k^*)\alpha\}}{h\hat{k}} &= \begin{cases} \frac{h}{16p}\cdot\frac{f(k^*)^2}{\hat{k}}, & \text{if } \frac{h}{4p}f(k^*) \le 1 (\alpha^* = \frac{h}{4p}f(k^*)) \\ \frac{f(k^*)/2 - p/h}{\hat{k}}, & \text{if } \frac{h}{4p}f(k^*) > 1 (\alpha^* = 1) \end{cases} \\
&\begin{cases} = \frac{p}{h}\cdot(\frac{h}{4p}f(k^*))^2/\hat{k}, & \text{if } \frac{h}{4p}f(k^*) \le 1 (\alpha^* = \frac{h}{4p}f(k^*)) \\ > \frac{p}{h}\cdot(1/\hat{k}), & \text{if } \frac{h}{4p}f(k^*) > 1 (\alpha^* = 1) \end{cases} \\
&\ge \frac{p}{h}(\min\{\frac{h}{4p}f(k^*), 1\})^2/\hat{k} \\
&\ge \frac{p}{h}(\min\{\frac{h}{4p}f(\lfloor\hat{k}\rfloor), 1\})^2/\hat{k}. \qquad (29)
\end{aligned}$$

Therefore by (26), (27), (28) and (29), the relative decrease in controllable costs satisfies

$$\frac{c^{ULS} - c^{ULS-PDF}}{c^{ULS}} \ge \frac{p}{h}(\min\{\frac{h}{4p}f(\lfloor\hat{k}\rfloor), 1\})^2/\hat{k} - \epsilon.$$

□

2.5 Generalizations

We present two generalizations of ULS-PDF. The first allows the discount factor to be different in each period and the second allows flexible demand to be delivered early as well as late.

2.5.1 Per-period discounts

The flexibility to offer per-period discounts can yield larger profits and therefore is of practical relevance. However, it also has some interesting algorithmic implications. In fact, as we will demonstrate soon, a relatively straightforward dynamic program can be developed to solve ULS-PDF with per-period discounts. The dynamic program is based on the following proposition which observes that for a given set up plan it is easy to compute the optimal per-period discounts. We continue to use the structural properties described in Theorem1 for fixed α because the theorem does not require α to be the same in each period. For any fixed values of α_i, the model remains the same.

Proposition 6 *For an optimal setup plan z to an an instance of ULS-PDF with per-period discounts, the optimal discount α_i^* for period i is* $\min\{1, \frac{c_i^z - C_i^z}{2p_i}\}$*, where c_i^z (C_i^z) is*

the unit production and holding cost for d_i (d_i^f) with respect to the optimal setup plan z. Furthermore, when $p_i = p$ for $i = 1, ..., n$, then the optimal discounts for periods within a production interval are nondecreasing. Moreover, when $g_i = g$ and $h_i = h$ for all $i = 1, ..., n$ as well, then the optimal discount for period i is

$$\alpha_i^* = \begin{cases} 0 & \text{if } i_t \le i < i_{t+1} - \Delta \text{ for some } i_t \in I^z \\ \min\{1, kh/2p\} & \text{otherwise,} \end{cases}$$

where i_t is the latest period with a setup at or before period i and i_{t+1} is the first period with a setup after period i and $k = i - i_t$.

Proof Let z be an optimal setup plan. Then by replacing the single discount α with a per-period discounts α_i in Theorem 2, we have that

$$\begin{aligned} P^z(\alpha) &= \sum_i (R_i^z(\alpha_i) + Q_i^z(\alpha_i) - e_i z_i) \\ &= \sum_i [-(D_i p_i)\alpha_i^2 + D_i(c_i^z - C_i^z)\alpha_i + D_i(p_i - c_i^z) - e_i z_i]. \end{aligned}$$

To maximize $P^z(\alpha)$ with $0 \le \alpha_i \le 1$ for $i = 1, ..., n$, the optimal discount in period i is

$$\alpha_i^* = \min\{1, \frac{D_i(c_i^z - C_i^z)}{2(D_i p_i)}\} = \min\{1, \frac{c_i^z - C_i^z}{2p_i}\}.$$

To see that within a production interval $c_i^z - C_i^z$ is nondecreasing, note that $c_{i+1}^z = c_i^z + h_i$ and $C_{i+1}^z \le C_i^z + h_i$, therefore $c_i^z - C_i^z \le c_{i+1}^z - C_{i+1}^z$. Thus, when $p_i = p$ for $i = 1, ..., n$, we have $\alpha_i^* \le \alpha_{i+1}^*$.

When $g_i = g$ and $h_i = h$ for $i = 1, ..., n$, as well, then, with $k = i - i_t$, we have $c_i^z = g + kh$ and

$$C_i^z = \begin{cases} g + kh & \text{if } i_t \le i < i_{t+1} - \Delta \text{ for some } i_t \in I^z \\ g & \text{otherwise,} \end{cases}$$

Thus we have that

$$c_i^z - C_i^z = \begin{cases} 0 & \text{if } i_t \le i < i_{t+1} - \Delta \text{ for some } i_t \in I^z \\ kh & \text{otherwise.} \end{cases}$$

Therefore

$$\alpha_i^* = \begin{cases} 0 & \text{if } i_t \le i < i_{t+1} - \Delta \text{ for some } i_t \in I^z \\ \min\{1, kh/2p\} & \text{otherwise.} \end{cases}$$

□

Using Theorem 1 and 2 and Proposition 6, we can define a dynamic program that solves ULS-PDF with per-period discounts in polynomial time. This algorithm takes advantage of the fact that we can find the optimal discount factor α_i once we know the unit production cost of d_i and d_i^f, i.e., the production periods of d_i and d_i^f.

Let $H(t, t_1, t_2)$ be the optimal profit associated with satisfying demand of periods $\{1, ..., t\}$, when d_t is produced in period t_1 and d_t^f is satisfied in period t_2. By Proposition 6, the optimal discount $\alpha_t^* = \min\{1, \frac{c_t - C_t}{2p_t}\}$, where $c_t = g_{t_1} + h_{t_1} + ... + h_{t-1}$ and $C_t = g_{t_2}$ if $t_2 > t$ and $C_t = g_{t_2} + h_{t_1} + ... + h_{t-1}$ if $t_2 \leq t$. (By Theorem 1, if $t_2 \leq t$, then $t_2 = t_1$ in an optimal solution.) As before, let $c(k, j) = g_k + \sum_{k \leq l < j} h_l$ for any $k < j$. The dynamic programming recursion is given by:

Case 1 $t_1 < t < t_2$, where $g_{t_2} < c(t_1, t)$ and $t_2 \leq t + \Delta$:

Note that in an optimal solution, d_t^f is produced in $t_2 \in \{t+1, ..., t+\Delta\}$ only if the profit is higher than in t_1. A necessary condition for that to be the case is that the unit cost for producing d_t^f in t_2, i.e., g_{t_2} is lower than for producing d_t^f in t_1, i.e., $c(t_1, t)$.

- If $t_2 < t + \Delta$,

$$H(t, t_1, t_2) = \max\{H(t-1, t_1, t_1), H(t-1, t_1, t_2)\} + (p_t - c_t)d_t + (p_t^f - C_t)d_t^f. \quad (30)$$

- If $t_2 = t + \Delta$,

$$H(t, t_1, t_2) = \max \left\{ \begin{array}{c} \max\limits_{t < t' < t_2 : g_{t'} > g_{t_2}} H(t-1, t_1, t') \\ H(t-1, t_1, t_1) \end{array} \right\} + (p_t - c_t)d_t + (p_t^f - C_t)d_t^f. \quad (31)$$

For both cases, $\alpha_t = \min\{1, (c_t - C_t)/2p_t\}$, where $c_t = c(t_1, t)$ and $C_t = g_{t_2}$.

To see (30) and (31), observe that since d_t is produced in $t_1 < t$, there is no setup at period t. Therefore periods $t-1$ and t are in the same production interval. From Theorem 1, it follows that d_{t-1} is produced in t_1. If $t_2 < t + \Delta$, then d_{t-1}^f can be

produced either in t_1 or t_2. When $t_2 = t+\Delta > t-1+\Delta$, then d^f_{t-1} cannot be satisfied in t_2, but has to be produced in t_1 or satisfied in some period $t' \in \{t+1, ..., t+\Delta-1\}$. Note that if such t' satisfies $g_{t'} < g_{t_2}$, then d^f_t would have been satisfied in t' too. Therefore, only t' with $t < t' < t_2$ and $g_{t'} > g_{t_2}$ are considered.

Case 2 $t_1 = t < t_2$, where $g_{t_2} < g_{t_1}$ and $t_2 \leq t + \Delta$:

Note that in an optimal solution, if d_t is satisfied in t, then d^f_t is not satisfied in t only if $g_{t_2} < g_t$.

- If $t_2 < t + \Delta$,

$$H(t, t_1, t_2) = \max \left\{ \begin{array}{c} \max\limits_{t' \leq t-1 : c(t', t-1) > g_{t_2}} H(t-1, t', t_2) \\ \max\limits_{t' \leq t-1} H(t-1, t', t') \end{array} \right\} - e_t + (p_t - c_t) d_t + (p^f_t - C_t) d^f_t. \tag{32}$$

- If $t_2 = t + \Delta$,

$$H(t, t_1, t_2) = \max \left\{ \begin{array}{c} \max\limits_{t' \leq t-1, t \leq t'' < t_2 : g_{t_2} < g_{t''} < c(t', t-1)} H(t-1, t', t'') \\ \max\limits_{t' \leq t-1} H(t-1, t', t') \end{array} \right\} - e_t + (p_t - c_t) d_t + (p^f_t - C_t) d^f_t. \tag{33}$$

For both cases, $\alpha_t = \min\{1, (c_t - C_t)/2p_t\}$, where $c_t = g_{t_1}$ and $C_t = g_{t_2}$.

To see (32), observe that d_{t-1} has to be produced in some period $t' \in \{1, ..., t-1\}$, and that d^f_{t-1} is produced in t' or satisfied in t_2. If d^f_{t-1} is satisfied in t_2, it must be the case that $g_{t_2} < c(t', t-1)$, otherwise profit can be increased by producing in t'. To see (33), observe that since d^f_{t-1} cannot be satisfied in $t_2 = t + \Delta$, similar reasoning as for (31) and (32) can be used.

Case 3 $t_1 = t = t_2$:

$$H(t, t, t) = \max \left\{ \begin{array}{c} \max\limits_{t' \leq t-1 : c(t', t-1) \leq g_t} H(t-1, t', t') \\ \max\limits_{t' \leq t-1 : c(t', t-1) > g_t} H(t-1, t', t) \end{array} \right\} - e_t + (p_t - g_t) D_t. \tag{34}$$

In this case, $\alpha_t = 0$.

To see (34), observe that d_{t-1} has to be produced in some period $t' \in \{1, ..., t-1\}$ and d^f_{t-1} is produced in t' or satisfied in t. If d^f_{t-1} is satisfied in some period $t' > t$, then $g_{t'} < g_t$, and d^f_t should have been satisfied in t' as well.

Case 4 $t_1 = t_2 < t$:

$$H(t, t_1, t_2) = H(t-1, t_1, t_2) + (p_t - c(t_1, t))D_t. \tag{35}$$

In this case, $\alpha_t = 0$.

To see (35), observe that $t-1$ and t are in the same production interval and D_{t-1} is produced in period t_1. If d^f_{t-1} is satisfied in some period $t' > t$, then $g_{t'} < c(t_1, t-1) \leq c(t_1, t)$, and d^f_t should have been satisfied in t' as well.

The dynamic program is initialized by setting $H(0, t_1, t_2) = 0$ for $1 \leq t_1, t_2 \leq n$, setting $H(1, 1, t_2) = -e_1 + (p_1 - g_1)d_1 + (p^f_1 - g_{t_2})d^f_1$ for $1 < t_2 \leq 1 + \Delta$ with $g_{t_2} < g_1$, in which case $\alpha_1 = \min\{1, (g_1 - g_{t_2})/2p_1\}$, and setting $H(1, 1, 1) = -e_1 + (p_1 - g_1)D_1$. Finally, $\max_{t \leq n}\{H(n, t, t)\}$ gives the value of the optimal solution.

To compute the optimal value, it takes $O(\Delta^2 n^2)$ time to evaluate Case 1 and Case 2 and $O(n^2)$ time to evaluate Case 3 and Case 4, therefore the dynamic program takes $O(\Delta^2 n^2)$ time.

2.5.2 Early and late delivery

When the flexible demand of period i is allowed to be delivered either early or late, the delivery periods are $\{i - \frac{\Delta}{2}, ..., i, ..., i + \frac{\Delta}{2}\}$ for $\Delta \in \{2, 4, 6, ...\}$. Let u_i be the total accumulated flexible demand that is satisfied early in period i, i.e., flexible demand of periods $\{i+1, ..., i + \Delta/2\}$ that is delivered in period i. This variant can be formulated as

$$\max \sum_{i=1}^{n} (p_i d_i + p^f_i d^f_i - e_i z_i - h_i s_i - g_i x_i)$$

subject to

$$
\begin{aligned}
x_1 &= s_1 + u_1 - r_1 + D_1 \\
x_i + s_{i-1} + u_{i-1} - r_{i-1} &= s_i + u_i - r_i + D_i,\ i = 2, ..., n \\
x_i &\leq D_{1n} z_i,\ i = 1, ..., n \\
u_{i-1} - u_i + r_i - r_{i-1} &\leq d_i^f,\ i = 2, ..., n \qquad (36) \\
r_i &\leq \sum_{j=\max\{1, i-\Delta/2+1\}}^{i} d_j^f,\ i = 1, ..., n \qquad (37) \\
u_i &\leq \sum_{j=i+1}^{\min\{i+\Delta/2, n\}} d_j^f,\ i = 1, ..., n \qquad (38) \\
s_n, u_n, r_n &= 0 \\
\alpha &\leq 1 \\
x, s, u, r, \alpha &\geq 0 \\
z &\in \{0, 1\}^n.
\end{aligned}
$$

Inequalities (36)-(38) ensure that any demand that is delivered early or late is flexible demand. For each feasible setup plan, the model is a network flow problem. It is easy to show that this generalization shares the properties of the simpler model given by (8)-(17).

2.5.3 An example

We illustrate the impact of these generalizations by means of an example. Let the demands be as shown in Figure 7 and the parameters be $e_i = 50$, $p_i = 20$, $g_i = 10$, and $h_i = 4$ for all i and $\Delta = 4$. Figure 8 shows three optimal production plans. The first for the setting in which there is no flexible demand, the second for the setting in which there is flexible demand, but with a single discount factor and late deliveries only, and the third setting in which there is flexible demand with per-period discounts and both early and late deliveries.

In the production plans for the first and second setting, production takes place in the same four periods. However, in the second setting, a 7.8% discount is offered (in every period) in order to have the flexibility to deliver a fraction of the demand of a period in a later period. The dashed arcs show how that flexibility is exploited; the tail of a dashed arc indicates the period in which delivery takes place for flexible demand of the period at the

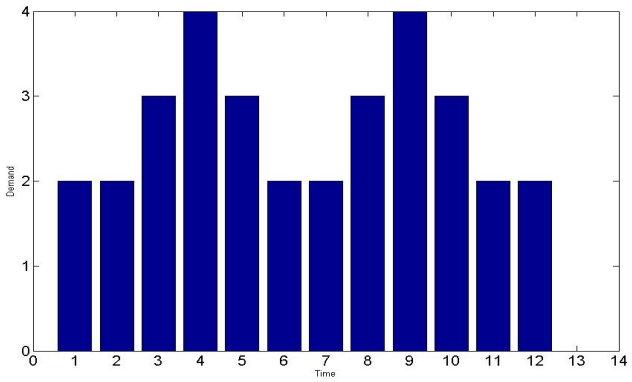


Figure 7: Demand of Instance

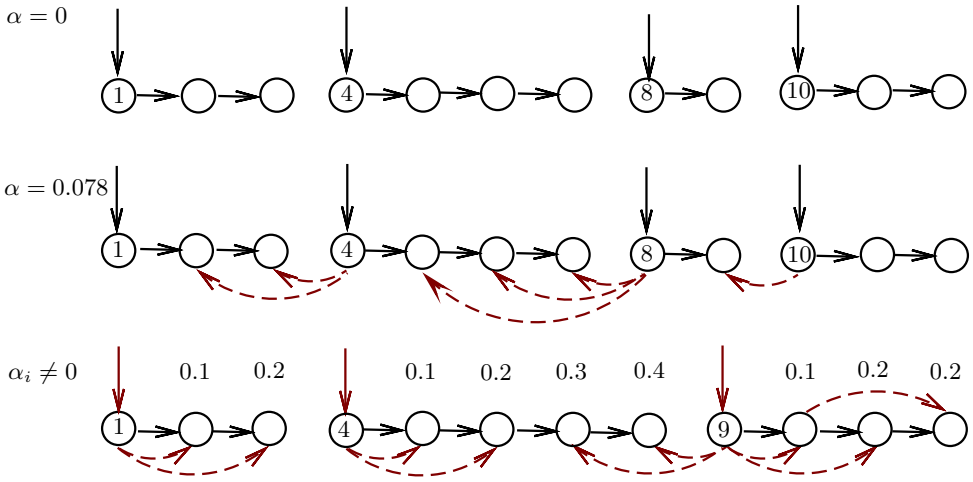


Figure 8: Example

head of the arc. In the third and most flexible setting, the additional flexibility is exploited to reduce the number of periods in which production takes place from four to three and to offer larger discounts (up to 40%) in periods where it is most advantageous to be able to deliver early or late. We see that except for a setup in the first period, the setups occur in periods with the highest demands and that in each production interval the discount factors monotonically increase. The reason for the latter is that within a production interval the per-unit holding cost increases for periods that are further away from the period in which production takes place and it thus becomes more and more advantageous to convert demand into flexible demand so as to reduce inventory holding costs.

2.6 A computational study

In Section 2.3, we have shown that ULS-PDF can be solved in polynomial time. However, we found that state-of-the-art quadratic integer programming solvers are able to solve medium-size instances in a reasonable amount of time. Therefore, we have used the quadratic integer programming solver of CPLEX 12.2 for all our computational experiments.

The main purpose of our computational study is to analyze the benefits of providing price discounts in return for delivery flexibility. Furthermore, we want to understand the difference between being able to deliver only in later periods or being able to deliver in both earlier and later periods, the impact of the size of the delivery window on instances with various periodicity, and the benefit of per-period price discounts versus a single price discount for all periods.

We use eight instances with constant demand over 36 time periods for our computational experiments. Four of the instances have demand $D_i = 2, 4, 6, 8$ for $i = 1, ..., n$ and four of the instances have the time between consecutive setups $TBS = 2, 4, 6, 9$ in their optimal solution of the ULS model. The base parameter settings for all instances are: $\Delta = 8$ and $p_i = 30$, $e_i = 100$, $g_i = 10$, and $h_i = 4$ for all periods i. The first set of instances allows us to examine the benefits of providing price discounts in return for delivery flexibility as a function of demand, and the second set of instances allows us to study the effect of the size of the delivery window, which intuitively is related to the time between consecutive setups.

Since we are interested in understanding the benefits of providing price discounts in return for delivery flexibility, we compare characteristics of optimal production plans of the variants of ULS introduced in this paper with the characteristics of an optimal production plan of the classical ULS. We are primarily interested in the profit associated with an optimal production plan, and because the profits associated with feasible production plans for a particular instance vary because of differences in setup costs, inventory holding costs, and discount costs, we provide the percentage decrease in the total of these controllable costs, i.e., $\delta_c^{rel} = \frac{c^{ULS} - c^{ULS-PDF}}{c^{ULS}}$, when we compare optimal production plans for different variants of ULS.

We have conducted a number of computational experiments with other demand profiles,

for example with a bi-modal demand profile, representing seasonal demand increases and peaks, but since the results were not that different from the constant demand case, we only present results for the constant demand case.

2.6.1 Pricing for delivery flexibility

In our first computational experiment, we study whether providing a discount in return for the option to deliver late is beneficial (a common discount factor in every period). A key characteristic of lot-sizing instances is the ratio of the per unit inventory holding cost and the setup cost since the primary challenge is to find the proper balance between setup costs and inventory holding costs. This ratio is reflected in an optimal production plan through the time between setups. The larger ratio of the per unit inventory holding cost and the setup cost, the smaller TBS in an optimal solution is to save on holding costs.

We first consider instances with varying ratios by keeping the setup cost fixed and varying the inventory holding cost. The results are shown in Figure 9.

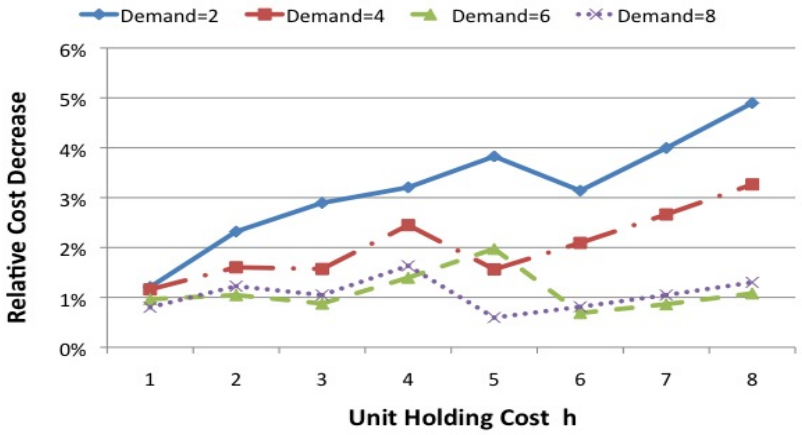


Figure 9: Relative cost decrease for varying holding cost

We see that the relative cost decreases reflect the jagged pattern observed in the lower bound derived in Section 2.4. As with the lower bound, the drops in the jagged pattern are the result of increases in the number of setups. The size of a drop (or the "jaggedness") is related to the number of additional setups. When the number of setups increases gradually, the value of δ_c^{rel} trends upward as expected.

Furthermore, we notice that for low demand levels, i.e., $D = 2$ and $D = 4$, the relative

cost decrease trends upward when holding costs increase, but that for high demand levels, i.e. $D = 6$ and $D = 8$, the relative cost decrease appears to be stable when holding costs increase. The reason that delivery flexibility looses some of its value for high demand levels is that there will be a relatively large number of setups and in each of the periods in which a setup occurs we "give away" revenue since we collect a discounted price for the flexible demand.

The above observation suggests that the size of an effective delivery window depends on the time between consecutive setups. Theorem 5 provides insight into this relationship. Theorem 5 shows that the benefits of offering a price discount in return for delivery flexibility depend on the optimal periodicity $\hat{k} = \sqrt{2e/Dh}$ and the relation between $\hat{k}$ and Δ, the delivery flexibility. Furthermore, it is established that the time between consecutive setups in an optimal solution is either $\lceil \hat{k} \rceil$ or $\lfloor \hat{k} \rfloor$. Next, we study this relationship empirically by varying the size of delivery window for instances with different times between consecutive setups. The results are shown in Figure 10.

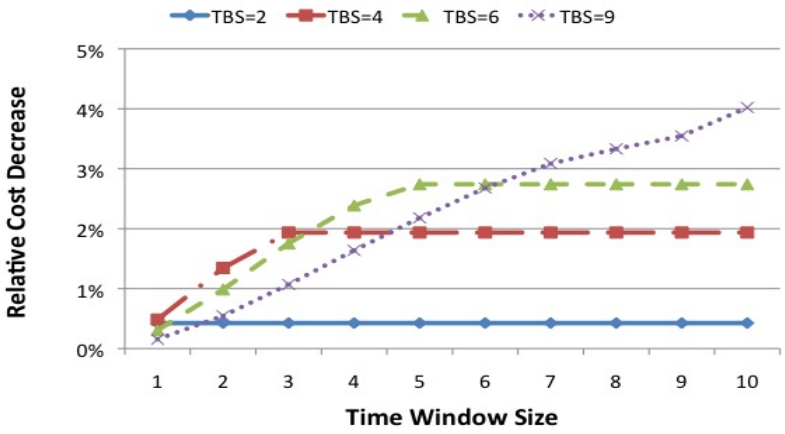


Figure 10: Relative cost decrease for varying delivery window Δ

As expected, the relative cost decrease initially improves when the size of the delivery window increases. However, increasing the size of the delivery window beyond the time between consecutive setups does not produce additional benefits for instances with $TBS = 2, 4, 6$. Note that as long as $\Delta < TBS$ increasing Δ increases the savings on holding cost while not affecting other costs or revenues, and therefore it always beneficial to use a larger

delivery window. When $\Delta \geq TBS$ and the setups in optimal solutions to ULS and ULS-PDF occur in the same periods, then none of the flexible demand will incur holding costs and thus increasing the delivery window has no effect. On the other hand, if the setups in optimal solutions to ULS and ULS-PDF do not occur in the same periods, increasing the size of the delivery window may create additional benefits, which depend on the setup cost, the unit holding cost and the price discount.

In line with our first experiment, we see that the largest benefits occur for instances with a large time between consecutive setups, which happens for small demands. Instances with a large time between consecutive setups have more periods that can benefit from delivery flexibility, and thus more holding costs can be saved.

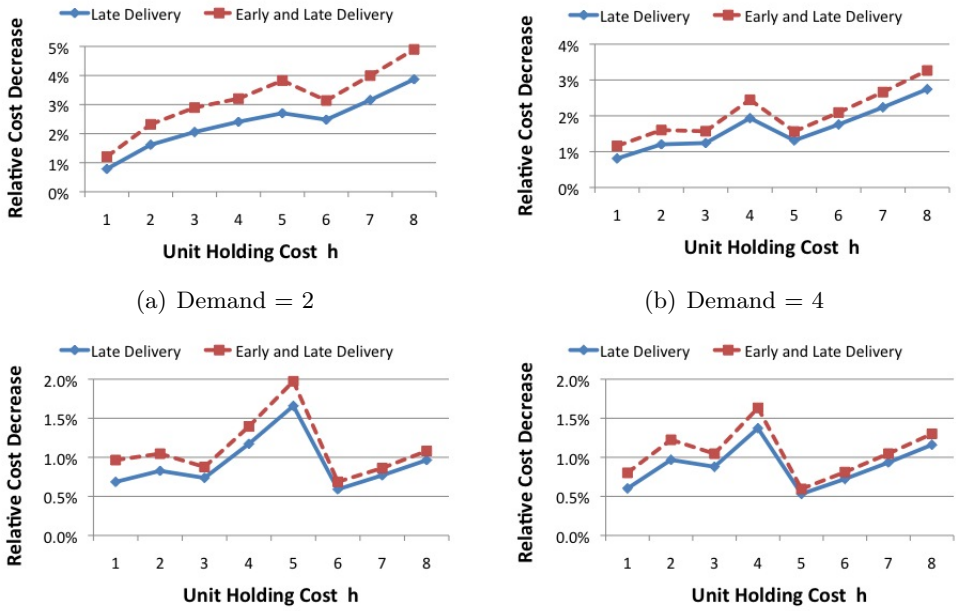
2.6.2 Delivery flexibility

Next, we study the value of being able to deliver in earlier and later periods as opposed to being able to deliver only in later periods. More specifically, we conduct experiments on the same instances with flexible demand d_i^f able to be delivered during periods $\{i - \Delta/2, ..., i + \Delta/2\}$, i.e. the lengths of the two delivery windows are the same.

In Figure 11 and Figure 12, we compare the results of delivering late with the results of delivering early and late for all eight instances.

The results show a clear benefit for being able to deliver early as well as late. The reason for this stems from the difference in benefits derived from delivering early and delivering late. Being able to deliver late only leads to a benefit if delivery takes place in a later period in which production occurs; in which case, we avoid inventory holding costs. If no such period exists, i.e., there is no period in the delivery window in which production occurs then there are no benefits. On the other hand, when we are able to deliver early, we can *always* reduce our inventory holding costs. If delivery takes place in an earlier period in which production occurs, we avoid inventory holding costs, but even if there is no period in the delivery window in which production occurs, we will still reduce our inventory holding costs by having delivery take place as early as possible.

In Figure 12, we see that for instances with $TBS = 4, 6, 9$, increasing the size of the



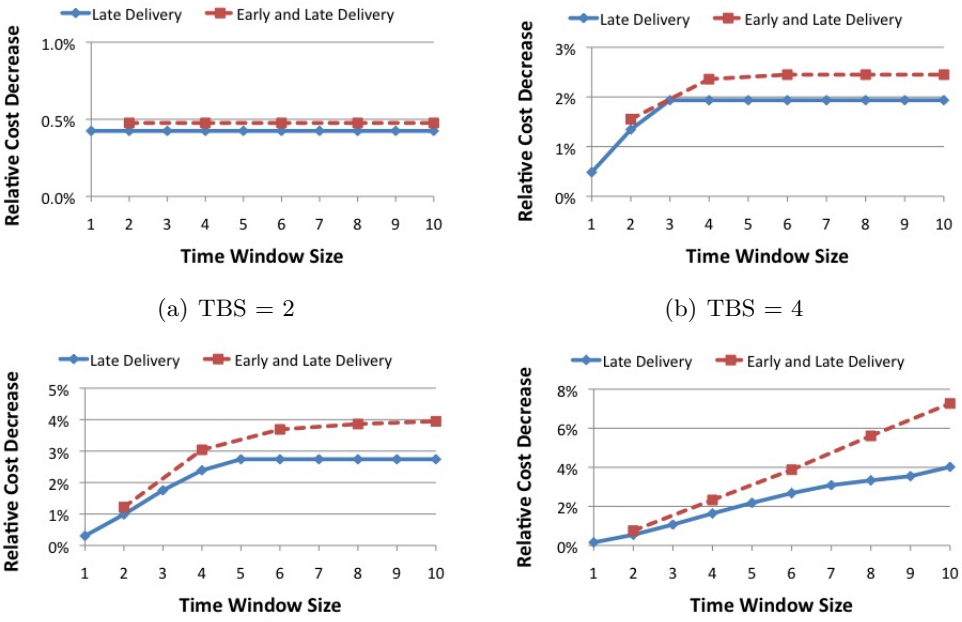
(a) Demand = 2

(b) Demand = 4

(c) Demand = 6

(d) Demand = 8

Figure 11: Delivery flexibility for varying holding cost



(a) TBS = 2

(b) TBS = 4

(c) TBS = 6

(d) TBS = 9

Figure 12: Delivery flexibility for varying delivery window size

delivery window continuous to be beneficial longer when delivery can take place early and late compared to when delivery can only take place late. When early and late deliveries are possible holding costs can be reduced in periods before and after a setup, thus whenever $\frac{\Delta}{2}$ is less than the time between consecutive setups.

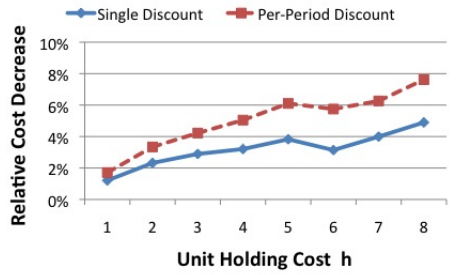
When we take a closer look at the optimal discount factors, we see that when deliveries can take place early and late, the price discount offered is always larger than the price discount offered when deliveries can only take place late. This is intuitive because there is always a benefit when being able to delivery early, but can also be seen as follows. For a fixed setup plan z, the profit function $P^z(\alpha)$ for early and late delivery is a quadratic function of α and has the same form as the profit function for late delivery only, i.e., as (20). In fact, the coefficients in the quadratic term and the constant term are exactly the same. The only difference is the coefficient in the linear term, i.e., the costs that are saved by exploiting delivery flexibility. Since in the last production interval there are no periods that can benefit from late delivery, the costs are smaller in the case with early and late deliveries. As a consequence, for each fixed z, the optimal price discount α is larger with early and late delivery. Since for all instances and for all delivery flexibilities, the optimal setup plans are the same, it follows that a larger discount is used when early and late deliveries can be made.

2.6.3 A single discount vs. per-period discounts

Finally, we study the value of offering per-period discounts as opposed to a single discount for all periods. We assume that the flexible demand of period i can be delivered during periods $\{i - \Delta/2, ..., i + \Delta/2\}$.

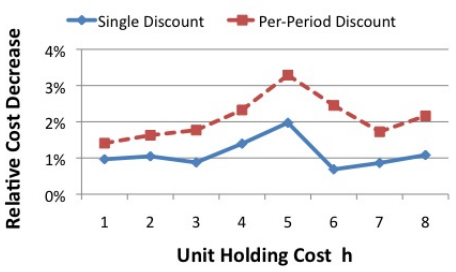
We compare the results of offering per-period discounts with the results of offering a single, common discount for all periods for all eight instances. The results are presented in Figure 13 and Figure 14.

The benefits of offering per-period discounts are substantial. The decrease in costs are significantly larger with per-period discounts. In all instances, using a per-period discount results in a relative cost decrease of at least 40% more than using a single discount factor.

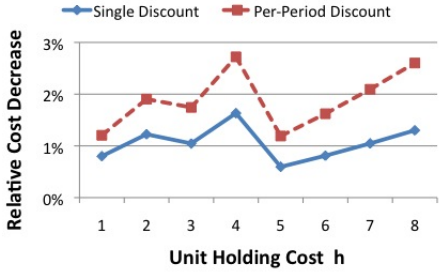


(a) Demand = 2

(b) Demand = 4

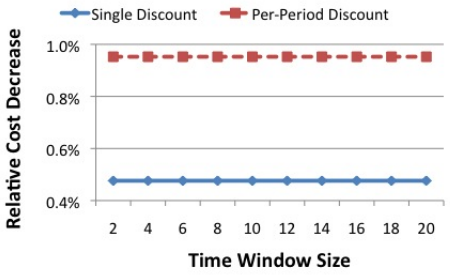


(c) Demand = 6

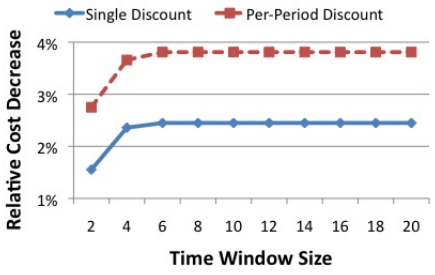


(d) Demand = 8

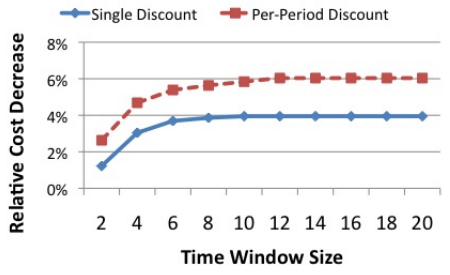
Figure 13: Discount factors for varying holding cost



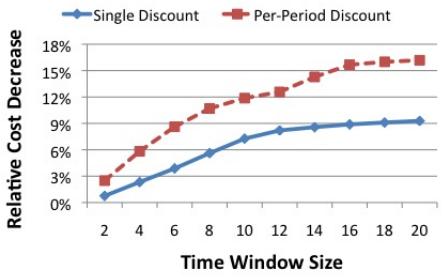
(a) TBS = 2



(b) TBS = 4



(c) TBS = 6



(d) TBS = 9

Figure 14: Discount factors for varying delivery window size

A closer examination reveals that, as expected, both the periods in which discounts are offered and the size of the discounts are chosen so as to make most effective use of periods in which production takes place. In all instances, the discount values monotonically increase within the production intervals so as to reduce inventory holding costs. By offering a larger discount, a larger portion of the demand becomes flexible and inventory holding costs are reduced. Also, even though the average of the per-period discounts is only slightly higher than the single discount, it leads to significantly larger relative controllable cost decreases. By choosing the discounts carefully, we no longer give away revenue unnecessarily and reduce holding costs when it is most advantageous.

CHAPTER III

CORNER RELAXATION FOR MKP

3.1 Introduction

Consider the multi-dimensional knapsack problem (MKP), which can be stated as:

$$\begin{aligned} f(b) = \min \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, i = 1, ..., m \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{39}$$

where $a_{ij} \in \mathbb{Z}_+$, $b = \{b_1, ..., b_m\}^T \in \mathbb{Z}_+^m$, $c = \{c_1, ..., c_n\} \in \mathbb{R}_+^n$, and $f(b)$ is the value function of MKP.

When $m = 1$, MKP is called the knapsack problem (KP). Let $a_j \in \mathbb{Z}_+$ be the weight of item j, $c_j \in \mathbb{R}_+$ be the cost of item j, for $j = 1, ..., n$, and $b \in \mathbb{Z}_+$ be the minimum requirement on the total weight. The objective is to minimize the total cost of items, such that the overall weight is at least b. As a special case of MKP, the problem can be formulated as:

$$\begin{aligned} F(b) = \min \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_j x_j \geq b \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{40}$$

where $c_1/a_1 \leq c_2/a_2 \leq ... \leq c_n/a_n$.

Methods for rapidly solving KP have been studied extensively, see Martello and Toth [52]. In particular, dynamic programming algorithms are often used to solve KP. Let $f_k(b)$ be the optimal objective value using only the first k items. Then an optimal solution to $f_k(b)$ either only uses the first $k-1$ items, which yields $f_k(b) = f_{k-1}(b)$, or uses item k at least once, which gives $f_k(b) = f_k(b - a_k) + c_k$. Thus the value of $f_k(b)$ can be obtained by

solving $f_i(t)$, for $i \leq k$ and $t \leq b$, where the recursive equation is given by:

$$f_k(y) = \min\{f_{k-1}(y), f_k(y - a_k) + c_k\}, \text{ and } f_0(x) = 0 \text{ for } x \leq 0,$$

for $k = 1, ..., n$ and $y = 0, ..., b$. Since $F(b) = f_n(b)$, the time complexity of the dynamic programming algorithm is $O(nb)$.

Assume that $c_1/a_1 < \min_{j=2,...,n}\{c_j/a_j\}$ and $b > 0$. Thus an optimal linear relaxation solution of KP is x^* with $x_1^* = b/a_1 > 0$ and $x_i^* = 0$ for $i = 2, ..., n$. The corner relaxation relaxes non-binding constraints at x^*, thus relaxing non-negativity constraint $x_1 \geq 0$ gives the corner relaxation of KP (CR-KP), which can be stated as

$$\begin{aligned} CR\text{-}KP(b) = \min \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_j x_j \geq b \\ & x_j \geq 0, j = 2, ..., n \\ & x \text{ integer}, \end{aligned}$$

where $CR\text{-}KP(b)$ is called the corner relaxation bound with right-hand side b of the constraint. Since the corner relaxation bound is a function of b, the largest relative difference between $CR\text{-}KP(b)$ and $F(b)$ for all $b \geq 0$, which is defined as

$$\max_{b \geq 0: F(b) \neq 0} \{|F(b) - CR\text{-}KP(b)|/F(b)\},$$

is called the worst-case corner relaxation gap. The smaller the worst-case gap, the tighter the corner relaxation.

The computational complexity of $F(b)$ can be reduced using two nice properties of KP:

- Gilmore and Gomory [27] described a *periodic property* of the value function $F(b)$, which can be stated as:

  $$\text{there exists a } b^* > 0, \text{ such that when } b \geq b^*, F(b) = c_1 + F(b - a_1).$$

  Thus, there is always an optimal solution with $x_1 \geq 1$ when b is large enough. Therefore, the computation of $F(b)$ can be reduced to $F(b')$, for some $b' < b^*$, by reducing multiple times of a_1 from b.

The periodic property of $F(b)$ implies that when b is large enough, $x_1 \geq 1$ is always satisfied by an optimal solution. Thus, the constraint $x_1 \geq 0$ is redundant for the optimal solution. However, the periodic property does not guarantee that constraint $x_1 \geq 0$ is redundant without showing that there is no better solution with $x_1 < 0$.

- Zhu [65] gave an *asymptotic property* of the knapsack problem, which can be stated as:

$$\text{there exists a } \bar{b}, \text{ such that when } b \geq \bar{b}, CR\text{-}KP(b) = F(b).$$

The asymptotic property implies that to solve the knapsack problem for large values of b, it is sufficient to solve its corner relaxation. That is, even when constraint $x_1 \geq 0$ is relaxed, there exists an optimal solution with $x_1 \geq 0$.

Since using the periodic property helps to reduce computational complexity for KP with large right-hand side, it is natural to ask if the periodic property can be extended to MKP? For MKP with a large number of constraints, the non-binding knapsack constraints at the optimal LP relaxation solution are relaxed by the corner relaxation. Thus when the asymptotic property holds, to solve MKP is equivalent to solving the corner relaxation with a smaller number of constraints. Therefore, it is meaningful to extend the asymptotic property to MKP.

In this chapter, we investigate conditions under which the periodic and asymptotic properties hold for MKP. For MKP with large right-hand side $b = (b_1, ..., b_m)^T$ satisfying certain conditions, we show that the value function $f(b)$ has a periodic property which is extended from KP, and can be calculated by solving MKP with a smaller right-hand side of the constraints. In addition, we define an asymptotic condition under which solving MKP is equivalent to solving its corner relaxation. Since the corner relaxation may contain a smaller number of constraints than the original problem, the computational complexity for solving MKP can be reduced. To understand the overall strength of the corner relaxation of MKP, we provide the worst-case corner relaxation gap when the asymptotic property does not hold.

Although the periodic and asymptotic properties have been partially explored by previous work including Gomory [32] and Zhu [65], our results provide alternative conditions that are specific for MKP. For example, Zhu [65] discussed conditions under which the corner relaxation for KP is tight by solving a sequence of IP problems. In our study, however, we provide an analytical condition that only depends on the constraint coefficients. Gomory [32] considered the asymptotic property in the context of general MIPs, which can also be applied to MKP. Here we take a different approach by considering the asymptotic property of MKP for some special cases, and provide alternative conditions under which the corner relaxation is tight. In addition, when the corner relaxation of MKP is not tight, we study the worst-case corner relaxation bounds, and give examples to demonstrate that the bounds we developed can be tight. Furthermore, we extend the periodic property from KP to MKP.

The remainder of the chapter is structured as follows: In Section 3.2, we present the periodic and asymptotic properties for the knapsack problem and conduct a worst-case analysis on the corner relaxation gap. In Section 3.3, we extend the periodic property to MKP with two constraints, which we call 2-KP, and provide a worst-case corner relaxation gap under certain conditions. Finally, in Section 3.4, we extend the periodic property to the general MKP.

3.2 The periodic property and corner relaxation for KP

We first make some clarifications on coefficients of the objective function and the constraints. Let $F(y) = \min\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \geq y, x \in \mathbb{Z}_+^n\}$ be the value function of KP with right-hand side y. If $c_j \leq 0$ and $a_j > 0$, we can make x_j arbitrarily large, therefore, assume that $c_j > 0$ for all j. If $a_j = 0$, there is no sense in using item j, thus assume $a_j > 0$ for all j. Denote the cost-to-weight ratio of item j by $\rho_j = \frac{c_j}{a_j}$ for $j = 1, ..., n$, and assume that $\rho_1 \leq \rho_2 \leq ... \leq \rho_n$. Obviously, $F(y) = 0$ when $y \leq 0$, since $x = 0$ is a feasible solution. Therefore, we are only interested in $F(y)$ with $y > 0$. We now formally state and prove the main result on the periodic property of $F(y)$ showed by Gilmore and Gomory [27].

Theorem 7 (Gilmore and Gomory [27]) *If $\rho_1 < \rho_2$, then for $y \geq y^* = c_1/(\rho_2 - \rho_1)$, $F(y) = F(y - a_1) + c_1$. That is, $F(y)$ is periodic with periodicity of a_1 and increment of c_1.*

Proof For $x_1 \geq 1$, $\bar{x}_1 = \lceil y/a_1 \rceil$, $\bar{x}_j = 0$ for $j = 2, ..., n$ is obviously a feasible solution to KP. Thus $c\bar{x} = c_1\lceil y/a_1 \rceil \leq c_1(y/a_1 + 1) = \rho_1 y + c_1$ is a valid upper bound for $F(y)$ when $x_1 \geq 1$.

If we force $x_1 = 0$, $\rho_2 y$ is a valid lower bound of $F(y)$, since $x_2^* = y/a_2, x_j^* = 0$ for $j = 3, ..., n$ is an optimal LP solution.

Therefore, there exists an optimal solution with $x_1 \geq 1$ if the upper bound of $F(y)$ when $x_1 \geq 1$ is less than or equal to the lower bound of $F(y)$ when $x_1 = 0$, that is, $\rho_1 y + c_1 \leq \rho_2 y$. This yields $y \geq c_1/(\rho_2 - \rho_1)$. □

Note that the value of y^* is determined by the first two items through the difference between ρ_2 and ρ_1. Thus, y^* is small when $\rho_1 \ll \rho_2$, which means that the first item should be used more often in an optimal solution in such a case.

In addition, Theorem 7 implies that when $y \geq y^*$, there exists an optimal solution with $x_1 \geq 1$, thus $x_1 \geq 0$ is redundant for the optimal solution. We next show that when constraint $x_1 \geq 0$ is relaxed, which leads to the corner relaxation $CR\text{-}KP(y)$, $CR\text{-}KP(y)$ is bounded for all $y \geq 0$.

Proposition 8 *$CR\text{-}KP(y)$ is bounded for all $y \geq 0$.*

Proof $CR\text{-}KP(y)$ is bounded if its linear relaxation $LCR(y)$ is bounded. $LCR(y)$ is unbounded if and only if there is an unbounded direction in the feasible region, that is, there exists x with $x_j \geq 0$ for $j = 2, ..., n$, such that

$$\sum_{j=1}^{n} a_j x_j \geq 0 \text{ and } \sum_{j=1}^{n} c_j x_j < 0.$$

This is equivalent to

$$\sum_{j=2}^{n} a_j x_j \geq -x_1 a_1 \text{ and } \sum_{j=2}^{n} c_j x_j < -x_1 c_1. \tag{41}$$

Note that $\rho_1 \leq \rho_2 \leq ... \leq \rho_n$, thus $c_j \geq a_j \rho_1$ for all $j = 2, ..., n$. Therefore, for all x satisfying $x_j \geq 0$ for $j = 2, ..., n$,

$$\sum_{j=2}^{n} c_j x_j \geq \rho_1 \sum_{j=2}^{n} a_j x_j \geq -\rho_1 x_1 a_1 = -x_1 c_1,$$

which contradicts (41). Therefore, $LCR(y)$ is always bounded, thus $CR\text{-}KP(y)$ is also bounded. □

Although this result excludes the possibility of incurring an unbounded corner relaxation of KP, it does not tell us much about the strength of the corner relaxation bound. We now consider the conditions under which the corner relaxation bound is tight.

Proposition 9 *If $\rho_1 < \rho_2$, then for $y \geq \bar{y} = a_1(2\rho_1 - \rho_2)/(\rho_2 - \rho_1)$, $CR\text{-}KP(y) = F(y)$, that is, the constraint $x_1 \geq 0$ is redundant.*

Proof For $x_1 \geq 0$, $\bar{x}_1 = \lceil y/a_1 \rceil$, $\bar{x}_j = 0$ for $j = 2, ..., n$ is obviously a feasible solution to the corner relaxation. Thus, $c\bar{x} = c_1\lceil y/a_1 \rceil \leq c_1(y/a_1 + 1) = \rho_1 y + c_1$ is a valid upper bound on $CR\text{-}KP(y)$ when $x_1 \geq 0$.

For $x_1 < 0$, let $\bar{x}$ be an optimal solution to the corner relaxation with $x_1 < 0$. Then a lower bound of $CR\text{-}KP(y)$ with $x_1 < 0$ can be obtained by solving the following problem:

$$\begin{aligned} c_1\bar{x}_1 + \min \ & \sum_{j=2}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=2}^{n} a_j x_j \geq y - a_1\bar{x}_1 \\ & x_j \geq 0, \ j = 2, ..., n. \end{aligned} \tag{42}$$

An optimal solution to problem (42) is $\hat{x}$ with $\hat{x}_2 = (y - a_1\bar{x}_1)/a_2$, $\hat{x}_j = 0$ for $j = 3, ..., n$. Thus a lower bound of $CR\text{-}KP(y)$ is $c_1\bar{x}_1 + c_2(y - a_1\bar{x}_1)/a_2 = c_1\bar{x}_1 + \rho_2(y - a_1\bar{x}_1) = a_1(\rho_1 - \rho_2)\bar{x}_1 + \rho_2 y \geq \rho_2 y + a_1(\rho_2 - \rho_1)$, since $\rho_1 - \rho_2 \leq 0$ and $\bar{x}_1 \leq -1$.

Therefore, if $\rho_2 y + a_1(\rho_2 - \rho_1) \geq \rho_1 y + c_1$, any optimal solution must satisfy $x_1 \geq 0$, thus relaxing $x_1 \geq 0$ does not affect the value of $F(y)$, and the corner relaxation is tight. This yields $y \geq c_1/(\rho_2 - \rho_1) - a_1 = a_1(2\rho_1 - \rho_2)/(\rho_2 - \rho_1)$ when $\rho_2 > \rho_1$. □

This result implies that for sufficiently large y, solving the corner relaxation is sufficient for solving the original KP. We next show that for smaller y, when the corner relaxation is not tight, the relative gap of the corner relaxation, which is defined by $\frac{F(y) - CR\text{-}KP(y)}{F(y)}$, is also bounded.

Proposition 10 *If $\rho_1 < \rho_2$, then for y satisfying $0 \le y \le \bar{y} = a_1(2\rho_1 - \rho_2)/(\rho_2 - \rho_1)$, the relative gap of the corner relaxation satisfies*

$$\frac{F(y) - CR\text{-}KP(y)}{F(y)} \le \frac{a_1}{y + a_1} - \frac{\rho_2 - \rho_1}{\rho_1}. \tag{43}$$

Proof Note that when $0 \le y \le a_1(2\rho_1 - \rho_2)/(\rho_2 - \rho_1)$, the right-hand side of inequality (43) is non-negative. If there exists an optimal solution to the corner relaxation satisfying $x_1 \ge 0$, then it is also an optimal solution to KP, thus $F(y) - CR\text{-}KP(y) = 0$. Otherwise, as shown in the proof of Proposition 9, $CR\text{-}KP(y) \ge \rho_2 y + a_1(\rho_2 - \rho_1)$. Since $F(y) \le \rho_1 y + c_1$ by rounding up the optimal linear relaxation solution, the corner relaxation gap satisfies

$$\frac{F(y) - CR\text{-}KP(y)}{F(y)} = 1 - \frac{CR\text{-}KP(y)}{F(y)} \le 1 - \frac{\rho_2 y + a_1(\rho_2 - \rho_1)}{\rho_1 y + c_1} = \frac{a_1}{y + a_1} - \frac{\rho_2 - \rho_1}{\rho_1}.$$

□

Proposition 9 and 10 provide a worst-case analysis on the quality of corner relaxation bound for all y. As y increases, the worst-case bound decreases, and when y is sufficiently large, the corner relaxation becomes tight. Both the value of $\bar{y}$ and the worst-case gap when $y \le \bar{y}$ are determined by the first two items only.

It is meaningful to compare the corner relaxation bound with the linear relaxation bound of KP, since the linear bound is the most often used lower bound. We do so by comparing the worst-case gap of the corner relaxation presented in (43) with the worst-case gap of the linear relaxation. Note that the linear relaxation has value function $LP(y) = \rho_1 y$, thus the linear relaxation gap satisfies

$$\frac{F(y) - LP(y)}{F(y)} \le \frac{(\rho_1 y + c_1) - \rho_1 y}{\rho_1 y + c_1} = \frac{a_1}{y + a_1}. \tag{44}$$

Since the linear relaxation of CR-KP and the linear relaxation of KP have the same optimal solution, we must have $\frac{F(y) - CR\text{-}KP(y)}{F(y) - LP(y)} \le 1$. The following example shows that the worst-case gaps of (43) and (44) can be tight, and the corner relaxation bound can be as weak as the linear relaxation bound.

Example Consider the problem

$$\begin{aligned} F(y) = \min\ & (n-1)x_1 + nx_2 \\ \text{s.t. } & nx_1 + (n+1)x_2 \geq y \\ & x_1, x_2 \geq 0,\ \text{integer}, \end{aligned}$$

with $y = 1$ and $n > 1$. Obviously, $y = 1 < a_1(2\rho_1 - \rho_2)/(\rho_2 - \rho_1)$, where $\rho_1 = (n-1)/n < \rho_2 = n/(n+1)$ for all $n > 1$. An optimal solution to the problem is $x_1 = 1$ and $x_2 = 0$, therefore $F(1) = n - 1$. The optimal linear solution is $x_1^* = 1/n$, $x_2^* = 0$, thus $LP(1) = (n-1)/n$. Therefore, the linear relaxation gap satisfies $\frac{F(1)-LP(1)}{F(1)} = \frac{n-1-(n-1)/n}{n-1} \to 1$ as $n \to +\infty$. Since $a_1/(y+a_1) = n/(n+1) \to 1$, this indicates that the worst-case linear relaxation gap in (44) is tight.

On the other hand, the optimal solution to the corner relaxation, which allows x_1 to be negative, is $\bar{x}_1 = -1$, $\bar{x}_2 = 1$, thus $CR\text{-}KP(1) = 1$. We have the corner relaxation gap $\frac{F(1)-CR\text{-}KP(1)}{F(1)} = (n-2)/(n-1) \to 1$ as $n \to +\infty$. Since $\frac{a_1}{y+a_1} - \frac{\rho_2-\rho_1}{\rho_1} = n/(n+1) - 1/(n^2-1) \to 1$ as $n \to +\infty$, the worst-case corner relaxation gap in (43) is also tight.

In addition, $LP(1) = (n-1)/n \to 1 = CR\text{-}KP(1)$, which shows that the corner relaxation bound is not stronger than the linear relaxation bound as $n \to +\infty$.

3.3 Periodic property and corner relaxation for 2-KP

Next, we extend results of the value function and corner relaxation to MKP with two constraints, which we call 2-KP. The formulation of 2-KP can be stated as:

$$\begin{aligned} f(y_1, y_2) = \min\ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t. } & \sum_{j=1}^{n} a_j^1 x_j \geq y_1 \\ & \sum_{j=1}^{n} a_j^2 x_j \geq y_2 \\ & x \in \mathbb{Z}_+^n. \end{aligned} \tag{45}$$

If $c_j = 0$ and $a_j^1 > 0$, the first constraint can be eliminated by using the first item as much as possible, thus the problem can be reduced to KP. Therefore, assume that $c_j > 0$ for all

j. Let $\rho_j^1 = c_j/a_j^1$ and $\rho_j^2 = c_j/a_j^2$ for all $j = 1, ..., n$ be the cost-to-weight ratios on the two constraints respectively. To simplify, assume that all constraint coefficients are positive so that ρ_j^1 and ρ_j^2 for $j = 1, ..., n$ are well defined, and we only consider the problem for $y_1 > 0$ and $y_2 > 0$.

For KP, the periodicity of the value function equals a_1, where $\rho_1 = \min_j\{\rho_j\}$, and the items with smaller cost-to-weight ratios are more likely to appear in an optimal solution. For 2-KP, we need to consider the impact of items on both constraints to evaluate the importance of items. A way to start is to study the solution of the linear relaxation under different optimal linear programming (LP) bases. An optimal LP basis is composed of two basic variables, including possibly one of the slack variables s_1 or s_2, where $\sum_{j=1}^n a_j^1 x_j - s_1 = y_1$, $\sum_{j=1}^n a_j^2 x_j - s_2 = y_2$, $s_1, s_2 \in \mathbb{Z}_+$. The variables not in the basis are non-basic variables.

Denote the 2 by 2 matrix (A_i, A_j) by B_{ij}, where A_i is the ith column of the constraint matrix. Let $B_{iy} = (A_i, y)$ and $B_{yi} = (y, A_i)$, where y is the right-hand-side vector of the constraints. We use B to indicate both the determinant of a matrix and the matrix itself interchangeably for simplicity. Assume that the optimal linear solution of 2-KP is non-degenerate, since when the linear solution is degenerate, it is much more complicated to analyze the conditions for a basis to be optimal.

3.3.1 Preliminaries

To prepare for the following analysis, in this section, we present the conditions for an LP basis to be feasible and optimal. Given any non-degenerate LP basis of the linear relaxation of 2-KP, there is a unique solution corresponding to the basis. The conditions that the objective coefficients and constraint coefficients must satisfy for the solution to be feasible and optimal are called feasibility and optimality conditions for the basis, respectively.

The feasibility and optimality conditions for a basis can be distinguished by the number of slack variables in the basis. We use basis $\{x_1, s_1\}$ to indicate that exactly one of the slack variables is in the basis, and use basis $\{x_1, x_2\}$ to indicate that no slack variables are in the basis. We next give the feasibility and optimality conditions for basis $\{x_1, s_1\}$ and basis $\{x_1, x_2\}$.

Proposition 11 • *$\{x_1, s_1\}$ is a non-degenerate optimal basis if the following conditions are satisfied:*

– *Feasibility condition, which can be stated as:*

$$y_1/y_2 \leq a_1^1/a_1^2. \tag{46}$$

That is, (y_1, y_2) must be in the cone spanned by A_1 and the y_2-axis.

– *Optimality condition, which can be stated as:*

$$\rho_1^2 = \min_{k=1,...,n} \rho_k^2. \tag{47}$$

That is, the column A_1 must have the smallest cost-to-weight ratio in the second constraint.

• *$\{x_1, x_2\}$ is a non-degenerate optimal basis if the following conditions are satisfied:*

– *Feasibility condition, which can be stated as:*

$$a_2^1/a_2^2 \leq y_1/y_2 \leq a_1^1/a_1^2. \tag{48}$$

That is, (y_1, y_2) must be in the cone spanned by A_1 and A_2.

– *Optimality condition, which can be stated as:*

$$\begin{gathered} \rho_1^1 \leq \rho_2^1 \ , \ \rho_2^2 \leq \rho_1^2, \\ \text{and } c_k \geq A_{12}^2 a_k^1 + A_{12}^1 a_k^2 \text{ for } k = 1, ..., n, \end{gathered} \tag{49}$$

where

$$\begin{aligned} A_{12}^1 &= \frac{c_2 a_1^1 - c_1 a_2^1}{B_{12}} = \frac{1/\rho_1^1 - 1/\rho_2^1}{B_{12}/c_1 c_2} \geq 0 \text{ and} \\ A_{12}^2 &= \frac{c_1 a_2^2 - c_2 a_1^2}{B_{12}} = \frac{1/\rho_2^2 - 1/\rho_1^2}{B_{12}/c_1 c_2} \geq 0. \end{aligned} \tag{50}$$

Thus, in the first constraint, column A_1 has a smaller cost-to-weight ratio and in the second constraint, column A_2 has a smaller cost-to-weight ratio.

Proof Given a non-degenerate basis B, the corresponding LP solution can be stated as $x_B = B^{-1}y$ and $x_i = 0$ for $i \notin B$. The solution is a feasible solution if $x_B \geq 0$. The solution is optimal if the reduced costs of all variables are no less than 0, that is, the objective value increases when a non-basic variable enters the basis.

- For basis $\{x_1, s_1\}$, the corresponding solution is feasible if

$$B^{-1}y = \begin{bmatrix} 0 & 1/a_1^2 \\ -1 & a_1^1/a_1^2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_2/a_1^2 \\ -y_1 + a_1^1 \cdot y_2/a_1^2 \end{bmatrix} \geq 0,$$

that is, $y_1/y_2 \leq a_1^1/a_1^2$.

The solution corresponding to basis $\{x_1, s_1\}$ is optimal if

$$\bar{c}_k = c_k - c_B B^{-1} A_k = c_k - \begin{bmatrix} c_1 & 0 \end{bmatrix} \begin{bmatrix} a_k^2/a_1^2 \\ -a_k^1 + a_1^1 \cdot a_k^2/a_1^2 \end{bmatrix} = c_k - a_k^2 \rho_1^2 \geq 0,$$

where $\bar{c}_k$ is the reduced cost of variable x_k and s_i, for $k = 1, ..., n$ and $i = 1, 2$. That is, $\rho_1^2 = \min_{k=1,...,n} \rho_k^2$.

Therefore $\{x_1, s_1\}$ is a non-degenerate optimal basis if and only if

$$y_1/y_2 \leq a_1^1/a_1^2 \text{ and } \rho_1^2 = \min_{k=1,...,n} \rho_k^2.$$

- For basis $\{x_1, x_2\}$, the corresponding solution is feasible if

$$B^{-1}y = \frac{1}{B_{12}} \begin{bmatrix} a_2^2 & -a_2^1 \\ -a_1^2 & a_1^1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{B_{12}} \begin{bmatrix} y_1 a_2^2 - y_2 a_2^1 \\ y_2 a_1^1 - y_1 a_1^2 \end{bmatrix} = \frac{1}{B_{12}} \begin{bmatrix} B_{y2} \\ B_{1y} \end{bmatrix} \geq 0,$$

which is equivalent to

$$a_2^1/a_2^2 \leq y_1/y_2 \leq a_1^1/a_1^2.$$

The solution corresponding to basis $\{x_1, x_2\}$ is optimal if

$$\begin{aligned} &\bar{c}_k = c_k - c_B B^{-1} A_k \geq 0 \\ &\Leftrightarrow c_k - \begin{bmatrix} c_1 & c_2 \end{bmatrix} \times \frac{1}{B} \begin{bmatrix} a_2^2 & -a_2^1 \\ -a_1^2 & a_1^1 \end{bmatrix} \begin{bmatrix} a_k^1 \\ a_k^2 \end{bmatrix} \geq 0 \\ &\Leftrightarrow c_k - \frac{B_{k2}}{B} c_1 - \frac{B_{1k}}{B} c_2 \geq 0, \end{aligned} \tag{51}$$

where $\bar{c}_k$ is the reduced cost of variable x_k and s_i, for $k = 1, ..., n$ and $i = 1, 2$.

For a slack variable, (51) is equivalent to

$$\bar{c}_{s_1} = 0 - \frac{1}{B}(-c_1 a_2^2 + c_2 a_1^2) \geq 0 \text{ and } \bar{c}_{s_2} = 0 - \frac{1}{B}(c_1 a_2^1 - c_2 a_1^1) \geq 0,$$

which can be rewritten as

$$\rho_1^1 \leq \rho_2^1 \text{ and } \rho_2^2 \leq \rho_1^2. \tag{52}$$

For an integer variable x_k, (51) can be rewritten as

$$\bar{c}_k = c_k - A_{12}^2 a_k^1 - A_{12}^1 a_k^2 \geq 0, \tag{53}$$

where

$$A_{12}^1 = \frac{c_2 a_1^1 - c_1 a_2^1}{B_{12}} = \frac{1/\rho_1^1 - 1/\rho_2^1}{B_{12}/c_1 c_2} \geq 0 \text{ and } A_{12}^2 = \frac{c_1 a_2^2 - c_2 a_1^2}{B_{12}} = \frac{1/\rho_2^2 - 1/\rho_1^2}{B_{12}/c_1 c_2} \geq 0. \tag{54}$$

Therefore $\{x_1, x_2\}$ is a non-degenerate optimal basis if and only if

$$a_2^1/a_2^2 \leq y_1/y_2 \leq a_1^1/a_1^2,$$
$$\text{and } \rho_1^1 \leq \rho_2^1 \text{ , } \rho_2^2 \leq \rho_1^2, c_k \geq A_{12}^2 a_k^1 + A_{12}^1 a_k^2 \text{ for } k = 1, ..., n.$$

□

Since the feasibility and optimality conditions only depend on the objective and constraint matrices, they can be used to divide the space $(y_1, y_2) \in \mathbb{R}_+^2$ into non-overlapping cones. In addition, each cone corresponds to a unique basis and is spanned by the corresponding column vectors of the constraints as shown in Figure 15. In Figure 15, for (y_1, y_2) in the cone spanned by A_3 and y_2-axis, the problem has an optimal basis $\{x_3, s_1\}$; for (y_1, y_2) in the cone spanned by A_3 and A_2, the problem has an optimal basis $\{x_3, x_2\}$; for (y_1, y_2) in the cone spanned by A_2 and y_1-axis, the problem has an optimal basis $\{x_2, s_2\}$.

We next discuss the periodic property and corner relaxation for 2-KP under the two cases, where one or none of the slack variables is in the basis using the corresponding feasibility and optimality conditions.

3.3.2 The optimal basis $\{x_1, s_1\}$

We first analyze the case where the optimal basis includes a slack variable. WLOG, assume that $y_1/y_2 \leq a_1^1/a_1^2$ and $\rho_1^2 < \rho_2^2 \leq ... \leq \rho_n^2$, then by Proposition 11, $\{x_1, s_1\}$ is the optimal basis.

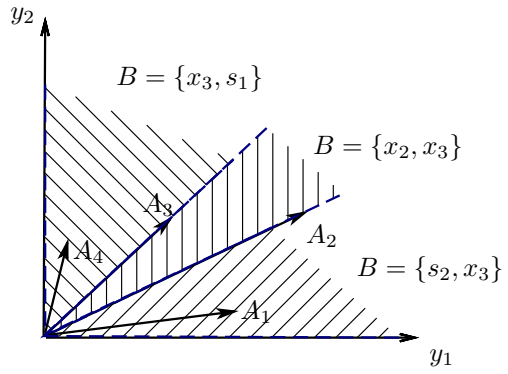


Figure 15: Cones and basis

Given a m by m matrix B, let K_B by a cone in $\mathbb{R}^m$, which is spanned by the column vectors of B. The periodic property of the value function $f(y_1, y_2)$ can be derived by similar techniques used in section 3.2. Thus, when the right-hand side of the constraints are sufficiently large, the value function is periodic with periodicity defined by a column with small cost-to-weigh ratio on one constraint.

Proposition 12 *For $y_2 \geq y^* = c_1/(\rho_2^2 - \rho_1^2)$ and $(y_1, y_2) \in K_{B_1}$, where B_1 is the optimal LP basis when x_1 is fixed to zero, $f(y_1, y_2)$ is periodic with periodicity of A_1 and increment of c_1, that is, $f(y_1, y_2) = c_1 + f(y_1 - a_1^1, y_2 - a_1^2) = c_1 + f(y - A_1)$. Thus $f(y_1, y_2)$ can be calculated by recursively reducing A_1 from $y = (y_1, y_2)$.*

Proof For $x_1 \geq 1$, $\bar{x}_1 = \lceil y_2/a_1^2 \rceil$, $\bar{x}_i = 0$, $i = 2, ..., n$ is a feasible solution to 2-KP. Therefore, a valid upper bound on $f(y_1, y_2)$ is $c_1 \lceil y_2/a_1^2 \rceil \leq c_1(y_2/a_1^2 + 1) = \rho_1^2 y_2 + c_1$.

If we force $x_1 = 0$, consider the linear relaxation of problem (45) with x_1 fixed at 0. Suppose an optimal linear relaxation solution is $x_i^* = B_{yj}/B_{ij}$, $x_j^* = B_{iy}/B_{ij}$ and $x_k^* = 0$, for $k \neq i, j$, where $B_{ij} > 0$ is the optimal LP basis. Thus by Proposition 11, we must have $y \in K_{B_{ij}} = K_{B_1}$, which is a cone spanned by A_i and A_j. Then a lower bound for $f(y_1, y_2)$ when $x_1 = 0$ is $c_i B_{yj}/B_{ij} + c_j B_{iy}/B_{ij}$.

Thus, to have an optimal solution with $x_1 \geq 1$, a sufficient condition is

$$\rho_1^2 y_2 + c_1 \leq c_i B_{yj}/B_{ij} + c_j B_{iy}/B_{ij},$$

which is equivalent to

$$\begin{aligned}
&(\rho_1^2 y_2 + c_1)(a_i^1 a_j^2 - a_j^1 a_i^2) \le c_i(a_j^2 y_1 - a_j^1 y_2) + c_j(a_i^1 y_2 - a_i^2 y_1) \\
\Leftrightarrow\ & c_i c_j(\rho_1^2 y_2 + c_1)(1/\rho_i^1\rho_j^2 - 1/\rho_j^1\rho_i^2) \le c_i c_j[(1/\rho_j^2 - 1/\rho_i^2)y_1 + (1/\rho_i^1 - 1/\rho_j^1)y_2] \\
\Leftrightarrow\ & c_1(1/\rho_i^1\rho_j^2 - 1/\rho_j^1\rho_i^2) \le (1/\rho_j^2 - 1/\rho_i^2)y_1 + [(\rho_j^2 - \rho_1^2)/\rho_i^1\rho_j^2 - (\rho_i^2 - \rho_1^2)/\rho_j^1\rho_i^2)]y_2.
\end{aligned}$$

Since by Propsition 11, $y_1 \ge y_2 a_j^1/a_j^2 = y_2\rho_j^2/\rho_j^1$ and $1/\rho_j^2 - 1/\rho_i^2 > 0$, it is sufficient to have

$$\begin{aligned}
c_1(1/\rho_i^1\rho_j^2 - 1/\rho_j^1\rho_i^2) &\le (1/\rho_j^2 - 1/\rho_i^2)y_2\rho_j^2/\rho_j^1 + [(\rho_j^2 - \rho_1^2)/\rho_i^1\rho_j^2 - (\rho_i^2 - \rho_1^2)/\rho_j^1\rho_i^2)]y_2 \\
&= (\rho_j^2 - \rho_1^2)(1/\rho_i^1\rho_j^2 - 1/\rho_j^1\rho_i^2)y_2.
\end{aligned}$$

Note that $1/\rho_i^1\rho_j^2 - 1/\rho_j^1\rho_i^2 = B_{ij}/c_i c_j > 0$, therefore it is equivalent to have $y_2 \ge c_1/(\rho_j^2 - \rho_1^2)$ if $\rho_j^2 > \rho_1^2$. Also by Proposition 11, we have $\rho_2^2 = \min_{k\ge 2}\{\rho_k^2\}$, thus it is sufficient to have $y_2 \ge c_1/(\rho_2^2 - \rho_1^2)$ if $\rho_2^2 > \rho_1^2$. □

This result shows that when only one constraint (the second constraint in this case) is binding at the linear relaxation solution, the periodicity of the value function is determined by the column with the smallest cost-to-weight ratio in this constraint. In addition, the value of y^* is also determined by the binding constraint.

Next, we study the tightness of the corner relaxation for 2-KP when $\{x_1, s_1\}$ is the optimal basis, which can be formulated as

$$\begin{aligned}
CR(y_1, y_2) = \ \min\ & \sum_{j=1}^{n} c_j x_j \\
\text{s.t.}\ & \sum_{j=1}^{n} a_j^2 x_j \ge y_2 \\
& x_j \ge 0, j = 2, ..., n \\
& x \text{ integer.}
\end{aligned}$$

Since the first constraint is relaxed, it is not surprising that the worst-case corner relaxation gap is also determined by the second constraint. As before, we first show the conditions under which the corner relaxation is tight.

Proposition 13 *When $y_2 \ge \bar{y} = \frac{a_1^2(2\rho_1^2 - \rho_2^2)}{\rho_2^2 - \rho_1^2}$, the corner relaxation is tight, that is $CR(y_1, y_2) = f(y_1, y_2)$.*

Proof For $x_1 \geq 0$, a feasible solution to the corner relaxation is $\bar{x}_1 = \lceil y_2/a_1^2 \rceil$, $\bar{x}_j = 0$ for $j = 2, ..., n$. Thus, an upper bound on $CR(y_1, y_2)$ is $c\bar{x} = c_1\lceil y_2/a_1^2 \rceil \leq \rho_1^2 y_2 + c_1$.

For $x_1 \leq -1$, let $\bar{x}$ be an optimal solution to the corner relaxation with $x_1 \leq -1$. Then a lower bound of $CR(y_1, y_2)$ with $x_1 \leq -1$ can be obtained by solving

$$\begin{aligned} c_1\bar{x}_1 + \ \min \ & \sum_{j=2}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=2}^{n} a_j^2 x_j \geq y_2 - a_1^2\bar{x}_1 \\ & x_j \geq 0 \ j = 2, ..., n. \end{aligned}$$

Since $\rho_2^2 \leq ... \leq \rho_n^2$, an optimal solution to the problem is $\bar{x}_2 = (y_2 - a_1^2\bar{x}_1)/a_2^2$, $\bar{x}_j = 0$ for $j = 3, ..., n$. Thus, the lower bound can be written as $LB(\bar{x}_1) = c_1\bar{x}_1 + c_2(y_2 - a_1^2\bar{x}_1)/a_2^2 = \rho_2^2 y_2 + a_1^2(\rho_1^2 - \rho_2^2)\bar{x}_1$. Since $\rho_1^2 < \rho_2^2$ and $\bar{x}_1 \leq -1$, a valid lower bound of $CR(y_1, y_2)$ with $x_1 \leq -1$ can be written as:

$$LB = \min_{\bar{x}_1 \leq -1, integer} \{LB(\bar{x}_1)\} = \rho_2^2 y_2 + a_1^2(\rho_2^2 - \rho_1^2). \tag{55}$$

Thus, a sufficient condition for an optimal solution to the corner relaxation satisfying $x_1 \geq 0$ is

$$\rho_1^2 y_2 + c_1 \leq \rho_2^2 y_2 + a_1^2(\rho_2^2 - \rho_1^2),$$

which yields $y_2 \geq a_1^2(2\rho_1^2 - \rho_2^2)/(\rho_2^2 - \rho_1^2)$ if $\rho_2^2 > \rho_1^2$. □

Comparing this condition with the more general result in Gomory [32] as shown in Figure 16, where the corner relaxation is tight over a cone that is strictly inside the area $\{(y_1, y_2) : y_1/y_2 \leq a_1^1/a_1^2, y_1 \geq 0, y_2 \geq 0\}$, our result defines a different area for the corner relaxation to be tight, which include all $(y_1, y_2) \in \{y_1/y_2 \leq a_1^1/a_1^2, y_1 \geq 0, y_2 \geq \bar{y}\}$. The worst-case analysis of the corner relaxation gap follows naturally.

Proposition 14 *When $y_2 \leq \bar{y} = a_1^2(2\rho_1^2 - \rho_2^2)/(\rho_2^2 - \rho_1^2)$, the relative gap of the corner relaxation for 2-KP satisfies*

$$\frac{f(y_1, y_2) - CR(y_1, y_2)}{f(y_1, y_2)} \leq \frac{a_1^2}{y_2 + a_1^2} - \frac{\rho_2^2 - \rho_1^2}{\rho_1^2}.$$

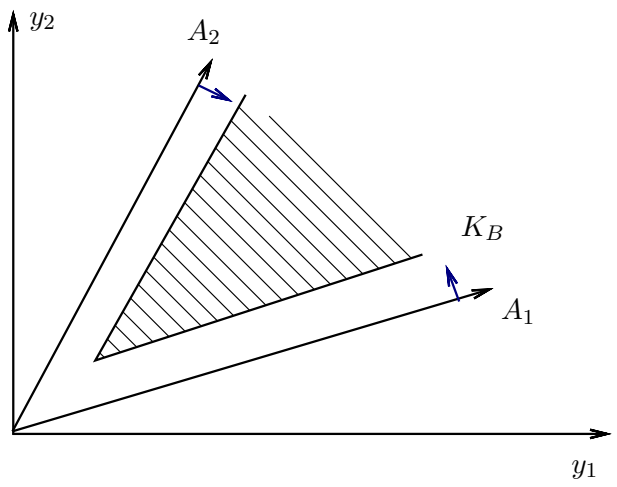


Figure 16: Gomory – Conditions for Tight Corner Relaxation

Proof If there exists an optimal solution to $CR(y_1, y_2)$ satisfying $x_1 \geq 0$, then $f(y_1, y_2) - CR(y_1, y_2) = 0$. Otherwise, using (55) we have $CR(y_1, y_2) \geq \rho_2^2 y_2 + a_1^2(\rho_2^2 - \rho_1^2)$, and $F(y) \leq \rho_1^2 y_2 + c_1$ by rounding up the optimal LP solution. Therefore, the relative gap satisfies

$$\frac{f(y_1, y_2) - CR(y_1, y_2)}{f(y_1, y_2)} = 1 - \frac{CR(y_1, y_2)}{f(y_2, y_2)} \leq 1 - \frac{\rho_2^2 y_2 + a_1^2(\rho_2^2 - \rho_1^2)}{\rho_1^2 y_2 + c_1} = \frac{a_1^2}{y_2 + a_1^2} - \frac{\rho_2^2 - \rho_1^2}{\rho_1^2}.$$

□

The results show that when only one knapsack constraint is binding at the linear relaxation solution, the periodic property and the worst-case corner relaxation gap generally depend on this constraint alone. Thus the cost-to-weight ratio of the columns in the binding constraint can be used to determine the periodicity of the value function. Therefore, under this condition, the problem is simplified.

3.3.3 The optimal basis $\{x_1, x_2\}$

We now consider the general case, where both of the basic variables are non-slack variables. Assume $B_{12} > 0$ and the constraint coefficients satisfy

$$\rho_1^1 \leq \rho_2^1 \ , \ \rho_2^2 \leq \rho_1^2 \ , \text{ and } c_k \geq A_{12}^2 a_k^1 + A_{12}^1 a_k^2 \text{ for } k = 1, ..., n, \tag{56}$$

where

$$A_{12}^1 = \frac{c_2 a_1^1 - c_1 a_2^1}{B_{12}} = \frac{1/\rho_1^1 - 1/\rho_2^1}{B_{12}/c_1 c_2} \geq 0 \text{ and } A_{12}^2 = \frac{c_1 a_2^2 - c_2 a_1^2}{B_{12}} = \frac{1/\rho_2^2 - 1/\rho_1^2}{B_{12}/c_1 c_2} \geq 0. \tag{57}$$

Consider $(y_1, y_2) \in \{(y_1, y_2) : a_2^1/a_2^2 \leq y_1/y_2 \leq a_1^1/a_1^2, (y_1, y_2) \in \mathbb{R}_+^2\}$, then by proposition (11), $\{x_1, x_2\}$ is the optimal LP basis of 2-KP.

We first derive a sufficient condition under which a periodicity of the value function exists, and show that such condition can be seen as a natural extension of KP.

Proposition 15 *If (y_1, y_2) satisfies $(A_{12}^2 - A_{ij}^2)y_1 + (A_{12}^1 - A_{ij}^1)y_2 + (c_1 + c_2) \leq 0$, where $\{x_i, x_j\}$ is an optimal basis when x_1 is fixed to zero, and $(y_1, y_2) \in K_{B_{ij}}$, the value function has periodicity of A_1, that is $f(y_1, y_2) = c_1 + f(y_1 - a_1^1, y_2 - a_1^2) = c_1 + f(y - A_1)$.*

Proof For $x_1 = 0$, suppose an optimal linear relaxation solution is $\bar{x}_i = B_{yj}/B_{ij}$, $\bar{x}_j = B_{iy}/B_{ij}$, and $\bar{x}_k = 0$ for $k \neq i, j$, where $B_{ij} > 0$. Thus we must have $y \in K_{B_{ij}}$. Then a lower bound for $f(y_1, y_2)$ when $x_1 = 0$ is $c_i B_{yj}/B_{ij} + c_j B_{iy}/B_{ij}$.

For $x_1 \geq 1$, define $\bar{x}$ by $\bar{x}_i = \lceil x_i^* \rceil$ when x_i^* is fractional, and $\bar{x}_i = x_i^* + 1$ when x_i^* is integer, for $i = 1, ..., n$, where x^* is the optimal linear relaxation solution of KP. Then $\bar{x}$ is a feasible integer solution. Thus, an upper bound for $f(y_1, y_2)$ when $x_1 \geq 1$ is $c_1(B_{y2}/B_{12} + 1) + c_2(B_{1y}/B_{12} + 1)$. Therefore, a sufficient condition for an optimal solution of the 2-KP to satisfy $x_1 \geq 1$ is

$$c_1(B_{y2}/B_{12} + 1) + c_2(B_{1y}/B_{12} + 1) \leq c_i B_{yj}/B_{ij} + c_j B_{iy}/B_{ij}. \tag{58}$$

Since multiple bases are concerned, we denote $\bar{c}_k(ij)$ to be the reduced cost of variable x_k with respect to basis x_i, x_j, for all $k, i, j = 1, ..., n$.

By replacing B_{yj}, B_{iy} with

$$B_{yj} = \frac{B_{1j}B_{y2} + B_{2j}B_{1y}}{B_{12}} \text{ and } B_{iy} = \frac{B_{i1}B_{y2} + B_{i2}B_{1y}}{B_{12}},$$

we find (58) equivalent to

$$\bar{c}_1(ij)B_{y2} + \bar{c}_2(ij)B_{1y} + B_{12}(c_1 + c_2) \leq 0, \tag{59}$$

where $\bar{c}_1(ij) = c_1 - c_i B_{1j}/B_{ij} - c_j B_{i1}/B_{ij}$ and $\bar{c}_2(ij) = c_2 - c_i B_{2j}/B_{ij} - c_j B_{i2}/B_{ij}$ are the reduced costs of x_1 and x_2 with respect to the basis $\{x_i, x_j\}$, respectively. Since the reduced costs can be formulated as

$$\bar{c}_1(ij) = c_1 - A_{ij}^2 a_1^1 - A_{ij}^1 a_1^2 \text{ and } \bar{c}_2(ij) = c_2 - A_{ij}^2 a_2^1 - A_{ij}^1 a_2^2,$$

where

$$A^1_{ij} = \frac{c_j a^1_i - c_i a^1_j}{B_{ij}} = \frac{1/\rho^1_i - 1/\rho^1_j}{B_{ij}/c_i c_j} \geq 0 \text{ and } A^2_{ij} = \frac{c_i a^2_j - c_j a^2_i}{B_{ij}} = \frac{1/\rho^2_j - 1/\rho^2_i}{B_{ij}/c_i c_j} \geq 0, \quad (60)$$

(59) is equivalent to

$$(a^2_2 \bar{c}_1(ij) - a^2_1 \bar{c}_2(ij))y_1 + (a^1_1 \bar{c}_2(ij) - a^1_2 \bar{c}_1(ij))y_2 + B_{12}(c_1 + c_2) \leq 0. \quad (61)$$

Because

$$\begin{aligned} a^2_2 \bar{c}_1(ij) - a^2_1 \bar{c}_2(ij) =& a^2_2(c_1 - A^2_{ij} a^1_1 - A^1_{ij} a^2_1) - a^2_1(c_2 - A^2_{ij} a^1_2 - A^1_{ij} a^2_2) \\ =& a^2_2 c_1 - a^2_1 c_2 - A^2_{ij} B_{12} = (A^2_{12} - A^2_{ij}) B_{12} \end{aligned}$$

and

$$\begin{aligned} a^1_1 \bar{c}_2(ij) - a^1_2 \bar{c}_1(ij) =& a^1_1(c_2 - A^2_{ij} a^1_2 - A^1_{ij} a^2_2) - a^1_2(c_1 - A^2_{ij} a^1_1 - A^1_{ij} a^2_1) \\ =& a^1_1 c_2 - a^1_2 c_1 - A^1_{ij} B_{12} = (A^1_{12} - A^1_{ij}) B_{12}, \end{aligned}$$

(61) can be rewritten as

$$(A^2_{12} - A^2_{ij})y_1 + (A^1_{12} - A^1_{ij})y_2 + (c_1 + c_2) \leq 0. \quad (62)$$

Although the periodicity of the value function exists on the half-plane defined by (62), we need to show that there exists $(y_1, y_2) \in \{(y_1, y_2) \in \mathbb{R}^2_+ : a^1_2/a^2_2 \leq y_1/y_2 \leq a^1_1/a^2_1\}$ such that the intersection of (62) and $\{(y_1, y_2) \in \mathbb{R}^2_+ : a^1_2/a^2_2 \leq y_1/y_2 \leq a^1_1/a^2_1\}$ is non-empty.

First, we have the reduced costs

$$\bar{c}_1(12) = c_1 - A^2_{12} a^1_1 - A^1_{12} a^2_1 = 0 \text{ and } \bar{c}_2(12) = c_2 - A^2_{12} a^1_2 - A^1_{12} a^2_2 = 0. \quad (63)$$

Since $\{x_i, x_j\}$ is not optimal when x_1 is not fixed to 0, the reduced costs satisfy

$$\bar{c}_1(ij) = c_1 - A^2_{ij} a^1_1 - A^1_{ij} a^2_1 < 0 \text{ and } \bar{c}_2(ij) = c_2 - A^2_{ij} a^1_2 - A^1_{ij} a^2_2 \geq 0. \quad (64)$$

Therefore, by combining (63) and (64), we have

$$c_1 = A^2_{12} a^1_1 + A^1_{12} a^2_1 < A^2_{ij} a^1_1 + A^1_{ij} a^2_1 \text{ and } c_2 = A^2_{12} a^1_2 + A^1_{12} a^2_2 \geq A^2_{ij} a^1_2 + A^1_{ij} a^2_2,$$

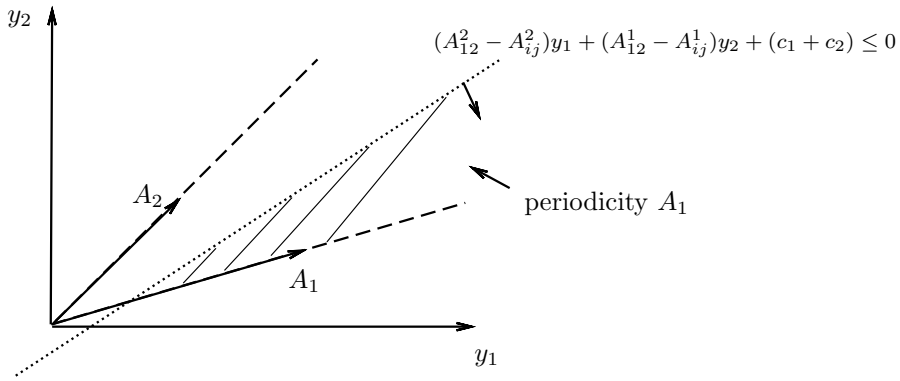


Figure 17: Periodicity of $f(y_1, y_2)$

which are equivalent to

$$(A^2_{12} - A^2_{ij})a^1_1 + (A^1_{12} - A^1_{ij})a^2_1 < 0 \Longrightarrow (A^2_{12} - A^2_{ij})a^1_1/a^2_1 + (A^1_{12} - A^1_{ij}) < 0$$

and

$$(A^2_{12} - A^2_{ij})a^1_2 + (A^1_{12} - A^1_{ij})a^2_2 \geq 0 \Longrightarrow (A^2_{12} - A^2_{ij})a^1_2/a^2_2 + (A^1_{12} - A^1_{ij}) \geq 0.$$

Since $a^1_2/a^2_2 \leq y_1/y_2 \leq a^1_1/a^2_1$, we know that the set $\{(y_1, y_2) \in \mathbb{R}^2_+ : (A^2_{12} - A^2_{ij})y_1 + (A^1_{12} - A^1_{ij})y_2 + (c_1 + c_2) \leq 0$ and $a^1_2/a^2_2 \leq y_1/y_2 \leq a^1_1/a^2_1\}$ is non-empty. □

Comparing (62) with the periodicity condition for KP, we find that (A^1_{ij}, A^2_{ij}) defined in (60) for $\{x_i, x_j\}$ plays a similar role as ρ_i for x_i in KP. However, the intuitive interpretation for (A^1_{ij}, A^2_{ij}) is not as straightforward as ρ_i. We illustrate the conditions where the value function has periodicity A_1 by the shaded area in Figure 17, which is a cone in the cone spanned by the basis matrix. Similar conditions can be derived for the value function with periodicity of A_2.

Next, we consider the corner relaxation for 2-KP when $\{x_1, x_2\}$ is the optimal basis, which relaxes the constraints $x_1 \geq 0$ and $x_2 \geq 0$. Although we do not specify the conditions under which the corner relaxation is tight since the conditions are more complex, the following example shows that the corner relaxation bound can be as weak as the linear relaxation bound.

Example Consider the problem

$$\begin{aligned} f(y_1, y_2) = \min \quad & x_1 + (1 + \tfrac{1}{n+1})x_2 \\ \text{s.t.} \quad & (n+1)x_1 + nx_2 \geq y_2 \\ & nx_1 + (n+1)x_2 \geq y_2 \\ & x_1, x_2 \geq 0, \text{ integer}, \end{aligned}$$

with $y_1 = y_2 = n + 1$. An optimal integer solution to the problem is $x_1 = 2, x_2 = 0$, with $f(n + 1, n + 1) = 2$. The linear relaxation has an optimal linear solution $x_1^* = x_2^* = (n+1)/(2n+1)$ with objective value $LP(n+1, n+1) = 1+2/(2n+1) \to 1$ as $n \to +\infty$. Corner relaxation relaxes constraints $x_1 \geq 0$ and $x_2 \geq 0$, and an optimal solution to the corner relaxation is $x_1 = n+1, x_2 = 1-n$, with objective value $CR(n+1, n+1) = 1+2/(n+1) \to 1$ as $n \to +\infty$. Therefore, the corner relaxation is no tighter than the linear relaxation as n becomes large.

3.4 The periodic property of the MKP

Finally, we extend the periodic property of the value function to MKP, which can be stated as:

$$\begin{aligned} f(y) = \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq y_i, \ i = 1...m \\ & x \in \mathbb{Z}_+^n. \end{aligned} \tag{65}$$

Assume that $c_j > 0$, $a_{ij} \geq 0$ for all i, j. If B is an optimal basis of the linear relaxation of MKP, then $y = (y_1, ..., y_m)$ must be in K_B, which is the cone spanned by columns in B. For a basic variable x_i, we show that $f(y)$ has a periodicity of A_i when y is in a cone intersecting a half-plane, which generalizes the results on the periodic property that we developed for 2-KP.

Claim 16 *Let B be a unique non-degenrate optimal basis for MKP with right-hand-side vector y^* of the constraints, and B_i be a non-degenerate optimal basis when a basic variable*

x_i is fixed to zero. Then for any $y \in K_B \cap K_{B_i} \cap \{s : (c_B B^{-1} - c_{B_i} B_i^{-1})^T s + |c_B| \leq 0\}$, the value function $f(y)$ has periodicity A_i, that is $f(y) = f(y - A_i) + c_i$.

Proof Let x^* be the optimal LP solution with respect to the basis B, and we must have $y^* \in K_B$. To satisfy $x_i \geq 1$, define $\bar{x}$ by $\bar{x}_i = \lceil x_i^* \rceil$ when x^* is fractional and $\bar{x} = x_i^* + 1$ when x_i^* is integral for $i \in B$, and $\bar{x}_i = 0$ for $i \notin B$, then $\bar{x}$ is a feasible integral solution to MKP with $x_i \geq 1$. Thus an upper bound of $f(y^*)$ with $x_i \geq 1$ is $UB \leq c_B(x_B^* + 1) = c_B B^{-1} y + |c_B|$, where $|c_B| = \sum_{i \in B} c_i$.

If we force $x_i = 0$, a lower bound of $f(y^*)$ when x_i fixed to zero is $LB = c_{B_i} B_i^{-1} y$. In such case, y^* must also be in the cone generated by columns in B_i, that is, $y^* \in K_{B_i}$.

Therefore, for an optimal solution of the MKP to have $x_i \geq 1$, a sufficient condition is that $c_B B^{-1} y^* + |c_B| \leq c_{B_i} B_i^{-1} y^*$, which is equivalent to

$$(c_B B^{-1} - c_{B_i} B_i^{-1}) y^* + |c_B| \leq 0. \tag{66}$$

Since the above statements hold for any $y \in K_B \cap K_{B_i}$, let $d = c_B B^{-1} - c_{B_i} B_i^{-1}$, then for any y satisfying

$$y \in K_B \cap K_{B_i} \cap \{s : d^T s + |c_B| \leq 0\}, \tag{67}$$

the value function $f(y)$ has periodicity A_i.

Now we show that the set (67) is non-empty. By definition of B and B_i, we have $y^* \in K_B \cap K_{B_i} \neq \emptyset$. Since B is the unique optimal basis, the corresponding objective value must be smaller than the objective value corresponding to basis B_i, that is $c_B B^{-1} y^* < c_{B_i} B_i^{-1} y^*$, thus $d^T y^* < 0$. Therefore, there exists $\lambda > 0$ such that $\lambda d^T y^* + |c_B| \leq 0$, and $\lambda y^* \in K_B \cap K_{B_i} \cap \{s : d^T s + |c_B| \leq 0\} \neq \emptyset$. □

This result can be seen as a generalization of the conditions for the periodic property to hold in 2-KP when $\{x_1, x_2\}$ is the optimal basis. However, the area of (y_1, y_2) where the periodic property holds is no longer guaranteed to be a cone.

CHAPTER IV

MULTI-DIMENSIONAL KNAPSACK INSTANCES

The purpose of this chapter is twofold. In Section 4.1, we review previous techniques on instance generation for several variations of the multi-dimensional knapsack problem (MKP). In Section 4.2, we generate a large number of MKP instances based on two parameters, which indicate the tightness of constraints and the correlation between the objective and constraint coefficients respectively, examine the impact of the parameters on problem hardness, and establish a testbed of instances for computational experiments in the remainder of this thesis.

4.1 Introduction

Consider the multi-dimensional knapsack problem (MKP):

$$\begin{aligned} \max \ & \sum_{i=1}^{m} \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, ..., m \\ & x \in \mathbb{Z}_+^n. \end{aligned} \tag{68}$$

Variations of MKP are distinguished by number of constraints, type of variables (unbounded, bounded integers or binary variables), and the form of objective and constraint coefficients. For $m = 1$, the problem can be reformulated as $\max\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \leq b, x \in \mathbb{Z}_+^n\}$, and is called the knapsack problem (KP); when all variables are binary, the problem is called the 0-1 multi-dimensional knapsack problem (0-1 MKP); when $m = 1$ and all variables are binary, the problem is called the 0-1 knapsack problem (0-1 KP). For other variations of MKP with special restrictions on constraint and objective coefficients, see Martello and Toth [52].

As computational improvement has been achieved on solving variations of MKP, a better understanding of the factors that impact the hardness of these problems is obtained.

Martello and Toth [52] generated 0-1 KP instances based on the correlation between objective and constraint coefficients. The objective–constraint correlation of column j is indicated by the ratio c_j/a_j. An instance with a strong objective–constraint correlation means a small difference between $\max_{j=1,...,n} c_j/a_j$ and $\min_{j=1,...,n} c_j/a_j$. Martello and Toth [52] showed computationally that strongly correlated instances are always the hardest to solve. In terms of the tightness of the constraint, they found that generating b with $\alpha \sum_{j=1}^{n} a_j$, where $\alpha < 1$, creates harder instances than generating b randomly.

The techniques used to generate 0-1 KP instances are also extended to higher dimensional knapsack problems. In particular, the OR-Library founded by Beasley [6] lists a number of 0-1 MKP benchmark instances, which are generated based on methods proposed by Fréville and Plateau [22], Chu and Beasley [12], and Osorio et al. [55]. As suggested by these studies, the two factors that influence the hardness of 0-1 MKP the most are the correlation between objective and constraint coefficients, and the tightness of constraints. The tightness of constraints is measured similarly as for the 0-1 KP by a real number α, where $b_i = \alpha \sum_{j=1}^{n} a_{ij}$ for all $i = 1, ..., m$. Since all variables are binary, instances with $\alpha \geq 1$ have an optimal solution $x_j = 1$ for $j = 1, ..., n$, thus only $\alpha < 1$ is considered. The objective-constraint correlation, however, is more complicated to define, since the constraint coefficients of each variable are a vector. A common way to capture the correlation is to control the objective coefficients by $c_j = \sum_{i=1}^{m} a_{ij}/m + K * u$, where u is a random number between zero and one and K is fixed. For small K, the objective and constraint coefficients are strongly correlated, and when K increases, the correlation becomes weaker.

Since the 0-1 MKP and MKP share similar constraint structures and are only different in types of variables, the factors impacting the hardness of 0-1 MKP have a similar impact on MKP, and the strategies used to generate 0-1 MKP instances can also be generalized to create MKP instances. We use two parameters to control the constraint tightness and

objective-constraint correlation respectively, and study their impact on MKP. For consistency, we use the minimization formulation for MKP, which can be stated as:

$$\begin{aligned} \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, i = 1, ..., m \\ & x \in \mathbb{Z}_+^n, \end{aligned} \tag{69}$$

and create a testbed composed of instances with different levels of constraint tightness and objective-constraint correlation. All tests in this chapter are done with CPLEX 12.2 with default setting on Linux machines with a 2.27GHz processor and 6 GB of memory.

4.2 Instance generation and computational experiments

Let us denote the number of unbounded integer variables by n, and the number of constraints by m. Since for the remainder of the thesis, our goal is to solve hard MIP problems by improving lower bounds with relaxation algorithms, we focus on instances with large sizes, especially those with a large number of constraints. Therefore, we choose $n \in \{50, 100\}$ and $m \in \{100, 500, 1000\}$. The constraint coefficients a_{ij} are randomly generated from the uniform distribution $U(0, 1000)$, for $i = 1, ..., m$ and $j = 1, ..., n$.

Let $K \geq 0$ be the parameter that controls the correlation between objective coefficients and constraint coefficients. The objective coefficients are generated by $c_j = \sum_{i=1}^{m} a_{ij}/m + K * u$, where u is a random number between zero and one. When $K = 0$, c_j is only a function of the constraint coefficients in column j, for all j. Thus the objective function and the constraints are strongly correlated. As K increases, the correlation deteriorates. Let $\alpha \geq 0$ be the tightness parameter controlling the tightness of all constraints. The right-hand side of the constraints are generated by $b_i = \alpha \sum_{j=1}^{n} a_{ij}$ for $i = 1, ..., m$. A large α indicates that the constraints are tight, while a small α means the constraints are loose. Note that we use the same tightness parameter on all constraints. The main reason is that when the constraints have different levels of tightness, tight constraints are needed to define the convex hull of the feasible region and loose constraints are likely to be "redundant", that is, a feasible solution satisfying the tight constraints is likely to

satisfy the loose constraints. In such case, the difficulty of solving the instances may be reduced. Therefore, we use a single tightness parameter to enforce a similar tightness on all constraints, thus no constraints are considered "redundant" in the problem generally. Since the variables are unbounded, α can be arbitrarily large, but we find that instances with $\alpha > 1$ are often easy to solve and not of computational interest. Therefore, we choose values for $K \in \{0, 100, 500, 1000\}$ and $\alpha \in \{0.25, 0.5, 0.75, 1\}$, which are sufficiently wide ranges for our computational experiments.

For each combination of n, m, α, and K, we generate a group of 10 instances. To test the impact of α on problem hardness, for a fixed combination of n, m, and K, we use the same constraint matrices and objective functions, and only vary α to change the right-hand sides of the constraints. Similarly, to test the impact of K, for a fixed combination of n, m, and α, we use the same constraints and only modify the objective functions by varying K. Thus, for each combination of n and m, there are 10 randomly generated constraint matrices in total. Therefore, our initial test set is composed of $2 * 3 * 4 * 4 * 10 = 960$ total instances in 96 groups, where $n \in \{50, 100\}, m \in \{100, 500, 1000\}, K \in \{0, 100, 500, 1000\}$ and $\alpha \in \{0.25, 0.5, 0.75, 1\}$.

The level of difficulty for solving the instances can be characterized using relative gaps, CPU processing time, and number of nodes that have been processed in the branch-and-cut algorithm. The relative gap is defined by $|UB - LB|/(LB + e^{-6})$, where UB is an upper bound of the optimal objective value obtained by using the best feasible solution found, and LB is the best lower bound of the objective value we have obtained. We use two stopping criteria for the branch-and-cut algorithm. The algorithm is terminated either when the relative gap is less than 0.1%, in which case we denote the instance as solved to "optimality", or when the algorithm has processed 10,000,000 nodes, which implies that the instance is very hard to solve. We present for each group the average relative gap (A. Gap) and range of gaps (GR) at the root node. The average is taken over the 10 instances in the group, and the range gives the difference between the maximum and minimum of the relative gaps. We show the number of instances that are solved to optimality (#). In addition, for instances that are solved to optimality, we show the average number of nodes

Table 1: Instances with $n = 50, m = 100$

Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	16.74%	14%-19%	0	-	-	2.67%	1.9%-3.8%	3008	2.8-3.3	9960
	100	13.84%	12%-17%	2	5534	1351	1.44%	0.6%-2.3%	2607	2.4-3.0	9949
	500	9.12%	7%-11%	10	99	18	-	-	-	-	-
	1000	7.85%	5%-10%	10	22	4	-	-	-	-	-
0.5	0	9.58%	7%-13%	0	-	-	2.35%	1.9%-2.6%	3708	3.5-4.0	9826
	100	7.70%	6%-9%	0	-	-	1.49%	1.0%-1.9%	3461	3.2-3.7	9841
	500	4.41%	2%-6%	10	649	110	-	-	-	-	-
	1000	3.72%	3%-5%	10	95	14	-	-	-	-	-
0.75	0	6.68%	5%-8%	0	-	-	1.95%	1.4%-2.2%	3774	3.6-4.0	9768
	100	5.93%	4%-8%	0	-	-	1.59%	1.4%-1.8%	3609	3.4-3.9	9827
	500	3.21%	2%-4%	10	1763	299	-	-	-	-	-
	1000	2.54%	1%-4%	10	154	25	-	-	-	-	-
1	0	0%	0%	10	0	0	-	-	-	-	-
	100	0%	0%	10	0	0	-	-	-	-	-
	500	2.19%	1%-3%	10	3509	592	-	-	-	-	-
	1000	1.84%	1%-3%	10	312	43	-	-	-	-	-

(in thousands) that are processed in the branch-and-cut tree (Nodes), and the average CPU time (in seconds) that are used by CPLEX to solve the instances (Time). For instances that are not solved to optimality, we present the results at node 10,000,000. Specifically, we show the average relative gap (A. Gap), the range of relative gaps at the node (GR), the average CPU time for processing 10,000,000 nodes (A. Time), and the range of CPU time (in thousand seconds) for CPLEX to terminate (TR). To see the progress of the branch-and-cut algorithm over time, we also give the average number of nodes (in thousands) processed when the relative gap at node 10,000,000 is first achieved (NN). We mark a group with $*$ if it contains instances not solved due to the memory limit. Tables 1 through 6 present the results for groups of instances with different sizes.

The results show that as the number of variables increases, the relative gaps at the root node and node 10,000,000 increase. Generally, the number of nodes that are used to first achieve the relative gaps at node 10,000,000 are smaller for $n = 100$ than $n = 50$, which implies that the instances with more variables are harder to solve. As the number of constraints increases, although the root node gap decreases, the relative gaps at node 10,000,000 increase and the time used for processing 10,000,000 nodes also increases, which indicate that the instances with more constraints are harder to solve. The time used by the branch-and-bound algorithm is consistent for instances within one group, and increasing problem sizes obviously increases the average solving time. In addition, as α increases,

Table 2: Instances with $n = 50, m = 500$

Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	24.69%	24%-26%	0	-	-	5.88%	5.5%-6.1%	15447	14-17	9940
	100	22.30%	19%-26%	0	-	-	3.55%	2.7%-4.1%	15242	14-17	9947
	500	11.34%	9%-14%	10	406	356	-	-	-	-	-
	1000	8.49%	6%-12%	10	30	22	-	-	-	-	-
0.5	0	14.12%	12%-16%	0	-	-	4.86%	4.6%-5.0%	16224	15-18	9895
	100	12.89%	11%-15%	0	-	-	3.32%	2.8%-3.7%	15425	14-16	9864
	500	5.98%	4%-8%	10	2731	2420	-	-	-	-	-
	1000	4.93%	3%-6%	10	262	166	-	-	-	-	-
0.75	0	9.59%	8%-11%	0	-	-	2.42%	2.3%-2.6%	16484	16-17	9853
	100	8.66%	7%-10%	0	-	-	1.73%	1.5%-2.0%	16024	14-17	9811
	500	4.08%	3%-6%	3	3599	3289	0.33%	0.1%-0.7%	8907	8-11	9867
	1000	3.15%	2%-5%	10	638	395	-	-	-	-	-
1	0	0%	0%	10	0	0	-	-	-	-	-
	100	0%	0%	10	0	0	-	-	-	-	-
	500	2.93%	2%-4%	1	2766	1865	0.42%	0.3%-0.7%	9400	8-11	9708
	1000	2.23%	1%-3%	9	527	309	0.24%	0.24%	6214	6.2	9768

Table 3: Instances with $n = 50, m = 1000$

Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	27.37%	25%-30%	0	-	-	9.33%	8.9%-9.6%	33239	28-45	9936
	100	25.53%	23%-28%	0	-	-	6.37%	5.8%-6.9%	37471	31-58	9953
	500	14.28%	12%-17%	10	388	942	-	-	-	-	-
	1000	9.86%	8%-12%	10	33	53	-	-	-	-	-
0.5	0	15.91%	15%-17%	0	-	-	5.08%	4.3%-7.0%	33914	31-37	9915
	100	14.46%	13%-16%	0	-	-	3.88%	3.4%-4.8%	31415	30-34	9915
	500	6.98%	6%-9%	6	2525	5475	0.70%	0.1%-1.4%	25009	20-30	5475
	1000	4.80%	2%-7%	10	313	407	-	-	-	-	-
0.75	0	10.21%	9%-11%	0	-	-	1.80%	1.7%-1.9%	40850	33-52	9811
	100	9.28%	8%-10%	0	-	-	1.06%	0.6%-1.2%	35711	31-48	9854
	500	4.73%	3%-7%	1	6677	13341	0.53%	0.1%-1.5%	25029	19-30	9862
	1000	3.15%	2%-4%	10	1103	1829	-	-	-	-	-
1	0	0%	-	10	0	0	-	-	-	-	-
	100	0%	-	10	0	0	-	-	-	-	-
	500	3.18%	0%-5%	5	1698	4235	0.40%	0.3%-0.5%	27564	22-31	9724
	1000	2.23%	1%-3%	10	1257	2080	-	-	-	-	-

Table 4: Instances with $n = 100, m = 100$

Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	9.72%	9%-11%	0	-	-	4.79%	4.4%-5.0%	10308	9-11	9830
	100	8.08%	7%-10%	0	-	-	3.56%	3.1%-4.0%	8785	8-10	9783
	500	4.96%	4%-6%	5	3303	877	0.49%	0.1%-0.8%	2805	2-4	9878
	1000	3.90%	2%-6%	10	539	137	-	-	-	-	-
0.5	0	5.67%	4%-8%	0	-	-	3.08%	2.8%-3.3%	9723	9-10	9633
	100	4.77%	4%-6%	0	-	-	2.30%	2.0%-2.7%	8734	8-10	9486
	500	2.77%	1%-4%	1	9752	2021	0.59%	0.4%-0.8%	2999	2-4	9560
	1000	1.96%	1%-3%	10	2734	639	-	-	-	-	-
0.75	0	3.90%	3%-5%	0	-	-	2.23%	2.1%-2.4%	9681	9-10	8771
	100	3.11%	2%-5%	0	-	-	1.62%	1.4%-1.9%	8639	8-10	8919
	500	1.71%	1%-3%	0	-	-	0.41%	0.1%-0.5%	2879	2-4	9389
	1000	1.41%	0.8%-3%	9	2272	544	0.22%	-	2841	-	9193
1	0	0%	0%	10	0	0	-	-	-	-	-
	100	2.15%	1%-3%	0	-	-	1.28%	1.1%-1.4%	8343	7-9	8429
	500	1.41%	1%-2%	0	-	-	0.32%	0.1%-0.5%	2791	2-4	9231
	1000	1.05%	0.6%-2%	6	3975	849	0.16%	0.1%-0.3%	2777	2-3	9115

Table 5: Instances with $n = 100, m = 500$

Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	15.92%	15%-17%	0	-	-	7.76%	7.5%-8.1%	62679	59-68	9840
	100	12.49%	11%-15%	0	-	-	6.05%	5.4%-6.9%	60580	56-65	9836
	500	6.61%	5%-9%	0	-	-	1.21%	0.5%-2.0%	18100	12-24	9859
	1000	5.29%	4%-7%	10	3160	2996	-	-	-	-	-
0.5	0	10.36%	9%-13%	0	-	-	5.15%	4.6%-5.4%	58729	54-62	9800
	100	7.60%	6%-10%	0	-	-	3.51%	3.1%-4.1%	57085	54-64	9485
	500	3.61%	2%-5%	0	-	-	0.98%	0.5%-1.7%	18766	14-24	9646
	1000	2.59%	1%-4%	3	6634	6102	0.3%	0.1%-0.4%	11529	9-16	9770
0.75	0	6.68%	5%-8%	0	-	-	2.86%	2.8%-2.9%	60883	58-63	9646
	100	5.42%	4%-7%	0	-	-	2.47%	2.2%-2.7%	57207	52-61	9401
	500	2.39%	1%-4%	0	-	-	0.79%	0.6%-1.2%	19171	15-27	9252
	1000	1.79%	1%-3%	0	-	-	0.31%	0.1%-0.5%	11242	7-16	9021
1	0	0%	-	10	0	0	-	-	-	-	-
	100	0%	-	10	0	0	-	-	-	-	-
	500	1.72%	1%-3%	0	-	-	0.62%	0.3%-0.8%	18249	15-23	8842
	1000	1.51%	1%-2%	0	-	-	0.28%	0.1%-0.4%	13598	8-17	8509

Table 6: Instances with $n = 100, m = 1000$

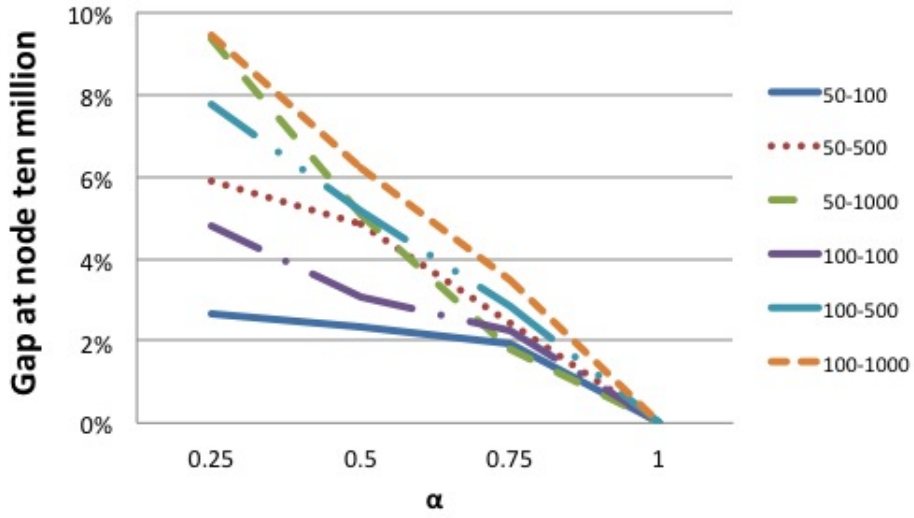
Instance		Root Node		Optimality			Node 10,000,000				
α	K	A.Gap	GR	#	Node(k)	A. Time	A. Gap	GR	A. Time	TR	NN(k)
0.25	0	18.82%	18%-21%	0	-	-	9.45%	9.1%-9.7%	129921	123-133	9834
	100	16.10%	14%-18%	0	-	-	7.20%	6.5%-7.8%	139650	131-148	9851
	500	7.36%	5%-9%	0	-	-	1.63%	1.2%-2.2%	56052	50-68	9856
	1000	6.08%	5%-7%	10	5105	12309	-	-	-	-	-
0.5	0*	11.63%	10%-14%	0	-	-	6.25%	5.4%-6.5%	118615	110-128	9777
	100*	9.13%	8%-11%	0	-	-	4.52%	4.1%-5.0%	120386	115-127	9720
	500	3.89%	3%-6%	0	-	-	1.21%	0.8%-1.7%	53714	45-64	9586
	1000	2.95%	1%-5%	2	4263	8936	0.63%	0.3%-0.8%	27510	21-31	9637
0.75	0	7.28%	6%-9%	0	-	-	3.47%	3.3%-3.6%	130580	116-139	9589
	100	6.42%	5%-7%	0	-	-	2.65%	2.4%-2.8%	123733	121-128	9556
	500	2.52%	1%-4%	0	-	-	0.94%	0.7%-1.1%	55417	42-68	9418
	1000	1.96%	1%-3%	1	6546	18049	0.47%	0.2%-0.7%	29522	21-52	9510
1	0	0%	-	10	0	0	-	-	-	-	-
	100	0%	-	10	0	0	-	-	-	-	-
	500	1.97%	1%-3%	0	-	-	0.76%	0.6%-1.0%	50745	45-58	9200
	1000	1.56%	1%-2%	0	-	-	0.35%	0.1%-0.5%	33428	21-51	9334

generally, the gaps at the root node and node 10,000,000 show a decreasing trend. When K increases, many instances show a trend of decreasing difficulty to solve; but for a small number of instances, increasing K may also increase the difficulty to solve. Next, we analyze the impact of varying α and K on the instance hardness in detail.

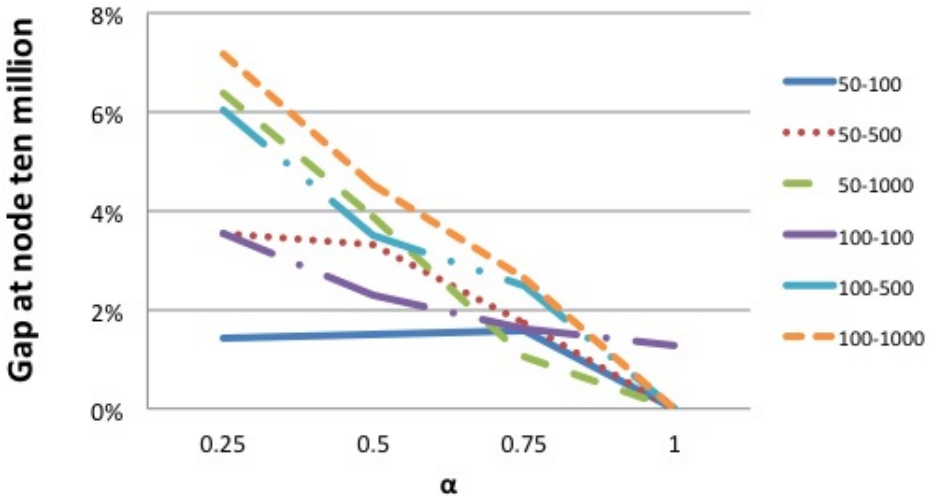
When K increases, since the second term in $c_j = \sum_{i=1}^{m} a_{ij}/m + K * u$ weighs more for all j (we use the same u when varying K), the relative difference between the objective coefficients of all columns becomes smaller, which means that the contributions to the objective value of the columns are similar. Therefore, columns with larger constraint coefficients should be used more often in a good feasible solution, and thus are more "important". On the other hand, for any feasible solution that uses the "important" columns, the constraints with small coefficients for these "important" columns become tight, thus more important in defining the lower bounds. Therefore, for larger K, it is easier to identify "important" columns and constraints to obtain good upper bounds and lower bounds, and the instances generally become easier to solve. One exception is when $\alpha = 1$, where instances with small K are often solved to optimality instantly. This is because for small K, most columns have similar importance, thus $x_j = 1$ for all j can be a good feasible solution.

To see the impact of the tightness ratio α, we use two types of measurements for the hardness of instances. For $K = 0$ and 100, most of the instances are not solved to optimality after processing 10,000,000 nodes, thus we use the average relative gaps at node 10,000,000 to indicate the hardness of instances as shown in Figure 18. For a small number of instances that are solved to optimality, we use 0.1% to be the gap at node 10,000,000. For $K = 500$ and 1000, most of the instances are solved to optimality. Therefore, the average number of nodes that are processed in the branch-and-bound tree to reach optimality can be used to measure the problem hardness as shown in Figure 19. For instances that are not solved to optimality, we use 10,000,000 to be the number of nodes used.

The impact of α is obviously opposite for small and large K. For instances with $K = 0$ and 100, increasing α leads to easier instances. But for $K = 500$ and 1000, increasing α yields more difficulty in solving the instances. In addition, for $K = 0$ and 100, the instances with more constraints generally have larger relative gaps at node 10,000,000, which shows

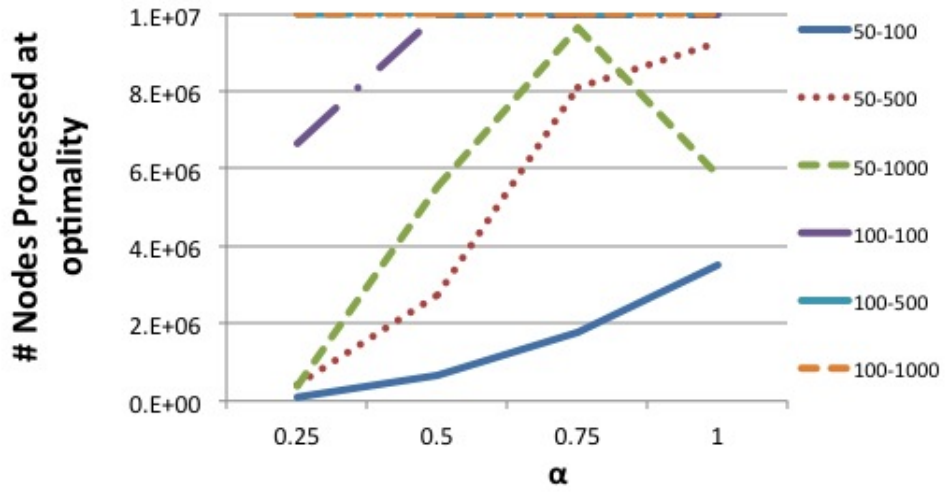


(a) K=0

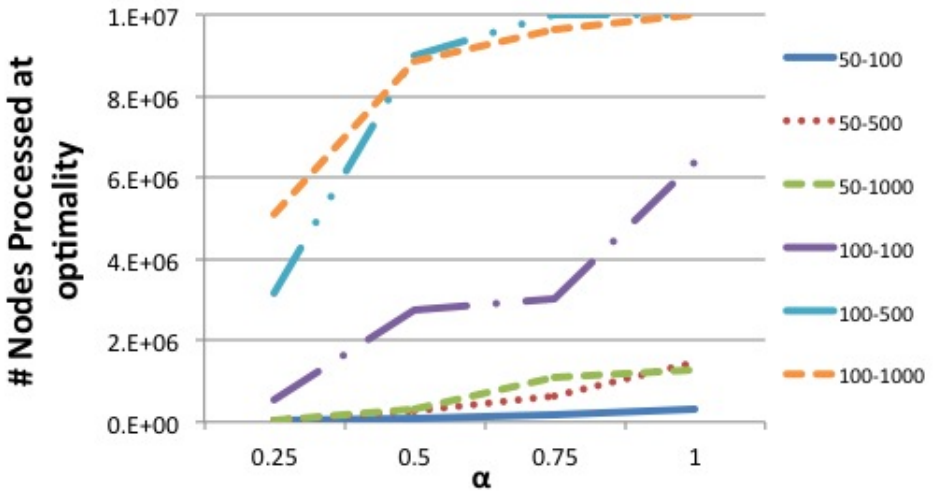


(b) K=100

Figure 18: Relative Gap at Node 10,000,000



(a) K=500



(b) K=1000

Figure 19: Number of Nodes Processed at optimality

that the number of constraints has a greater impact on the problem hardness than the number of variables. But for $K = 500$ and 1000, it requests a larger number of nodes to solve the instances with more variables to optimality, thus the number of variables has a greater impact than the number of constraints.

After examining the best feasible solutions found, we give the following explanation. Note that for instances with $K = 0$ and 100, most of the variables have value zero or one in the solution, which means there are no columns that are much more important than other columns. Therefore, when the right-hand side increases, we have more options in choosing columns to use in the solution, which makes the instances easier to solve. However, for $K = 500$ and 1000, the solutions have a much smaller number of positive components, which correspond to the "important" columns. As α increases, the scales of the "important" components in a good solution increases, thus the problem is harder to solve.

We point out that the impact of α and K on the hardness of the instances depend on the size of the instances. For example, when $K = 500$, 86 out of 120 instances with $n = 50$ are solved to optimality, but only 6 out of 120 instances with $n = 100$ are solved to optimality. When $\alpha = 1$, 105 out of 120 instances with $n = 50$ are solved to optimality, but only 56 out of the 120 instances with $n = 100$ are solved to optimality. Therefore, the impact of α and K should be considered as relative to the chosen sizes of the instances.

At last, we select a testbed of instances that will be used for the computational experiments in the remainder of the thesis from the 960 instances. Since varying K and α gives similar trends in results for instances with the chosen sizes, we choose those with a fixed size of $n = 50$ and $m = 500$. Furthermore, since for small K, smaller α leads to harder instances and for large K, larger α yields to harder problems, we select two sets of instances, which feature different characteristics in constraint tightness and objective-constraint correlation.

- *Set I*: The set has 10 instances with $n = 50$, $m = 500$, $K = 0$, and $\alpha = 0.25$. This set represents the instances with loose constraints, and the objective and constraint coefficients are strongly correlated.

- *Set II*: The set has 10 instances with $n = 50$, $m = 500$, $K = 500$, and $\alpha = 1$. This

set represents the instances with tight constraints, and the objective and constraint coefficients are less correlated.

Set I contains instances where the "importance" of all columns is similar, and thus the problem hardness is caused by choosing the right columns to use in the solution. Set II contains instances where some columns are more important than others, thus the problem hardness is caused by the large right-hand side of the constraints. Therefore, the two sets of instances characterize different causes of hardness for MKP, and we use both sets for the computational experiments in the remainder of the thesis.

CHAPTER V

RELAXATION ALGORITHMS

5.1 Introduction

In this chapter, we study using relaxation algorithms to obtain lower bounds for the multi-dimensional knapsack problem (MKP), which can be stated as:

$$\begin{aligned} z(b) = \min\ & cx \\ \text{s.t. } & A_1 x \geq b_1 \\ & A_2 x \geq b_2 \\ & x \geq 0,\ \text{integer}, \end{aligned} \tag{70}$$

with $A_1 \in \mathbb{Z}^+_{m_1 \times n}$, $A_2 \in \mathbb{Z}^+_{m_2 \times n}$, $m_1 + m_2 = m$, $b \in \mathbb{Z}^+_m$ and $c \in \mathbb{R}^+_n$.

We consider relaxing constraints $A_2 x \geq b_2$. We call the relaxed constraints inactive and the remaining constraints active.

Generally, more active constraints lead to better lower bounds obtained from a relaxation. However, since the convex hull of the feasible solutions to problem (70) is unknown, the number of active constraints that are needed to obtain good lower bounds is also unknown. In addition, there is a tradeoff between the number of active constraints and the ease in solving the relaxation, as a large number of active constraints may lead to relaxations that are harder to solve, while a small number may give poor lower bounds. Therefore, our first goal is to understand the impact of the number of active constraints on lower bounds obtained from relaxations.

At the same time, the lower bounds from relaxations with different active constraints may vary drastically for a fixed m_1. Some relaxation algorithms rely on choosing active constraints that lead to easy to solve relaxations. Alternatively, active constraints can be chosen to better approximate the convex hull of the feasible region to obtain good lower bounds. For MKP, since all the constraints are knapsack constraints, it is generally hard to choose active constraints that lead to fast solutions, we use linear relaxation information to

select active constraints. This study helps us to understand the benefit of using relaxations to obtain lower bounds when a large number of constraints exist.

Given the set of active constraints, problem relaxation can be achieved by simply dropping, dualizing, or aggregating inactive constraints, which are called constraint, lagrangian and surrogate relaxations, respectively. Constraint relaxation can be stated as:

$$\begin{aligned} z_{CR} = \min\ & cx \\ \text{s.t. } & A_1 x \geq b_1 \\ & x \in \mathbb{Z}_n^+. \end{aligned} \tag{71}$$

Thus, the constraint relaxation (71) is also an MKP with m_1 constraints.

The lagrangian relaxation problem can be formulated as:

$$\begin{aligned} z_{LD}(\lambda) = \min\ & cx + \lambda^T(b_2 - A_2 x) \\ \text{s.t. } & A_1 x \geq b_1 \\ & x \in \mathbb{Z}_n^+, \end{aligned} \tag{72}$$

where $\lambda \in \mathbb{R}_{m_2}^+$ are *lagrangian multipliers*. Since $z_{LD}(\lambda)$ is a valid lower bound of $z(b)$ for all $\lambda \in \mathbb{R}_{m_2}^+$, lagrangian relaxation algorithms iteratively solve problem (72) and search for a $\lambda \in \mathbb{R}_{m_2}^+$ that maximizes $z_{LD}(\lambda)$. Let $\{x^k\}$ be the set of solutions satisfying $\{x \in \mathbb{Z}_n^+ : A_1 x \geq b_1\}$, then the lagrangian relaxation bound can be stated as

$$z_{LD}(\lambda) = \min_k cx^k + \lambda^T(b_2 - A_2 x^k). \tag{73}$$

Therefore, $z_{LD}(\lambda)$ is a piecewise linear concave function of λ. To determine the optimal λ^*, a subgradient method which rapidly converges to λ^* can be used. Given $\lambda \in \mathbb{R}_{m_2}^+$ and an optimal solution x^* to (72), a subgradient method iteratively replaces λ with $\lambda + t(b_2 - A_2 x^*)$, where $t \in \mathbb{R}^+$ is a positive step size usually defined as

$$t = \frac{\alpha(UB - LB)}{||A_2 x - b_2||}$$

with $||s|| = (\sum_i s_i^2)^{1/2}$ being the norm of vector s. The scalar α is initialized by 2 and halved when the lower bounds are not improved in a number of consecutive iterations. UB is an upper bound on $z(b)$ which can be obtained with a primal heuristic algorithm, and

LB is the lower bound obtained in the last iteration. We refer to Fisher (2004) [20] for an in-depth discussion of the lagrangian relaxation method.

Surrogate relaxation of (70) can be formulated as

$$\begin{aligned} z_{SD}(\lambda) = \min\ & cx \\ \text{s.t. } & A_1 x \geq b_1 \\ & \lambda^T A_2 x \geq \lambda^T b_2 \\ & x \in \mathbb{Z}_n^+, \end{aligned} \tag{74}$$

where $\lambda \in \mathbb{R}_{m_2}^+$ are *surrogate multipliers*, and $\lambda^T A_2 x \geq \lambda^T b_2$ is the *surrogate constraint*. Since any feasible solution to (70) is also feasible to (74), $z_{SD}(\lambda)$ is a valid lower bound of $z(b)$ for all $\lambda \in \mathbb{R}_{m_2}^+$. Surrogate relaxation algorithms search for $\lambda \in \mathbb{R}_{m_2}^+$ that maximize $z_{SD}(\lambda)$ and iteratively solve (74). Subgradient methods can also be used to search for good surrogate multipliers. In this thesis, we use a subgradient algorithm that is suggested in Galvao et al. [26]. The surrogate multipliers are initialized with the normalized dual solution $\pi/||\pi||$, where π is the optimal LP dual solution to (70), and updated by

$$\lambda_j = \max\{0, \lambda_j - ts_j\}, \tag{75}$$

where s_j is initialized by 0 and updated by $s_j = l_j + 0.5s_j$, and l_j is the slack of constraint j for the best feasible solution obtained in the last iteration. The step size t is determined by $t = \alpha(UB - LB)/||s||$, where LB is the lower bound obtained in the last iteration and UB is the best upper bound obtained so far. α is initialized with 2, and if the lower bounds obtained in the relaxation problems are not improved in a number of consecutive iterations, α is halved. For more discussion of surrogate relaxation algorithms, see Karwan and Rardin [41], and Freville and Plateau [23].

The lower bounds obtained with different relaxation algorithms may vary, depending on the structure of the problem. In this chapter, we compare the lower bounds obtained from constraint, lagrangian and surrogate relaxations of MKP. In addition, we consider the lazy relaxation, which is used mainly computationally with a Branch and Bound (B&B) algorithm. The set of active constraints in a lazy relaxation may be extended during B&B,

while the inactive constraints are treated as lazy constraints. A lazy constraint is relaxed (as in constraint relaxation) until found violated by a solution to the relaxation, it is then added back to the relaxation problem as a cutting plane in B&B. Because adding lazy constraints back to the problem can be computationally expensive, the lazy constraints are usually chosen because they are the ones less likely to be violated, thus they can be applied lazily—that is, only when needed. When using CPLEX 12.2 as the IP solver, the lazy constraints are checked each time an integer solution is found. Therefore, distinct from other relaxations, the feasible solutions to a lazy relaxation are also feasible to the original problem. In addition, since lazy constraints may be added back to the problem as cutting planes, the lazy relaxation may eventually be as tight as the original problem.

Note that the constraint, lagrangian and surrogate relaxations relax a fixed set of constraints, which can be a barrier for obtaining good lower bounds if the active constraints are not chosen properly. On the other hand, the lazy relaxation provides the flexibility of modifying active constraints during the algorithm. However, since the lazy relaxation algorithm monotonically increases the number of active constraints, the size of the relaxation problem may increase significantly. Therefore, we consider a new heuristic algorithm that solves relaxation problems with various active constraints to obtain lower bounds.

This idea is analogous to the neighborhood search based heuristic algorithms, where neighborhoods of feasible solutions are explored by fixing some variables to obtain better solutions, while our algorithm relaxes a number of constraints and solves relaxations to obtain lower bounds. Since fixing variables in a primal problem corresponds to relaxing constraints in its dual problem, we call the algorithm of heuristically relaxing constraints a *dual heuristic algorithm*. The dual heuristic algorithm chooses active constraints using information obtained from linear relaxations of the B&B nodes, thus relaxations with different sets of active constraints can be explored. As far as we know, this is the first heuristic algorithm that uses a neighborhood search idea in the dual space. In addition, we study adjusting parameters to control the implementation of the dual heuristic algorithms to improve lower bounds.

For computational experiments, four sets of instances generated in Chapter 4 are used,

with each set indicated by $n\_m\_K\_\alpha$, where n is the number of variables, m is the number of constraints, α indicates the tightness of constraints, and K represents the correlation between the objective function and the constraint matrix. Specifically, we use 10 instances in each set of 50_500_0_0.25, 50_500_500_0, 100_500_0_0.25, and 100_500_500_0. All tests are computed with CPLEX 12.2 on Linux machines with 2.27GHz processor and 6 GB of memory.

The rest of the chapter is organized as follows. In section 5.1.1, we study strategies for selecting active constraints using constraint relaxation. In section 5.1.2, we compare lower bounds obtained from various relaxation algorithms on MKP. In section 5.1.3, we develop a dual heuristic algorithm that solves relaxations in B&B to obtain lower bounds. In section 5.1.4, we study strategies that dynamically control the dual heuristic algorithm to achieve better performance.

5.1.1 Choosing Active Constraints

We first study methods to choose active constraints to produce good lower bounds for MKP using constraint relaxation.

Information from linear relaxation can be used to choose active constraints. Since an optimal linear dual solution can be interpreted as the marginal price of constraints, constraints with larger dual values are more important for solving the LP. In addition, constraints with small positive slack values are tight constraints at an optimal linear solution, thus are also important for solving the LP. Due to weak duality, the dual values of constraints with positive slack values are zero. Therefore, we can choose active constraints first in non-increasing order of the dual values that are nonzero, and then in non-decreasing order of the slack values when the dual solutions are zero. The lower bounds obtained from relaxations with active constraints that are chosen by this order are indicated by "LO" through Tables 7 to 10. In each table, we report the results for 10 instances in column 1 to 10 and the average of results in column "Ave". For comparison, we obtain lower bounds from ten different relaxations using randomly chosen active constraints of the same size, and report the minimum, maximum, and average of the lower bounds obtained, indicated by "min", "max",

Table 7: Choosing active constraints: Lower bounds of 50_500_0_0.25 instances

Instance		1	2	3	4	5	6	7	8	9	10	Ave
	LP	6246	6211	6223	6219	6212	6195	6260	6286	6231	6255	6234
#	UB	7931	7879	7912	7894	7861	7878	7943	7972	7900	7933	7910
40	min	6390	6341	6357	6399	6335	6331	6439	6379	6384	6395	6375
	max	6497	6435	6530	6470	6487	6458	6562	6526	6493	6497	6496
	ave	6436	6394	6413	6433	6393	6400	6488	6461	6437	6459	6432
	LO	6697	6661	6679	6672	6658	6650	6682	6773	6694	6705	6687
80	min	6824	6852	6867	6832	6824	6813	6876	6923	6878	6899	6859
	max	6942	6919	6937	6904	6938	6937	6974	6990	6931	6961	6943
	ave	6911	6882	6903	6880	6885	6858	6928	6966	6904	6935	6906
	LO	6954	6910	6909	6914	6892	6857	6939	6977	6926	6931	6921
100	min	6923	6914	6924	6898	6906	6893	6959	6978	6874	6957	6923
	max	7025	6988	7001	7035	7004	6961	7041	7053	7024	7045	7018
	ave	6963	6959	6955	6957	6954	6932	6995	7020	6979	7002	6972
	LO	7061	6905	6910	6892	6940	6933	7027	7030	6938	7006	6964
200	min	7154	7107	7124	7117	7109	7095	7166	7180	7121	7167	7134
	max	7182	7132	7160	7146	7149	7120	7192	7210	7153	7196	7164
	ave	7172	7123	7141	7135	7130	7106	7177	7199	7142	7176	7151
	LO	7165	7125	7151	7130	7130	7089	7194	7203	7147	7191	7153
300	min	7227	7188	7195	7194	7186	7176	7233	7264	7203	7239	7211
	max	7246	7206	7230	7212	7212	7195	7256	7288	7225	7268	7234
	ave	7238	7199	7211	7205	7198	7184	7244	7278	7216	7250	7223
	LO	7227	7193	7215	7204	7202	7171	7251	7288	7206	7239	7220
400	min	7304	7259	7276	7258	7254	7241	7303	7335	7279	7308	7282
	max	7322	7275	7291	7282	7274	7257	7316	7349	7291	7326	7298
	ave	7311	7268	7283	7273	7264	7249	7311	7345	7287	7316	7291
	LO	7305	7268	7277	7276	7267	7243	7319	7349	7275	7316	7290
All		7353	7318	7332	7321	7309	7294	7355	7386	7323	7357	7335

and “ave”.

In order to understand the impact of the number of active constraints, we report the lower bounds obtained within 1 hour and vary the number of active constraints (#) from 40 to 400. In addition, we present the lower bounds obtained by solving the original problems in row ”All”, that is, all constraints are active. Furthermore, we report the linear relaxation bounds indicated by “LP” and the best known upper bounds UB of $z(b)$ obtained within 10 hours using CPLEX to solve the original instances.

The results show that with only 40 active constraints, the lower bounds obtained from relaxations using “LO” order are usually better than the LP bounds. For instances with low correlation between the objective function and constraints, that is, $K = 500$, the lower bounds with “LO” order are always significantly better than the random bounds. For $K = 0$, the lower bounds with “LO” order are often better than or close to the maximum of lower bounds for the random case, which implies that the constraints chosen by “LO” order are relative more important constraints.

For all instances, the differences between the maximum and minimum lower bounds

Table 8: Choosing active constraints: Lower bounds of 50_500_500_1 instances

Instance		1	2	3	4	5	6	7	8	9	10	Ave
	LP	36,412	35,786	34,635	36,588	35,620	34,812	36,072	36,453	37,824	35,577	35,978
#	UB	37,085	36,575	35,416	37,407	36,336	35,579	36,840	37,213	38,572	36,168	36,719
40	min	29,799	30,459	28,682	30,698	28,731	29,614	30,049	30,500	32,127	30,123	30,079
	max	33,145	32,373	30,838	32,906	31,877	31,774	32,737	32,716	33,966	31,299	32,363
	ave	31,302	31,219	30,052	32,028	30,908	30,508	31,529	31,864	32,699	30,853	31,297
	LO	36,817	36,151	34,944	36,866	35,935	35,045	36,453	36,963	38,137	36,052	36,336
80	min	32,180	32,683	31,152	33,177	32,258	31,164	32,763	33,246	33,133	31,722	32,348
	max	34,049	33,818	33,180	34,646	34,067	33,064	33,959	34,846	35,494	33,535	34,066
	ave	33,333	33,322	32,254	34,143	32,999	32,434	33,494	33,867	34,855	32,633	33,334
	LO	36,927	36,361	35,253	37,128	36,117	35,268	36,672	37,079	38,362	36,165	36,533
100	min	32,838	33,210	31,917	33,678	32,236	31,858	32,828	33,200	35,066	32,123	32,896
	max	35,117	34,992	33,300	35,499	34,214	33,257	34,712	34,726	36,151	33,884	34,585
	ave	34,103	33,914	32,520	34,477	33,372	32,661	33,846	34,022	35,690	33,130	33,774
	LO	36,962	36,411	35,292	37,159	36,171	35,318	36,705	37,110	38,438	36,165	36,573
200	min	34,947	34,968	33,681	34,982	33,928	33,514	35,099	34,715	36,202	33,245	34,529
	max	35,921	35,404	34,527	36,324	35,156	34,650	35,549	36,134	37,236	35,269	35,617
	ave	35,406	35,175	33,980	35,790	34,671	34,221	35,322	35,649	36,676	34,477	35,137
	LO	37,017	36,451	35,276	37,174	36,182	35,340	36,707	37,075	38,424	36,165	36,581
300	min	36,023	35,648	34,333	36,207	35,142	34,431	35,531	35,842	37,254	34,739	35,516
	max	36,481	36,071	34,735	36,843	35,811	34,888	36,234	36,635	37,963	35,792	36,145
	ave	36,311	35,872	34,611	36,437	35,530	34,675	35,869	36,417	37,620	35,332	35,868
	LO	36,897	36,422	35,242	37,157	36,150	35,329	36,665	37,042	38,399	36,165	36,547
400	min	36,495	35,968	34,804	36,590	35,566	34,774	36,296	36,571	37,861	35,524	36,045
	max	36,742	36,321	35,094	37,055	35,950	35,210	36,538	36,941	38,293	36,129	36,427
	ave	36,612	36,170	34,923	36,815	35,844	35,078	36,400	36,748	38,093	35,899	36,259
	LO	36,877	36,403	35,217	37,142	36,143	35,310	36,660	37,029	38,392	36,165	36,534
All		36,882	36,388	35,207	37,130	36,120	35,293	36,650	37,013	38,365	36,132	36,518

Table 9: Choosing active constraints: Lower bounds of 100_500_0_0.25 instances

Instance		1	2	3	4	5	6	7	8	9	10	Ave
	LP	12,445	12,476	12,448	12,466	12,475	12,495	12,480	12,485	12,481	12,463	12,472
#	UB	14,413	14,497	14,397	14,401	14,465	14,460	14,416	14,438	14,466	14,440	14,439
40	min	11,708	11,664	11,526	11,652	11,716	11,727	11,630	11,755	11,687	11,690	11,676
	max	11,871	11,914	11,886	11,913	11,888	11,981	11,890	11,922	11,932	11,899	11,910
	ave	11,799	11,809	11,730	11,751	11,810	11,829	11,766	11,824	11,823	11,813	11,796
	LO	12,256	12,276	12,270	12,241	12,230	12,287	12,191	12,286	12,258	12,281	12,258
80	min	12,273	12,337	12,192	12,338	12,308	12,385	12,245	12,311	12,324	12,354	12,307
	max	12,412	12,462	12,466	12,427	12,449	12,461	12,467	12,437	12,463	12,440	12,448
	ave	12,350	12,389	12,326	12,374	12,381	12,417	12,359	12,392	12,390	12,398	12,378
	LO	12,596	12,626	12,601	12,626	12,628	12,656	12,646	12,642	12,645	12,626	12,629
100	min	12,454	12,428	12,483	12,484	12,475	12,501	12,529	12,504	12,502	12,434	12,480
	max	12,538	12,560	12,538	12,561	12,565	12,609	12,601	12,578	12,575	12,583	12,571
	ave	12,497	12,510	12,509	12,528	12,535	12,556	12,554	12,545	12,540	12,507	12,529
	LO	12,659	12,701	12,660	12,686	12,704	12,722	12,720	12,710	12,709	12,690	12,696
200	min	12,851	12,862	12,853	12,862	12,875	12,896	12,886	12,878	12,886	12,869	12,872
	max	12,881	12,932	12,890	12,906	12,903	12,930	12,922	12,923	12,913	12,903	12,910
	ave	12,869	12,897	12,869	12,885	12,891	12,918	12,900	12,906	12,901	12,888	12,893
	LO	12,897	12,901	12,882	12,895	12,902	12,923	12,926	12,910	12,900	12,889	12,903
300	min	13,000	13,030	13,011	13,018	13,032	13,056	13,036	13,044	13,030	13,023	13,028
	max	13,022	13,047	13,019	13,039	13,048	13,071	13,054	13,059	13,052	13,045	13,046
	ave	13,013	13,039	13,015	13,032	13,040	13,062	13,044	13,050	13,041	13,035	13,038
	LO	13,009	13,035	13,010	13,028	13,027	13,059	13,048	13,057	13,034	13,019	13,033
400	min	13,091	13,124	13,100	13,111	13,117	13,140	13,127	13,135	13,123	13,114	13,119
	max	13,101	13,133	13,109	13,120	13,128	13,150	13,136	13,149	13,132	13,127	13,129
	ave	13,095	13,126	13,103	13,116	13,124	13,146	13,132	13,139	13,126	13,120	13,123
	LO	13,095	13,118	13,100	13,120	13,104	13,138	13,135	13,135	13,124	13,106	13,118
All		13,149	13,183	13,146	13,166	13,182	13,199	13,186	13,192	13,179	13,171	13,175

Table 10: Choosing active constraints: Lower bounds of 100_500_500_1 instances

Instance		1	2	3	4	5	6	7	8	9	10	Ave
	LP	67,239	67,178	69,164	66,777	68,340	71,377	66,713	67,885	65,541	65,278	67,549
#	UB	67,981	67,925	69,866	67,427	69,031	72,217	67,294	68,576	66,189	65,980	68,249
40	min	56,905	55,583	56,241	56,670	55,576	59,877	55,146	56,392	55,710	52,855	56,096
	max	59,463	59,822	60,881	60,320	62,219	61,872	59,847	61,596	59,365	57,499	60,288
	ave	57,876	57,556	58,430	58,233	58,357	60,939	57,588	58,816	57,223	55,742	58,076
	LO	67,318	67,163	69,255	66,920	68,511	71,245	66,925	68,033	65,640	65,450	67,646
80	min	59,687	60,119	60,356	58,963	60,896	63,349	59,074	60,663	59,040	58,706	60,086
	max	63,256	61,675	64,312	62,926	63,876	65,922	62,242	62,880	60,960	60,830	62,888
	ave	61,097	61,126	62,241	61,067	62,322	64,536	61,368	61,925	60,093	59,767	61,555
	LO	67,499	67,380	69,378	67,010	68,664	71,611	67,036	68,152	65,730	65,559	67,802
100	min	60,407	61,227	61,054	60,481	61,801	64,689	60,937	61,137	59,616	59,499	61,085
	max	62,396	63,219	65,075	62,687	64,510	67,173	63,217	64,414	61,967	62,344	63,700
	ave	61,414	62,138	62,998	61,946	63,229	65,824	61,973	63,056	60,999	60,595	62,418
	LO	67,503	67,406	69,377	67,032	68,672	71,629	67,030	68,154	65,736	65,570	67,811
200	min	63,628	64,269	63,980	63,886	64,606	68,420	64,352	64,362	62,513	62,141	64,216
	max	65,357	66,105	67,234	65,728	66,493	69,868	65,373	66,431	64,398	63,418	66,041
	ave	64,389	64,929	65,623	64,561	65,598	69,108	64,797	65,179	63,477	62,869	65,053
	LO	67,494	67,391	69,368	67,014	68,663	71,623	67,016	68,150	65,729	65,556	67,800
300	min	65,138	65,418	66,917	65,084	66,360	68,892	65,240	66,303	63,832	63,817	65,701
	max	66,286	66,528	67,860	66,554	67,940	70,735	66,203	67,178	64,876	64,660	66,882
	ave	65,846	65,979	67,379	65,689	67,129	69,965	65,883	66,793	64,311	64,392	66,337
	LO	67,483	67,389	69,355	67,001	68,641	71,616	67,005	68,135	65,720	65,544	67,789
400	min	66,536	66,519	67,941	65,883	67,415	70,409	65,961	67,310	64,809	64,774	66,756
	max	67,272	67,055	68,940	66,872	68,473	71,279	66,778	67,948	65,543	65,351	67,551
	ave	66,956	66,819	68,571	66,560	68,046	70,804	66,335	67,593	65,136	65,034	67,186
	LO	67,476	67,377	69,351	66,993	68,630	71,604	66,999	68,127	65,715	65,534	67,781
All		67,470	67,375	69,343	66,988	68,627	71,600	66,988	68,121	65,711	65,531	67,776

obtained with randomly chosen active constraints decrease when the number of active constraints increases, indicating that the choice of active constraints becomes less important as the number of active constraints increases. At the same time, for $K = 0$, the minimum lower bounds obtained with a larger number of active constraints are often greater than the maximum lower bounds obtained with fewer active constraints, which shows that the impact of the number of active constraints is stronger than the choice of active constraints.

In addition, for instances with $K = 0$, note that the lower bounds generally increase when the number of active constraints increases. While for instances with $K = 500$, the lower bounds first increase then decrease as the number of active constraints increases, which indicates that further extending the set of active constraints does not help to improve the lower bounds. This, of course, is a consequence of the time limit.

Furthermore, when $K = 0$, the solutions of the relaxation problems are always infeasible for the original problems. On the other hand, for $K = 500$, the solutions to the relaxation problems are often feasible for the original problem. This is because for instances with

$K = 0$, the LP solutions are highly degenerate, thus ordering constraints according to slack values may not be an accurate indication of importance. However, for instances with $K = 500$, the optimal LP solutions usually have a small number of positive components, and the slack values vary drastically for constraints, thus the LP information is an accurate indication of importance.

5.1.2 Comparing Lower Bounds of Relaxations

Next, we further examine the lower bounds obtained from constraint relaxation, surrogate relaxation, lagrangian relaxation and lazy relaxation. As suggested in section 5.1.1, we choose active constraints using "LO" order for all relaxations. We compare the lower bounds obtained from various relaxations with the number of active constraints varying from 20 to 400. Since the results for instances with 50 variables and 100 variables are similar, we use instances with 50 variables. Table 11 and 12 display the lower bounds computed in one hour (we observe that the results are similar over a five-hour computational time).

Row "All" shows the lower bounds obtained from solving the original problems, that is, all 500 constraints are active. We display the difference between the lower bounds obtained from solving the relaxations and the bounds in row "All". Therefore, a positive number indicates that the lower bound is improved by using the relaxation. Row "NI" presents the results from constraint relaxation. Furthermore, we consider three types of surrogate relaxations. Row "AI" displays the lower bounds obtained from surrogate relaxation, where all surrogate multipliers are fixed to 1, thus the surrogate constraint is formulated as:

$$\sum_{j} \sum_{i=m_1+1}^{m_1+m_2} a_{ij} x_j \geq \sum_{i=m_1+1}^{m_1+m_2} b_i. \tag{76}$$

Row "SAI" shows the results of surrogate relaxation using a scaled surrogate constraint, that is, $\sum_j \lceil \sum_{i=m_1+1}^{m_1+m_2} a_{ij}/|m_2| \rceil x_j \geq \lfloor \sum_{i=m_1+1}^{m_1+m_2} b_i/|m_2| \rfloor$. The surrogate constraints for "SAI" are weaker than surrogate constraints for "AI", but the size of the constraints are scaled down, which may be advantageous. The row "SR" shows the results from a surrogate relaxation algorithm that iterates with a subgradient method suggested in Galvao et al. [26] as described in Section 5.1. For each fixed surrogate multiplier, the time limit for solving the surrogate relaxation is 10 minutes. Row "LR" represents the results of the

lagrangian relaxation with the subgradient method as described in section 5.1. For each fixed lagrangian multiplier, we use a time limit of 10 minutes for solving $z_{LD}(\lambda)$. Row "Lazy" includes the results of lazy relaxations.

Since the solutions found in lazy relaxations are feasible to the original instances, we also compare the upper bounds obtained in Table 13 and 14. Again, row "All" represents the upper bounds obtained from solving the original instances, and the rest of the rows display the difference between the upper bounds obtained from lazy relaxations and the upper bounds in row "All". Therefore, a negative number means that a better feasible solution is found in the lazy relaxation.

For $K = 0$, except for the lazy relaxation, the lower bounds of all relaxations increase when the number of active constraints increases. For instances with $K = 500$, the lower bounds of all relaxations first increase then decrease when the number of active constraints increases, which indicates again that for such instances, a large number of constraints are not helpful in improving the lower bounds. The results confirm that using "LO" order to indicate the importance of constraints is meaningful for instances with $K = 500$. However, for instances with $K = 0$, because of the degeneracy of the optimal LP solution, the number of active constraints seems to be more important for obtaining good lower bounds for all relaxation algorithms.

For most instances, the lower bounds obtained from surrogate and lagrangian relaxations with iterative updates on multipliers are often the smallest among all lower bounds, because the number of iterations were limited. Since the success of both relaxation algorithms depend on searching through a large number of multipliers, rapidly solving the relaxation problems for each fixed multiplier is required. However, for MKP, the relaxation problem with only 20 active constraints may still be hard to solve, which prevents the algorithms from conducting a large number of iterations to find a good multiplier. By further reducing the time limit for each iteration, we find that the lower bounds obtained are not any better. Since with 100 active constraints, the number of iterations for all instances with lagrangian relaxations is around 10, further increasing the time limit for each iteration will decrease the number of iterations. Therefore, the lower bounds improvements obtained by increasing

Table 11: Lower bounds obtained from various relaxations: 50_500_0_0.25 instances

	Instance	1	2	3	4	5	6	7	8	9	10	Ave
# Active	All	7353	7318	7332	7321	7309	7294	7355	7386	7323	7357	7335
20	NI	-970	-963	-973	-932	-973	-951	-947	-946	-947	-954	-955
	AI	-970	-963	-973	-932	-973	-951	-947	-946	-947	-954	-955
	SAI	-970	-963	-973	-932	-973	-951	-947	-946	-947	-954	-955
	SR	-932	-921	-921	-917	-906	-908	-931	-900	-898	-912	-914
	LR	-970	-963	-956	-961	-953	-950	-961	-939	-945	-954	-955
	Lazy	-36	-35	-31	-30	-15	-17	-23	-11	-3	-11	-21
40	NI	-651	-659	-657	-644	-653	-643	-672	-609	-624	-648	-646
	AI	-658	-654	-659	-642	-650	-644	-675	-613	-630	-650	-647
	SAI	-660	-656	-657	-644	-658	-644	-670	-615	-628	-653	-648
	SR	-747	-748	-752	-735	-741	-739	-759	-715	-722	-741	-739
	LR	-754	-751	-751	-743	-742	-738	-764	-718	-727	-744	-743
	Lazy	-12	-19	-16	-22	-15	-17	-16	-14	-20	-12	-16
80	NI	-399	-408	-410	-407	-417	-437	-416	-409	-397	-426	-412
	AI	-399	-408	-380	-407	-417	-437	-416	-409	-397	-426	-409
	SAI	-399	-408	-407	-407	-417	-437	-416	-409	-397	-426	-412
	SR	-535	-525	-542	-515	-518	-540	-509	-536	-503	-554	-527
	LR	-529	-537	-504	-541	-540	-547	-547	-542	-515	-535	-533
	Lazy	-5	-19	-17	-13	-6	-4	-6	-7	-4	-4	-8
100	NI	-292	-413	-420	-425	-369	-361	-328	-357	-384	-351	-370
	AI	-365	-419	-362	-356	-369	-361	-328	-334	-395	-351	-364
	SAI	-339	-413	-392	-356	-369	-361	-328	-355	-390	-351	-365
	SR	-485	-511	-456	-468	-475	-476	-446	-465	-489	-484	-475
	LR	-500	-505	-487	-468	-479	-488	-503	-466	-501	-507	-490
	Lazy	-14	-4	-15	-7	0	4	5	2	3	3	-2
200	NI	-188	-192	-181	-188	-179	-215	-163	-187	-179	-167	-183
	AI	-187	-190	-182	-184	-175	-199	-161	-167	-180	-181	-180
	SAI	-194	-185	-175	-184	-181	-205	-161	-176	-175	-182	-181
	SR	-307	-306	-302	-299	-293	-321	-289	-297	-292	-291	-299
	LR	-325	-305	-317	-305	-300	-314	-287	-308	-298	-306	-306
	Lazy	-17	-11	-18	-19	-2	-11	-2	-2	0	-4	-8
300	NI	-128	-126	-120	-116	-115	-126	-101	-100	-121	-120	-117
	AI	-122	-120	-124	-118	-105	-127	-114	-94	-116	-110	-115
	SAI	-120	-113	-113	-117	-107	-128	-114	-101	-121	-108	-114
	SR	-239	-236	-236	-236	-222	-238	-224	-214	-234	-232	-231
	LR	-241	-248	-249	-250	-238	-249	-219	-231	-236	-236	-239
	Lazy	-15	-19	-10	-19	-11	-12	-6	-9	-2	-9	-11
400	NI	-47	-50	-58	-65	-38	-49	-39	-37	-56	-48	-48
	AI	-54	-54	-57	-48	-43	-50	-41	-37	-54	-46	-48
	SAI	-54	-60	-62	-50	-52	-45	-41	-53	-54	-45	-51
	SR	-179	-178	-184	-173	-167	-178	-174	-163	-172	-173	-174
	LR	-182	-177	-183	-187	-171	-177	-167	-174	-182	-166	-176
	Lazy	-3	-11	-11	-10	-8	-8	-6	0	5	0	-5

Table 12: Lower bounds obtained from various relaxations: 50_500_500_1 instances

	Instance	1	2	3	4	5	6	7	8	9	10	Ave
# Active	All	36882	36388	35207	37130	36120	35293	36650	37013	38365	36132	36518
20	NI	-665	-999	-1083	-806	-1256	-1084	-922	-958	-1068	-449	-929
	AI	-660	-997	-1072	-799	-1183	-1006	-923	-956	-1059	-449	-910
	SAI	-667	-996	-1082	-810	-1191	-1023	-922	-956	-1060	-449	-916
	SR	-424	-601	-559	-542	-500	-481	-577	-560	-542	-453	-524
	LR	-470	-602	-572	-542	-500	-481	-578	-560	-542	-519	-537
	Lazy	79	72	83	57	27	42	77	85	70	33	63
40	NI	-68	-237	-262	-263	-187	-250	-198	-48	-227	-80	-182
	AI	-62	-234	-263	-259	-188	-245	-196	-49	-231	-80	-181
	SAI	-63	-248	-262	-261	-185	-246	-192	-46	-227	-80	-181
	SR	-115	-260	-293	-292	-230	-277	-201	-118	-250	-187	-222
	LR	-131	-297	-312	-317	-239	-292	-252	-125	-282	-200	-245
	Lazy	60	52	57	44	48	51	47	62	52	33	51
80	NI	46	-26	44	-3	-4	-28	23	64	-3	33	15
	AI	84	-27	50	-6	-2	-32	33	73	0	33	21
	SAI	52	-31	52	0	-4	-25	28	75	1	33	18
	SR	-38	-113	-47	-80	-83	-93	-65	-38	-66	11	-61
	LR	-38	-123	-58	-91	-84	-92	-67	-41	-85	15	-66
	Lazy	91	40	51	40	38	38	39	56	46	33	47
100	NI	79	23	84	30	52	25	56	94	69	33	55
	AI	63	52	75	53	46	16	55	88	74	33	56
	SAI	48	51	78	57	38	17	50	95	80	33	55
	SR	-23	-71	-37	-66	-48	-59	-53	-31	-39	33	-39
	LR	-26	-71	-36	-68	-49	-58	-53	-33	-38	33	-40
	Lazy	58	47	48	31	42	25	33	63	53	33	43
200	NI	108	59	63	44	59	47	56	62	59	33	59
	AI	46	56	67	41	54	40	55	64	59	33	52
	SAI	55	57	65	45	55	39	51	52	45	33	50
	SR	-38	-70	-54	-56	-50	-50	-55	-55	-63	-29	-52
	LR	-38	-67	-52	-58	-49	-50	-56	-55	-62	-28	-52
	Lazy	39	47	48	36	34	21	40	55	37	33	39
300	NI	16	32	34	27	32	33	13	24	30	33	27
	AI	34	30	39	20	34	29	25	41	26	33	31
	SAI	45	28	38	12	29	28	15	37	36	33	30
	SR	-73	-93	-79	-76	-75	-62	-77	-84	-81	-68	-77
	LR	-73	-92	-80	-78	-79	-63	-76	-82	-82	-67	-77
	Lazy	13	33	37	24	28	32	12	34	30	33	28
400	NI	-1	13	13	9	22	15	7	14	23	33	15
	AI	10	12	16	8	23	16	20	21	13	33	17
	SAI	27	1	16	10	13	10	-1	16	10	33	14
	SR	-95	-107	-90	-88	-81	-77	-92	-94	-88	-92	-90
	LR	-76	-106	-92	-91	-79	-79	-95	-94	-88	-85	-89
	Lazy	-2	11	15	8	16	19	15	9	24	33	15

Table 13: Upper bounds obtained from the lazy relaxations: 50_500_0_0.25 instances

Instance	1	2	3	4	5	6	7	8	9	10	Ave
All	7943	7879	7917	7916	7879	7907	7945	8008	7941	7958	7930
20	-12	-4	-19	-3	-1	-24	17	-19	-29	-17	-11
40	-4	5	1	-6	11	-26	-33	-10	-46	-27	-13
80	-3	14	-46	-13	22	-33	-17	-21	-4	4	-9
100	17	36	26	16	-17	-10	35	5	-34	-5	7
200	37	38	5	15	24	-21	3	20	-15	3	11
300	-16	38	-33	-21	19	-18	20	-10	-21	8	-3
400	-35	22	9	-8	21	8	21	-17	-29	3	0

Table 14: Upper bounds obtained from the lazy relaxations: 50_500_500_1 instances

Instance	1	2	3	4	5	6	7	8	9	10	Ave
All	37085	36575	35437	37414	36336	35620	36840	37213	38572	36168	36726
20	-65	0	-15	-41	0	-41	0	-6	0	0	-16
40	-65	0	-21	-41	0	-61	0	-6	0	0	-19
80	-65	0	-21	-41	0	-61	0	-6	0	0	-19
100	-65	0	-21	-22	0	-41	0	-6	0	0	-15
200	36	0	-21	-41	0	28	0	-6	40	0	3
300	-65	0	-21	-41	0	-61	0	-6	0	0	-19
400	36	1	-21	-41	22	-41	0	0	0	0	-4

the time limit for each iteration is also limited.

On the other hand, the lower bounds obtained by lazy relaxations are often the best among all relaxations. Particularly for instances with $K = 500$, lazy relaxation shows a strong advantage when a small number of active constraints are used, which indicates that many inactive constraints are actually never violated by solutions to the relaxation problems, and thus can be eliminated without sacrificing the lower bounds. In addition, we see that for instances with $K = 0$, the upper bounds of lazy relaxations are better when a small number of active constraints are used. This indicates that using a small number of active constraints may also help to obtain better upper bounds.

Furthermore, note that for instances with $K = 0$, the lower bounds from relaxation problems almost always increase as the number of active constraints increases. Therefore, it is natural to ask that for MKP with a larger number of constraints, is there an optimal number of active constraints such that further increasing it does not help increase the lower bounds? To further explore this question, we generate two sets of instances with 1000 and 5000 constraints, respectively, using methods in Chapter 4. All instances have 50 unbounded integer variables, $\alpha = 0.25$, and $K = 0$, with 10 instances in each group. Since the lazy relaxation gives the best lower bounds among all relaxations, we use lazy relaxations to obtain lower bounds, which are shown in Tables 15 and 16.

For instances with 1000 constraints, the lower bounds obtained from relaxing 400 or 500 constraints are often better than the lower bounds obtained in the original problems. For instances with 5000 constraints, using more than 500 or 1000 active constraints usually produce good lower bounds. In addition, the improvement on lower bounds always fluctuates when the number of active constraints increases, that is, increasing the number of active

Table 15: LB using Lazy Constraints: 50_1000_0_0.25

Instance	1	2	3	4	5	6	7	8	9	10	Ave
All	7454	7434	7431	7409	7470	7454	7460	7462	7449	7418	7442
20	-40	-21	-30	-39	-34	-32	-47	-36	-68	-18	-36
40	-40	-34	-32	-26	-24	-28	-40	-61	-80	-37	-40
80	-37	-24	-49	-29	-33	-43	-40	-52	-55	-47	-40
100	-17	-26	-17	-23	-21	-22	-14	-24	-34	-27	-22
200	-21	-1	-12	3	1	0	0	-13	-6	-11	-6
300	-3	8	9	-5	-2	-17	-13	-11	-14	12	-3
400	0	16	10	2	1	10	12	-17	-10	16	4
500	9	18	8	20	8	-4	-10	-14	3	16	6
600	2	16	13	1	-4	-1	-6	-12	-16	7	0
700	-13	4	12	-4	4	-6	-6	-7	-12	0	-2
800	-5	10	14	4	-1	-11	-10	-25	-13	11	-2
900	-1	10	17	-4	5	-5	-8	-14	-7	3	0

Table 16: LB using Lazy Constraints: 50_5000_0_0.25

Instance	1	2	3	4	5	6	7	8	9	10	Ave
All	7535	7524	7495	7515	7507	7491	7511	7542	7497	7572	7519
20	6	7	7	28	1	19	28	0	7	-45	6
40	-13	0	18	1	13	20	10	-5	9	-59	0
80	-10	-7	4	-1	18	22	13	-11	11	-49	-1
100	-19	-4	20	-7	11	2	6	-10	8	-64	-5
500	41	60	68	61	60	69	50	27	65	-4	50
1000	15	26	37	14	32	42	30	15	31	-20	23
1500	24	21	35	25	31	40	32	3	32	-39	21
2000	7	28	35	26	43	35	14	-4	17	-32	17
2500	0	22	39	23	40	38	23	0	23	-43	17
3000	8	28	41	25	36	18	34	13	45	-47	21
3500	31	44	36	14	38	41	24	9	32	-16	26
4000	30	22	47	18	30	33	19	1	26	-32	20
4500	4	4	40	3	33	23	17	2	36	-42	12

constraints does not have a uniform impact on lower bounds, since adding more active constraints may not help tighten the feasible region. Therefore, our results show that for most instances, solving the relaxations can help to improve lower bounds. However, note that for Instance 10 with 5000 constraints, the lower bounds obtained from the relaxations never outperform the lower bounds of the original problem. It demonstrates that the number of active constraints needed varies.

The advantage of the lazy relaxation algorithm is in its ability to add active constraints to the relaxation. Especially for instances with $K = 0$, determining good active constraints can be challenging, thus adding more active constraints can be useful to strengthen the relaxation problems and obtain better lower bounds. Next, we study solving relaxations with diversified sets of active constraints to improve lower bounds.

5.1.3 A Dual Heuristic Algorithm

We consider a dual heuristic algorithm, which solves relaxation problems to improve the linear relaxation bounds for nodes in the B&B tree. Generally, the integer programming problem at a node of the B&B tree for problem (70) can be stated as:

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & A_1 x \geq b_1 \\ & A_2 x \geq b_2 \\ & l_i \leq x_i \leq u_i, i = 1, ..., n, \\ & x \in \mathbb{Z}_n^+, \end{aligned} \tag{77}$$

where l_i and u_i are bounds for variable x_i that may be derived from previous branching decisions. The lower bound of the node is obtained by solving the linear relaxation of (77).

At a chosen node, we solve a relaxation of the associated subproblem. We call a node *special* if a relaxation is solved, and *processed* if it is pruned or two children nodes are generated from it in the B&B algorithm.

Here we use a surrogate relaxation, where the surrogate constraint is obtained from summing up all inactive constraints, as it balances time efficiency and bound quality as shown in Section 5.1.2. Thus, the relaxation for (77) with active constraints $A_1 x \geq b_1$ can

be written as:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } A_1 x \geq b_1 \\ &\quad (\sum_{i=m_1+1}^{m_1+m_2} a_i)x \geq \sum_{i=m_1+1}^{m_1+m_2} b_i, \\ &\quad l_i \leq x_i \leq u_i, i = 1, ..., n, \\ &\quad x \in \mathbb{Z}_n^+, \end{aligned} \tag{78}$$

where a_i for $i = m_1 + 1, ..., m_1 + m_2$ are the row vectors of matrix A_2.

As suggested in Section 5.1.1, active constraints can be selected according to the dual solutions and slack values. Because additional constraints may be added during B&B, the linear dual solutions for the associated subproblems may not be applicable. Therefore, we use the slack values of the constraints to select active constraints in the relaxations. In particular, we choose active constraints with slack values less than a threshold value, indicated by *Slack*.

Let LB indicate the lower bound obtained from solving the relaxation subproblem, lb be the LP bound of the node, and UB be the best upper bound obtained in B&B at the time when the node is processed. If LB is larger than UB, then the node can be pruned. Otherwise, if LB is greater than lb, the lower bound is improved. To impose the new lower bound in B&B, we add a cutting plane $c^T x \geq LB$ to the child nodes. However, adding such cutting planes increases the sizes of the problems for children nodes and may have a negative influence on the efficiency of the B&B algorithm. Thus, we consider adding the associated cutting plane only when the improvement on the lower bound is larger than a threshold value Δ, that is when $LB - lb > \Delta$.

In addition, since the formulations for problems of a node and its child nodes are not much different, solving the relaxation subproblems at both nodes may be unnecessary. Thus, we restrict the depth between two special nodes in the same subtree to be greater than a threshold value *Depth*.

For an efficient heuristic, the selection of the special nodes is important. Note that even if the relaxations can be solved efficiently, it is not realistic to solve a relaxation at every

node of the B&B tree. Therefore, we use a threshold value $Check\_Percent$ to indicate the largest total percentage of special nodes over all processed nodes in the B&B tree. In addition, to avoid spending too much time on one node, we use a time limit indicated by $Time$ for solving the relaxation subproblems.

Furthermore, we use the relative gaps to choose special nodes, where the relative gap of a node is defined as $\frac{|UB-lb|}{UB}$. Our experiments show that with two types of ranges for the relative gaps, the dual heuristic algorithm obtains the best lower bounds. The first type only includes special nodes with small relative gaps, because such nodes are more likely to be pruned by the relaxations, and thus the size of the B&B tree is likely to be reduced. The second type includes special nodes with large relative gaps. Since the lower bounds of the B&B algorithm are determined by the minimum lower bounds of the remaining nodes in the B&B tree, improving on the worst lower bounds may help improve the overall lower bounds. We use Gap to indicate the range of relative gaps of special nodes. In next section, we consider choosing nodes with relative gaps in both types of ranges.

The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1 A Dual Heuristic Algorithm

For each node v in the B&B tree:

if $\frac{\#\text{ special nodes}}{\#\text{ Processed Nodes}} < Check\_Percent$ and $\frac{|UB-lb|}{UB} \in Gap$ **then**

Solve relaxation subproblem at v with time limit $Time$ and obtain LB:

if $LB > UB$ **then**
Node v is pruned.
else if $LB > lb$ **then**
Lower bound of node v is improved.
if $LB - lb > \Delta$ **then**
Generate two children nodes with cutting plane $c^T x \geq LB$.
else
Generate two children nodes.
end if
else
Lower bounds not improved, generate two children nodes.
end if
end if

We conducted all experiments on 10 instances with 50 variables, 500 constraints, $\alpha = 0.25$ and $K = 0$. For all tests, we use a time limit of 10 hours for the B&B algorithm,

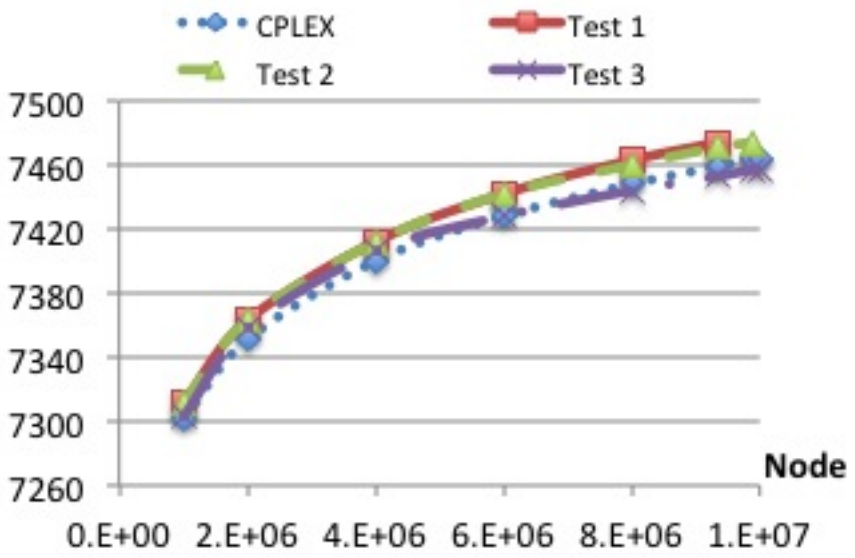
$Slack = 1$ and $Depth = 5$. We use $Time = 0.05$ or 1 second, since we found that extending $Time$ to 2 or 5 seconds gives worse lower bounds. We compare the results from CPLEX 12.2 with the results from the dual heuristic algorithm using three sets of parameters:

Test 1: $Check\_Percent$=100%, $Time$=1 second, Gap = 0% - 2%, $\Delta = 1$. Thus for all nodes with relative gaps less than 2%, a relaxation is solved with time limit 1 second, a cutting plane is added to children nodes if the lower bound obtained from the relaxation is greater than the LP relaxation by one.

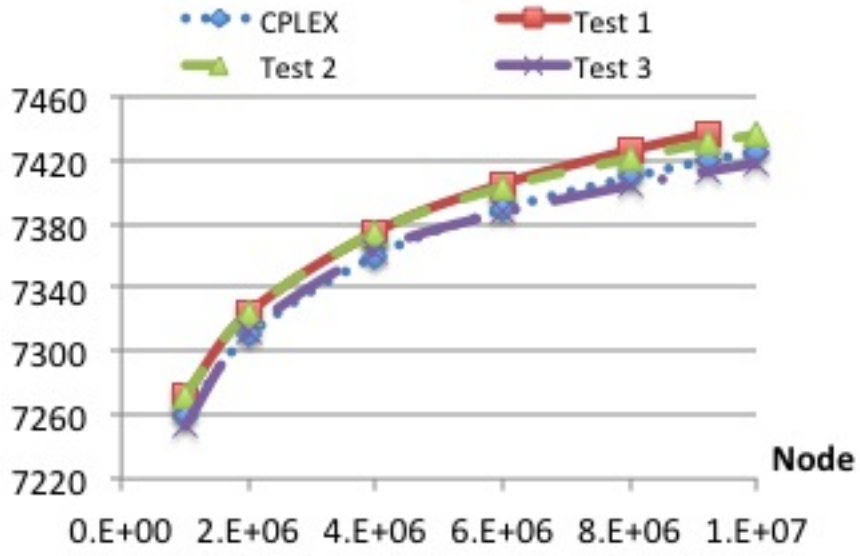
Test 2: $Check\_Percent$=20%, $Time$= 1 second, Gap = 0 %- 2%, $\Delta = 1$. Thus for nodes with relative gaps less than 2% and total percentage of special nodes less than 20%, a relaxation is solved with time limit 1 second, a cutting plane is added to children nodes if the lower bound obtained from the relaxation is greater than the LP relaxation by one.

Test 3: $Check\_Percent$=2%, $Time$= 0.05 second, Gap = 8%-100%, $\Delta = 0$. Therefore, for nodes with relative gaps greater than 8% and total percentage of special nodes less than 2%, a relaxation is solved with time limit 0.05 second, a cutting plane is added to children nodes if the lower bound obtained from the relaxation is greater than the LP relaxation.

To compare the results, we study the lower bounds, number of remaining nodes in the B&B tree, relative gaps, and average time spent on each node in the B&B tree. We report the results after a certain number of nodes have been processed in the B&B tree in Figures 20 to 39.

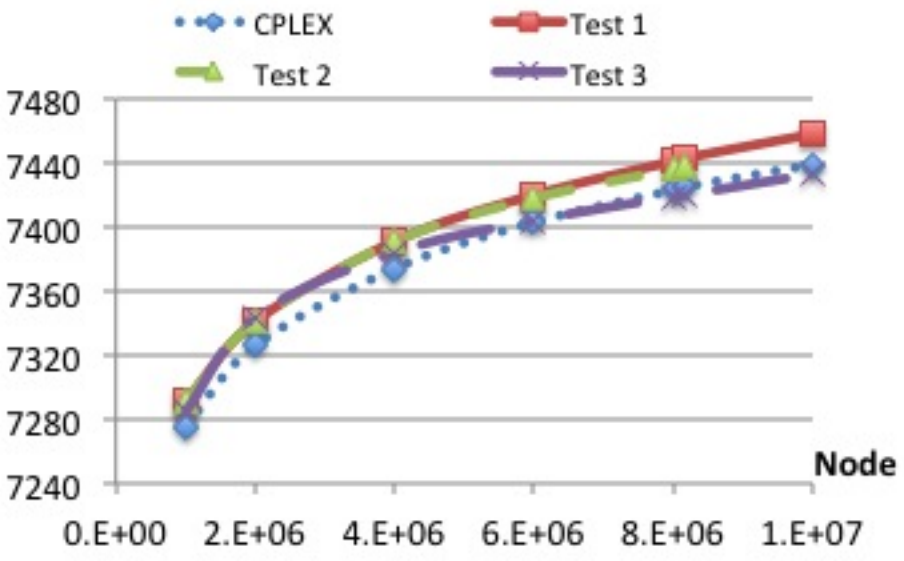


(a) Instance 1

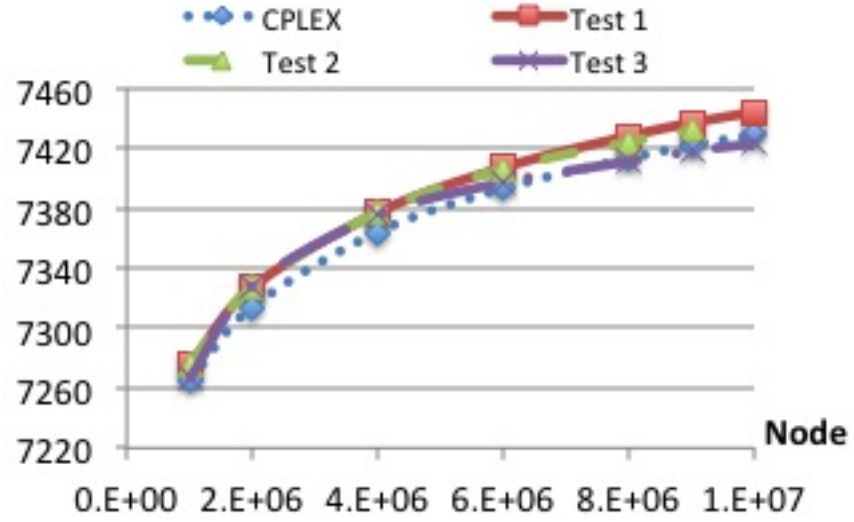


(b) Instance 2

Figure 20: Lower bounds

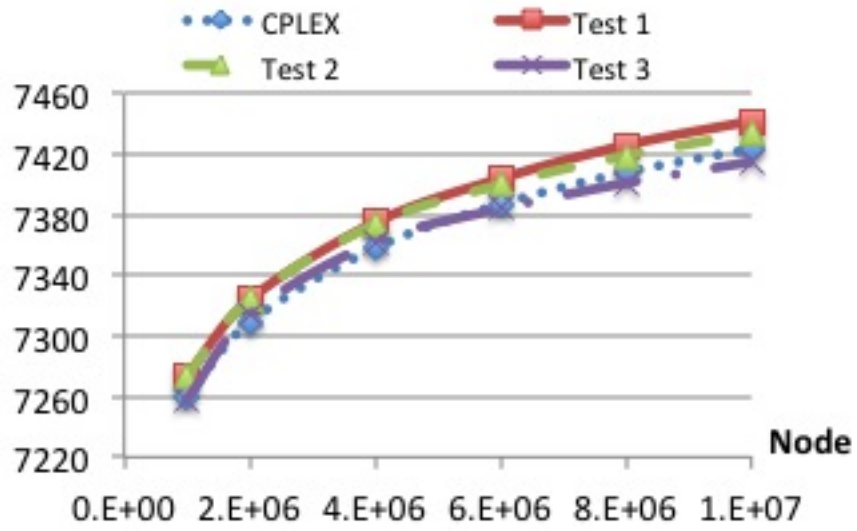


(a) Instance 3

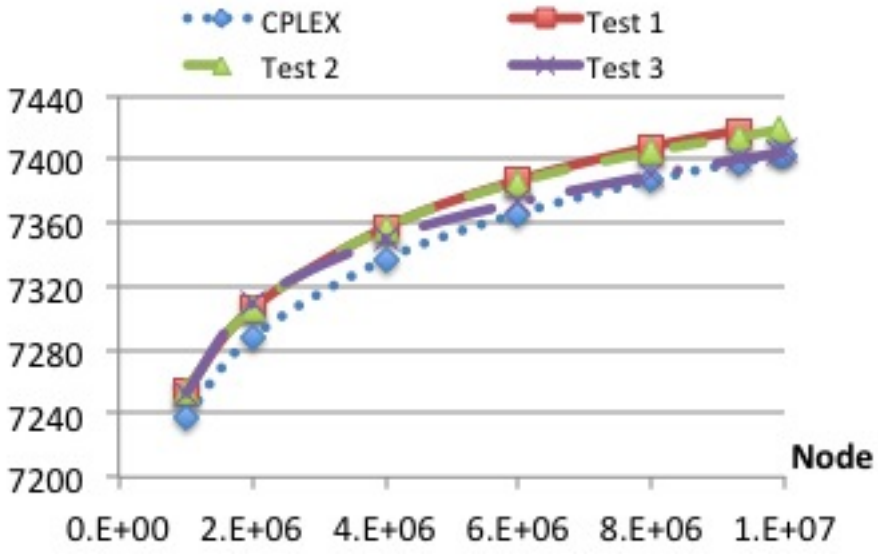


(b) Instance 4

Figure 21: Lower bounds

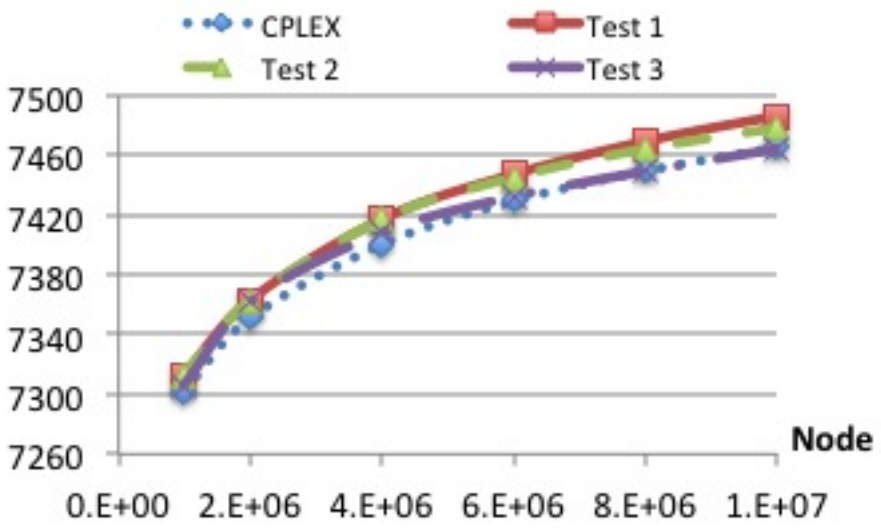


(a) Instance 5

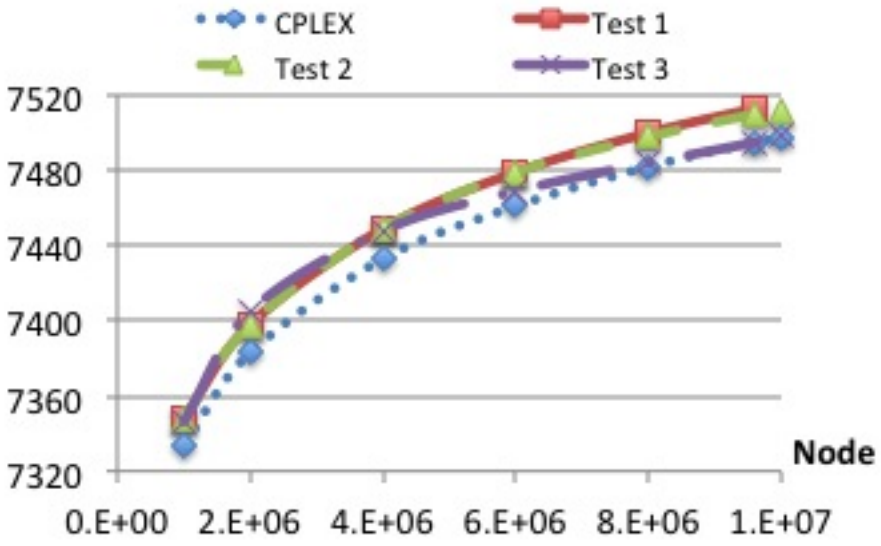


(b) Instance 6

Figure 22: Lower bounds

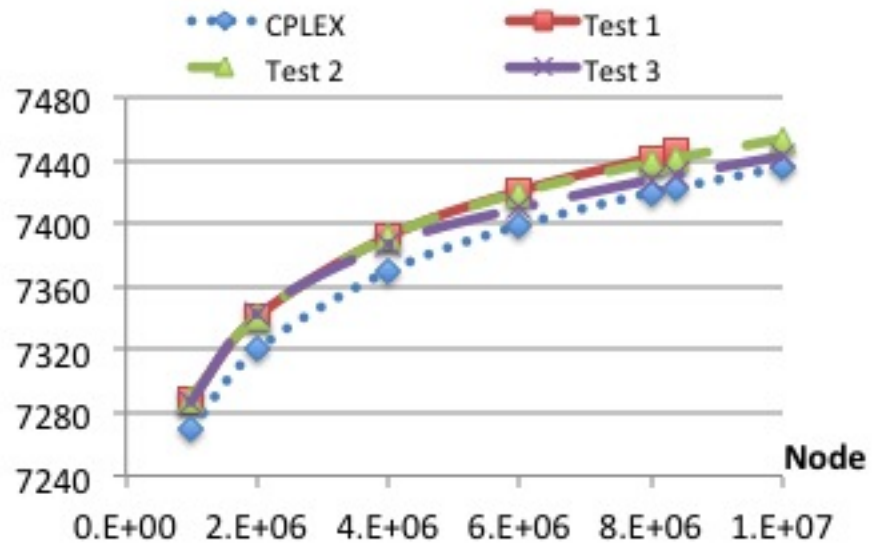


(a) Instance 7

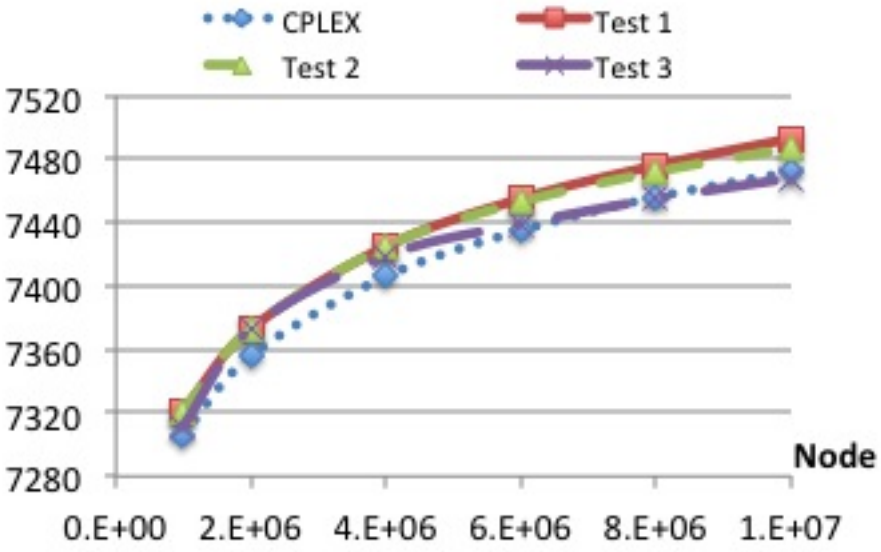


(b) Instance 8

Figure 23: Lower bounds

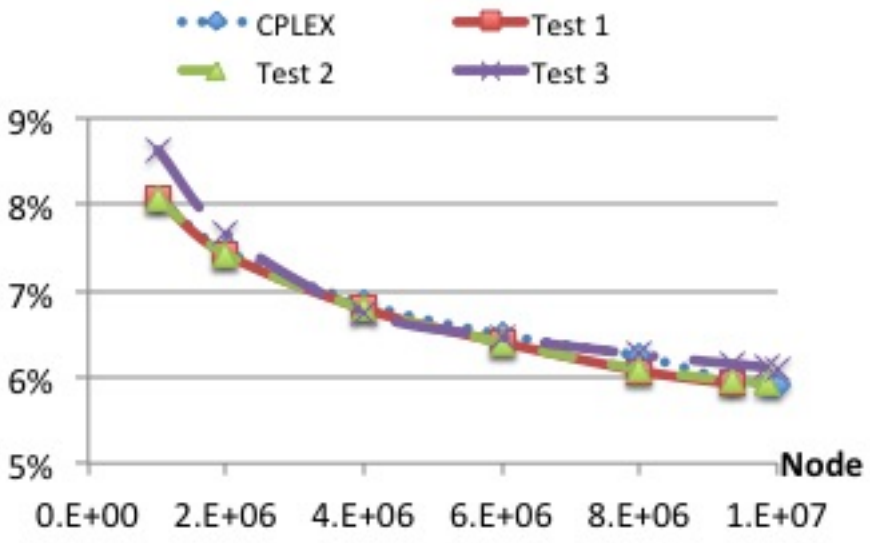


(a) Instance 9

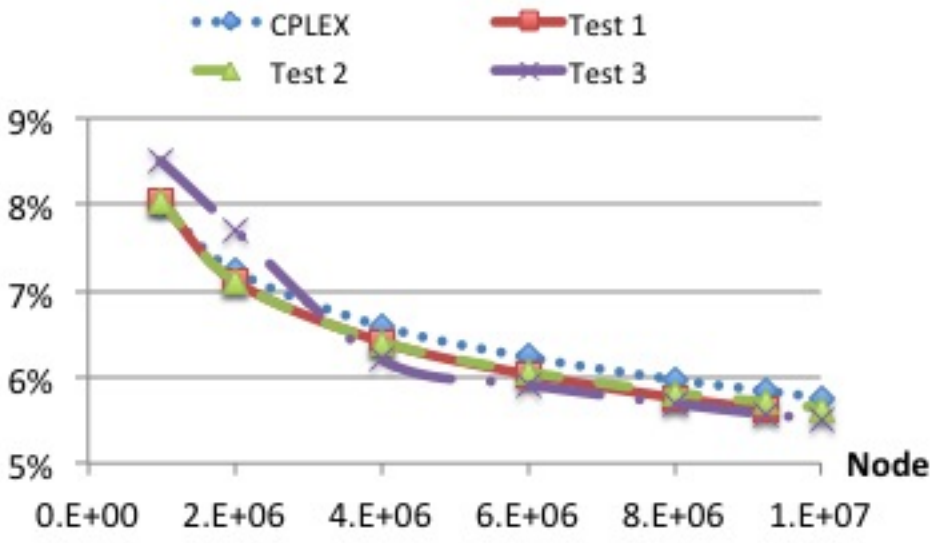


(b) Instance 10

Figure 24: Lower bounds

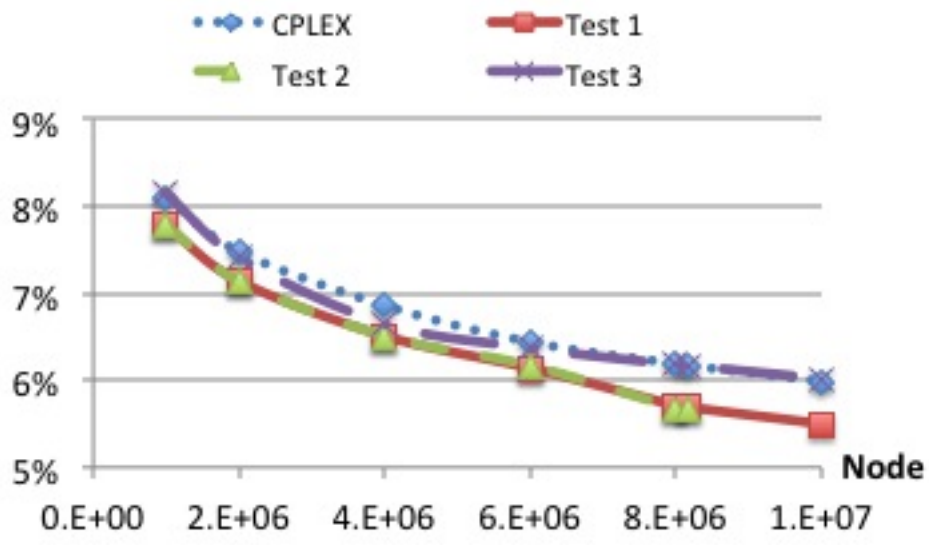


(a) Instance 1

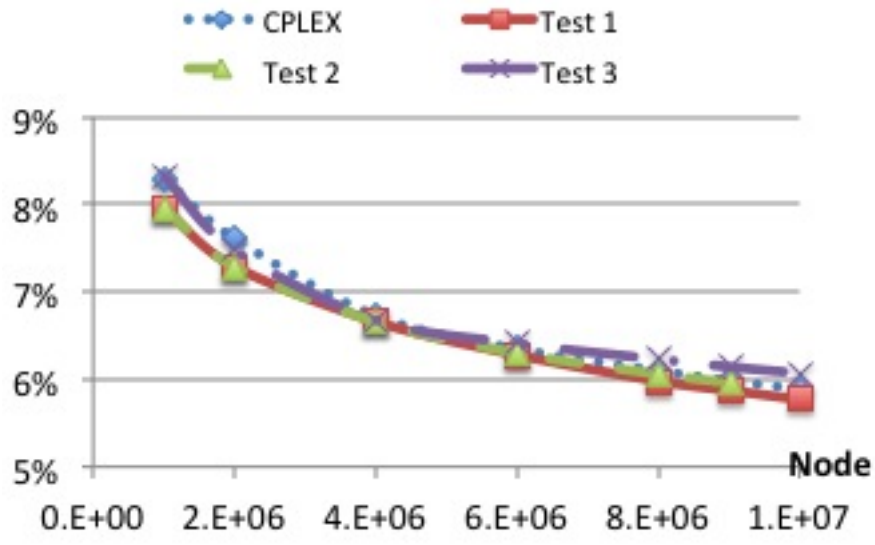


(b) Instance 2

Figure 25: Relative gaps

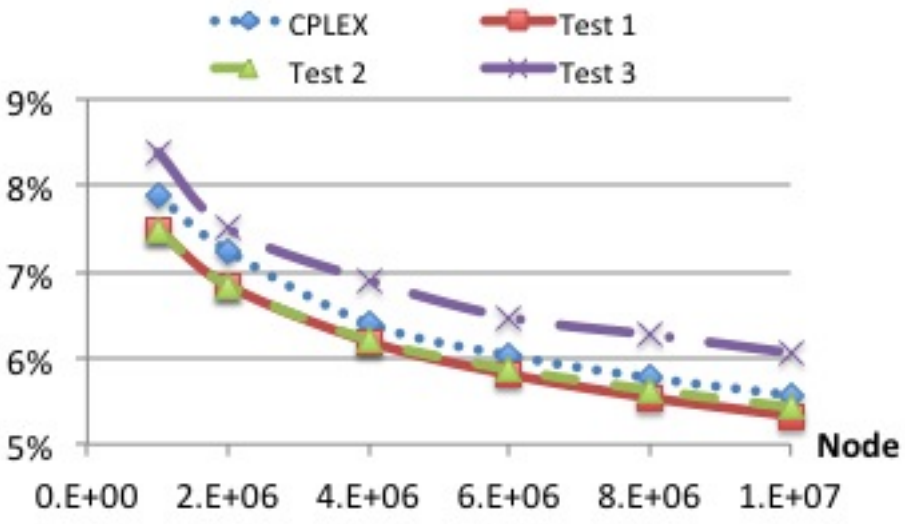


(a) Instance 3

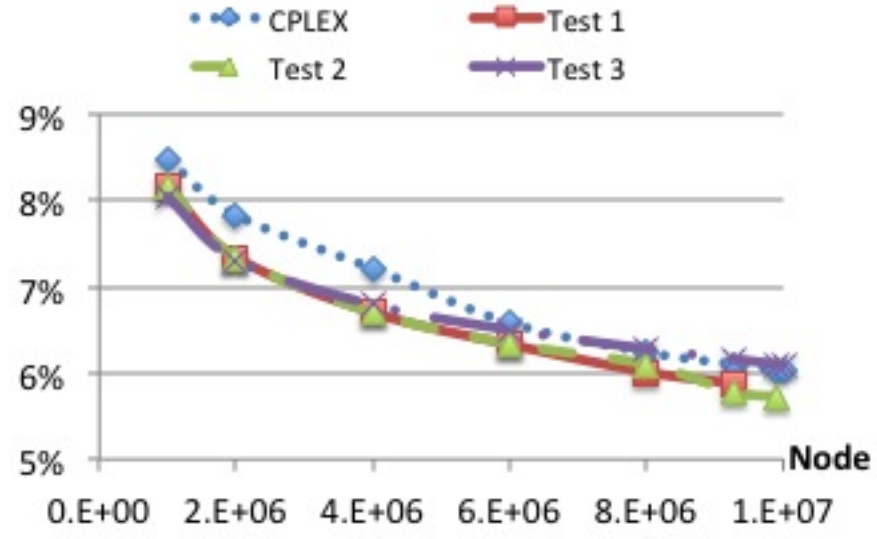


(b) Instance 4

Figure 26: Relative gaps

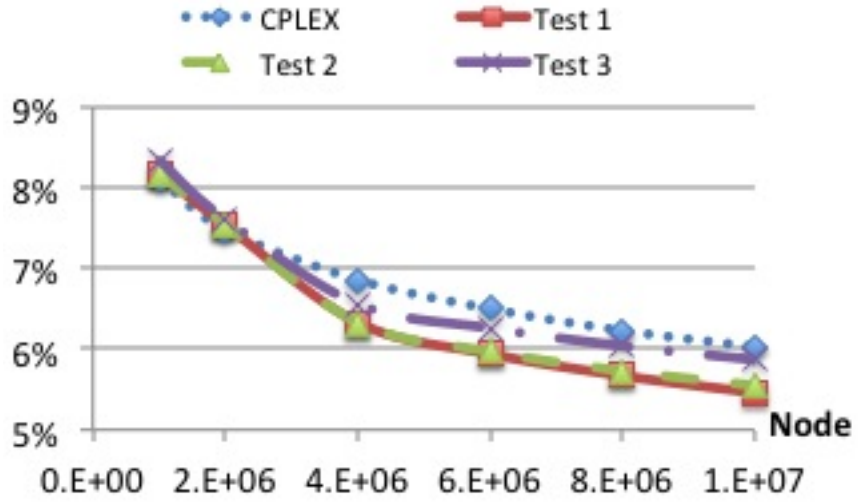


(a) Instance 5

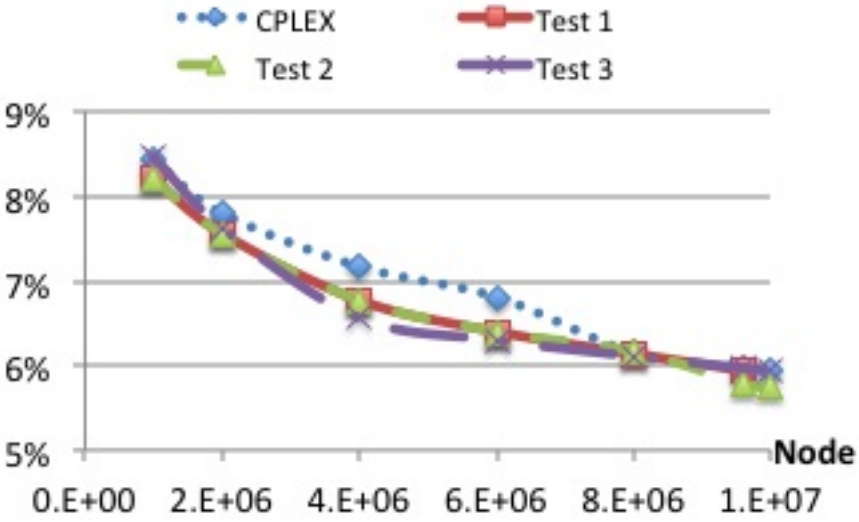


(b) Instance 6

Figure 27: Relative gaps

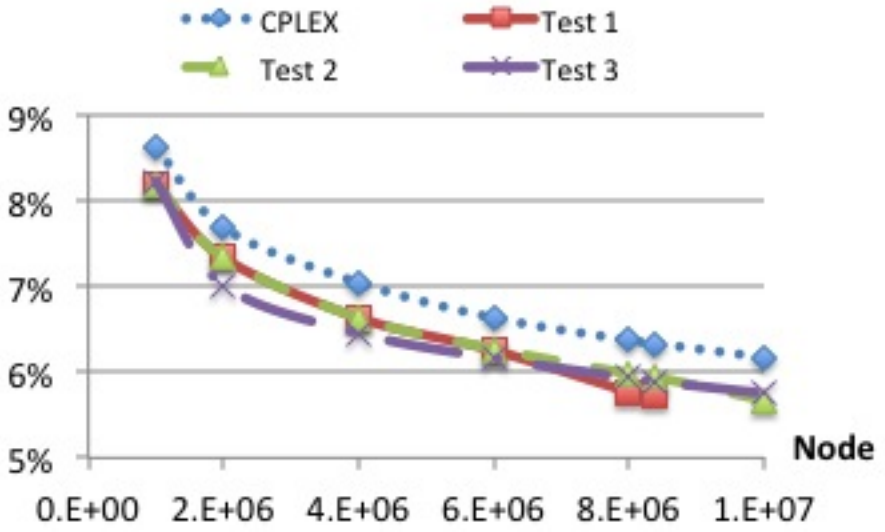


(a) Instance 7

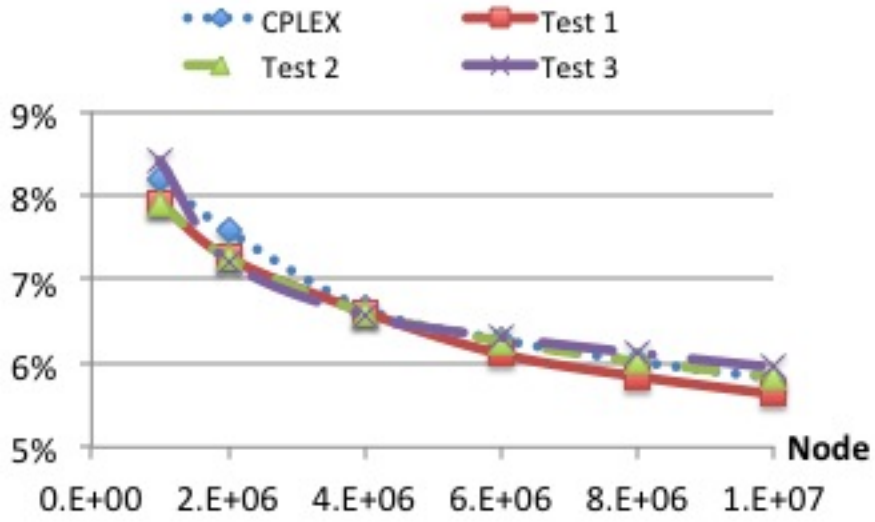


(b) Instance 8

Figure 28: Relative gaps

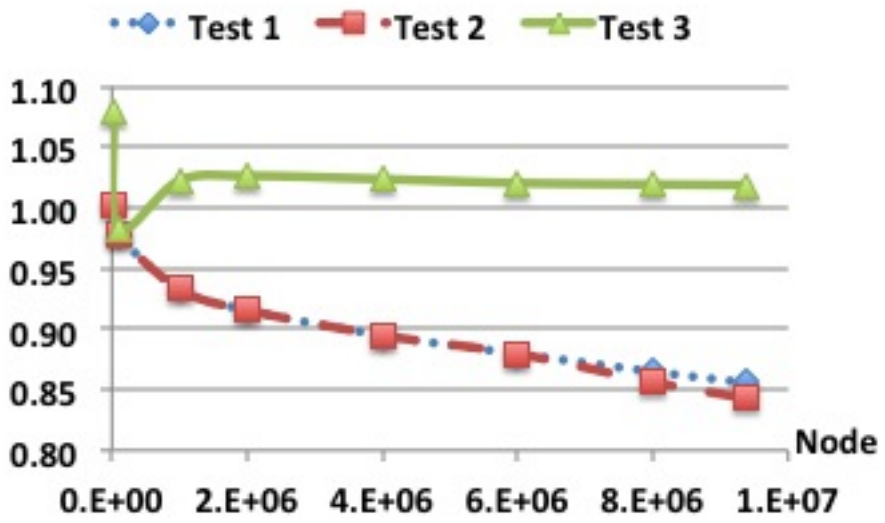


(a) Instance 9

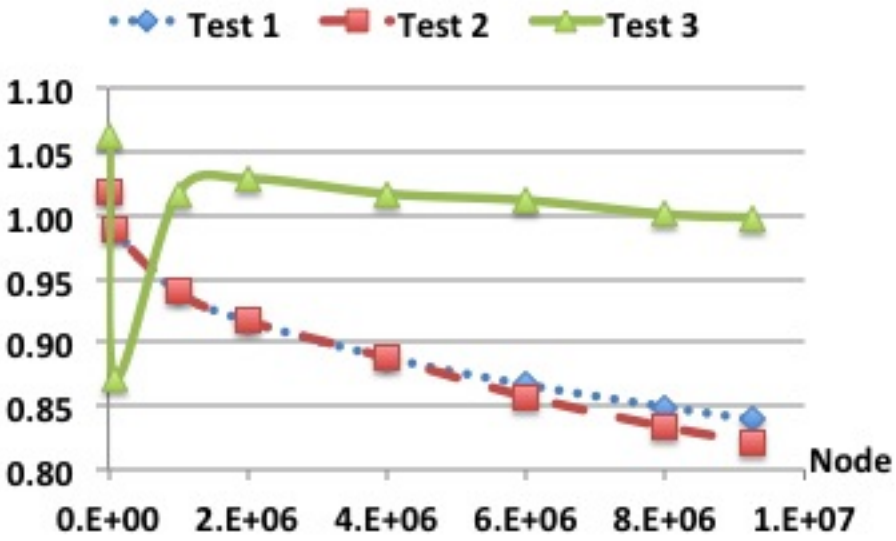


(b) Instance 10

Figure 29: Relative gaps

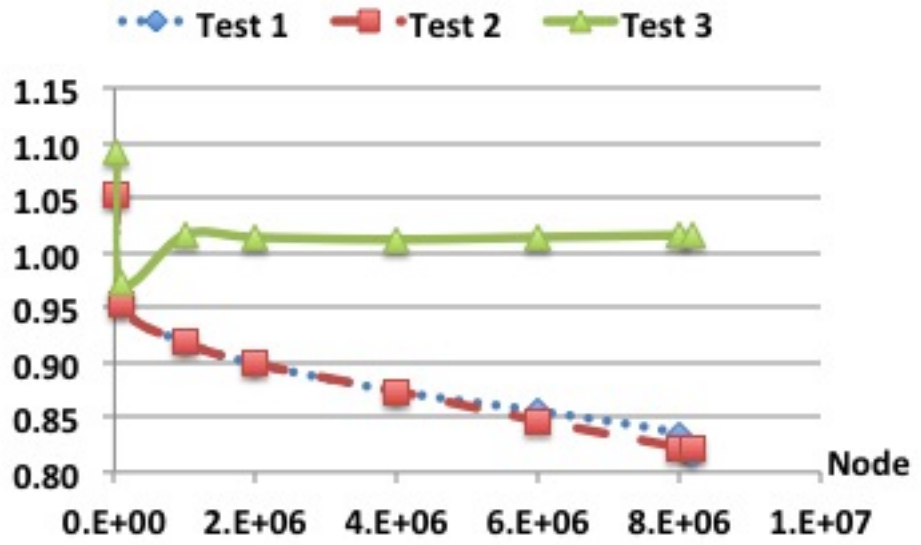


(a) Instance 1

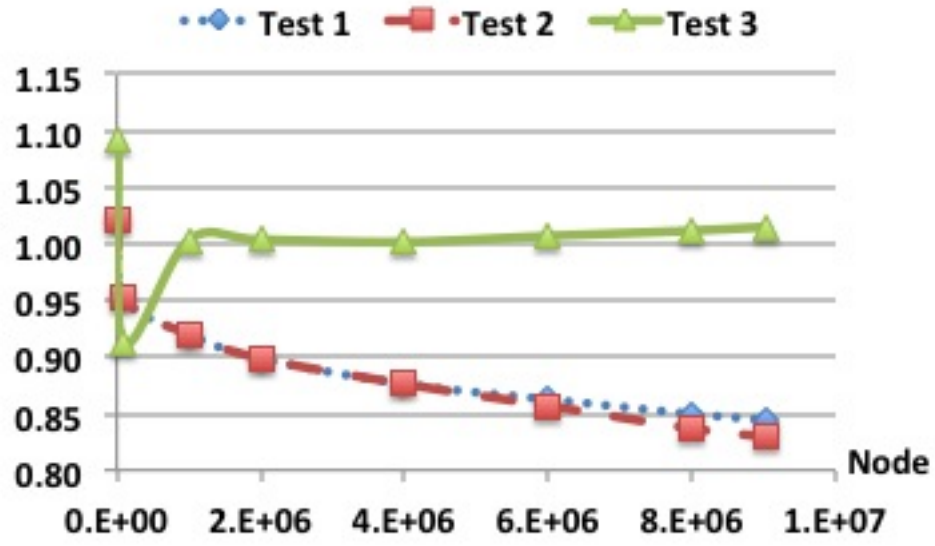


(b) Instance 2

Figure 30: # Remaining nodes in B&B tree compared with CPLEX

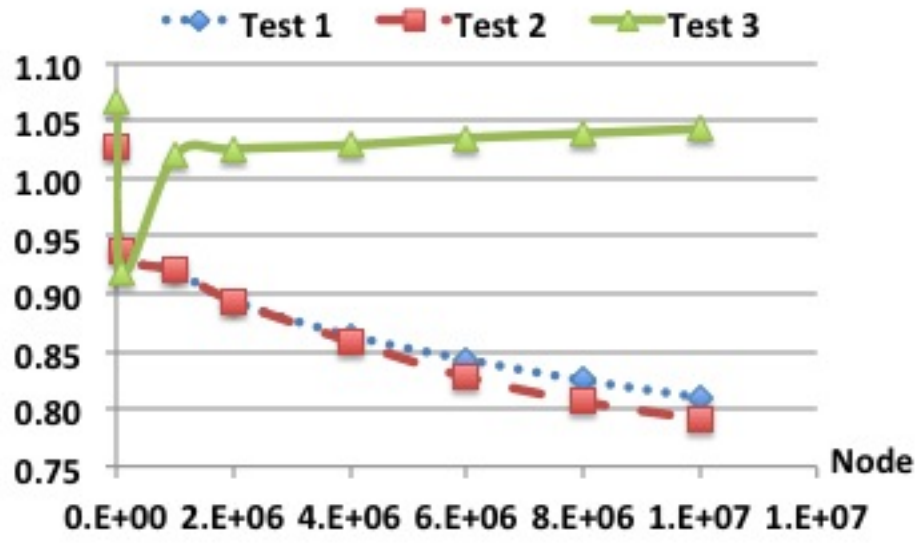


(a) Instance 3

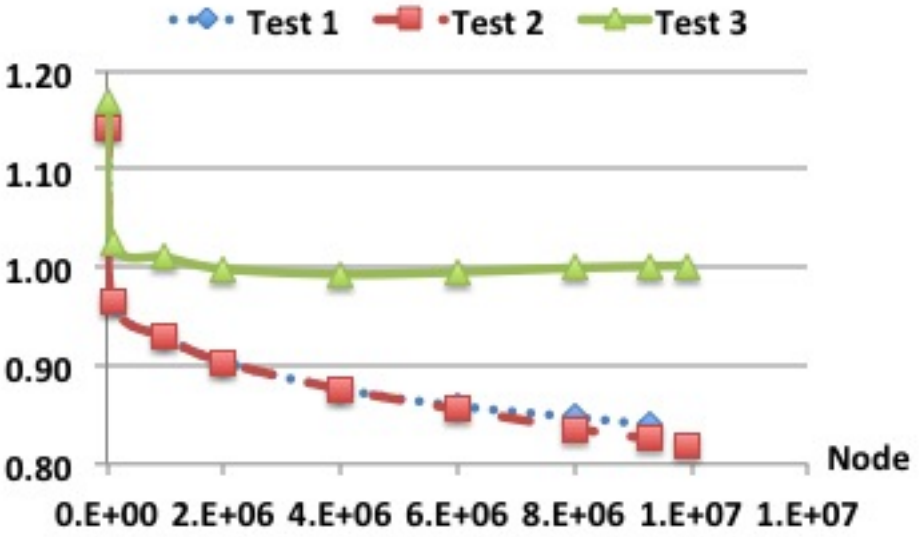


(b) Instance 4

Figure 31: # Remaining nodes in B&B tree compared with CPLEX

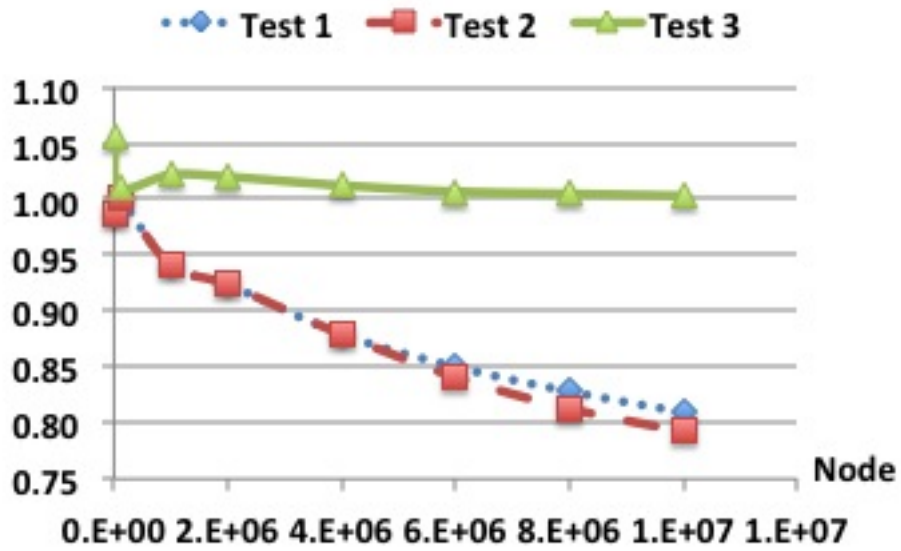


(a) Instance 5

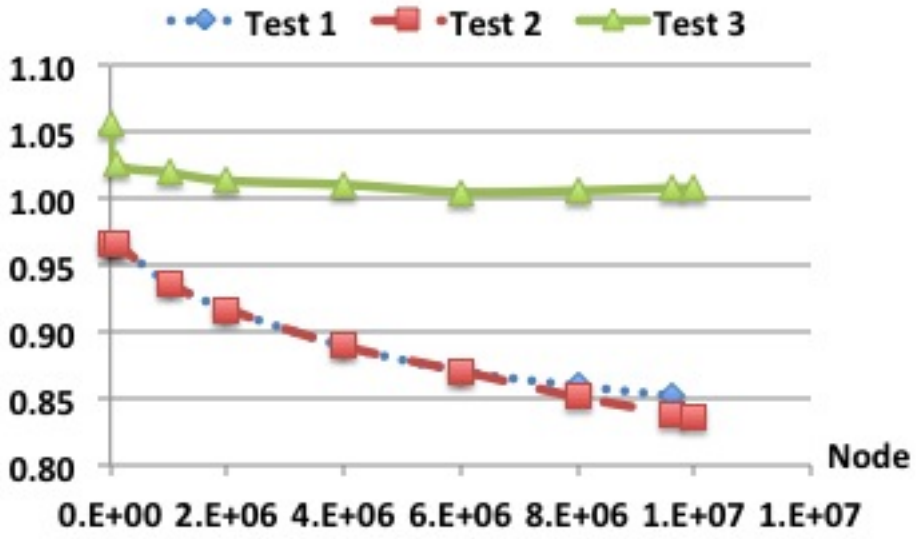


(b) Instance 6

Figure 32: # Remaining nodes in B&B tree compared with CPLEX

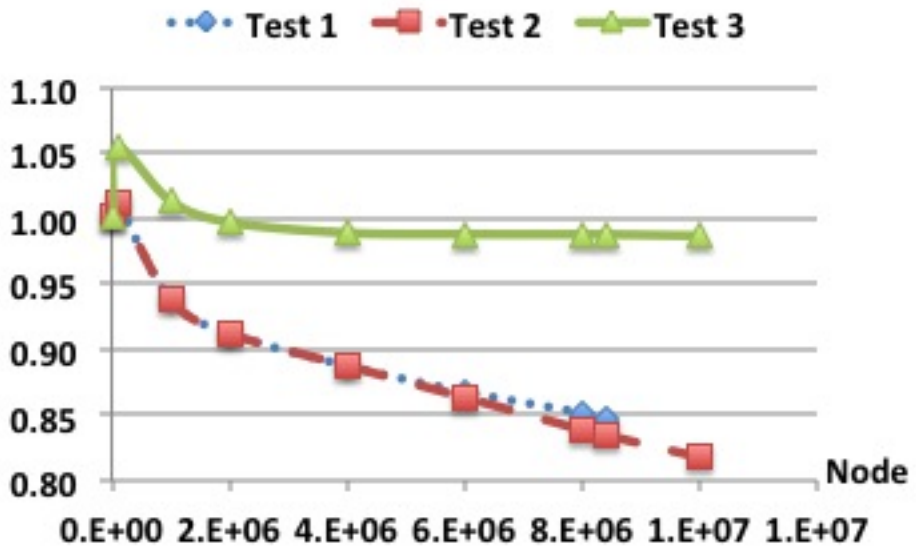


(a) Instance 7

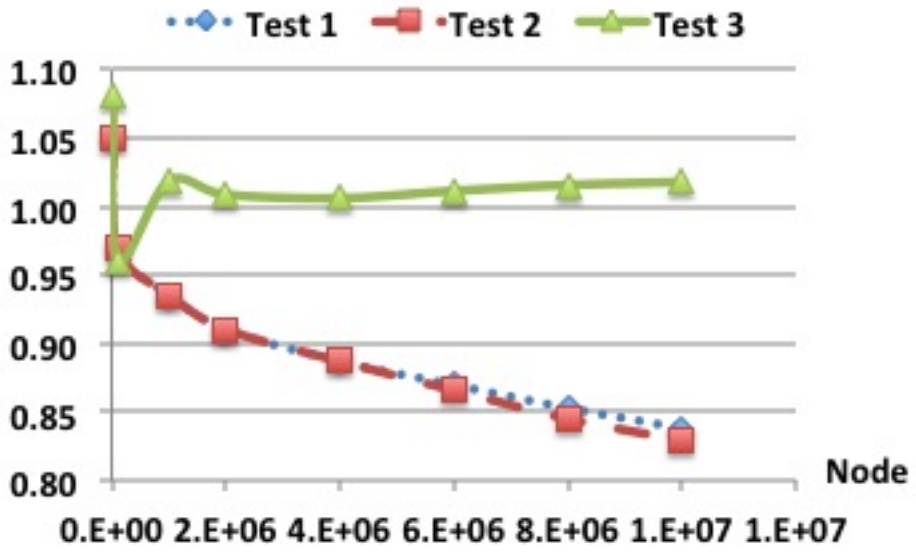


(b) Instance 8

Figure 33: # Remaining nodes in B&B tree compared with CPLEX



(a) Instance 9



(b) Instance 10

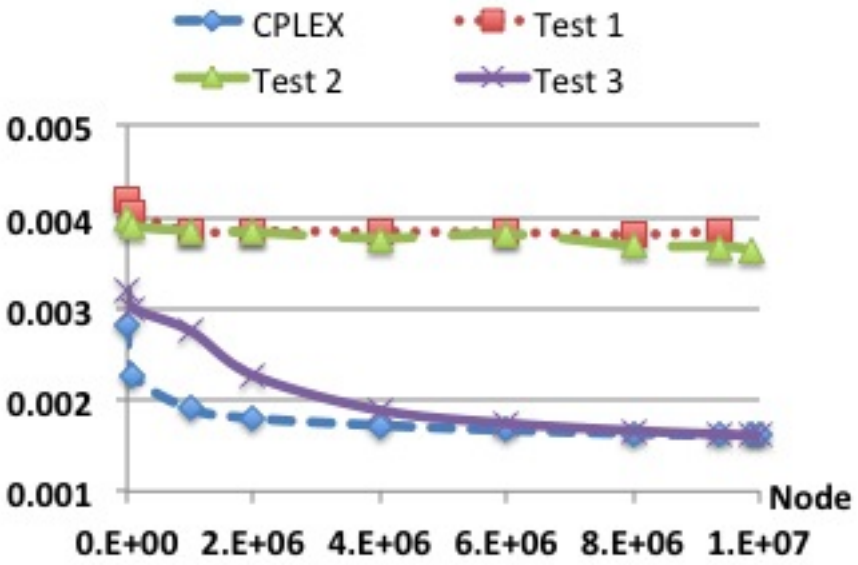
Figure 34: # Remaining nodes in B&B tree compared with CPLEX

Figures 20 - 24 show that the lower bounds can be improved using the dual heuristic algorithm for processing the same number of nodes in the B&B tree. In particular, the improvements on the lower bounds are greater for Tests 1 and 2, where special nodes have smaller relative gaps. For Test 3, the improvements are usually observed in the earlier stages of B&B, since special nodes are those with large relative gaps that appear in the early stages. When the percentage of special nodes decreases as the relative gap falls, the improvement on lower bounds also decreases. We also observe that over 99% of the special nodes are pruned by solving the relaxations in Tests 1 and 2 for all instances. However, nodes are rarely pruned in Test 3. Although the nodes pruned in Tests 1 and 2 are the ones with small relative gaps, the lower bounds can still be improved. The pruning of the special nodes helps to maintain a smaller B&B tree, where most of the unpruned nodes have larger relative gaps. Therefore, the B&B algorithm can focus on dealing with the "hard" nodes.

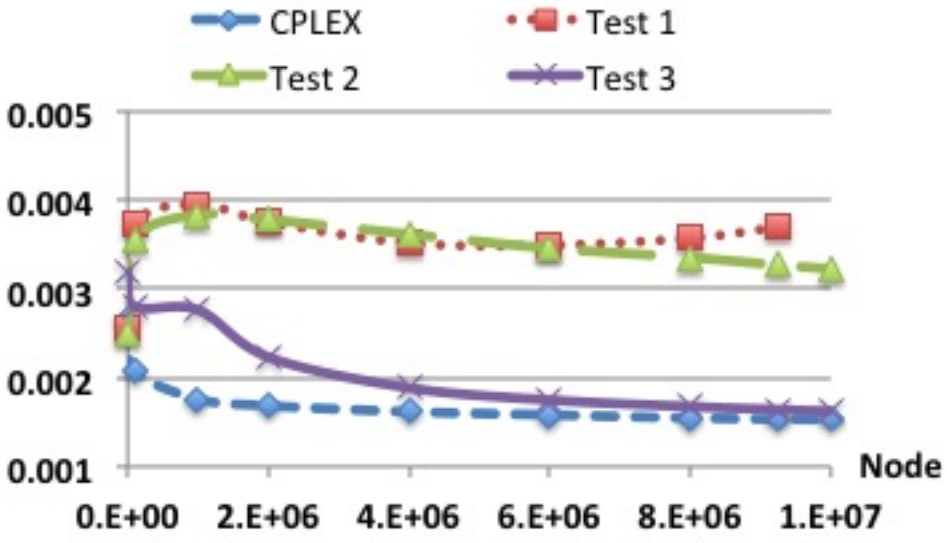
Figures 25 - 29 show the relative gaps in all tests. Since the relative gaps are determined by both lower bounds and upper bounds, the advantage of the dual heuristic algorithm is not significant. However, Tests 1 and 2 show better relative gaps than Test 3 in general.

In Figures 30 - 34, we show the ratio between the number of remaining nodes in the B&B tree for the dual heuristic algorithms and CPLEX. Note that for Tests 1 and 2, the number of remaining nodes is greatly reduced, that is, the size of the B&B tree is reduced. This is because a large percent of special nodes are pruned using the lower bounds obtained from the relaxations. However, for Test 3, since the special nodes have large relative gaps and are almost never pruned, the size of the B&B tree is not reduced much.

Figures 35 - 39 show the average time spent on each node in the B&B tree for all tests. The average time for Tests 1 and 2 is much larger than CPLEX, since the percentage of special nodes is very high, the time spent on solving the relaxations accumulates. For Test 3, the average time gradually decreases, since a smaller number of relaxations are solved. Thus, although focusing on special nodes with small relative gaps helps to reduce the size of the B&B tree and improve the lower bounds, the time used for solving the relaxations overwhelms the time saved for CPLEX to handle the special nodes. However, focusing on nodes with large relative gaps helps to improve the lower bounds of "hard" nodes, which

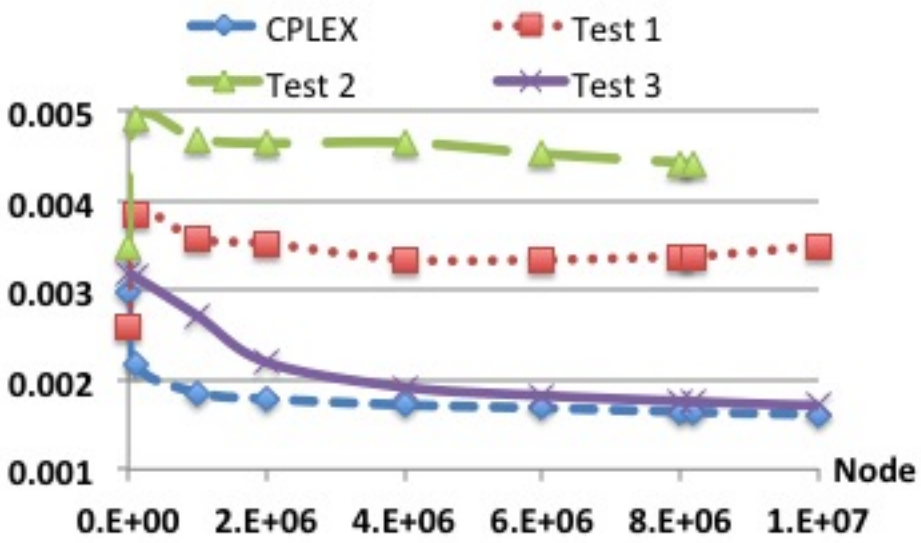


(a) Instance1

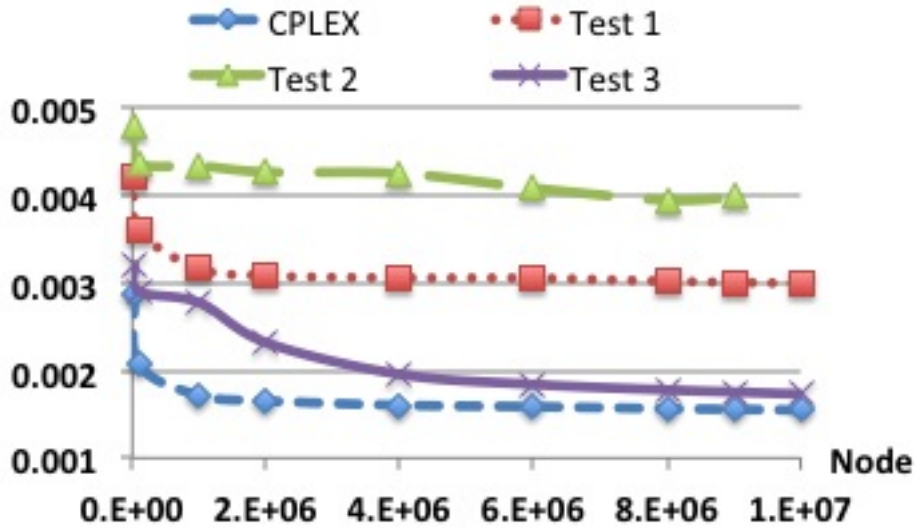


(b) Instance2

Figure 35: Average time spent on each node in seconds

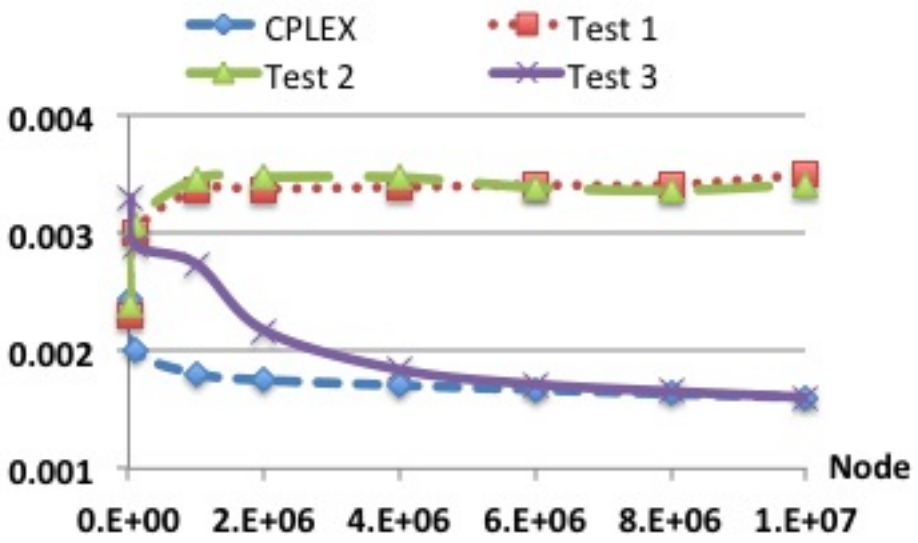


(a) Instance3

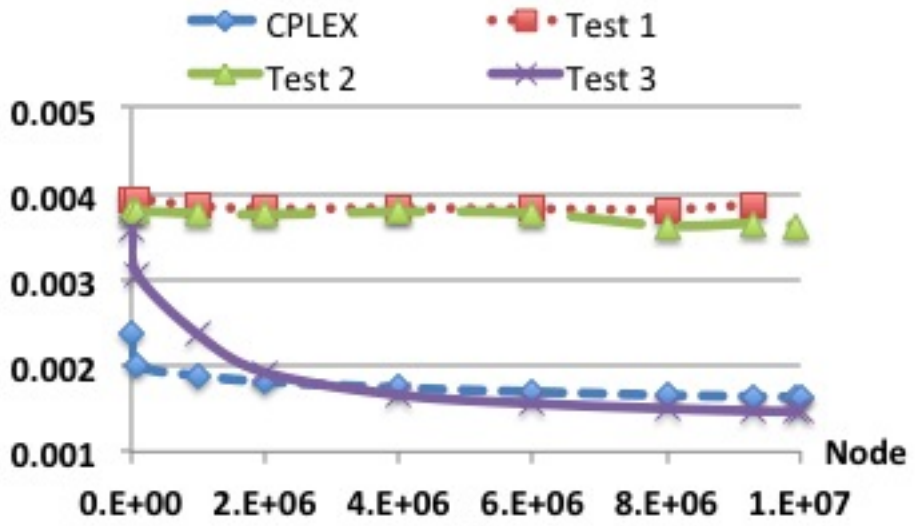


(b) Instance4

Figure 36: Average time spent on each node in seconds

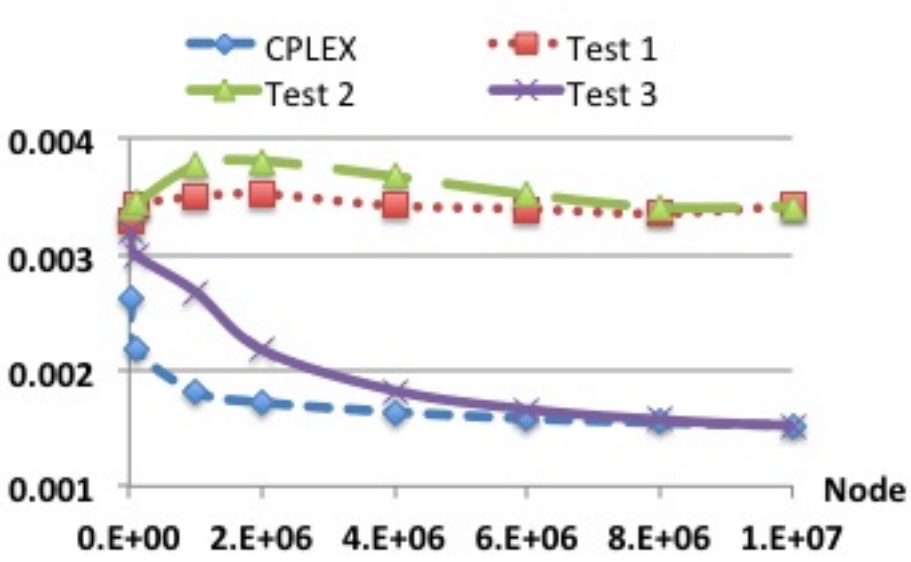


(a) Instance5

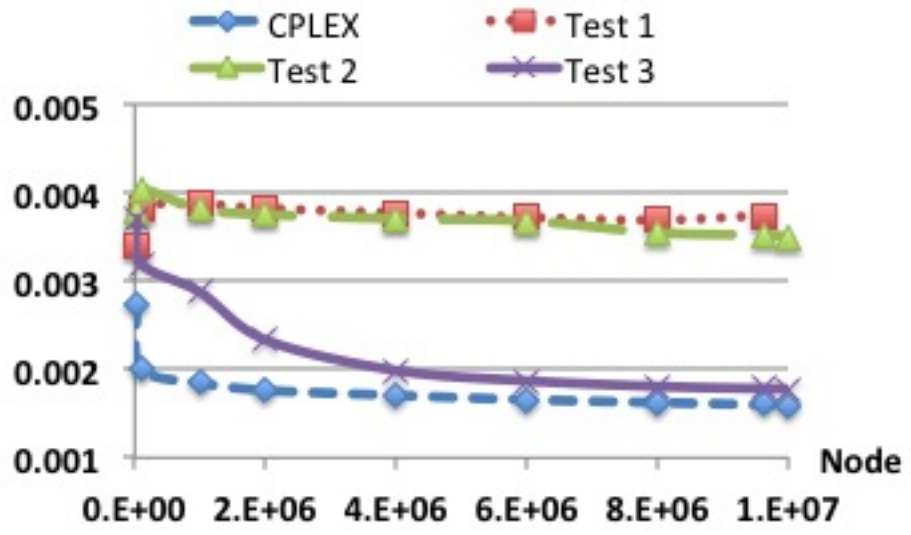


(b) Instance6

Figure 37: Average time spent on each node in seconds

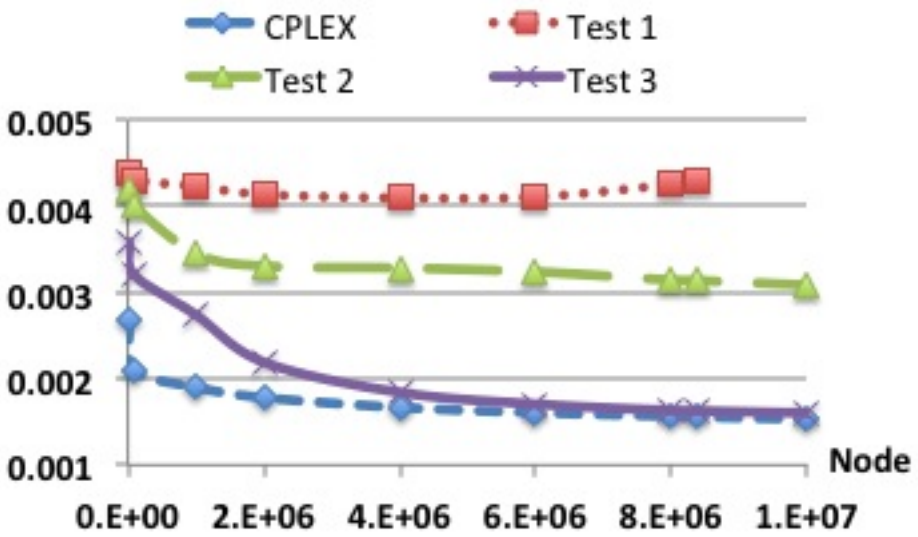


(a) Instance7

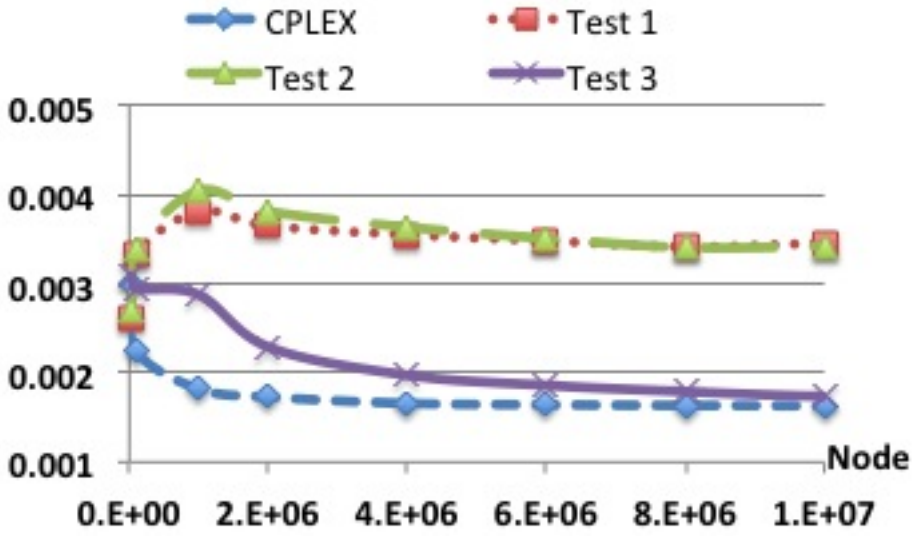


(b) Instance8

Figure 38: Average time spent on each node in seconds



(a) Instance9



(b) Instance10

Figure 39: Average time spent on each node in seconds

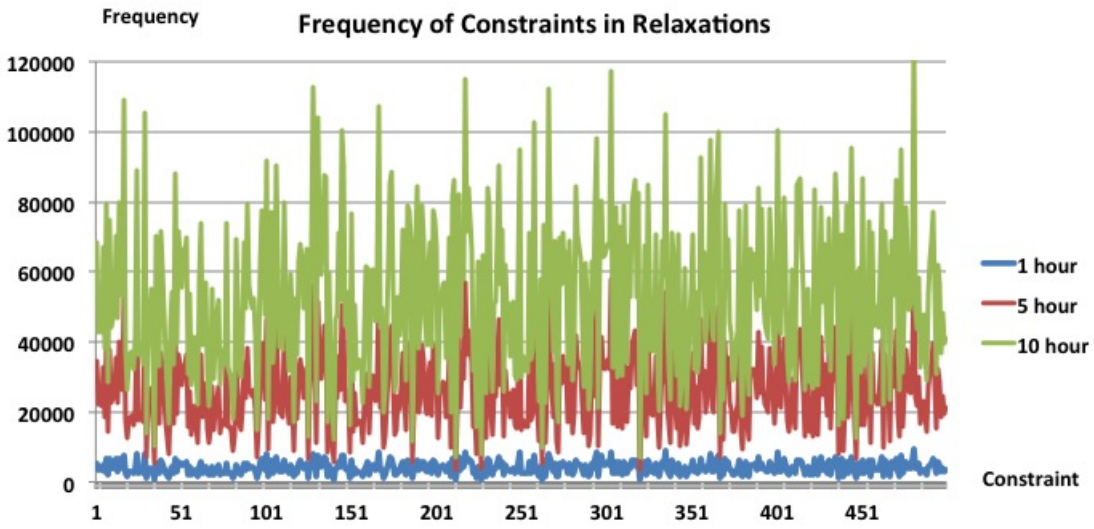


Figure 40: Frequency of constraints in relaxations

could lead to a large subtree in the B&B algorithm, thus the efficiency of Test 3 is better then Tests 1 and 2.

Finally, to understand which constraints are used in the relaxations, we show the number of times that each constraint is used as an active constraint in the relaxations. Since the results show similar trend, we use instance 2 for demonstration. Figure 40 shows the frequency of constraints in Test 2 at time 1 hour, 5 hours, and 10 hours.

The results indicate that at the beginning of the B&B algorithm, the constraints are chosen as active constraints with similar frequency. However, as the B&B algorithm proceeds, the frequency of constraints varies more. Specifically, in 10 hours, about 10%(50) constraints were chosen as active constraints in more than 80000 special nodes, about 36% (180) constraints were chosen in more than 60000 special nodes, and about 66% (330) constraints were chosen in more than 40000 special nodes. However, further investigation shows that the constraints used most often are usually not the constraints with the largest dual values in the linear relaxation of the original problems, which implies that the LP information of the original problem is less useful for the indication of importance of the constraints in later stages of the B&B algorithm.

Our results indicate that the dual heuristic algorithm can be effective in improving

lower bounds for the B&B algorithm for properly chosen parameters. As more relaxations are solved, better lower bounds can be obtained. However, the efficiency of the algorithm decreases when a large number of relaxations is solved. In the next section, we study modifying parameters that control the dual heuristic algorithm to balance its time efficiency and the effectiveness in improving lower bounds.

5.1.4 Modifying Parameters

To improve the performance of the dual heuristic algorithm by applying proper parameters, we now consider a dynamic control mechanism that modifies parameters of the algorithm using information collected during the B&B algorithm.

One factor that impacts the efficiency of the algorithm is the number of special nodes. Too many special nodes may lead to inefficiency in computing, while too few special nodes may not be sufficient to improve lower bounds. Since solving relaxations to reduce the size of the B&B tree and to improve lower bounds is more effective in the earlier stages of B&B, we consider gradually decreasing the percentage of special nodes in B&B to reduce the time spent on the subproblems.

In addition, we recognize that the range of relative gaps of the special nodes is important to the overcall efficiency of the algorithm. Specifically, by choosing special nodes with small relative gaps, the size of the B&B tree can be well controlled, since a large number of special nodes can be pruned. However, choosing special nodes with large relative gaps has a direct impact on the overall lower bounds of the B&B algorithm, and the potential improvement on lower bounds can be larger. We thus consider a strategy that benefits from choosing both types of nodes by modifying *Gap*.

We also consider modifying the level of relaxation for the subproblems, which is controlled by the largest slack value of active constraints, that is, *Slack*. Therefore, for nodes that are close to the top of the B&B tree, we may include more constraints for a tighter relaxation subproblem; Furthermore, for nodes that have larger depth, a smaller number of active constraints should be included since the bounds on variables are also tighter.

For all tests, we fix $\Delta = 0.5$, that is, the cutting plane $c^T x \geq LB$ is added only if the

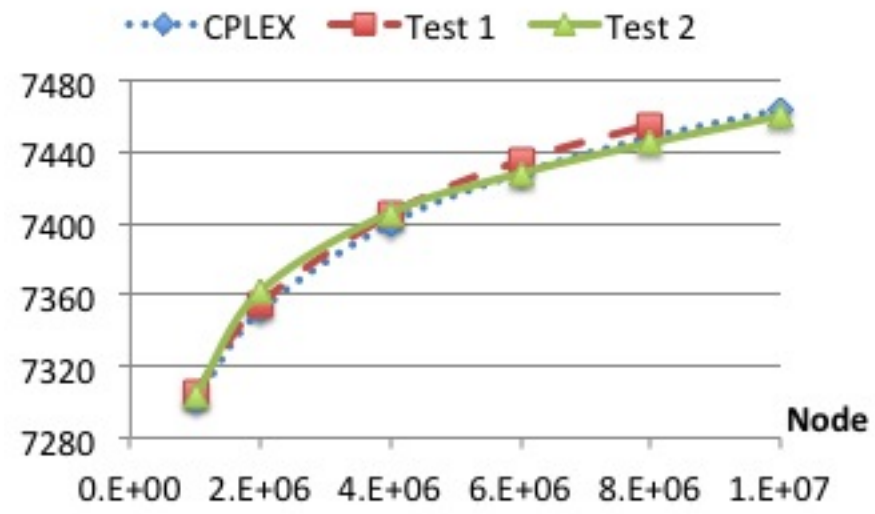
lower bound obtained from the relaxation is at least 0.5 larger than the LP bound of the node. We use two different time limits for solving the relaxations. For nodes with relative gaps less than 2%, we use a time limit of 1 second, since previous results show that such nodes are most likely to be pruned in a short time. Otherwise, we use a time limit of 0.02 seconds, since the nodes are harder to prune. We use a total time limit of 10 hours for all tests.

We conduct three experiments on dynamically controlling *Check_Percent*, *Gap* and *Slack*. For each experiment, we compare the results of CPLEX's default settings with the results obtained from the dual algorithm using two sets of initial parameters. The first set, indicated by Test 1, initializes *Check_Percent* with 20%, and *Gap* with $0 - 2\%$, that is, the special nodes have relative gaps less than 2%. The second set, indicated by Test 2, initializes *Check_Percent* with 2%, and *Gap* with 8%-100%, that is, the special nodes have relative gaps greater than 8%. Note that in Test 1, since a larger number of relaxations are solved and a longer time may be spent on special nodes with smaller relative gaps, the total number of nodes processed within the time limit is smaller.

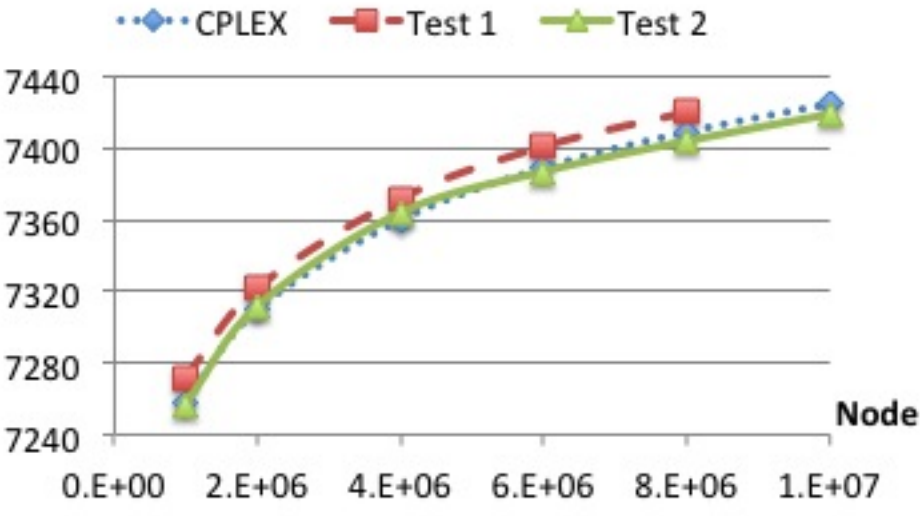
In the first experiment, we examine varying *Slack* to modify the relaxation level. Specifically, for nodes with depth less than 20, we use $Slack = 50$; for nodes with depth between 20 and 30, we use $Slack = 20$; for nodes with depth between 30 and 40, we use $Slack = 10$, and for nodes with depth greater than 40, we use $Slack = 1$. Figures 41 - 45 display the lower bounds obtained from the algorithm and CPLEX for the same number of nodes processed in the B&B tree, while Figures 46 - 50 shows the average time per node for both algorithms.

Note that the average time per node is slightly increased compared with tests in Section 5.1.3, where $Slack = 1$. However, lower bounds obtained from Test 1 are still better than from Test 2 in general. Compared with the ones using a fixed $Slack = 1$ as in previous tests, the lower bounds do not change much with varying *Slack* values. Since the time limit for solving the relaxations is too small, increasing the number of active constraints may not boost the lower bounds significantly in a very short time.

In the next experiment, we adjust *Check_Percent* to modify the largest percentage

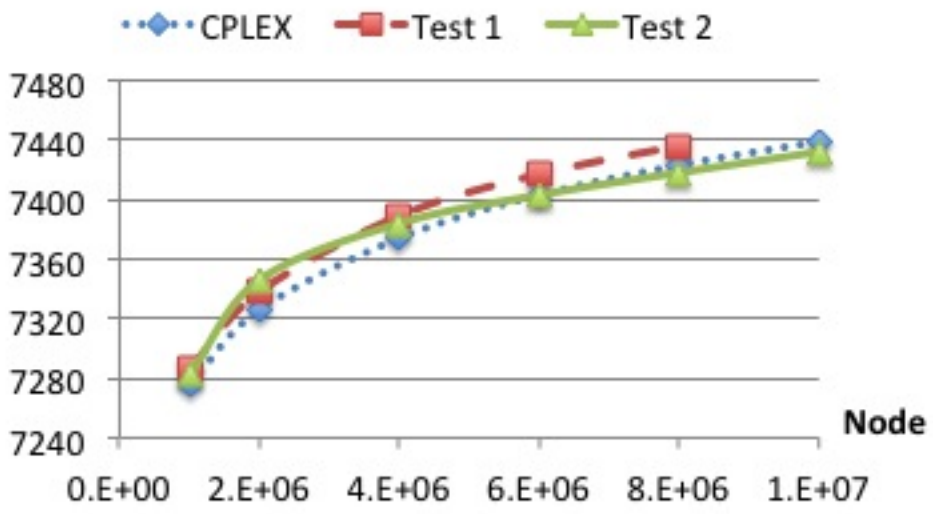


(a) Instance 1

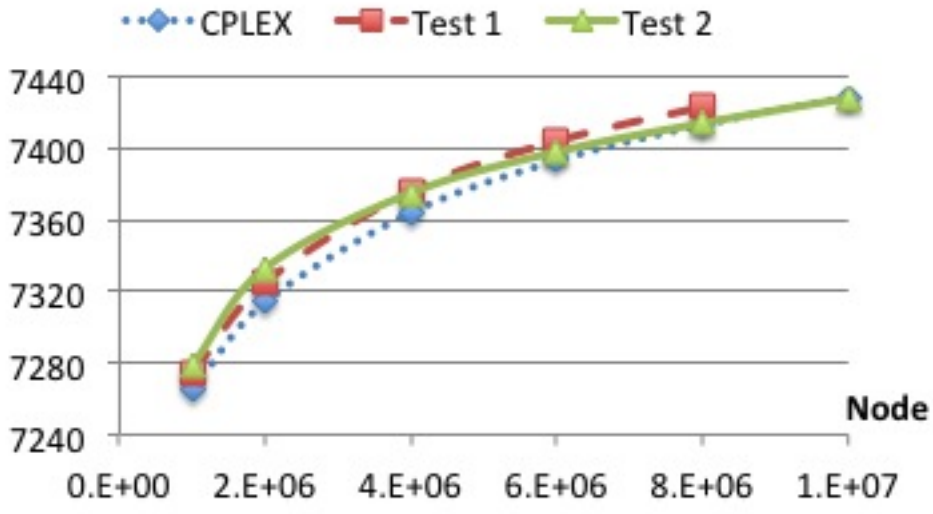


(b) Instance 2

Figure 41: Lower bounds for modifying *Slack*

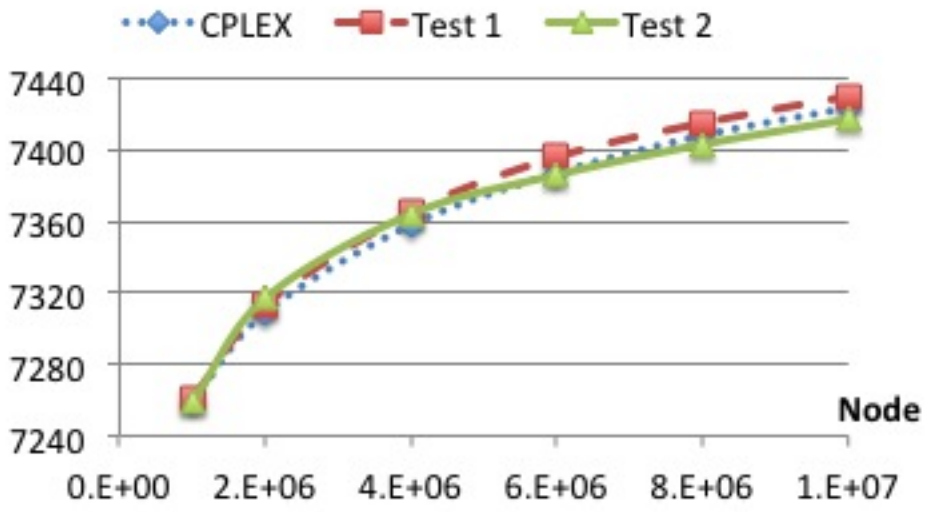


(a) Instance 3

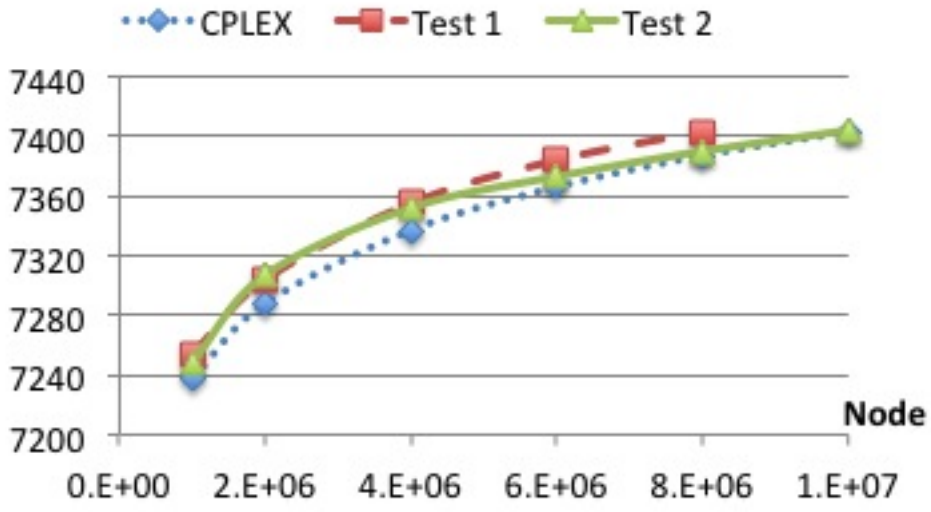


(b) Instance 4

Figure 42: Lower bounds for modifying *Slack*

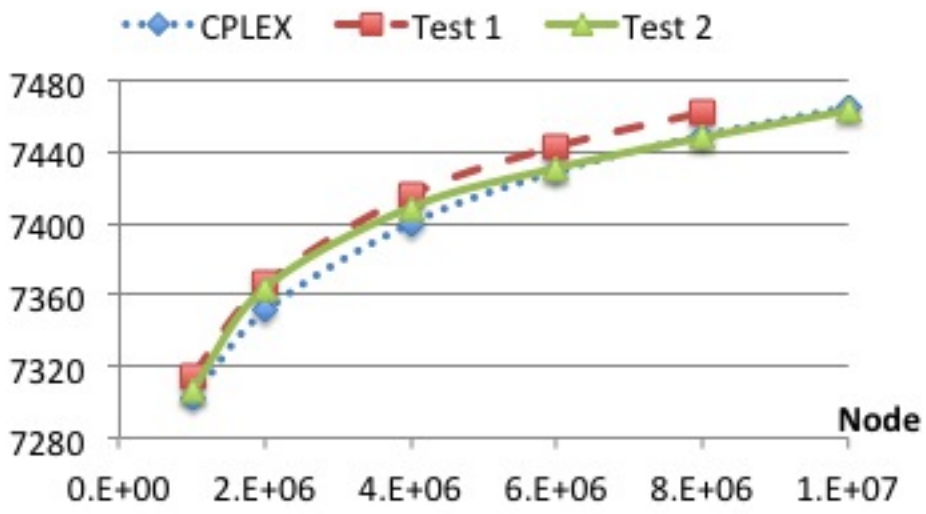


(a) Instance 5

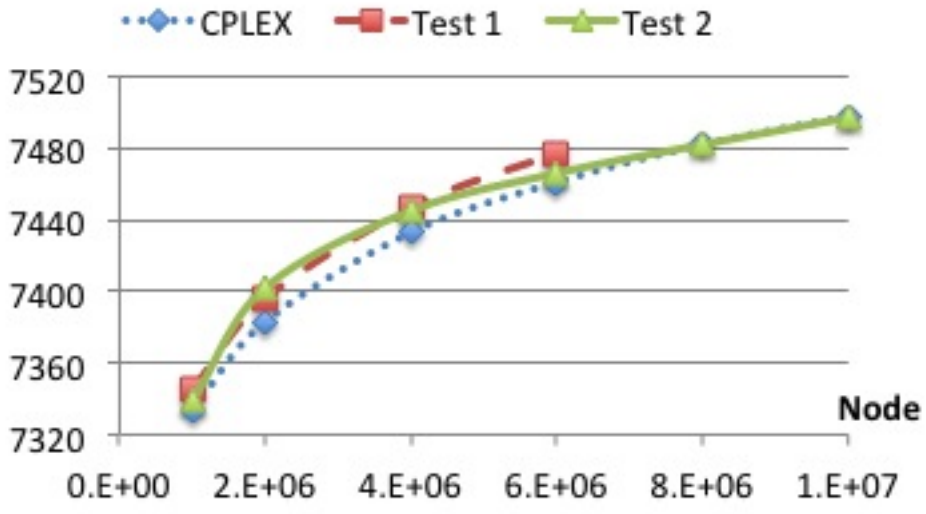


(b) Instance 6

Figure 43: Lower bounds for modifying *Slack*

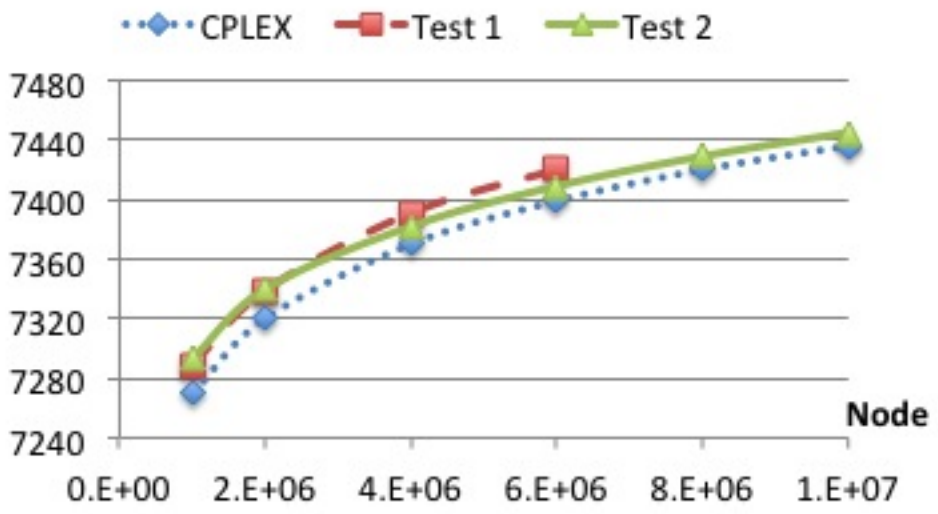


(a) Instance 7

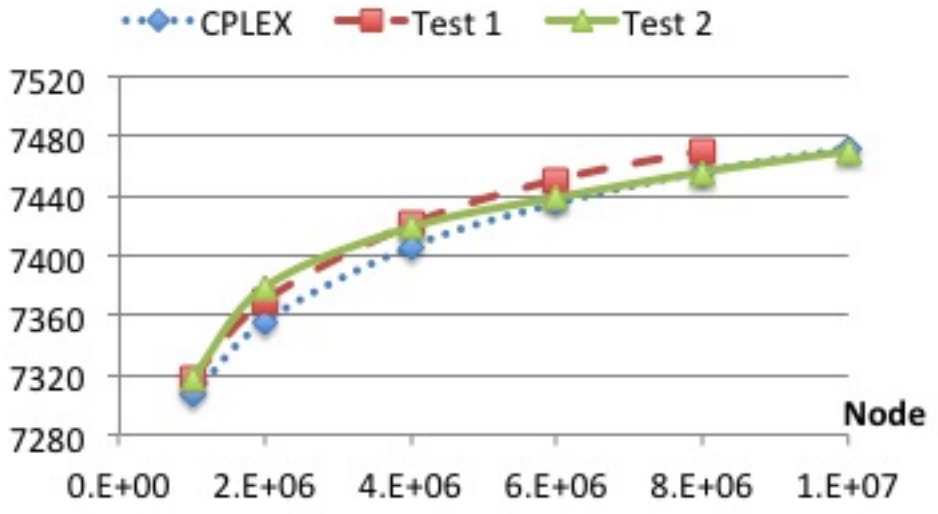


(b) Instance 8

Figure 44: Lower bounds for modifying *Slack*

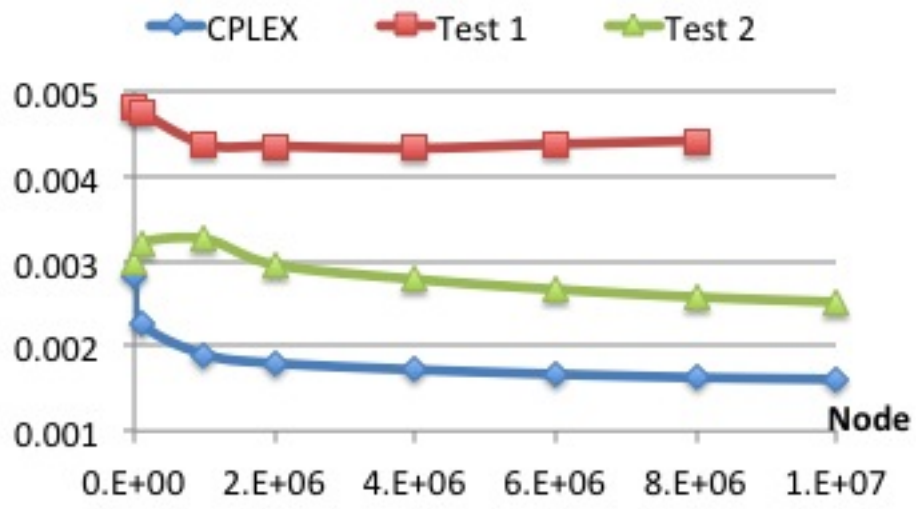


(a) Instance 9

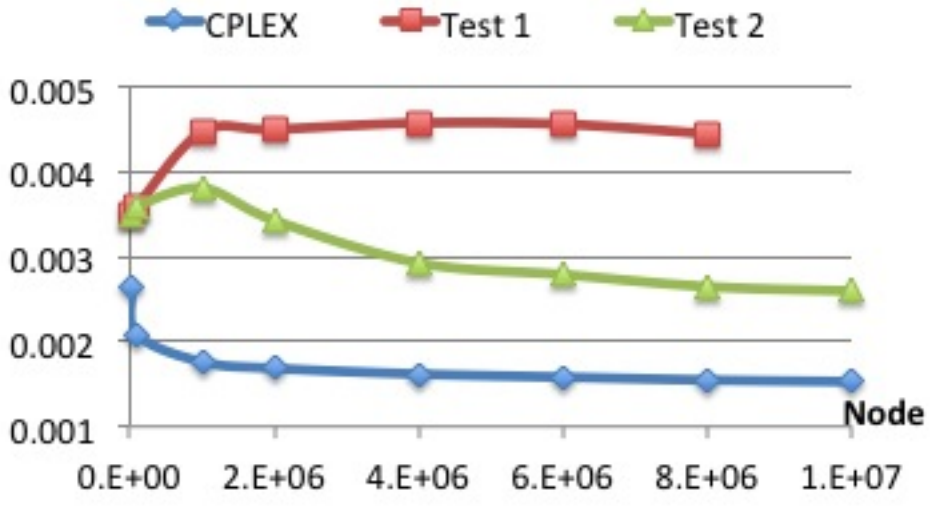


(b) Instance 10

Figure 45: Lower bounds for modifying *Slack*

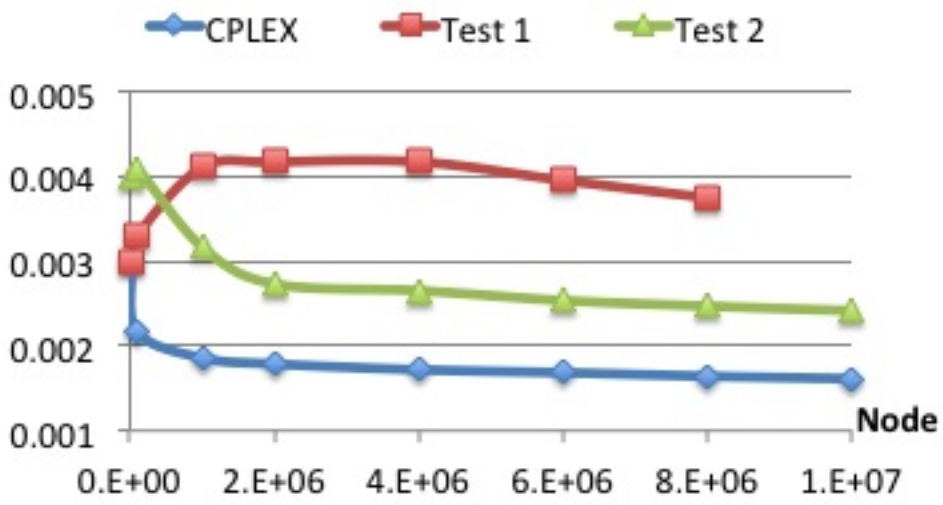


(a) Instance 1

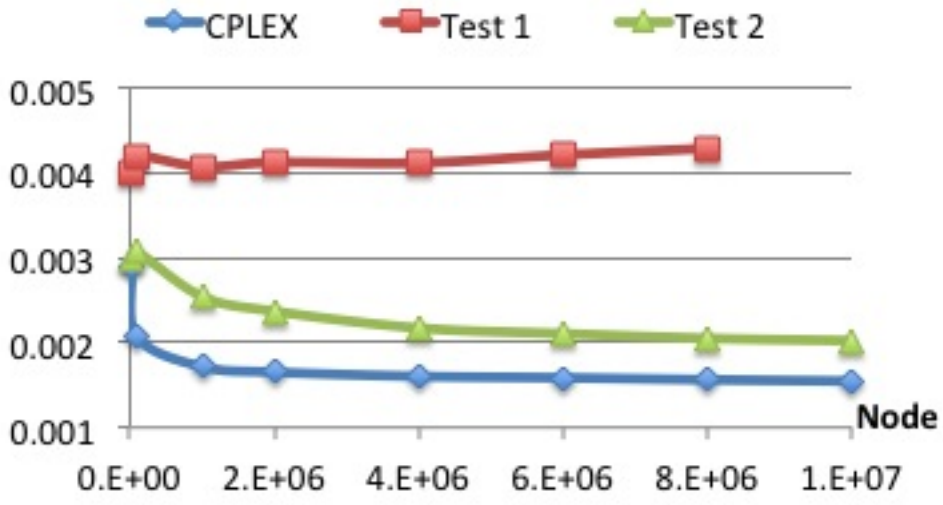


(b) Instance 2

Figure 46: Average time per node for modifying *Slack*

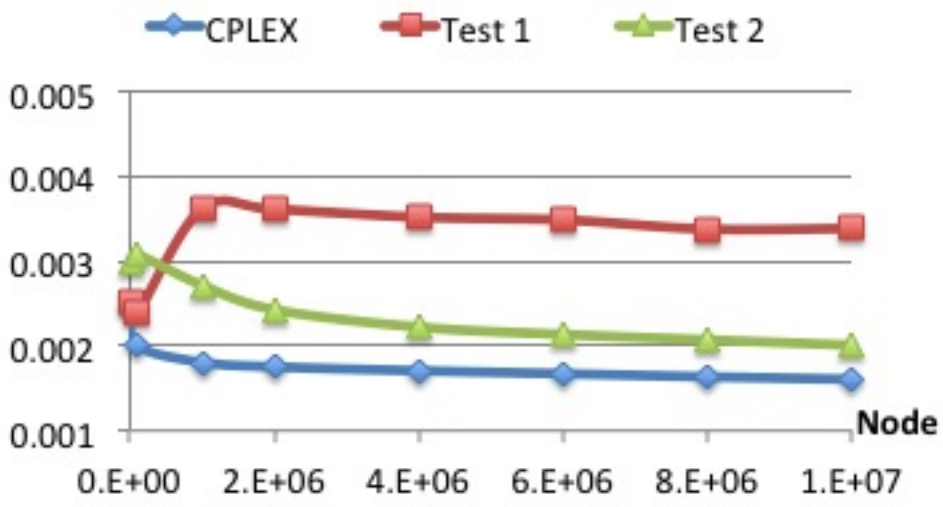


(a) Instance 3

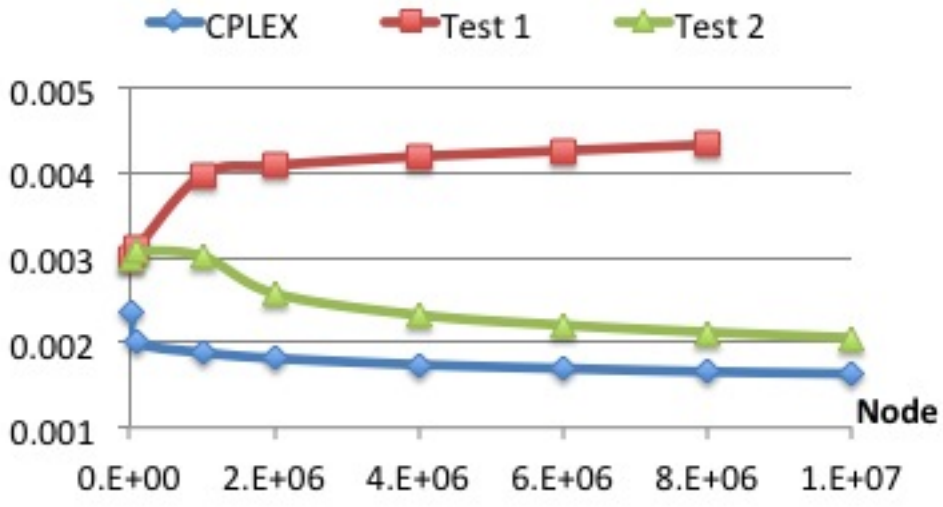


(b) Instance 4

Figure 47: Average time per node for modifying *Slack*

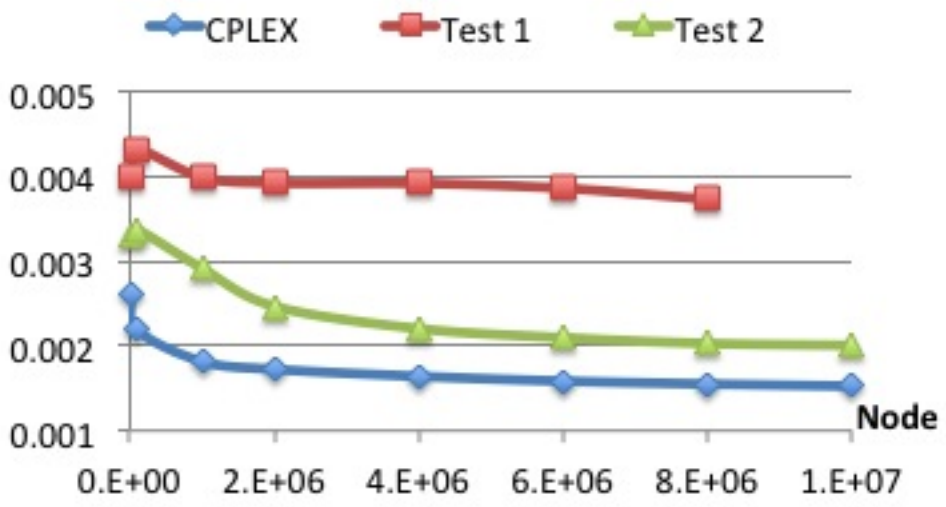


(a) Instance 5

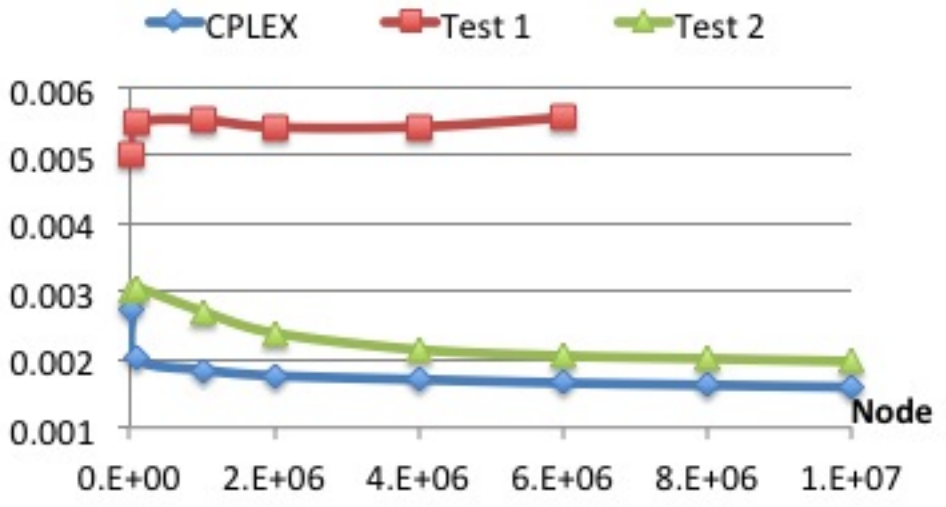


(b) Instance 6

Figure 48: Average time per node for modifying *Slack*

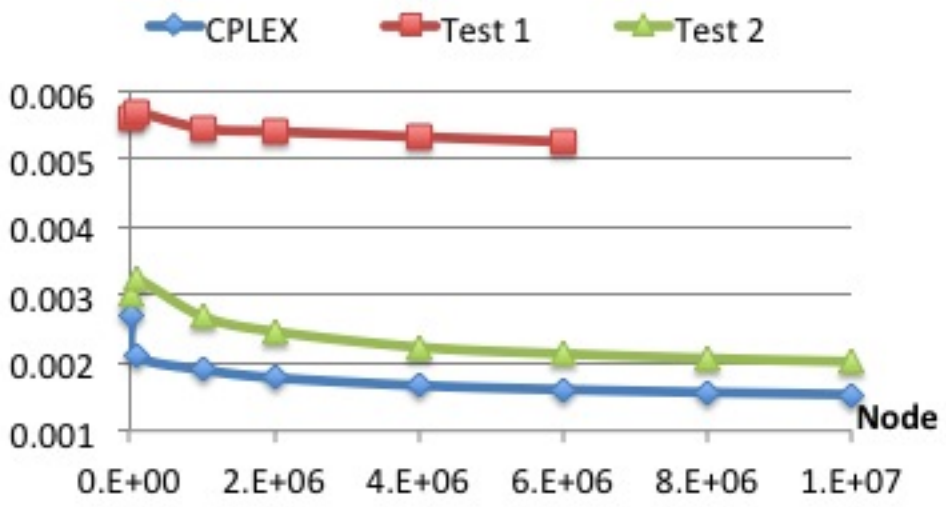


(a) Instance 7

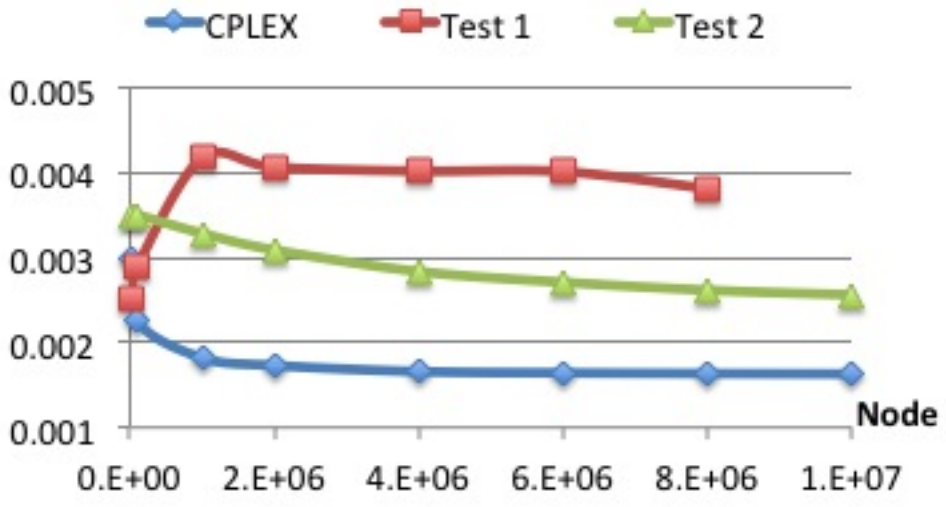


(b) Instance 8

Figure 49: Average time per node for modifying *Slack*



(a) Instance 9



(b) Instance 10

Figure 50: Average time per node for modifying *Slack*

of special nodes over the nodes that have been processed in the B&B tree. We make the modification for every 10,000 nodes processed in the B&B tree. Let IMP be the percentage of special nodes that are pruned, or where lower bounds obtained from the relaxations are larger than the LP relaxation bounds. We reduce the value of $Check\_Percent$ by the minimum of $1 - IMP$ and 10%, That is, if fewer than 90% of the special nodes are pruned or with the lower bounds improved, we decrease the threshold value of the total percentage of special nodes by 10%. By doing this, less time will be spent on solving the ineffective subproblems.

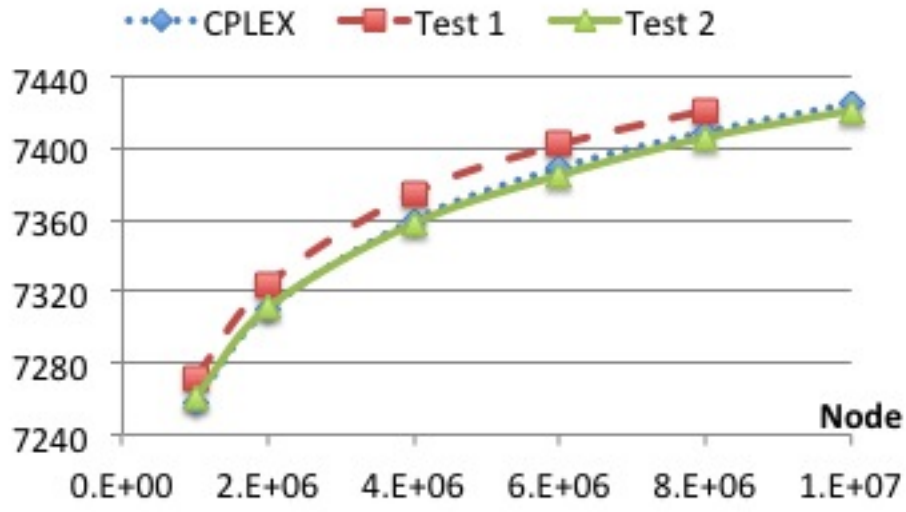
Figures 51 - 55 display the lower bounds obtained. In Figures 56 - 60, we present the actual percentage of special nodes over all processed nodes in the B&B tree, indicated by "Test 1 - TP" and "Test 2 -TP". The values of $Check\_Percent$ are indicated by "Test 1 - TP thresh" and "Test 2 -TP thresh", respectively.

Note that for both tests, the actual percentage of special nodes converges to the threshold value. However, for Test 2 where only nodes with large relative gaps are chosen, the values of $Check\_Percent$ converge to zero rapidly, which indicates that the lower bounds are only improved for a small percentage of special nodes. However, for Test 1, the overall decrease of $Check\_Percent$ is less than 1% for all instances, which implies that for almost all special nodes, the lower bounds are improved by solving the relaxations. In addition, although the percentage of special nodes is decreased in Test 1, the quality of lower bounds is not affected much. On the other hand, for Test 2, the lower bounds are slightly decreased compared with previous tests with fixed parameters, as a smaller number of relaxations are solved.

Finally, we study adjusting the range of relative gaps for special nodes. For Test 1, since the initial relative gaps of special nodes are small, the algorithm tries to choose special nodes with larger relative gaps. For Test 2, since the initial relative gaps of special node are large, the algorithm tries to choose special nodes with smaller relative gaps.

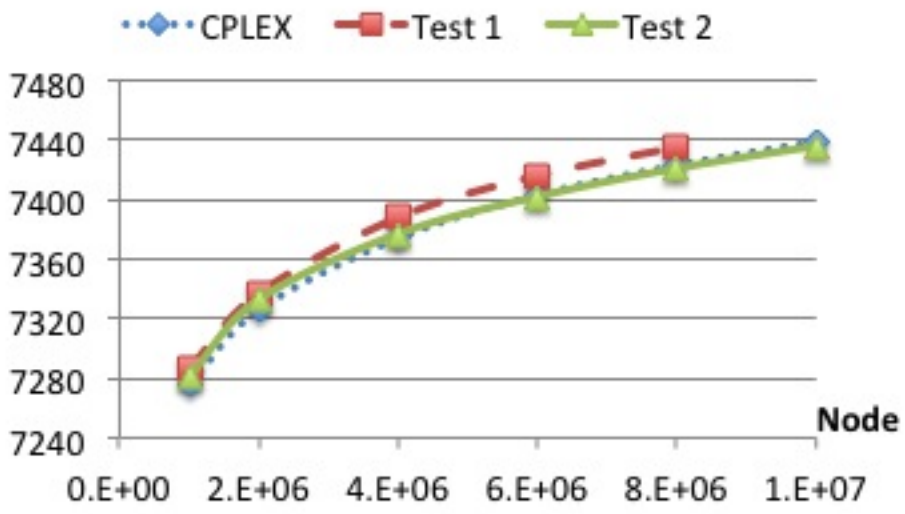
Let *check* be the actual accumulated percentage of special nodes over all processed nodes in the B&B tree, and $Gap = (Lo, Hi)$ be the range of relative gaps for special nodes, thus the relative gap of any special node is between Lo and Hi. We examine the value of *check* for every 10,000 nodes and make the following modification. If the actual percentage of special

(a) Instance 1

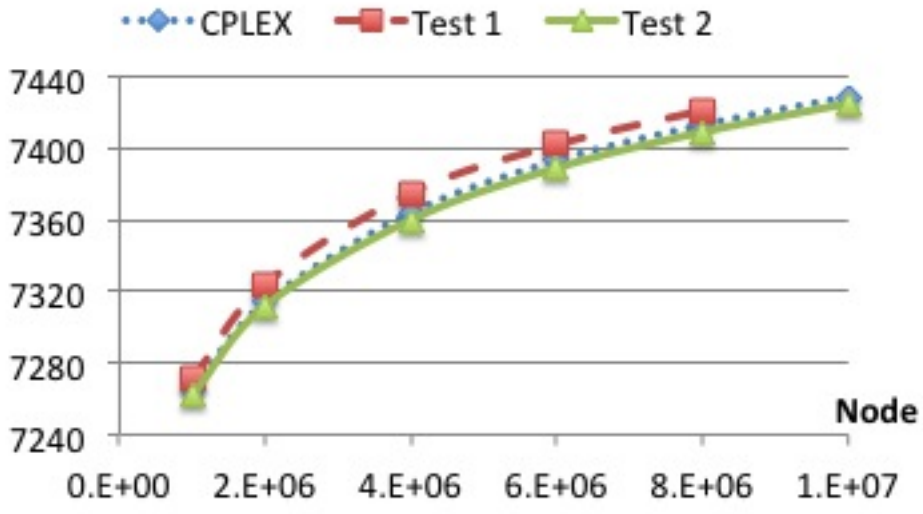


(b) Instance 2

Figure 51: Lower bounds for modifying percentage of special nodes



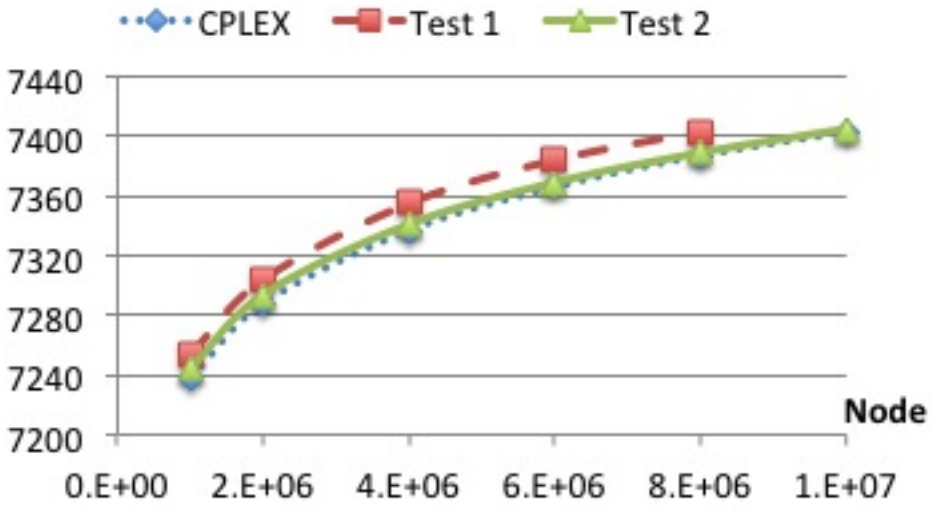
(a) Instance 3



(b) Instance 4

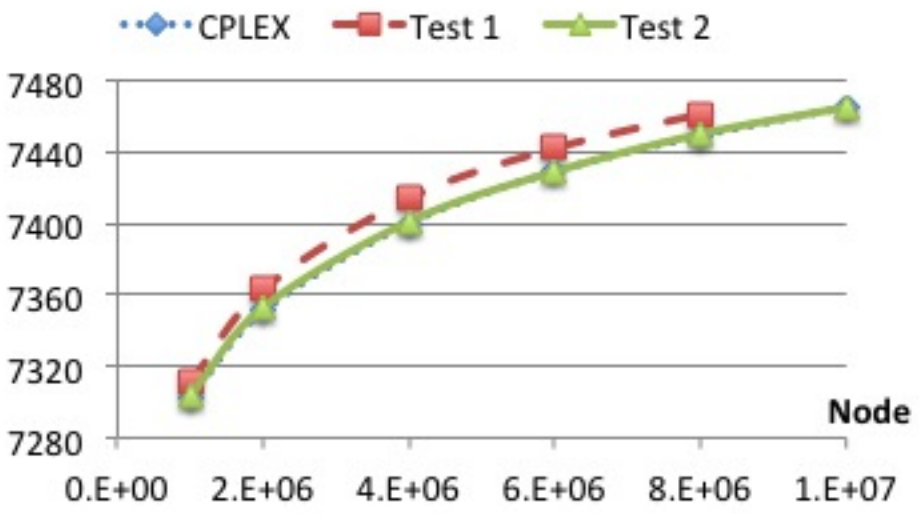
Figure 52: Lower bounds for modifying percentage of special nodes

(a) Instance 5

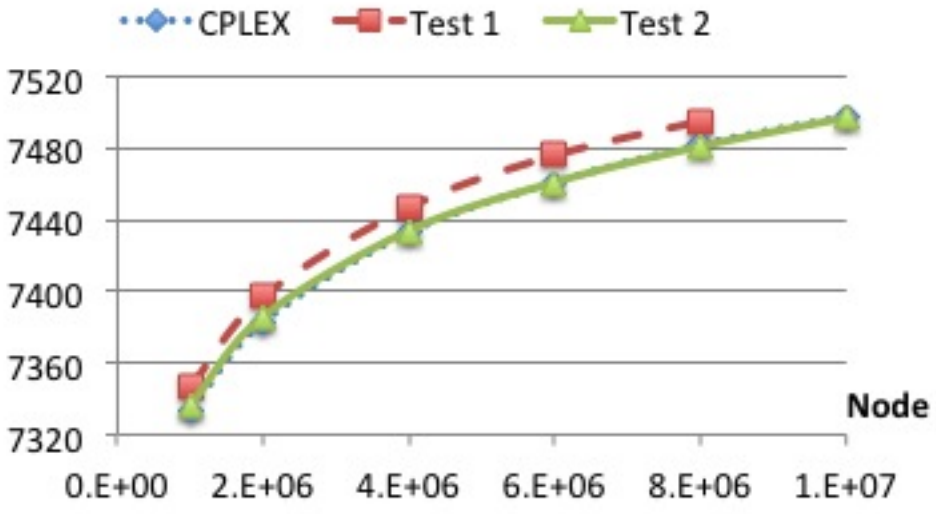


(b) Instance 6

Figure 53: Lower bounds for modifying percentage of special nodes

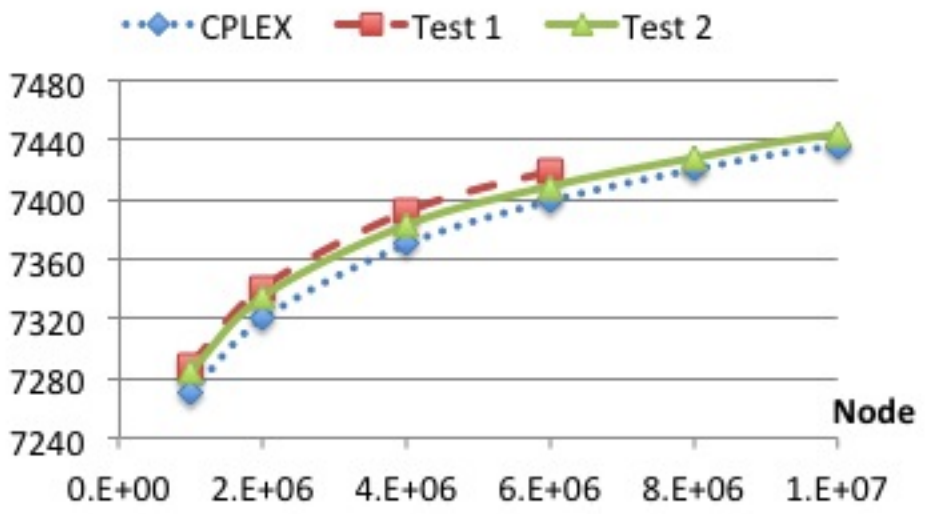


(a) Instance 7

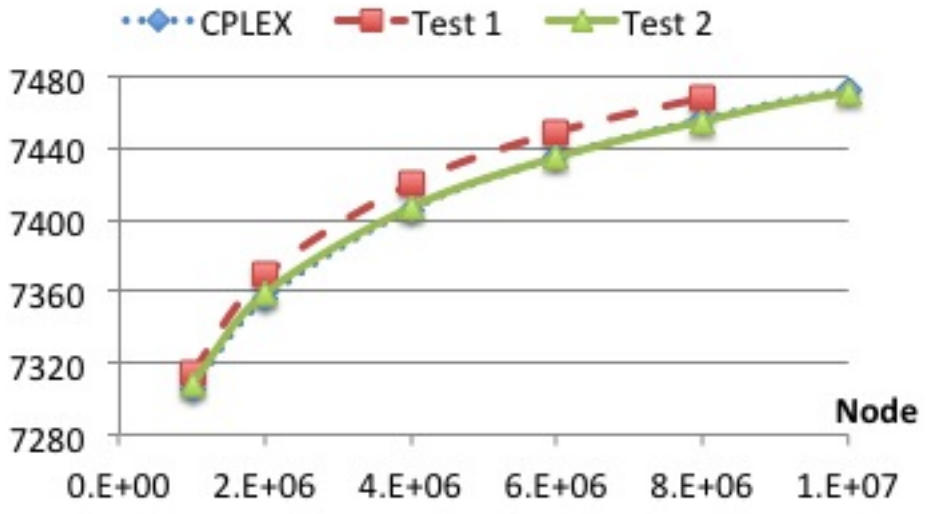


(b) Instance 8

Figure 54: Lower bounds for modifying percentage of special nodes

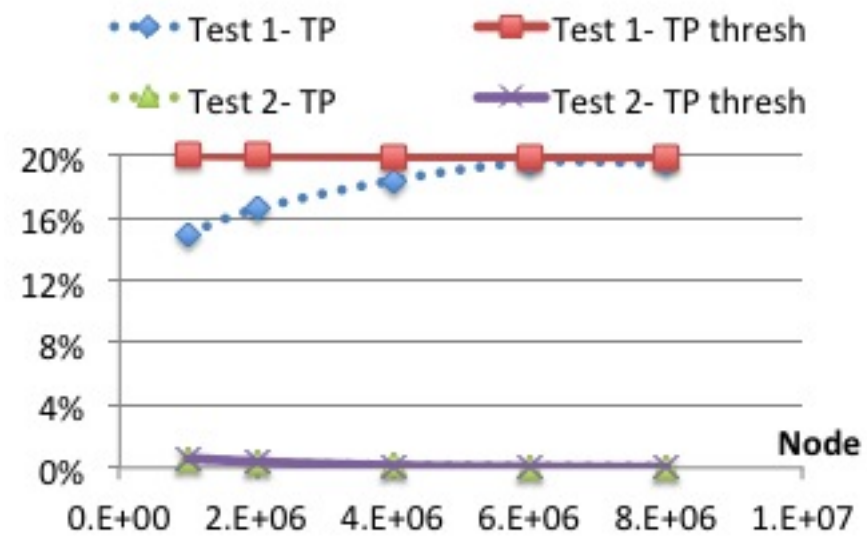


(a) Instance 9

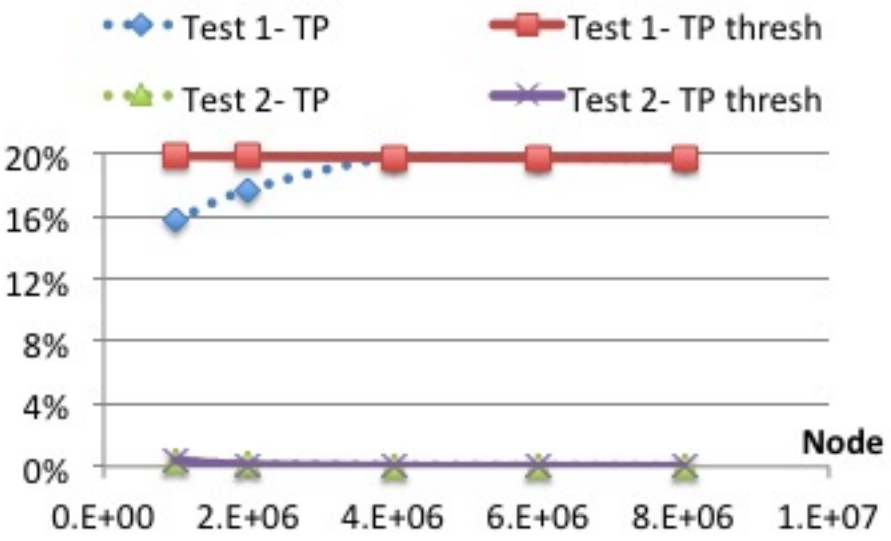


(b) Instance 10

Figure 55: Lower bounds for modifying percentage of special nodes

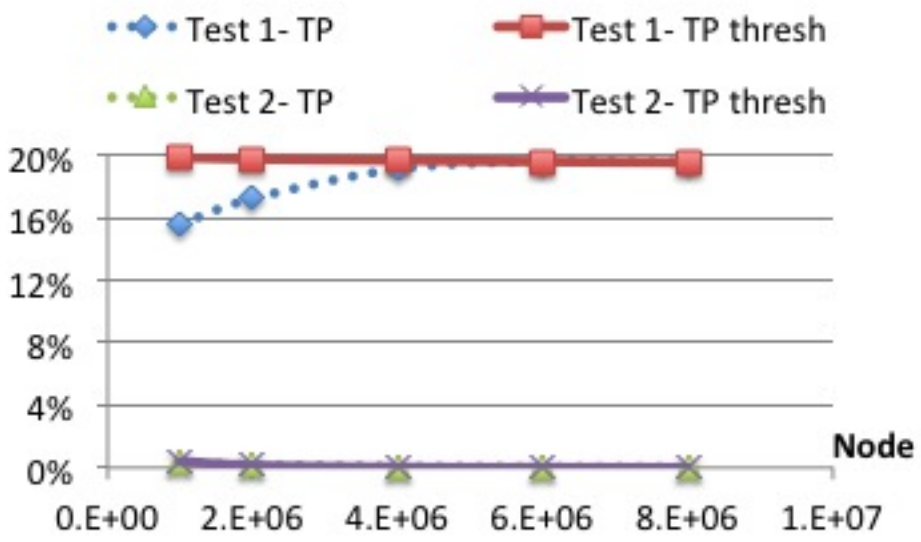


(a) Instance 1

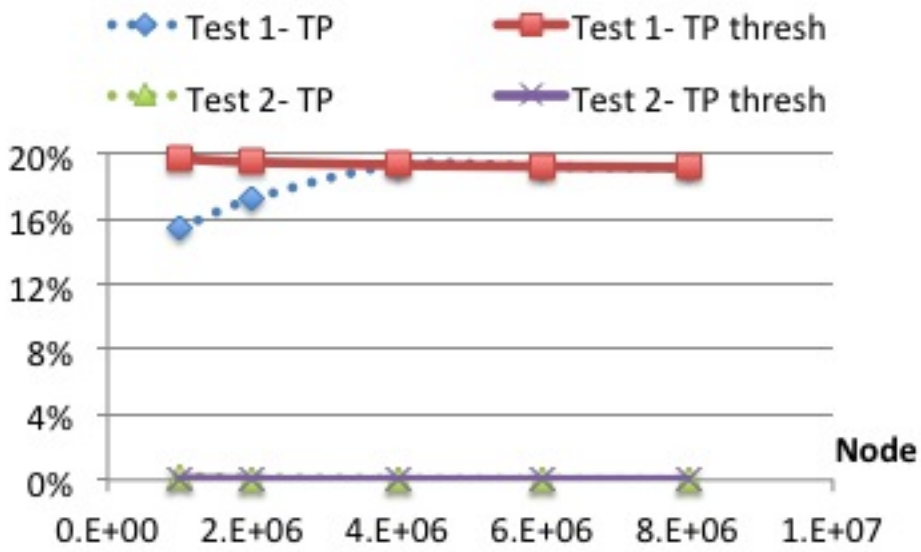


(b) Instance 2

Figure 56: *Check_Percent* for modifying percentage of special nodes

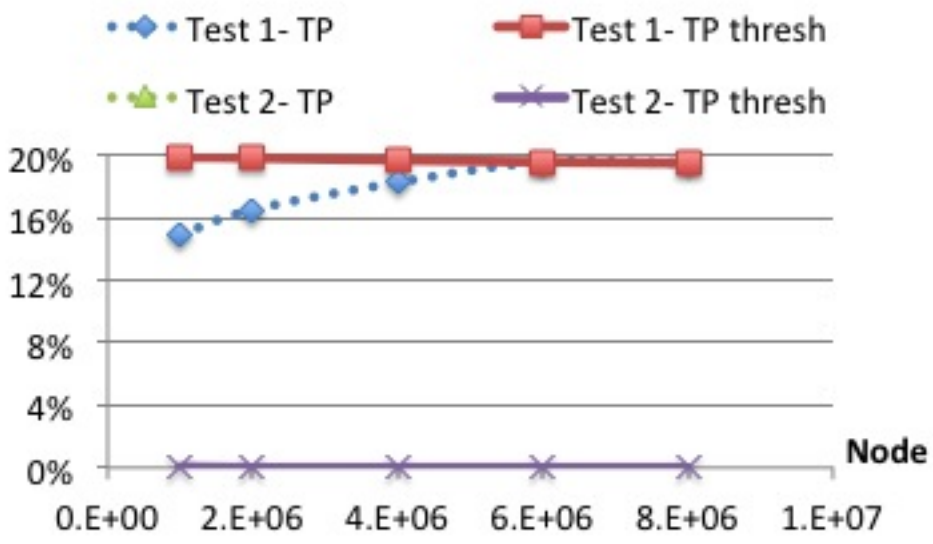


(a) Instance 3

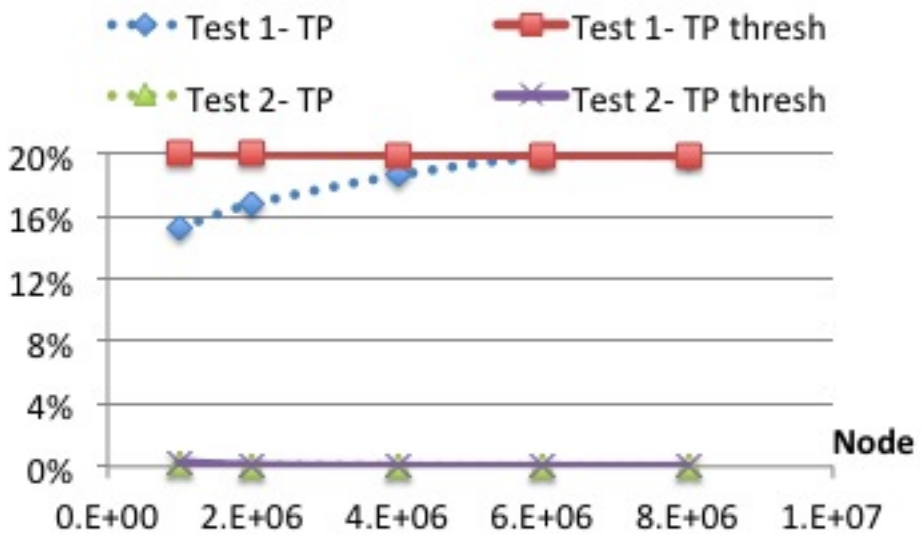


(b) Instance 4

Figure 57: *Check_Percent* for modifying percentage of special nodes

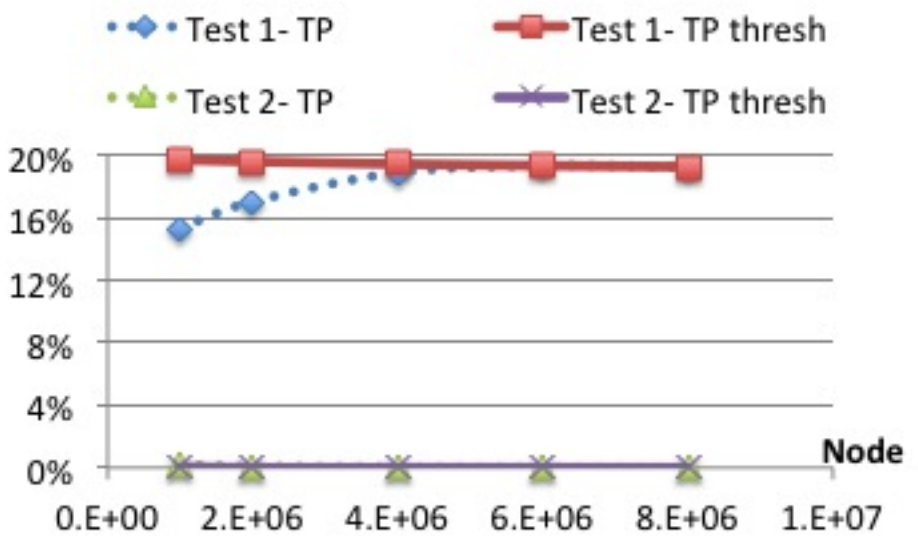


(a) Instance 5

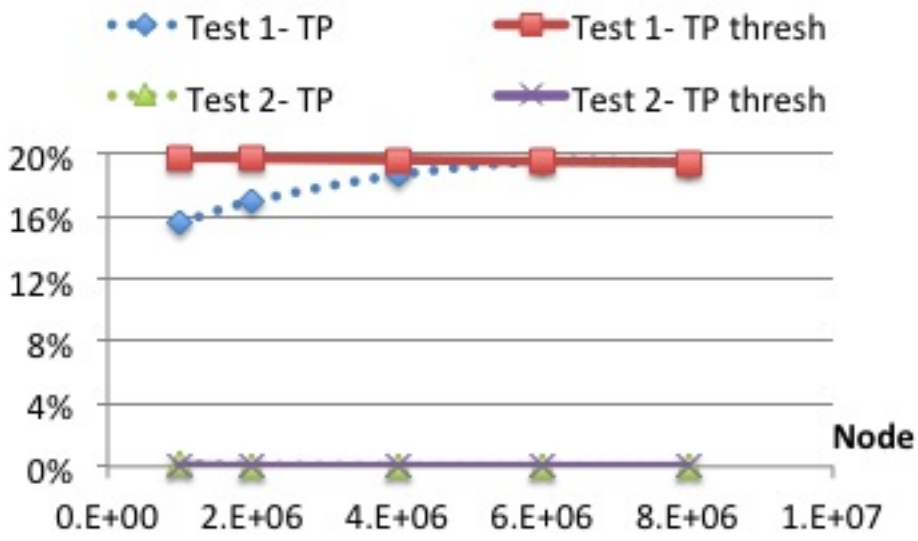


(b) Instance 6

Figure 58: *Check_Percent* for modifying percentage of special nodes

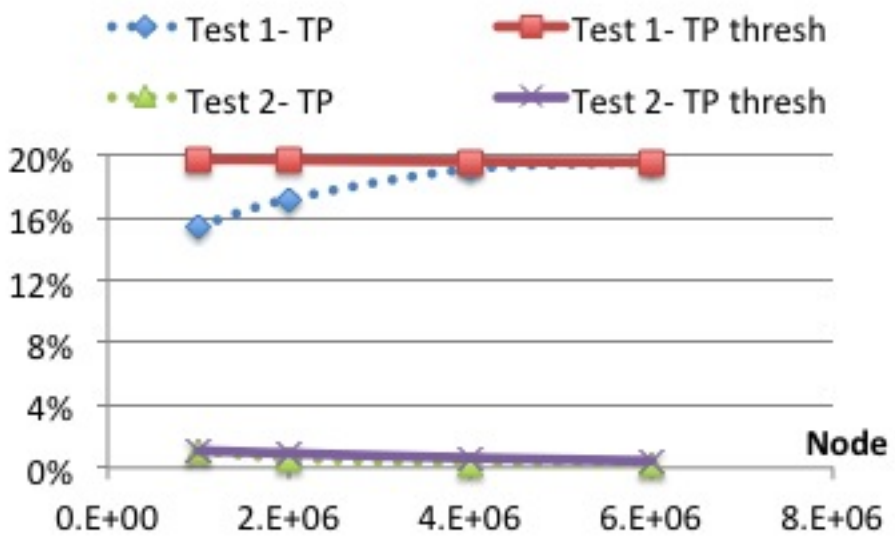


(a) Instance 7

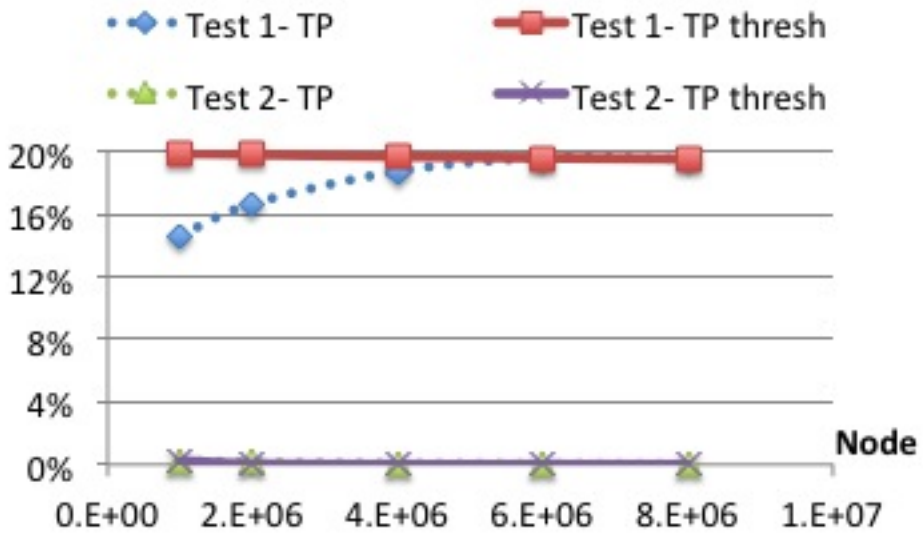


(b) Instance 8

Figure 59: *Check_Percent* for modifying percentage of special nodes



(a) Instance 9



(b) Instance 10

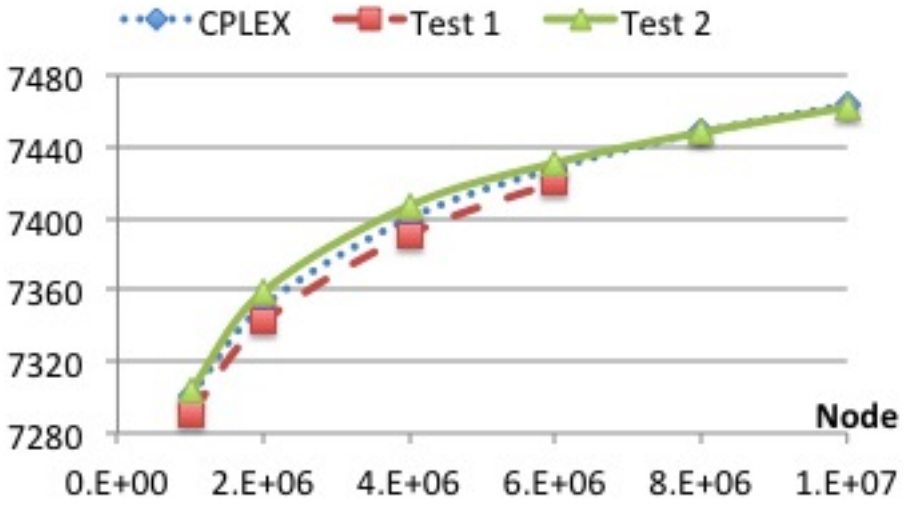
Figure 60: *Check_Percent* for modifying percentage of special nodes

nodes over all processed nodes in the B&B tree is less than 80% of $Check\_Percent$, that is, a relatively small number of special nodes are chosen, we decrease Lo by 0.5%, so that more nodes with smaller relative gaps can be chosen. Similarly, if the total percentage of special nodes that are pruned is greater than 80%, indicating that too much time is spent on special nodes with small relative gaps, we increase Hi by 0.5%, so that more special nodes with larger relative gaps can be considered.

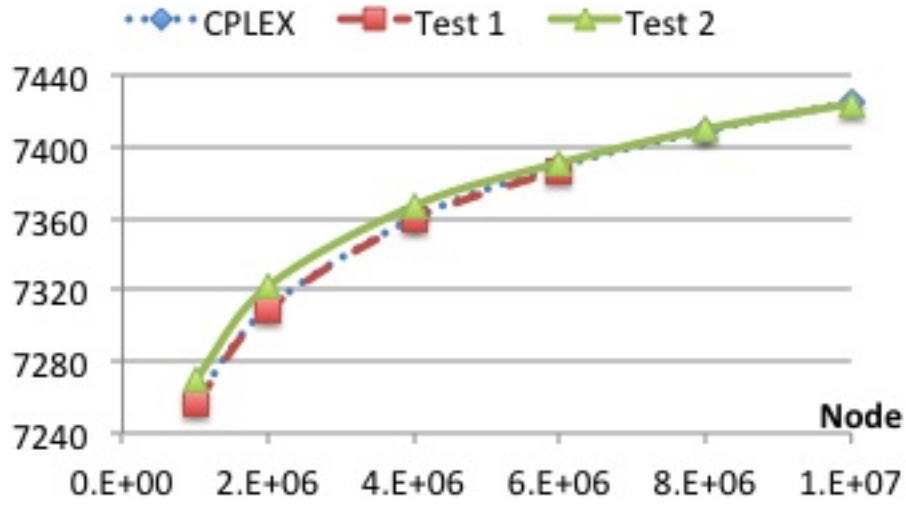
Figures 61 - 65 present the lower bounds obtained from both settings of the algorithm compared with CPLEX. In Figures 66 - 70, we show the relative gaps of B&B indicated by "Test 1 - Gap" and "Test 2 - Gap", respectively. For Test 1, since the percentage of special nodes that are pruned is high, we only see an increase in the value of Hi, thus we show the values of the largest relative gap for an special node indicated by "Test 1 - High". On the other hand, for Test 2, since the percentage of special nodes decreases, we only see an decrease in the value of Lo. Therefore, we present the values of the smallest relative gap for an special node indicated by "Test 2 - Low".

For Test 1, as we include more special nodes with larger relative gaps, the lower bounds are weakened when compared with only including special nodes with smaller relative gaps. However, the lower bounds of Test 2 are clearly increased, which indicates that including more special nodes with smaller relative gap is effective in improving lower bounds. Note that the values of Lo indicated by "Test 2 - Low" decrease stepwise, and are very close to the B&B relative gaps indicated by "Test 2 - Gap", indicating that the special nodes considered in Test 2 are always those with the largest relative gaps in the B&B tree. This implies that for properly chosen special nodes, even solving relaxations on a small percent of nodes can be effective in improving the lower bounds. In addition, since the percentage of special nodes is low, the efficiency of the algorithm in Test 2 is close to CPLEX. Therefore, among all the tests with the dual heuristic algorithm, Test 2 gives the best results by balancing of the quality of the lower bounds and time efficiency.

The results of tests indicate that always choosing special nodes with the largest relative gaps and maintaining a low level of $Check\_Percent$ are effective in improving lower bounds. However, including more active constraints in the relaxations at nodes with small depths

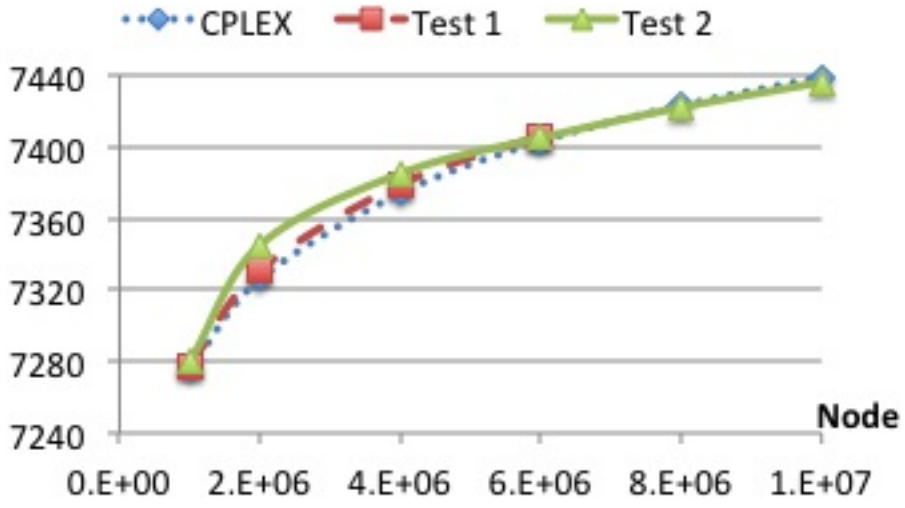


(a) Instance 1

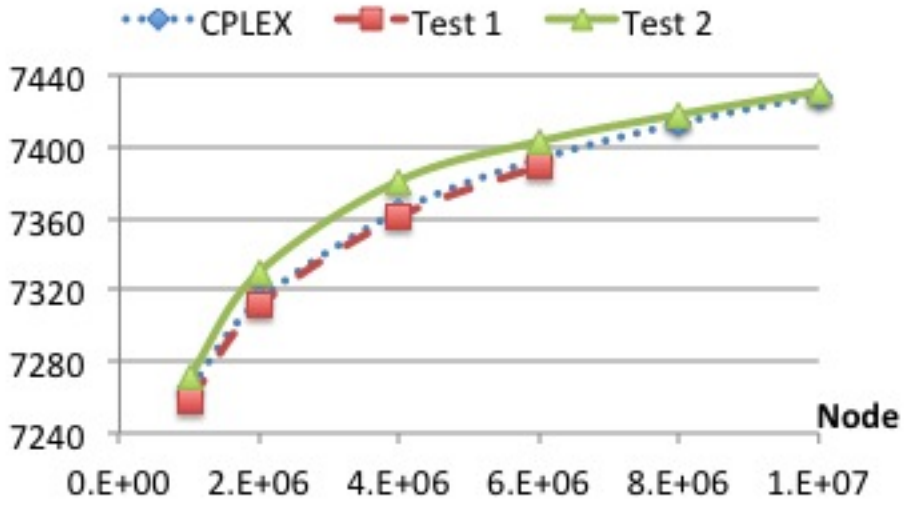


(b) Instance 2

Figure 61: Lower bounds for modifying the relative gap ranges

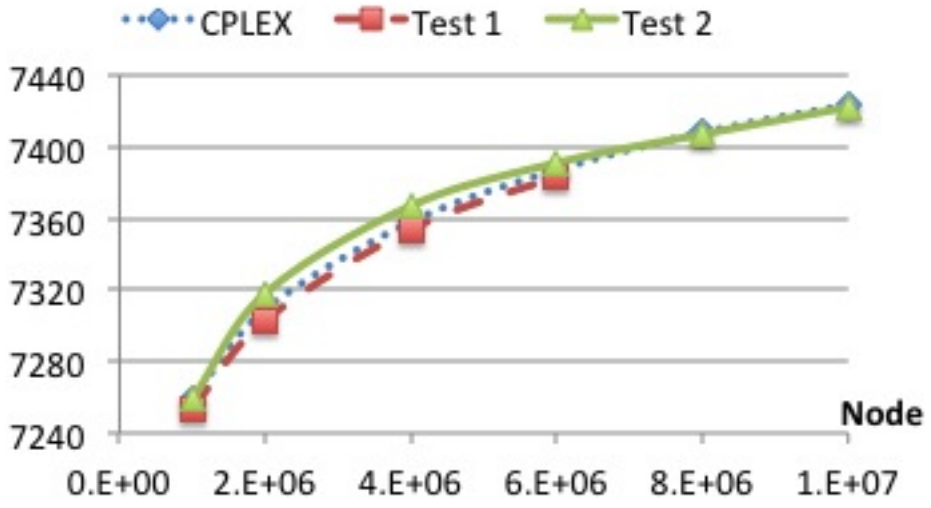


(a) Instance 3

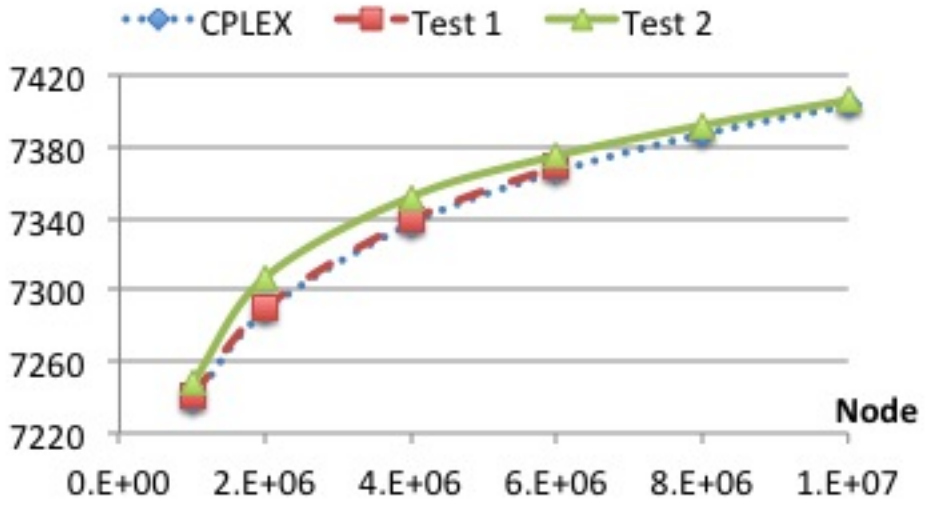


(b) Instance 4

Figure 62: Lower bounds for modifying the relative gap ranges

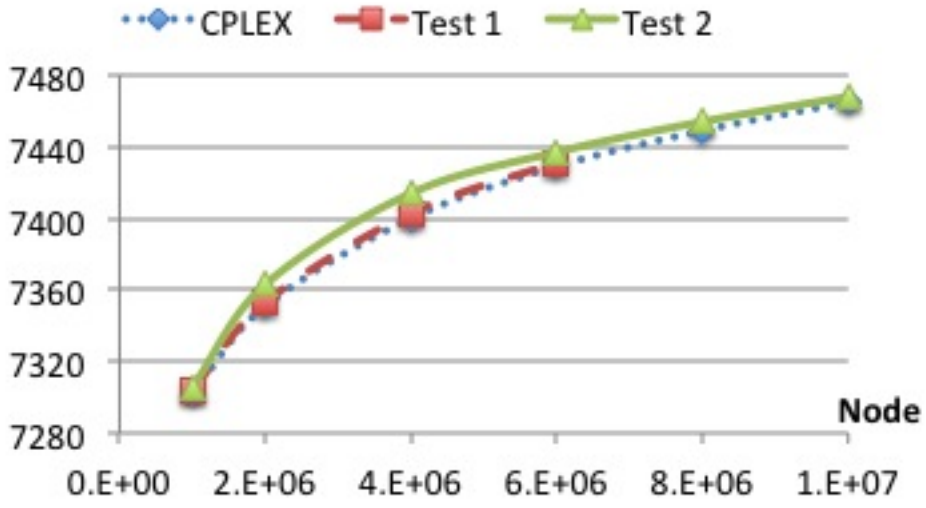


(a) Instance 5

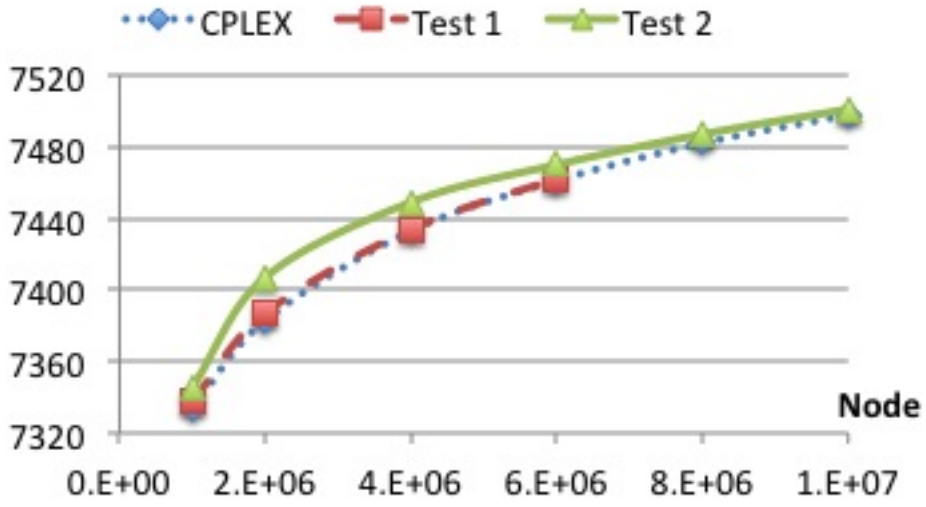


(b) Instance 6

Figure 63: Lower bounds for modifying the relative gap ranges

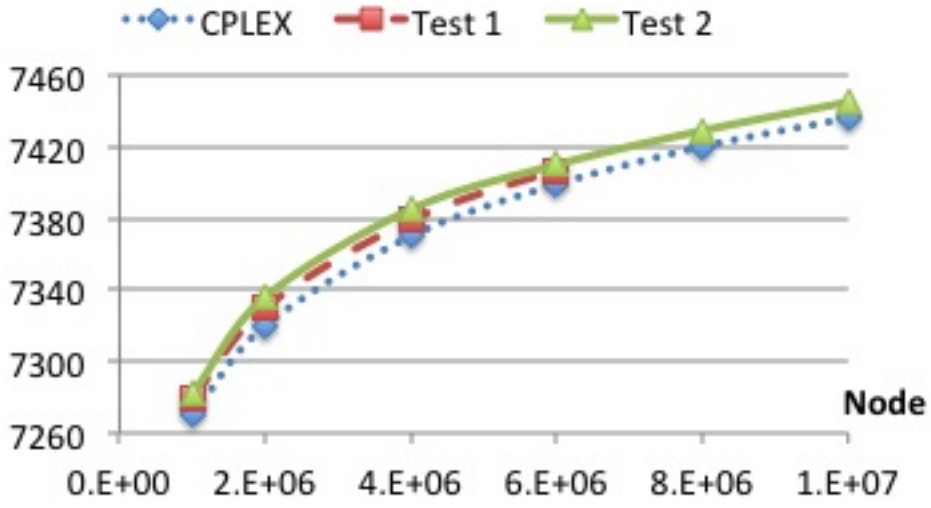


(a) Instance 7

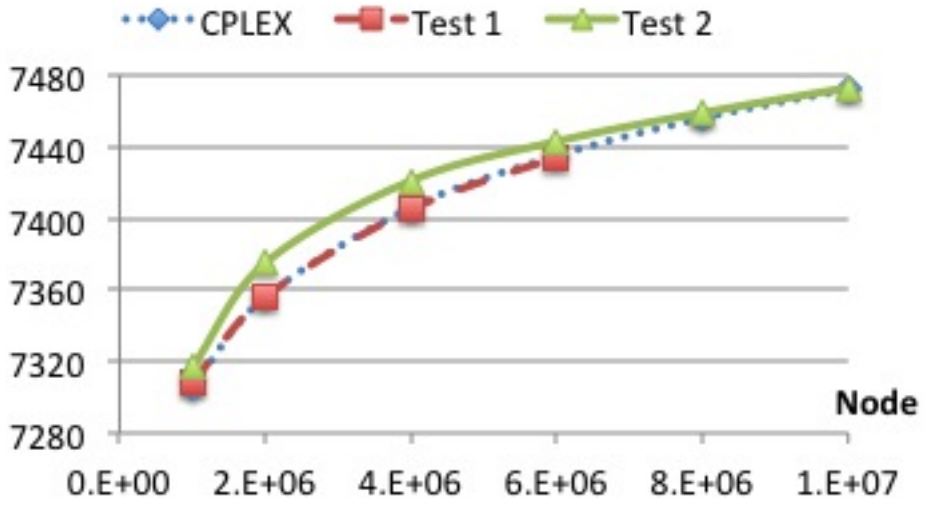


(b) Instance 8

Figure 64: Lower bounds for modifying the relative gap ranges

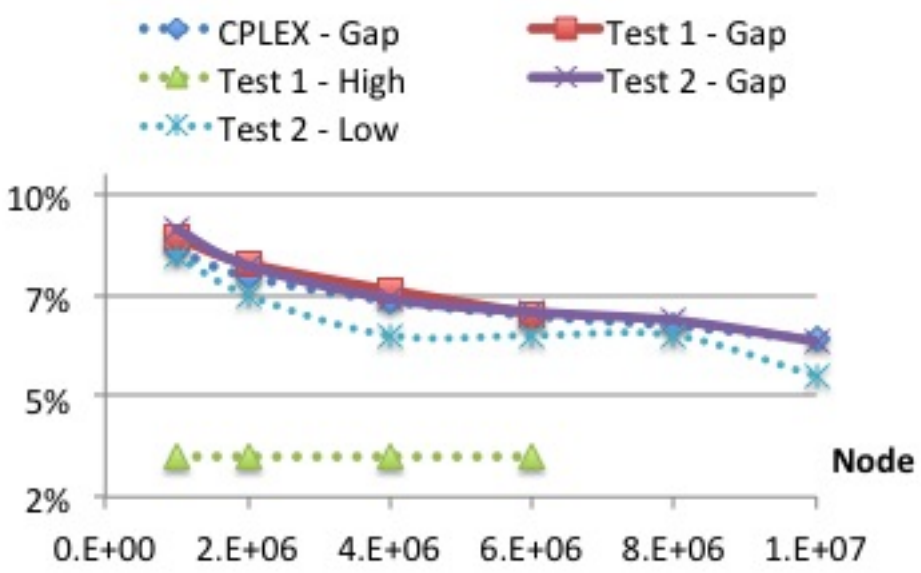


(a) Instance 9

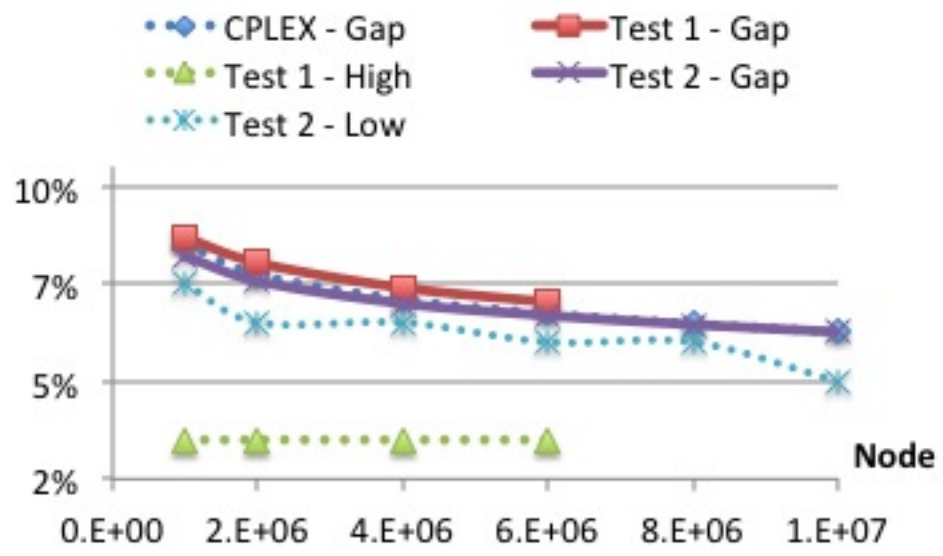


(b) Instance 10

Figure 65: Lower bounds for modifying the relative gap ranges

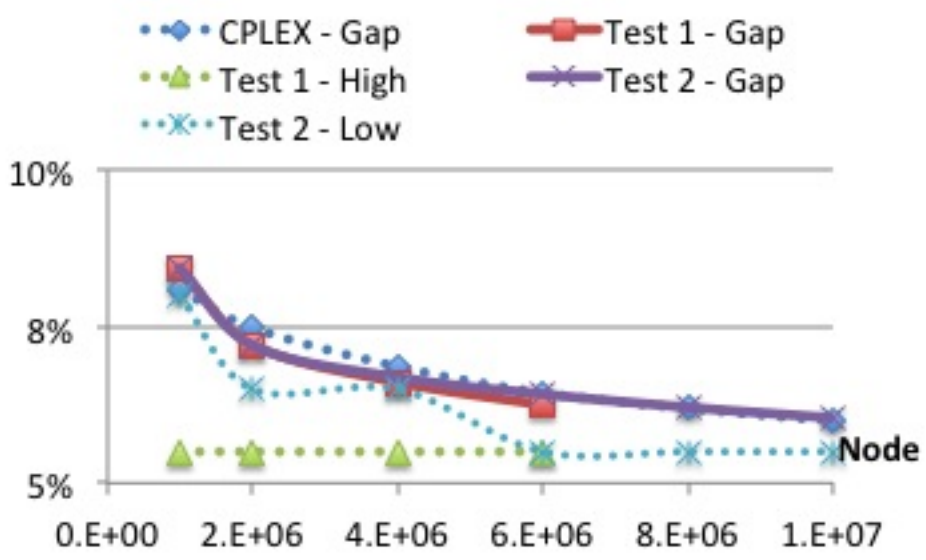


(a) Instance 1

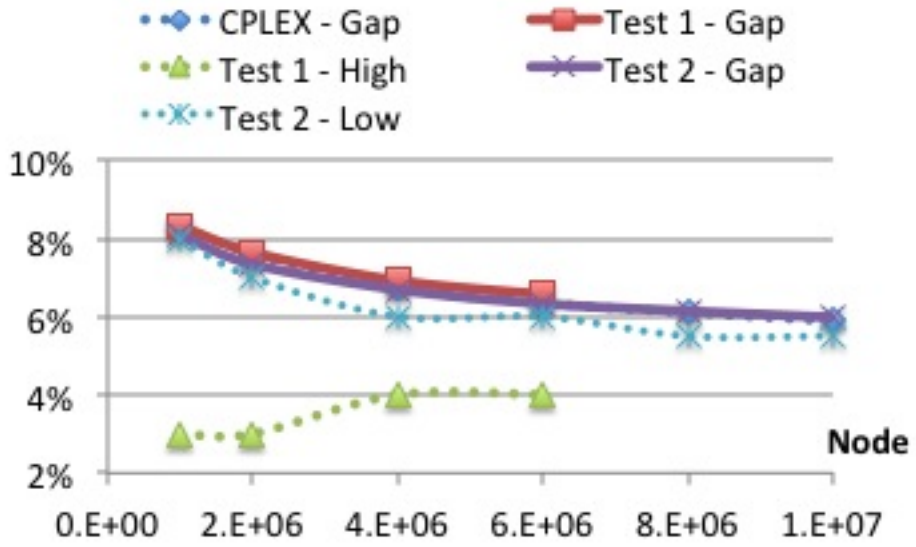


(b) Instance 2

Figure 66: Gap bounds for modifying the relative gap ranges

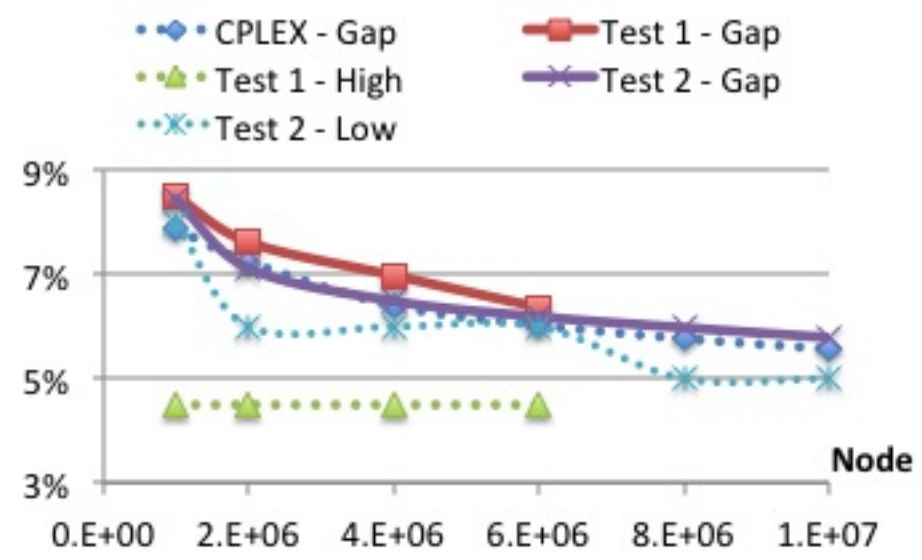


(a) Instance 3

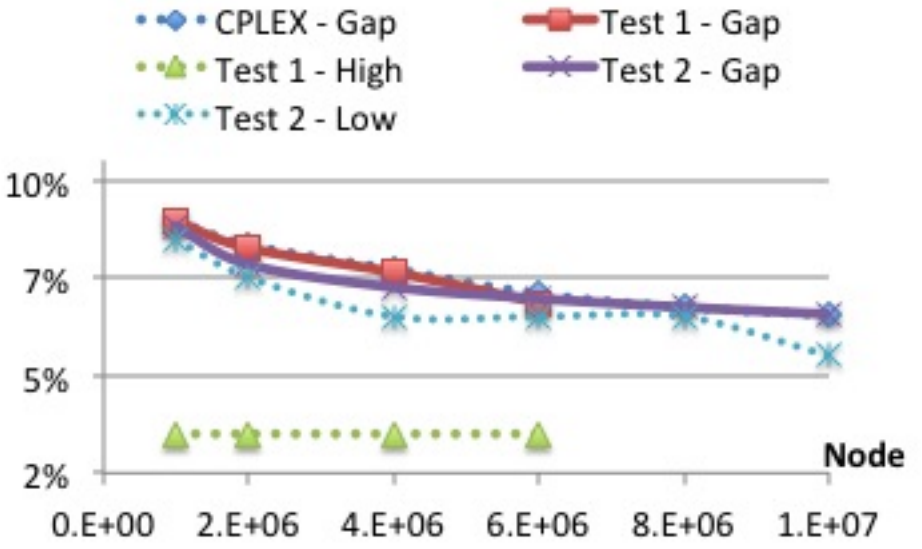


(b) Instance 4

Figure 67: Gap bounds for modifying the relative gap ranges

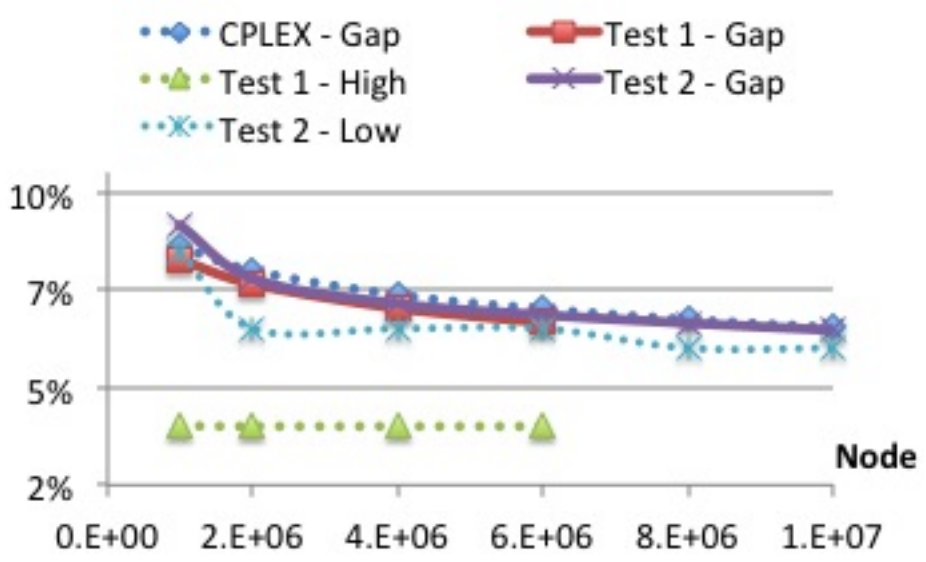


(a) Instance 5

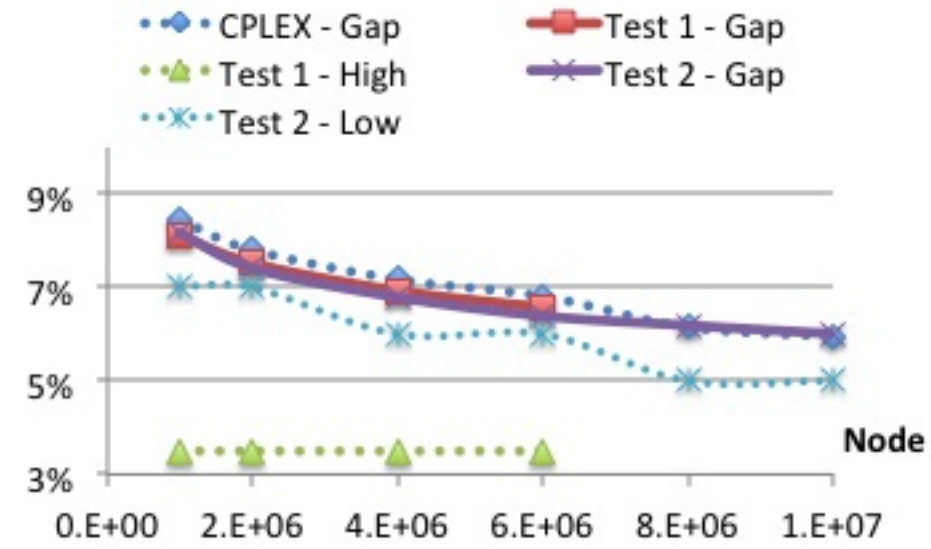


(b) Instance 6

Figure 68: Gap bounds for modifying the relative gap ranges

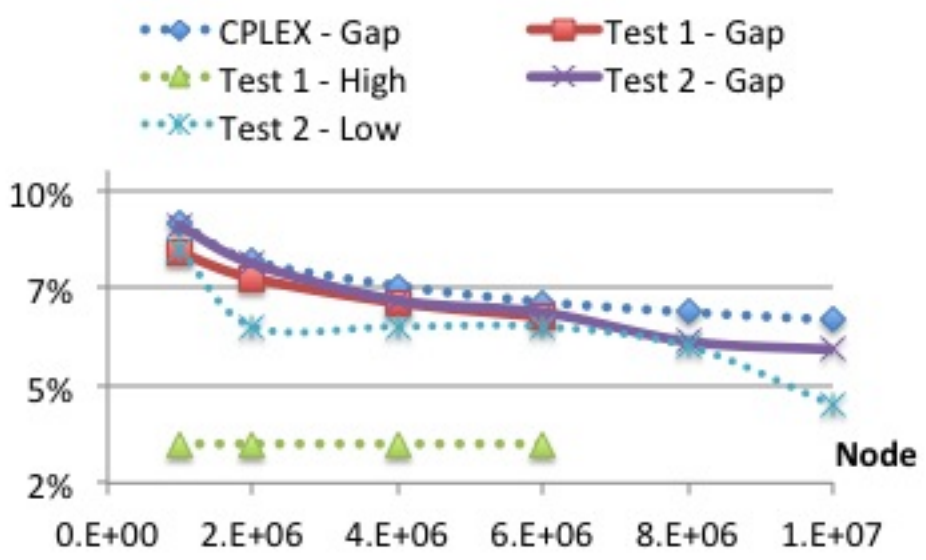


(a) Instance 7

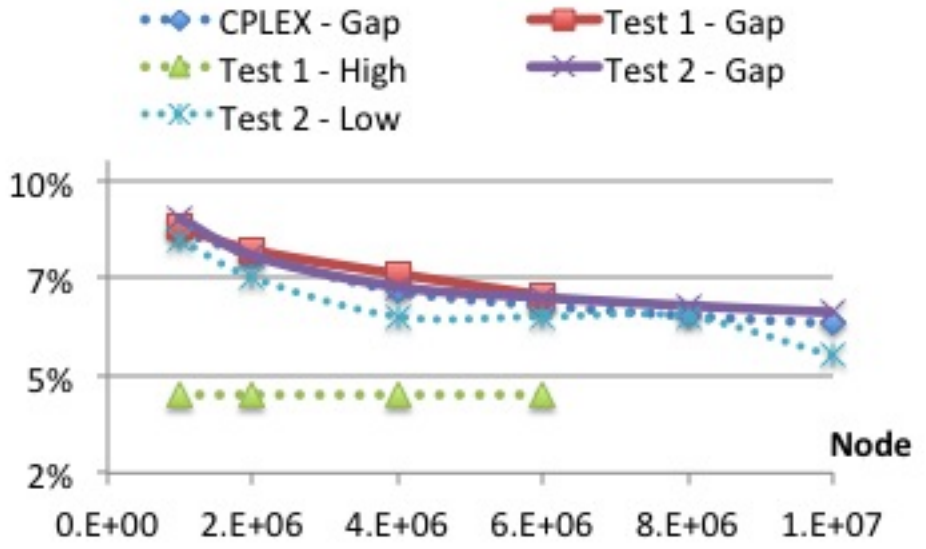


(b) Instance 8

Figure 69: Gap bounds for modifying the relative gap ranges



(a) Instance 9



(b) Instance 10

Figure 70: Gap bounds for modifying the relative gap ranges

does not help much in obtaining better lower bounds.

5.1.5 Conclusions and future research

In this chapter, we studied using various relaxation algorithms to obtain lower bounds for MKP, and examined the impact of the number of active constraints on the lower bounds obtained from the relaxations. Our results indicated that for MKP, the quality of lower bounds depends heavily on the choice of active constraints and also the number of active constraints.

In addition, we developed a dual heuristic algorithm which solves relaxations during the execution of the B&B algorithm with various active constraints to improve lower bounds. Furthermore, we explored methods in improving the algorithm by modifying parameters that control the implementation. The results indicated that by always choosing special nodes with the largest relative gaps among the remaining nodes in the B&B tree, the quality of the lower bounds can be improved, since the percentage of special nodes does not decrease. We concluded that for the improvement on lower bounds per node, our algorithm is effective. However, the efficiency of the algorithm is limited by the efficiency of solving the relaxation subproblems. We see two directions for future work to improve the algorithm.

As shown in Figure 40, "important" constraints in different stages of the B&B algorithm may vary depending on the nodes. Therefore, a strategy that chooses diversified active constraints may be needed. Exploring diversified neighborhood to find better feasible solutions has been considered in some effective primal heuristics, a similar idea for the dual heuristic is to limit the number of the same active constraints that are used in two special nodes. Specifically, let v be a special node where the lower bound is not improved by solving the relaxation. Then for special nodes that are in the subtree rooted at v, we may require the use of a somewhat different set of active constraints. This strategy reduces the dependency of our algorithm on the linear relaxation solutions, and thus may help when the LPs are highly degenerate.

On the other hand, we notice that for almost all special nodes, the number of tight constraints included in the relaxation is less than 25. This indicates that with only a small

number of active constraints, the lower bounds of the nodes can still be improved by solving the relaxations. This provides us hope of improving the efficiency of the algorithm by using a single surrogate constraint in relaxations, where only a small number of constraints can have large weights. That is, we obtain lower bounds by solving problem:

$$\max\{c^T x : \lambda A x \geq \lambda b, x \in \mathbb{Z}_+^n\}.$$

The weights of the constraints, indicated by λ, can be updated using information obtained during the B&B algorithm. However, for tight constraints at the special nodes, the weights can be made larger. Since the number of tight constraints is usually very small at all special nodes, the quality of the lower bounds obtained with the single surrogate constraint may not be much weaker, but the efficiency in solving the relaxation as a knapsack problem may be improved.

CHAPTER VI

A SUBADDITIVE ALGORITHM AND SHORTEST PATH ALGORITHM FOR MKP

So far, we have focused on relaxation algorithms that can be used to obtain lower bounds for general integer programming problems. In this chapter, we examine two special-purposed algorithms to obtain lower bounds for the multi-dimensional knapsack problem (MKP). We first develop a subadditive dual algorithm, then reformulate MKP as a shortest path problem and implement a shortest path algorithm to obtain lower bounds.

6.1 Introduction

Let $z(b)$ be the value function of MKP, which can be stated as:

$$\begin{aligned} z(b) = \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ i = 1, ..., m \\ & x \in \mathbb{Z}_n^+, \end{aligned} \tag{79}$$

where $A = (a_{ij})_{m \times n} \in \mathbb{Z}_{m \times n}^+$, $b \in \mathbb{Z}_m^+$ and $c \in \mathbb{R}_n^+$.

The dual problem of MKP can be generally defined as

$$z_D = \max\{g(b) : g(\cdot) \leq z(\cdot), g : \mathbb{R}_m^+ \to \mathbb{R}\}. \tag{80}$$

A feasible solution to the dual problem is called a *dual feasible solution*. Since the dual feasible solutions are functions on $\mathbb{R}_m^+$, they are also referred to as *dual feasible functions*. The set of dual feasible solutions are called the *dual (functional) space*. Since any dual feasible solution $g(\cdot)$ satisfies $g(b) \leq z(b)$, which yields $z_D \leq z(b)$, the dual problem (80) satisfies weak duality. In addition, the value function $z(\cdot)$ itself is a dual feasible solution, therefore, the dual problem (80) satisfies strong duality, that is, $z_D = z(b)$. Thus, obtaining the value of $z(b)$ is equivalent to solving the dual problem (80).

An advantage of solving the dual problem to obtain lower bounds is that any dual feasible solution gives a valid lower bound. Thus optimality is not required in (80) to obtain a lower bound on $z(b)$ in (79). However, the dual feasible solutions defined in problem (80) only depend on function $z(\cdot)$, where solving $z(t)$ for each $t \in \mathbb{R}^+_m$ is an $\mathcal{NP}-hard$ problem. Note that the value function $z(\cdot)$ is subadditive, where a *subadditive* function $F(\cdot)$ satisfies

$$F(x) + F(y) \geq F(x+y) \text{ for all } x, y \in \mathbb{R}^m_+,$$

because if x_1, x_2 are optimal solutions for $z(b_1)$ and $z(b_2)$ respectively, then $x_1 + x_2$ is a feasible solution for $z(b_1 + b_2)$, thus $z(b_1 + b_2) \leq z(b_1) + z(b_2)$. Then the subadditive dual problem of MKP can be defined as:

$$\begin{aligned} z_{SD} = & \max F(b) \\ & \text{s.t. } F(A_j) \leq c_j, j = 1, ..., n && (81) \\ & F(0) = 0 && (82) \\ & F : \mathbb{R}^m_+ \to \mathbb{R} \text{ is non-decreasing and subadditive.} && (83) \end{aligned}$$

Inequality (81) and subadditivity guarantee that $F(Ax) \leq \sum_{j=1}^n F(a_j)x_j \leq \sum_{j=1}^n c_j x_j = c^T x$ for all $x \in \mathbb{Z}^+_n$. Because $F(\cdot)$ is non-decreasing, we have that $F(b) \leq F(Ax) \leq c^T x$ for all x satisfying $Ax \geq b$. Therefore the subadditive dual problem satisfies weak duality. Jeroslow (1978) [37] showed that the subadditive dual satisfies strong duality.

During the 1970s and 80s, integer programming duality theory was studied and developed, see Johnson (1973, 1979) [39][40], Burdet and Johnson (1977) [10], and Wolsey (1981) [63]. Llewellyn and Ryan (1993) [50] developed a primal-dual algorithm which solves the dual problem of general IP but is not computationally efficient. Several papers present constructive algorithms that produce subadditive dual functions. Klabjan (2004, 2006, 2007) [43][44][45] designed a family of subadditive functions on general and specific integer programming problems.

In particular, for the partitioning problem, Johnson (1980) [38] developed a subadditive lifting method, which constructs a series of subadditive dual functions and terminates with an optimal dual solution. He also designed three types of subadditive functions that can be implemented in the algorithm.

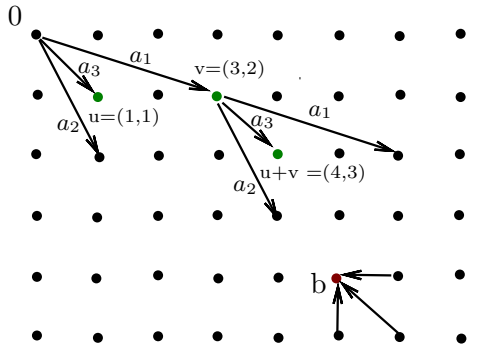


Figure 71: Shortest Path Problem for MKP

Because the subadditive lifting method continuously improves the lower bound and can be implemented with various subadditive functions, we use it as a framework and develop a modified subadditive dual algorithm which constructs a series of subadditive dual functions for MKP. The algorithm is based on an alternative formulation of MKP, that is, a shortest path formulation of MKP.

It is well known that the knapsack problem, which is a special case of MKP with $m = 1$, can also be formulated as a shortest path problem. With a similar technique, MKP can be transformed to the shortest path problem.

The shortest path problem for MKP with data (A, b, c) is defined on graph $G(A, b, c) = (V, E)$ as shown in Figure 71. Each node $v \in V$ corresponds to an integer vector $v \in \mathbb{Z}^+_m$, where node $u + v$ is denoted by the integer vector $(u_1 + v_1, ..., u_m + v_m) \in \mathbb{Z}^+_m$. Nodes $u, v \in V$ satisfy $u \geq v$ if $u_i \geq v_i$ for $i = 1, ..., m$. The set of nodes V can be partitioned into two sets: S and $V \setminus S$, where $S = \{v \in V : v \geq b\}$. For each $v \in V \setminus S$, there are n outgoing arcs $e_j(v) = (v, v + a_j) \in E$ with length c_j for $j = 1, ..., n$. For each $v \in S$, there is an outgoing arc $e_0(v) = (v, b) \in E$ with cost 0. Note that to find a path from node 0 to node b, only nodes satisfying $v_j \leq \sum_{i=1}^{m} b_i$ for $j = 1, ..., m$ need to be considered, therefore the size of set $V \setminus S$ is finite and depends on b, the right-hand side of the constraint.

Each path from node 0 to node b corresponds to a feasible solution of problem (79), the number of arcs of the form $(v, v + a_j)$ in a shortest path gives the value of x_j in an optimal solution to (79). Therefore, solving problem (79) is equivalent to finding a shortest path

from 0 to b on graph $G(A, b, c)$.

The shortest path problem has been studied extensively and still receives considerable attention because of its practical uses. There exists a large body of literature on efficient algorithms for solving the problem exactly. The well known Dijkstra's Algorithm (1959) [16] solves the problem in polynomial time as a function of the number of nodes and number of arcs in the graph. See Gallo and Pallottino (1986) [25] for a detailed survey on the shortest path algorithms. The shortest path reformulation for the knapsack problem first appeared in Gilmore and Gomory (1966)[27]. Frieze (1976) [24] proposed a labeling algorithm for the shortest path formulation of the knapsack problem.

Approximation algorithms for solving the shortest path problem have been considered because many applications involve very large scale graphs. Nemhauser (1972) [53] developed an algorithm using an estimated distance to label nodes. The algorithm is similar to the A^*- search technique for the artificial intelligence field, which also uses estimated distances for node selection, see Doran et al. (1967) [17] and Nilsson et al. (1968) [34]. Recently, Goldberg and Harrelson (2005) [29] developed an algorithm using A^*- search in combination with a graph lower-bounding technique that uses estimated lower bounds for true distances, which is very successful in solving large practical shortest path problems.

For MKP with large right-hand sides, the associated shortest path problem on graph $G(A, b, c)$ has a large number of nodes and arcs, thus algorithms with estimated distances can be very useful in efficiently obtaining lower bounds on the length of a shortest path. The second part of this chapter focuses on obtaining lower bounds for MKP using approximation algorithms on the corresponding shorting path problem. We propose an algorithm based on the work of Nemhauser (1972) [53] and calculate the estimated distances by solving linear programming problems. We use a special ordering on the arcs to reduce the total number of paths considered and conduct computational tests with the algorithm to examine the quality of lower bounds obtained.

The remainder of the chapter is structured as follows. In section 6.2, we introduce the subadditive lifting method by Johnson (1980) [38] and present a subadditive dual algorithm

using an alternative formulation of MKP. We analyze the algorithm using information obtained from the linear relaxation of MKP and give an example demonstrating the algorithm. In addition, we present an approximation algorithm to obtain lower bounds for the shortest path formulation of MKP. In Section 6.3, we show computational results using the shortest path algorithm.

6.2 Subadditive Dual and Shortest Path Algorithms

6.2.1 The Subadditive Lifting Method

We first introduce the framework of the subadditive lifting method by Johnson (1980) [38]. The method was developped for the partitioning problem, but can be modified and implemented on MKP. The partitioning problem can be stated as:

$$\begin{aligned} \bar{z}(b) = \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t. } & \sum_{j=1}^{n} a_{ij} x_j = b_i, \ i = 1, ..., m \\ & x \in \mathbb{Z}_n^+. \end{aligned} \tag{84}$$

Since the constraints in the partitioning problem are satisfied as equalities, the subadditive dual of problem (84) is defined as:

$$\begin{aligned} & \max F(b) \\ \text{s.t. } & F(A_j) \leq c_j, j = 1, ..., n \\ & F(0) = 0 \\ & F : \mathbb{R}_+^m \to \mathbb{R} \text{ is subadditive.} \end{aligned} \tag{85}$$

Therefore, a subadditive feasible function for the partitioning problem does not have to be non-decreasing.

The subadditive lifting method constructs a series of feasible subadditive dual functions and terminates with a function $F(\cdot)$ satisfying $F(b) = \bar{z}(b)$. The framework provides a general scheme to obtain an optimal subadditive dual function. Various types of subadditive functions can be implemented to build an optimal dual solution.

Three elements are maintained throughout the algorithm: a subadditive dual feasible function $F(\cdot)$, a fixed set $H \subset \mathbb{Z}_m^+$ and a candidate set $C \subset \mathbb{Z}_m^+$. The values of $F(\cdot)$ on

certain points are changed at each step while the subadditivity of the function is maintained. For each point $x \in H$, the value $F(x)$ can not be changed any more. For each point $x \in C$, we maintain an upper bound $U(x)$ of $F(x)$, such that the function satisfies

$$F(x) \leq U(x) \text{ for } x \in C. \tag{86}$$

The upper bounds are obtained from constraints (81) and (83). In addition, $F(b-x) = F(b) - F(x)$ is maintained for $x \in H$.

For initialization, F can be any feasible subadditive dual function. For example, a linear function or the constant function $F(x) = 0$ for all x. H is initialized by $H = \{0\}$, and the candidate set is initialized by $C = \{A_j\}_{j=1}^{n}$ with $U(A_j) = c_j$ for $j = 1, ..., n$.

The lifting step and the hitting step are then executed sequentially. In the lifting step, an increment is made on values of $F(\cdot)$ for all points that are not in H, such that (83) and inequalities (86) are satisfied. The various methods to "lift" the values of function $F(\cdot)$ determine the subadditive functions that are constructed.

In the hitting step, at least one point $x^* \in C$ satisfies $F(x) = U(x)$, and the point is marked as fixed. That is, $H = H \cup \{x^*\}$ and $C = C \setminus \{x^*\}$. To maintain subadditivity, the candidate set is then updated with $C = C \cup \{x^* + y : \text{ for all } y \in H\}$ and $U(x+y) = F(x) + F(y)$, thus $F(x+y) \leq U(x+y) = F(x) + F(y)$ is always guaranteed.

After an iterative step of lifting and hitting, the algorithm terminates when $F(b)$ can not be lifted any more, that is when $x, b-x \in H$ for some x, therefore $F(b) = F(b-x) + F(x)$ is the optimal value of the objective function.

The algorithm can be implemented with different methods for "lifting" function F. Johnson (1980) [38] proposed three types of subadditive functions for solving the knapsack problem, where $m = 1$. Next, we provide a subadditive dual algorithm which works with an alternative formulation for MKP based on this framework.

6.2.2 A Subadditive Dual Algorithm

We first define an alternative formulation for MKP, which allows our algorithm to use information obtained from the linear relaxation to the problem.

Proposition 17 *An alternative formulation for MKP can be stated as:*

$$\begin{aligned} &\max\ F(b) \\ &s.t.\ F(x+A_j)-F(x)\le c_j,\ \textit{for } x\not\geq b, j=1,...,n, \\ &\quad F(x)\ge F(b)\ \textit{for } x\ge b, \end{aligned} \tag{87}$$

Proof Let z_{IP} and z_{SD} be the optimal objective values of problem (79) and its subadditive dual problem respectively.

The shortest path problem for MKP can be formulated as:

$$\begin{aligned} z_{SP}=\min\ &\sum_{j=1...n}\sum_{d} c_j y_j(d) \\ \text{s.t. }\ &-\sum_{j=1}^{n} y_j(0)=-1, \\ &\sum_{d\ge b, d\neq b} y_0(d)+\sum_{j=1}^{n} y_j(b-A_j)=1, \\ &\sum_{j=1}^{n} y_j(d-A_j)-\sum_{j=1}^{n} y_j(d)=0,\ \text{for } d\in V\setminus S, \\ &\sum_{j=1}^{n} y_j(d-A_j)-y_0(d)=0,\ \text{for } d\in S, \\ &y_j(d)\in\{0,1\}\ \text{for all } j=1,...,n, d\in V, \end{aligned} \tag{88}$$

where $V=\mathbb{Z}^+_m$ and $S=\{v\in\mathbb{Z}^+_m : v\ge b\}$ as shown in section 6.1. The constraints represent flow conservation for nodes 0, b, nodes in $V\setminus S$ and S respectively, where one unit of flow is sent from node 0 to node b.

The dual problem of the linear relaxation of problem (88) can be written as

$$\begin{aligned} z_{LP}=\max\ &F(b)-F(0) \\ \text{s.t. }\ &F(d+A_j)-F(d)\le c_j,\ \text{for } d\not\geq b, j=1,...,n, \\ &F(d)\ge F(b)\ \text{for } d\ge b. \end{aligned} \tag{89}$$

Since the optimal value of z_{LP} only depends on the difference between $F(b)$ and $F(0)$, we can set $F(0)=0$.

By duality and definitions of the problems, we must have $z_{LP}\le z_{SP}$. Since the shortest path problem is an alternative formulation for MKP, we must have $z_{SP}=z_{IP}=z_{SD}$. By

comparing the problem formulations for z_{SD} and z_{LP}, we find that the subadditivity and constraint $F(A_j) \leq c_j$ lead to $F(x + A_j) \leq F(x) + F(A_j) \leq F(x) + c_j$, thus $F(x + A_j) - F(x) \leq c_j$. Therefore, problem (89) is a relaxation of the subadditive dual problem, that is, $z_{LP} \geq z_{SD}$. Combining the inequalities, we have $z_{SD} = z_{LP}$, thus problem (87) is an alternative formulation for MKP. □

With the new formulation of MKP, the non-decreasing value constraint on the subadditive dual function can be replaced by $F(x) \geq F(b)$ for $x \geq b$.

In addition, note that in the hitting step of the framework, when x^* is fixed, $x^* + y$ for all $y \in H$ are put into the candidate set to make sure that the subadditivity of the function is maintained. Thus the size of the candidate set increases exponentially. By Proposition 17, we modify the hitting step of the framework by adding $x^* + A_j$ to the candidate set with $U(x^* + A_j) = F(x^*) + c_j$ for $j = 1, ..., n$. Since in each step, at most n candidates can be added, then the size of the candidate set increases linearly.

The pseudocode of the dual algorithm is shown in Algorithm 2.

Algorithm 2 A Subadditive Dual Algorithm

Initialize:

$F^0(x) = \pi^T x$, $H^0 = \{0\}$, $C^0 = \{b, A_j\}_{j=1}^n$ with $U^0(A_j) = c_j, j = 1, ..., n$ and $U^0(b) = +\infty$.

Lift:

$$F^{i+1}(x) = \begin{cases} F^i(x) & x \in H^i, \\ F^i(x) + \alpha_i & x \notin H^i, \end{cases} \tag{90}$$

where $\alpha_i = \min_{x \in C^i} \{U^i(x) - F^i(x)\}$.

Hit:

Let $x^* = arg \min_{x \in C^i} \{U^i(x) - F^i(x)\}$, $H^{i+1} = H^i \cup \{x^*\}$.

$C^{i+1} = C^i \setminus \{x^*\} \cup \{x^* + A_j | j = 1, ..., n\}$ with:

$U^{i+1}(x^* + A_j) = \min\{U^i(x^* + A_j), F^i(x^*) + c_j\}$ and

$U^{i+1}(b) = \min\{U^i(b), \min_{d \gneq b} U^{i+1}(d)\}$.

Terminate when $F^i(b) = U^i(b)$.

We initialize $F(\cdot)$ by a linear function $F(x) = \pi^T x$ where π is the optimal linear dual solution of MKP, and increase the function value by the same amount on all points that are not fixed. Note that the points satisfying $v \gneq b$ do not need to be added to the candidate or the fixed set, since the values of $F(v)$ are never used except for the upper bound of $F(b)$.

Therefore by using $F(b) \leq \min_{d \gtrdot b}\{U(d)\}$, $F(v) \geq F(b)$ for $v \geq b$ is always satisfied.

Next, we show that although we use the algorithm on the alternative formulation of MKP, the functions constructed are subadditive.

Proposition 18 *The functions $F^i(\cdot)$ constructed in Algorithm 2 are subadditive.*

Proof We use induction to show that the functions constructed are all subadditive. For $i = 0$, F^0 is a linear function which is clearly subadditive.
Suppose F^{i-1} is a subadditive function, for some $i \geq 1$. To show $F^i(x+y) \leq F^i(x) + F^i(y)$, we consider the following cases:

- If neither of x or y is in H^{i-1}, then $F^i(t) = F^{i-1}(t) + \alpha_{i-1}$, for $t = x, y$. Thus we have $F^i(x+y) \leq F^{i-1}(x+y) + \alpha_{i-1} < F^{i-1}(x+y) + 2\alpha_{i-1} \leq F^{i-1}(x) + \alpha_{i-1} + F^{i-1}(y) + \alpha_{i-1} = F^i(x) + F^i(y)$.

- If exactly one of x or y is in H^{i-1}, say $y \in H^{i-1}$, then $F^{i-1}(y) = F^i(y)$ and $F^i(x) = F^{i-1}(x) + \alpha_{i-1}$. Thus $F^i(x+y) \leq F^{i-1}(x+y) + \alpha_{i-1} \leq F^{i-1}(y) + F^{i-1}(x) + \alpha_{i-1} = F^i(y) + F^i(x)$.

- If both x and y are in H^{i-1}, and if $x+y \in H^{i-1}$, then $F^{i-1}(t) = F^i(t)$ for $t = x, y, x+y$. Thus $F^i(x+y) \leq F^i(x) + F^i(y)$ since F^{i-1} is subadditive. If $x + y \notin H^{i-1}$, suppose that x is fixed at step $t \leq i-1$, thus $F^i(x) = F^t(x) = F^0(x) + \sum_{k=0}^{t-1} \alpha_k$. Similarly, let $F^i(y) = F^0(y) + \sum_{k=0}^{s-1} \alpha_k$ where $s \leq i-1$. Note that $F^i(x+y) = F^0(x+y) + \sum_{k=0}^{i-1} \alpha_k$. Therefore $F^i(x+y) \leq F^i(x) + F^i(y)$ if and only if $\sum_{k=0}^{t-1} \alpha_k + \sum_{k=0}^{s-1} \alpha_k \leq \sum_{k=0}^{i-1} \alpha_k$, i.e. $\sum_{k=0}^{s-1} \alpha_k \leq \sum_{k=t}^{i-1} \alpha_k$. Note that if this does not hold, i.e. $\sum_{k=0}^{s-1} \alpha_k > \sum_{k=t}^{i-1} \alpha_k$, $x+y$ would have been fixed before y. Thus this is a contradiction with $y \in H^{i-1}$ but $x + y \notin H^{i-1}$.

Therefore we have shown that the functions F^i for $i = 0, 1, ...$ are subadditive. □

Since the value of the functions $F^i(\cdot)$ are increased by the same amount at all points, keeping track of the fixed set in each step is sufficient for recording the subadditive function constructed in each step. For all $x \notin H^i$, $F^i(x) = F^0(x) + \sum_{k=0}^{i-1} \alpha_k$. If $x \in H^t \setminus H^{t-1}$ for

some $t < i$, that is, x is added to the fixed set at step $t-1$, then $F^i(x) = F^0(x) + \sum_{k=0}^{t-1} \alpha_k$. Also note that $F^i(b) = F^0(b) + \sum_{k=0}^{i-1} \alpha_k$ is a valid lower bound on the optimal solution of MKP for each i, since at each iteration F^i is a feasible subadditive dual solution.

We illustrate the algorithm with the following example.

Example Consider the knapsack problem

$$\begin{aligned} \min\ & 2x_1 + 3x_2 + 4x_3 + 5x_4 + 7x_5 \\ \text{s.t. } & x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 \geq 10 \\ & x \geq 0,\ \text{integer.} \end{aligned} \tag{91}$$

The optimal linear dual solution is $\pi = \frac{5}{4}$.

In step 0, $F^0(x) = \frac{5}{4}x$, for all $x \geq 0$. $H^0 = \{0\}$, $C^0 = \{1,2,3,4,5,10\}$ with $U(1) = 2, U(2) = 3, U(3) = 4, U(4) = 5, U(5) = 7, U(10) = +\infty$. $\alpha_0 = \min\{2 - \frac{5}{4} \times 1, 3 - \frac{5}{4} \times 2, 4 - \frac{5}{4} \times 3, 5 - \frac{5}{4} \times 4, 7 - \frac{5}{4} \times 5\} = 0$. Therefore, 4 is added to the fixed set and $LB = F^0(b) + \alpha_0 = \frac{25}{2}$.

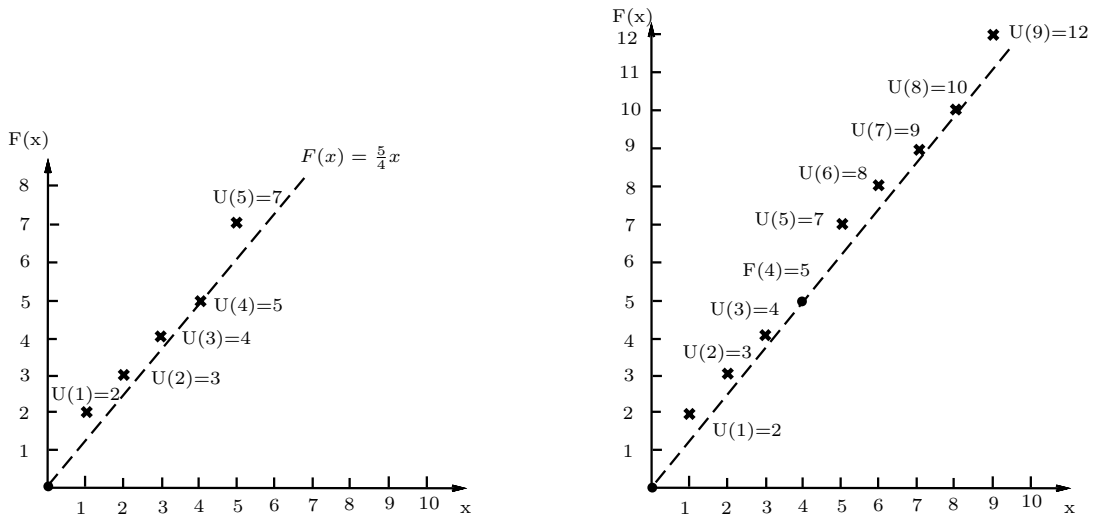
In step 1, $F^1(x) = \frac{5}{4}x$. $H^1 = \{0,4\}$, $C^1 = \{1,2,3,5,6,7,8,9,10\}$ with $U(1) = 2, U(2) = 3, U(3) = 4, U(5) = 7, U(6) = 8, U(7) = 9, U(8) = 10, U(9) = 12, U(10) = +\infty$. $\alpha_1 = \min\{2 - \frac{5}{4} \times 1, 3 - \frac{5}{4} \times 2, 4 - \frac{5}{4} \times 3, 7 - \frac{5}{4} \times 5, 8 - \frac{5}{4} \times 6, 9 - \frac{5}{4} \times 7, 10 - \frac{5}{4} \times 8, 12 - \frac{5}{4} \times 9\} = 0$. 8 is added to the fixed set and $LB = F^1(b) + \alpha_1 = \frac{25}{2}$.

In step 2, $F^2(x) = \frac{5}{4}x$. $H^2 = \{0,4,8\}$, $C^2 = \{1,2,3,5,6,7,9,10\}$ with $U(1) = 2, U(2) = 3, U(3) = 4, U(5) = 7, U(6) = 8, U(7) = 9, U(9) = 12, U(10) = 13$. $\alpha_2 = \min\{2 - \frac{5}{4} \times 1, 3 - \frac{5}{4} \times 2, 4 - \frac{5}{4} \times 3, 7 - \frac{5}{4} \times 5, 8 - \frac{5}{4} \times 6, 9 - \frac{5}{4} \times 7, 12 - \frac{5}{4} \times 9, 13 - \frac{5}{4} \times 10\} = \frac{1}{4}$, where 3, 7 are added to the fixed set and $LB = F^2(b) + \alpha_2 = \frac{51}{4}$.

In step 3, $F^3(x) = \frac{5}{4}x$ for $x \in H^2$ and $F^3(x) = \frac{5}{4}x + \frac{1}{4}$ for $x \notin H^2$. $H^3 = \{0,3,4,7,8\}$, $C^3 = \{1,2,5,6,9,10\}$ with $U(1) = 2, U(2) = 3, U(5) = 7, U(6) = 8, U(9) = 12, U(10) = 13$. $\alpha_3 = \min\{2 - \frac{5}{4} \times 1 - \frac{1}{4}, 3 - \frac{5}{4} \times 2 - \frac{1}{4}, 7 - \frac{5}{4} \times 5 - \frac{1}{4}, 8 - \frac{5}{4} \times 6 - \frac{1}{4}, 12 - \frac{5}{4} \times 9 - \frac{1}{4}, 13 - \frac{5}{4} \times 10 - \frac{1}{4}\} = \frac{1}{4}$, where 2, 6, 10 are added to the fixed set and $LB = F^3(b) + \alpha_3 = 13$.

Since $F^3(10) = 13 = U(10)$, optimality is obtained. Note that $U(10)$ is obtained at step 2 with $U(10) = F^2(8) + c_2 = \frac{5}{4} \times 8 + 3 = 13$, an optimal solution is $x_4 = 2$, $x_2 = 1$ and $x_j = 0$ for $j \neq 2, 4$.

Figure 72 and 73 display the steps of the algorithm applied on the example. The solid



Subadditive Algorithm: step 0 Subadditive Algorithm: step 1

Figure 72: Subadditive Algorithm: step 0-1

dots represent the points in the fixed set, the crosses represent the points in the candidate set and the corresponding upper bounds, and the dotted line represents the subadditive function constructed. It is easy to see that the subadditive functions constructed are linear functions with possible exceptions on points in the fixed set.

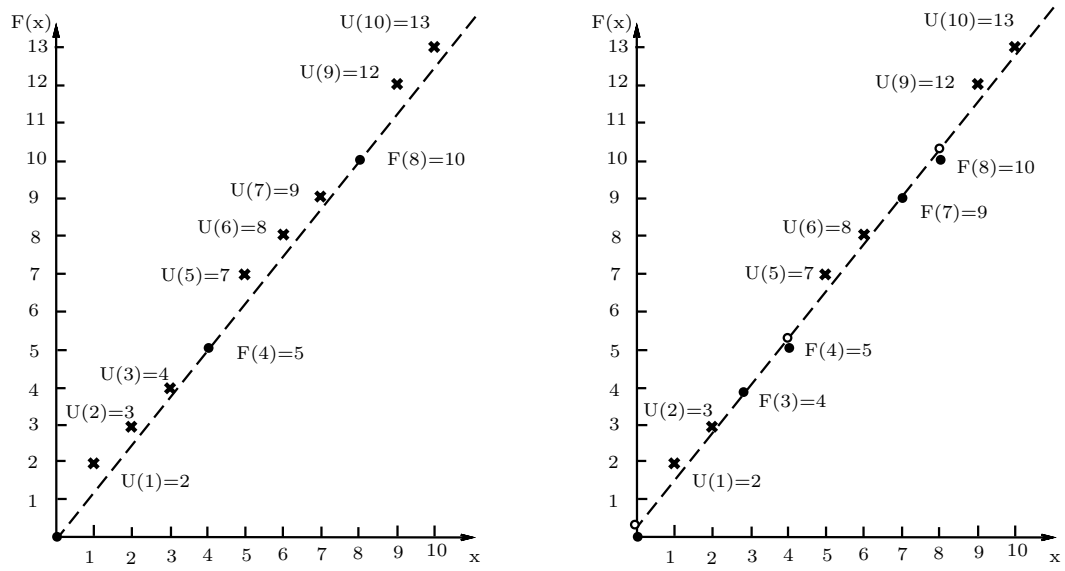
□

A major stage in the subadditive dual algorithm is to find the points to add to the fixed set and to compute the increment α_i in each step i. This stage can be done efficiently using the reduced costs of variables.

Let $\delta_j = c_j - F^0(A_j) = c_j - \pi^T A_j$ denote the reduced cost of x_j in the linear relaxation of MKP, where π is the optimal linear dual solution. The reduced cost of x_j indicates the unit increase of the objective value for x_j to be positive in the solution. Therefore, all basic variables have reduced costs of 0.

Assume that the variables are in order of nondecreasing reduced costs, that is, $\delta_1 \leq \delta_2 \leq ... \leq \delta_n$. Then we can make the following statements.

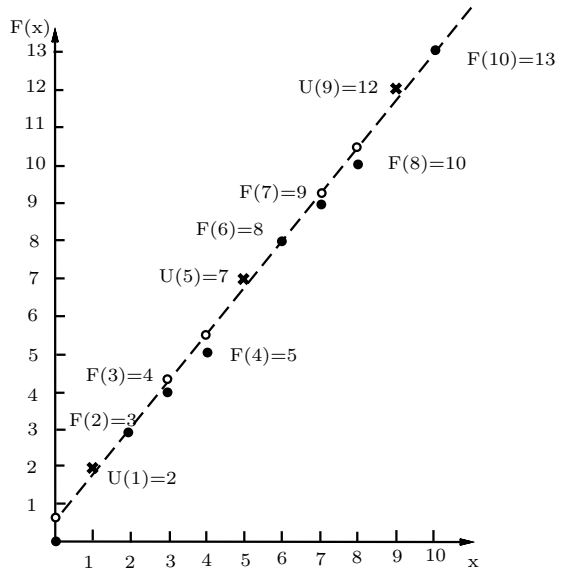
Proposition 19 *The increment made in the function value in step i, α_i, depends on the increments in previous steps $\alpha_0, ..., \alpha_{i-1}$ and the reduced costs of variables.*



Subadditive Algorithm: step 2

Subadditive Algorithm: step 3

Figure 73: Subadditive Algorithm: step 2-3



Subadditive Algorithm: final step

Figure 74: Subadditive Algorithm: step 4

Proof Suppose $x \in H^t \setminus H^{t-1}$ and $x + A_j \in C^i$ where $i \geq t$. Therefore $F^i(x + A_j) = F^0(x + A_j) + \sum_{k=0}^{i-1} \alpha_k$ and $U^i(x + A_j) = F^t(x) + c_j = F^0(x) + \sum_{k=0}^{t-1} \alpha_k + c_j$, thus the maximal increment for $x + A_j$ is

$$\alpha_i(x+A_j) = U^i(x+A_j) - F^i(x+A_j) = F^0(x) + \sum_{k=0}^{t-1} \alpha_k + c_j - F^0(x+A_j) - \sum_{k=0}^{i-1} \alpha_k = \delta_j - \sum_{k=t}^{i-1} \alpha_k. \tag{92}$$

That is to calculate the maximal increment of $x+A_j$, it is sufficient to look at the difference of the reduced cost of x_j and the increments made after x is fixed. Since $\alpha_i = \min_{x \in C^i}\{\alpha_i(x)\}$, α_i depends on $\alpha_0, ..., \alpha_{i-1}$ and δ_j for $j = 1, ..., n$. □

Proposition 19 shows that calculating the increment made in the function values can be done efficiently by using the reduced costs, the fixed set and the increments in previous steps. Therefore, the time spent on calculating the increment α_i is $O(|C^i|)$, where $|C^i|$ is the size of the candidate set at step i.

Proposition 19 also implies that the order of points being fixed are closely related to the reduced costs. Since $\delta_1 \leq \cdots \leq \delta_n$, we must have $\alpha_i(x + A_1) \leq \cdots \leq \alpha_i(x + A_n)$ at step i, given that $x + A_1, ..., x + A_n$ are not fixed yet. Thus, for any $x^* \in H$, $\{x^* + A_j\}_{j=1}^n$ should be fixed in a nondecreasing order of j. The following proposition extends the results on the order of points being fixed.

Proposition 20 *If $x \in H^t \setminus H^{t-1}$ and $y \in H^s \setminus H^{s-1}$ for some $t < s$. Then $x + A_j$ will be fixed before $y + A_j$ for $j = 1, ..., n$.*

Proof If $x + A_j$ is fixed before step s, then it is obvious that $x + A_j$ is fixed before $y + A_j$. Otherwise, by (92), $\alpha_i(x + A_j) = \delta_j - \sum_{k=t}^{i-1} \alpha_k \leq \alpha_i(y + A_j) = \delta_j - \sum_{k=s}^{i-1} \alpha_k$, for any $i > s$. Thus $x + A_j$ is fixed before $y + A_j$. □

Proposition 20 shows that if x is fixed before y, $x + A_j$ should be fixed before $y + A_j$ for $j = 1, .., n$ generally. Thus to calculate the increment $\alpha_i = \min_{x \in C^i}\{U^i(x) - F^i(x)\}$, although the size of the candidate set at step i is $|C^i| = O(n|H^i|)$, where $|H^i|$ is the size of the fixed set, it is sufficient to consider only a small portion of the candidate points using the pre-determined order indicated by Propositions 19 and 20.

The next proposition shows that the total increment in the sequential steps can be obtained using the reduced costs.

Proposition 21 *If x is fixed at step t_1 and $x+A_j$ is fixed at step t_2, then the total increment during steps between t_1 and t_2 is δ_j, that is, $\sum_{k=t_1}^{t_2-1} \alpha_k = \delta_j$.*

Proof Using (92), we know that if $x + A_j$ is fixed at step t_2, then $\alpha_{t_2}(x + A_j) = \delta_j - \sum_{k=t_1}^{t_2-1} \alpha_k = 0$. Thus the total increments since step t_1 equals to δ_j. □

Proposition 21 indicates that the total increments between the steps where x and $x+A_j$ are added to the fixed set respectively, can be calculated directly using the reduced cost δ_j.

Although the above efforts are made to reduce the computational complexity of the algorithm, for highly degenerate problems with small reduced costs of variables, the improvement on lower bounds can be limited, since these depend on the summation of the reduced costs as shown in Proposition 19. However, the subadditive dual algorithm shares similar components with a shortest path labeling algorithm, where the fixed set is analogous to a set of permanently labeled nodes and the candidate set is similar to the temporarily labeled nodes. In addition, the implementation of the algorithm is based on the shortest path formulation of MKP. This similarity motivate us to develop a shortest path algorithm that provides lower bounds for MKP.

6.2.3 A Shortest Path Algorithm

In this section, we consider solving the shortest path problem to obtain lower bounds for MKP. We first revisit the shortest path algorithm proposed by Nemhauser(1972) [53], which is a generalization of the Dijkstra's algorithm and uses estimated distances that are lower bounds of the true shortest distances between nodes.

Let C be a set of nodes that are temporarily labeled, and P be the set of nodes that are permanently labeled. Let f_v be the distance from node 0 to $v \in V$, which is updated throughout the algorithm until v is permanently labeled.

Define function h_v for each node $v \in V$ to be a lower bound of the shortest distance

from v to b, satisfying the triangular inequality, which can be stated as:

$$\text{For any } e = (v, w) \in E, h_v \leq h_w + c_e, \text{ and } h_b \leq 0. \tag{93}$$

Specifically, $h_v \leq h_{v+a_j} + c_j$ for all $v \in V \setminus S$, $j = 1, ..., n$, and $h_v \leq h_b$ for all $v \in S$, where $S = \{v \in V : v \geq b\}$ is used as defined in section 6.1.

Let $L(Q)$ denote the length of path Q, which equals to the sum of the lengths of all arcs on the path. For a path $Q = \{v = v_1, e_1, v_2, e_2, ..., e_{k-1}, v_k = b\}$, since $h_{v_i} \leq h_{v_{i+1}} + c_{e_i}$ for $i = 1, ..., k-1$, by summing the inequalities, we have $h_v \leq h_b + L(Q) \leq L(Q)$. Therefore, h_v is an underestimate of the shortest distance from node v to node b.

The pseudocode of the algorithm on graph $G(A, b, c)$ is given in Algorithm 3.

Algorithm 3 A Shortest Path Algorithm for Lower Bounding MKP

Initialize: $C = \{0\}$, $P = \emptyset$. $f_0 = 0$, $f_v = +\infty$ for all $v \neq 0$. $LB = +\infty$.
while $b \notin P$ **do**
  $v^* = \arg\min_{v \in C}\{f_v + h_v\}$.
  $P = P \cup \{v^*\}$, $LB = f_{v^*} + h_{v^*}$.
  for $j = 1, ..., n$ **do**
    $u = v^* + a_j$. If $u \notin P$, $C = C \cup \{u\}$, $f_u = \min\{f_u, f_{v^*} + c_j\}$
  end for
end while

In each step, the algorithm picks a node from set C with the smallest $f_v + h_v$ value and labels it permanently. When node v is permanently labeled, f_v is the shortest distance from node 0 to node v. Since h_v is an underestimate of the shortest distance from node v to node b, $f_v + h_v$ is an underestimate of the shortest distance from node 0 to node b for any path passing through node v. The algorithm terminates with $f_b + h_b$ being the shortest distance from 0 to b. Since the value of $f_{v^*} + h_{v^*}$ is non-decreasing, where v^* is the permanently labeled node in each step, $f_{v^*} + h_{v^*}$ is a valid lower bound of the shortest distance.

If h_v is the shortest distance from v to b for all $v \in V$, the lower bounds obtained always equal to the shortest distance from 0 to b. Therefore, the quality of the lower bounds obtained from the algorithm is determined by the quality of the estimated distance h_v.

We next introduce a new implementation of the algorithm, which obtains the estimated distance h_v by solving a linear programming problem for each v. Define h_v by solving the

following linear programming (LP) problem:

$$\begin{aligned} h_v = \min\ & c^T x \\ & Ax \geq b - v \\ & x \geq 0, \end{aligned} \tag{94}$$

where $b - v = (b_1 - v_1, ..., b_m - v_m) \in \mathbb{Z}_m$. It is easy to see that h_v defined by (94) satisfies the triangular inequality (93) and $h_b = 0$.

The estimated distance h_v defined by (94) is easy to compute, since only the right-hand side of the constraints change when varying v. However, note that the outgoing arcs are the same for all nodes in $V \setminus S$, thus there is a repeated structure in the graph. The LP formulation for h_v estimates the distance from node v to node b without using the recursive structure and the information obtained from computing f_v, thus it can be improved. The idea is to restrict the paths to be considered while not eliminating the shortest path. Inspired by previous discussions on the subadditive dual algorithm, we consider paths that use arcs in a fixed order. For example, consider path $P = \{v_1, e_{i_1}, v_2, e_{i_2}, ..., e_{i_{k-1}}, v_k\}$ with $i_1 \leq i_2 \leq ... \leq i_{k-1}$, that is, the arcs along the path have a nondecreasing order of index. Since the paths that use the same composition of arcs have the same length, the length of the shortest path remains the same.

The reduced cost is an indicator of the importance of variables, thus we order the variables in a nondecreasing order of reduced costs. For the paths that are considered in the algorithm, the arcs on the paths have nondecreasing reduced costs.

To enforce this order on paths, for each node v with finite f_v value, we keep the smallest index i of the incoming arcs of v, such that $f_v = f_{v-a_i} + c_i$. By adding constraints $x_j = 0$ for all $j < i$ in problem (94), we make sure that the estimated distance from v to b is based on paths that do not use arcs with index smaller than i, thus that the estimated distance from 0 to b via node v satisfies the requirements on the order of arcs. In addition, the modified LP formulation for computing h_v is tighter than (94) and the values of h_v can be increased. If node u is permanently labeled and $v = u + a_i$ is temporarily labeled, then the temporary

incoming arc for node v is a_i. Then the modified LP for h_v can be stated as:

$$\begin{aligned} h_v = \min\ & c^T x \\ & Ax \geq b - v \\ & x_j = 0, \text{ for } j = 1, ..., i-1 \\ & x \geq 0, \end{aligned} \tag{95}$$

Note that the modified LP problem for calculating h_v can not be determined until the arc that goes into the node can be determined, that is when node v is temporarily labeled. Therefore, if $(w, v) \in E$ with $v = w + a_j$ and w is permanently labeled, then the LP formulation for computing h_v must include constraints $x_i = 0$ for $i = 1, .., j-1$. Since the index of the arc going into node w is no greater than the index of the arc going into node v, we must have $h_w \leq h_v + c_j$. Therefore, the modified formulation for h_v satisfies the triangular inequalities.

6.3 Computational Results

In this section, we report the results of the computational tests we conducted with the shortest path algorithm for obtaining lower bounds for the MKP.

For consistency, we continue to use instances generated in Chapter 4. The instances are indicated by (n, m, α, K), where n is the number of variables, m is the number of constraints, α is the tightness indicator of the constraints and K is the correlation between the objective function and constraints. We test on four groups of instances, where each group contains 10 instances. Two groups have instances with 50 variables and 500 constraints, and two groups have instances with 50 variables and 1000 constraints. To include instances with different features, we use two groups of instances with $\alpha = 0.25$ and $K = 0$, and two groups with $\alpha = 1$ and $K = 500$.

We compare the results of the shortest path algorithm with the result we obtained from CPLEX 12.2 with default settings. We report the relative gaps and lower bounds obtained by both methods with time limits of 1 hour and 5 hours.

Note that for each node v, the value of $f_v + h_v$ is a lower bound on the shortest distance from 0 to b passing through node v. Therefore, with an upper bound UB for the shortest

Table 17: Lower Bounds of the Shortest path Algorithm: 50_500

instance	1 hour				5 hour			
	SP algorithm		Cplex		SP algorithm		Cplex	
$\alpha = 0.25$ $K = 0$	LB	Gap	LB	Gap	LB	Gap	LB	Gap
1	7191	9.07%	7349	7.49%	7247	8.36%	7468	5.85%
2	7163	9.04%	7314	7.17%	7220	8.32%	7435	5.64%
3	7168	8.89%	7328	7.45%	7225	8.16%	7445	5.91%
4	7162	9.27%	7322	7.51%	7218	8.56%	7436	5.80%
5	7157	8.96%	7311	7.22%	7214	8.23%	7429	5.50%
6	7137	9.36%	7290	7.81%	7196	8.61%	7409	5.96%
7	7202	8.97%	7353	7.46%	7258	8.27%	7904	5.45%
8	7242	9.16%	7384	7.80%	7292	8.53%	7505	5.87%
9	7171	9.17%	7324	7.63%	7226	8.47%	7448	6.02%
10	7204	9.17%	7358	7.54%	7262	8.44%	7477	5.70%
$\alpha = 1$ $K = 500$	LB	Gap	LB	Gap	LB	Gap	LB	Gap
1	36642	1.02%	36881	0.55%	36681	0.92%	37017	0.01%
2	36131	1.21%	36390	0.51%	36180	1.08%	36572	0.01%
3	34976	1.24%	35210	0.64%	35020	1.12%	35336	0.23%
4	36915	1.23%	37136	0.63%	36955	1.12%	37241	0.35%
5	35904	1.19%	36122	0.59%	35948	1.07%	36235	0.28%
6	35087	1.33%	35295	0.91%	35127	1.21%	35394	0.52%
7	36395	1.21%	36653	0.51%	36436	1.10%	36781	0.16%
8	36762	1.20%	37016	0.53%	36810	1.07%	37164	0.12%
9	38143	1.11%	38369	0.53%	38188	1.00%	38502	0.18%
10	35837	0.92%	36165	0.01%*	35877	0.80%	-	-

distance from 0 to b with $UB < f_v + g_v$, the node v needs not be labeled, since no shortest path passes through it. In the computational tests, we obtain such an upper bound from CPLEX. The instances marked with $*$ are solved within the time limit.

All tests are implemented on Linux machines with 2.27GHz processor and 6 GB of memory. The results are shown in Tables 17 and 18.

The empirical results show that for almost all instances, the lower bounds obtained from the shortest path algorithm are no better than the lower bounds obtained from CPLEX. With a time limit of one hour, the relative differences between the gaps of CPLEX and the shortest path algorithm are within 30% for instances with $\alpha = 0.25$ and $K = 0$, and go up to more than 50% for instances with $\alpha = 1$, and $K = 500$.

In addition, we notice that the ratio between the number of temporarily labeled nodes

Table 18: Lower Bounds of the Shortest path Algorithm: 50_1000

instance	1 hour				5 hour			
	SP algorithm		Cplex		SP algorithm		Cplex	
$\alpha = 0.25$ $K = 0$	LB	Gap	LB	Gap	LB	Gap	LB	Gap
1	7361	12.15%	7454	11.18%	7429	11.34%	7582	9.51%
2	7348	12.68%	7434	11.69%	7414	11.90%	7563	10.13%
3	7324	12.89%	7431	11.72%	7398	12.01%	7556	10.08%
4	7291	12.69%	7409	11.35%	7364	11.82%	7536	9.74%
5	7361	12.77%	7470	11.66%	7434	11.91%	7593	9.82%
6	7345	12.67%	7454	11.59%	7414	11.85%	7576	9.92%
7	7351	12.77%	7460	11.60%	7422	11.93%	7586	9.98%
8	7337	12.97%	7462	11.62%	7409	12.11%	7582	10.00%
9	7318	12.77%	7449	11.21%	7390	11.91%	7571	9.73%
10	7317	12.72%	7418	11.71%	7383	11.93%	7548	9.88%
$\alpha = 1$ $K = 500$	LB	Gap	LB	Gap	LB	Gap	LB	Gap
1	36823	1.27%	37038	0.70%	36879	1.12%	37145	0.41%
2	37373	0.61%	37565	0.1%*	37427	0.47%	-	-
3	37677	0%*	37677	0%*	-	-	-	-
4	35420	1.37%	35649	0.83%	35472	1.23%	35753	0.45%
5	37802	1.07%	38019	0.50%	37857	0.92%	38162	0.12%
6	38207	0.02%*	38160	0.14%*	-	-	-	-
7	36569	1.40%	36770	0.86%	36622	1.26%	36878	0.57%
8	37382	1.28%	37572	0.81%	37430	1.15%	37668	0.53%
9	37872	0.01%*	37869	0.02%*	-	-	-	-
10	37331	1.22%	37537	0.75%	37391	1.06%	37652	0.44%

and the number of the permanently labeled nodes is much smaller for instances with $\alpha = 1$ and $K = 500$ than instances with $\alpha = 0.25$ and $K = 0$. This indicates that for each permanently labeled node, a much smaller number of nodes on average get temporarily labeled for instances with $\alpha = 1$ and $K = 500$. Since the paths use a nonincreasing order of the arcs in terms of importance, nodes incident to incoming arcs with a large index are likely to be eliminated by the upper bound, because the values of h_v can be large for these nodes. The conclusion is consistent with the results in Chapter 4 that for instances with $\alpha = 1$ and $K = 500$, some columns are less important and less likely to be used in an optimal solution.

The advantage of CPLEX increases as time increases since the number of temporarily labeled nodes increases exponentially. Thus searching for nodes to label permanently takes more time and the memory used explodes. For most instances, the lower bounds obtained by the shortest path algorithm within 5 hours are not as good as the lower bounds of CPLEX within 1 hour. To improve the efficiency of the algorithm, better data structures and graph reduction techniques may be needed.

6.4 Conclusion

In this chapter, we reviewed a subadditive method and developed a new implementation for the subadditive lifting method on MKP. We then presented an approximation algorithm on a shortest path problem that is equivalent to MKP. Both algorithms provide an approximation for the optimal value of MKP from below and utilize the shortest path formulation of MKP. Our computational results show that the quality of lower bounds obtained from the shortest path algorithm is limited, which may be improved by using advanced data structures and additional techniques for solving large-scale shortest path problems.

CHAPTER VII

CONCLUSIONS AND FUTURE RESEARCH

This thesis studies the development of models and design of algorithms for integer programming problems that involve real-life applications and theoretical and computational innovations.

Chapter 2 presents a production planning model that uses discounts in pricing to reduce costs and further increase profit. We use time windows to increase flexibility of production and delivery, and develop a polynomial-time algorithm that simultaneously determines optimal discount and production planning decisions. In addition, we discuss variations of the model that further extend the flexibility of production and delivery. The computational experiments demonstrate that the benefit of using flexibility in production and delivery is closely related to the demand level, density of setups and the size of time windows. The results confirm that profit can be increased not only by increasing revenues, but also by focusing on reducing costs. We believe this is a fertile area for further research, which complements the traditional research on pricing to increase revenues with wide applicability in production planning and distribution management.

Chapter 3 through Chapter 6 study theoretical and computational methods for obtaining lower bounds for minimization integer programming problems. The research specifically focuses on the multidimensional knapsack problems (MKP), and considers both general and special-purpose methods.

Chapter 3 studies the corner relaxation for MKP and examines the worst-case gap of the lower bounds obtained from the corner relaxation. We also provide a sufficient condition under which the corner relaxation is tight. In addition, a periodic property of the value function is extended from the knapsack problem to MKP, thus the complexity for solving MKP with large right-hand side can be reduced using the periodicity. The results on the corner relaxation motivate further investigations on the quality of lower bounds from a

computational perspective.

Chapter 4 establishes meaningful benchmark instances for the computational experiments with MKP. We conduct computational tests to analyze the characteristics that affect the hardness of MKP, including the correlation between the objective function and constraints and the size of right-hand side values.

Chapter 5 compares the lower bounds obtained from various relaxation algorithms, and presents a dual heuristic algorithm that solves relaxations to improve lower bounds. Our results demonstrate that using information based on linear relaxations to choose active constraints is generally effective. However, for many hard instances, choosing the right constraints to relax may still be challenging, thus a large number of active constraints are needed to obtain good lower bounds. Furthermore, we compare the lower bounds obtained from constraint relaxations (which include the corner relaxation as a special case), lagrangian relaxation algorithms, surrogate relaxation algorithms, and lazy relaxations. The results indicate that lazy relaxations perform the best, since it allows adding the active constraints during the B&B algorithm. This motivates the design of a dual heuristic algorithm, which solves relaxed subproblems and uses the obtained lower bounds to improve linear relaxation bounds in the B&B algorithm. The computational results show that the dual heuristic algorithm has a significant potential in improving lower bounds for B&B and effectively reduces the size of the B&B tree. By modifying the parameters of the algorithm to properly select special nodes and formulate the relaxations, we demonstrate that there is a significant potential for improving the lower bounds by only solving a small number of relaxations. Since our goal is to develop an efficient algorithm to obtain lower bounds, further improvements on the efficiency of solving relaxations can be beneficial. In addition, investigations on how to collect information during the algorithm and use that to guide constraint selection in relaxations may also help to improve the quality of lower bounds. We propose two directions for future works that focus on these two aspects.

Finally, Chapter 6 examines two special-purpose algorithms for obtaining lower bounds for MKP: 1) a new implementation of the subadditive lifting method, which produces an exact optimal dual solution, and 2) a shortest path algorithm using the shortest path

formulation for MKP. We adopt a scheme that uses approximate distances for node labeling. The computational tests show that the quality of lower bounds obtained from the shortest path algorithm is limited. However, as more efficient large-scale shortest path algorithms are developed, it will be interesting to see if using these techniques and combining the recursive network structure can produce better performance.

REFERENCES

[1] Aarts, E. and Lenstra, J. K., *Local search in combinatorial optimization*. John Wiley & Sons, Inc., 1997.

[2] Applegate, D., Bixby, R., Chvatal, V., and Cook, W., *The traveling salesman problem: a computational study*. Princeton University Press, 2007.

[3] Ata, B. and Olsen, T. L., "Near-optimal dynamic lead-time quotation and scheduling under convex-concave customer delay costs," *Operations Research*, vol. 57, no. 3, pp. 753–768, 2009.

[4] Balas, E., "Disjunctive programming," *Annals of Discrete Mathematics*, vol. 5, pp. 3–51, 1979.

[5] Balas, E., Ceria, S., and Cornuéjols, G., "A lift-and-project cutting plane algorithm for mixed 0–1 programs," *Mathematical programming*, vol. 58, no. 1, pp. 295–324, 1993.

[6] Beasley, J., "OR - library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[7] Benichou, M., Gauthier, J. M., Girodet, P., Hentges, G., Ribiere, G., and Vincent, O., "Experiments in mixed-integer linear programming," *Mathematical Programming*, vol. 1, no. 1, pp. 76–94, 1971.

[8] Brahimi, N., Dauzère-Pérès, S., and Najid, N., "Capacitated multi-item lot-sizing problems with time windows," *Operations Research*, vol. 54, no. 5, p. 951, 2006.

[9] Brahimi, N., Dauzere-Peres, S., and Wolsey, L., "Polyhedral and lagrangian approaches for lot sizing with production time windows and setup times," *Computers & Operations Research*, vol. 37, no. 1, pp. 182–188, 2010.

[10] Burdet, C. and Johnson, E., "A subadditive approach to solve linear integer programs," *Annals of Discrete Mathematics*, vol. 1, pp. 117–144, 1977.

[11] Celik, S. and Maglaras, C., "Dynamic pricing and lead-time quotation for a multiclass make-to-order queue," *Management Science*, vol. 54, no. 6, pp. 1132–1146, 2008.

[12] Chu, P. and Beasley, J., "A genetic algorithm for the multidimensional knapsack problem," *Journal of heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[13] Chvátal, V., "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics*, vol. 4, no. 4, pp. 305–337, 1973.

[14] Danna, E., Rothberg, E., and Pape, C., "Exploring relaxation induced neighborhoods to improve mip solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[15] Dauzère-Pérès, S., Brahimi, N., Najid, N., and Nordli, A., "The single-item lot sizing problem with time windows," tech. rep., Technical Report, 02/4/AUTO, Ecole des Mines de Nantes, France, 2002.

[16] Dijkstra, E., "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[17] Doran, J., "An approach to automatic problem-solving," *Machine Intelligence*, vol. 1, no. 105-124, p. 35, 1967.

[18] Fischetti, M., Glover, F., and Lodi, A., "The feasibility pump," *Mathematical Programming*, vol. 104, no. 1, pp. 91–104, 2005.

[19] Fischetti, M. and Lodi, A., "Local branching," *Mathematical Programming*, vol. 98, no. 1, pp. 23–47, 2003.

[20] Fisher, M., "The lagrangian relaxation method for solving integer programming problems," *Management science*, vol. 50, no. 12 supplement, pp. 1861–1871, 2004.

[21] Fréville, A. and Hanafi, S., "The multidimensional 0-1 knapsack problembounds and computational aspects," *Annals of Operations Research*, vol. 139, no. 1, pp. 195–227, 2005.

[22] Freville, A. and Plateau, G., "Hard 0-1 multiknapsack test problems for size reduction methods," *Investigation Operativa*, vol. 1, pp. 251–270, 1990.

[23] Fréville, A. and Plateau, G., "An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem," *Discrete Applied Mathematics*, vol. 49, no. 1, pp. 189–212, 1994.

[24] Frieze, A., "Shortest path algorithms for knapsack type problems," *Mathematical Programming*, vol. 11, no. 1, pp. 150–157, 1976.

[25] Gallo, G. and Pallottino, S., "Shortest path methods: a unifying approach," in *Netflow at Pisa*, pp. 38–64, Springer, 1986.

[26] Galvao, R., Gonzalo Acosta Espejo, L., and Boffey, B., "A comparison of lagrangean and surrogate relaxations for the maximal covering location problem," *European Journal of Operational Research*, vol. 124, no. 2, pp. 377–389, 2000.

[27] Gilmore, P. C. and Gomory, R. E., "The theory and computation of knapsack functions," *Operations Research*, vol. 14, no. 6, pp. 1045–1074, 1966.

[28] Glover, F., "Surrogate constraints," *Operations Research*, vol. 16, no. 4, pp. 741–749, 1968.

[29] Goldberg, A. and Harrelson, C., "Computing the shortest path: A search meets graph theory," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 156–165, Society for Industrial and Applied Mathematics, 2005.

[30] Gomory, R., "An algorithm for the mixed integer problem," tech. rep., DTIC Document, 1960.

[31] Gomory, R., "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[32] Gomory, R., "Some polyhedra related to combinatorial problems," *Linear Algebra and its Applications*, vol. 2, no. 4, pp. 451–558, 1969.

[33] Guzelsoy, M., *Dual Methods in Mixed Integer Linear Programming*. Lehigh University, 2009.

[34] Hart, P., Nilsson, N., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[35] Held, M. and Karp, R., "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

[36] Held, M. and Karp, R., "The traveling-salesman problem and minimum spanning trees: Part ii," *Mathematical programming*, vol. 1, no. 1, pp. 6–25, 1971.

[37] Jeroslow, R., "Cutting-plane theory: Algebraic methods," *Discrete mathematics*, vol. 23, no. 2, pp. 121–150, 1978.

[38] Johnson, E. L., "Subadditive lifting methods for partitioning and knapsack problems," *Journal of Algorithms*, vol. 1, no. 1, pp. 75–96, 1980.

[39] Johnson, E., "Cyclic groups, cutting planes and shortest paths," *Mathematical programming*, pp. 185–211, 1973.

[40] Johnson, E., "On the group problem and a subadditive approach to integer programming," *Annals of Discrete Mathematics*, vol. 5, pp. 97–112, 1979.

[41] Karwan, M. and Rardin, R., "Surrogate dual multiplier search procedures in integer programming," *Operations Research*, vol. 32, no. 1, pp. 52–69, 1984.

[42] Kellerer, H., Pferschy, U., and Pisinger, D., *Knapsack problems*. Springer, 2004.

[43] Klabjan, D., "A practical algorithm for computing a subadditive dual function for set partitioning," *Computational optimization and applications*, vol. 29, no. 3, pp. 347–368, 2004.

[44] Klabjan, D., "A new subadditive approach to integer programming," in *Integer Programming and Combinatorial Optimization*, pp. 384–400, Springer, 2006.

[45] Klabjan, D., "Subadditive approaches in integer programming," *European journal of operational research*, vol. 183, no. 2, pp. 525–545, 2007.

[46] Kunreuther, H. and Schrage, L., "Joint pricing and inventory decisions for constant priced items," *Management Science*, vol. 19, no. 7, pp. 732–738, 1973.

[47] Land, A. H. and Doig, A. G., "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, vol. 28, no. 3, pp. 497–520, 1960.

[48] Lee, C., Çetinkaya, S., and Wagelmans, A., "A dynamic lot-sizing model with demand time windows," *Management Science*, vol. 47, no. 10, pp. 1384–1395, 2001.

[49] Linderoth, J. and Savelsbergh, M., "A computational study of search strategies for mixed integer programming," *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 173–187, 1999.

[50] Llewellyn, D. and Ryan, J., "A primal dual integer programming algorithm," *Discrete Applied Mathematics*, vol. 45, no. 3, pp. 261–275, 1993.

[51] Lodi, A., "Mixed integer programming computation," in *50 Years of Integer Programming 1958-2008*, pp. 619–645, Springer, 2010.

[52] Martello, S. and Toth, P., *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[53] Nemhauser, G., "A generalized permanent label setting algorithm for the shortest path between specified nodes," *Journal of Mathematical Analysis and Applications*, vol. 38, no. 2, pp. 328–334, 1972.

[54] Nemhauser, G. and Wolsey, L., *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[55] Osorio, M., Glover, F., and Hammer, P., "Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions," *Annals of Operations Research*, vol. 117, no. 1, pp. 71–93, 2002.

[56] Pochet, Y. and Wolsey, L., *Production planning by mixed integer programming*. Springer, 2006.

[57] Richard, J. and Dey, S., "The group-theoretic approach in mixed integer programming," in *50 Years of Integer Programming 1958-2008*, pp. 727–801, Springer, 2010.

[58] Savasaneril, S., Griffin, P. M., and Keskinocak, P., "Dynamic lead-time quotation for an m/m/1 base-stock inventory queue," *Operations Research*, vol. 58, no. 2, pp. 383–395, 2010.

[59] Shapiro, J., "Generalized lagrange multipliers in integer programming," *Operations Research*, vol. 19, no. 1, pp. 68–76, 1971.

[60] Thomas, J., "Price-production decisions with deterministic demand," *Management Science*, vol. 16, no. 11, pp. 747–750, 1970.

[61] Van den Heuvel, W. and Wagelmans, A., "A polynomial time algorithm for a deterministic joint pricing and inventory model," *European Journal of Operational Research*, vol. 170, no. 2, pp. 463–480, 2006.

[62] Wagner, H. M. and Whitin, T. M., "Dynamic version of the economic lot size problem," *Management Science*, vol. 5, no. 1, pp. 89–96, 1958.

[63] Wolsey, L., "Integer programming duality: Price functions and sensitivity analysis," *Mathematical Programming*, vol. 20, no. 1, pp. 173–195, 1981.

[64] Wolsey, L., "Lot-sizing with production and delivery time windows," *Mathematical Programming*, vol. 107, no. 3, pp. 471–489, 2006.

[65] Zhu, N., "A relation between the knapsack and group knapsack problems," *Discrete applied mathematics*, vol. 87, no. 1, pp. 255–268, 1998.