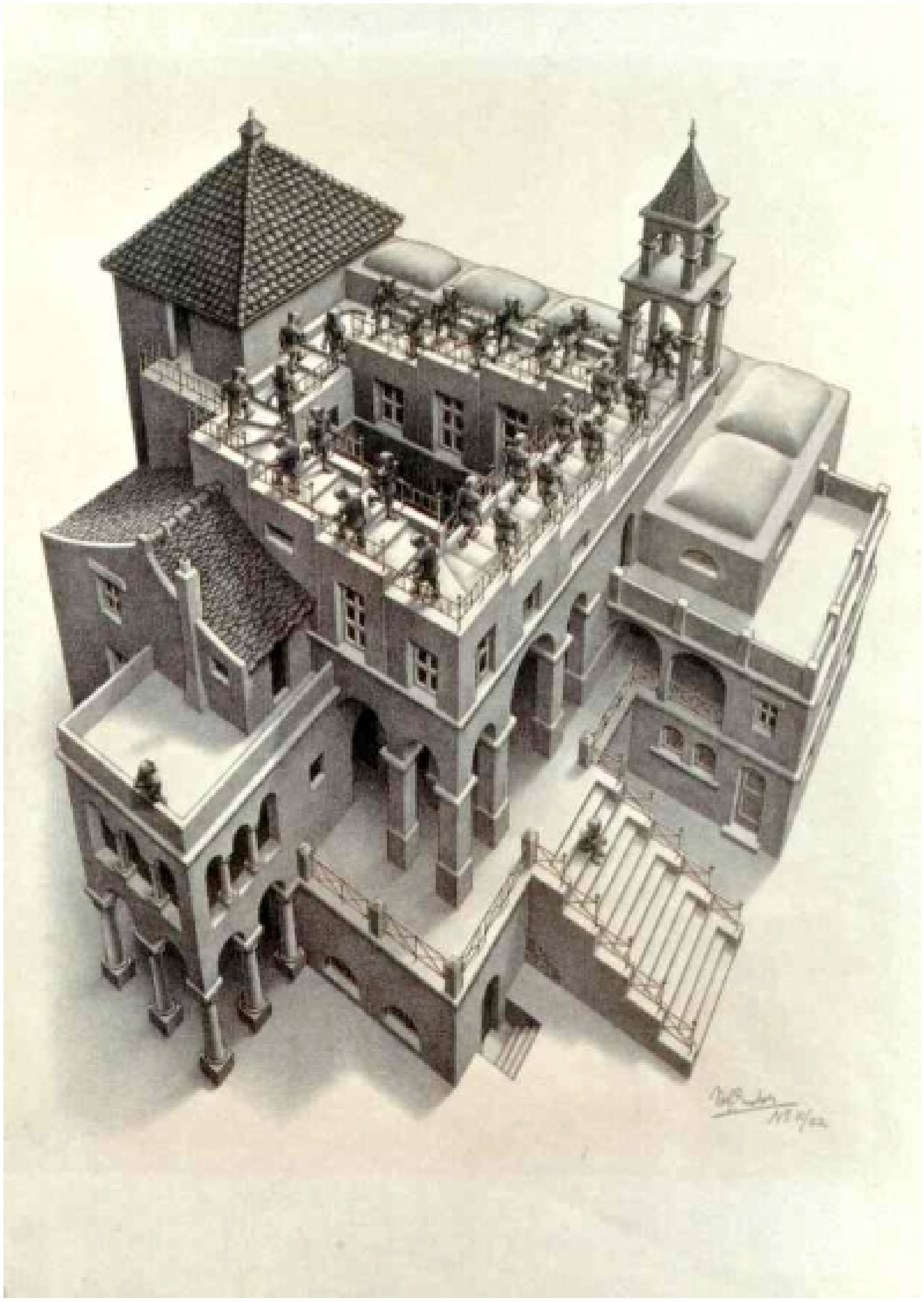

Optimal Allocation of Resources among Disjoint Sets of Discrete and Continuous Activities

A Dissertation Presented

by

Georgios Konstantinou Kozanidis

to

The Department of Mechanical, Industrial and Manufacturing Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the field of

Industrial Engineering

Northeastern University

Boston, Massachusetts

August 19, 2002

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Dissertation Title: Optimal Allocation of Resources among Disjoint Sets of Discrete and Continuous Activities

Author: Georgios Konstantinou Kozanidis

Department: Mechanical, Industrial and Manufacturing Engineering

Approved for Dissertation Requirement of the Doctor of Philosophy Degree

Dissertation Advisor, Dr. Emanuel S. Melachrinoudis	Date
Committee Member, Dr. Marius Solomon	Date
Committee Member, Dr. Waleed M. Meleis	Date
Department Chair, Dr. John W. Cipolla Jr.	Date
Director of the Graduate School, Dr. Yaman Yener	Date

To my family:

My father Konstantinos,

my mother Theodora,

my sister Ioanna

and my nephew Tasos

"The roots of education are bitter, but the fruit is sweet." **Aristotle**

"The secret to success is to know something nobody else knows." **Aristotle Onassis**

"The important thing is not to stop questioning." **Albert Einstein**

"When you are courting a nice girl an hour seems like a second. When you sit on a red-hot cinder a second seems like an hour. That's relativity." **Albert Einstein, On relativity**

"Don't be afraid to take a big step. You can't cross a chasm in two small jumps."
David Lloyd George

"To steal ideas from one person is plagiarism, to steal ideas from many is research."**Anon**

"Education is the ability to listen to almost anything without losing your temper or your self-confidence." **Robert Frost**

"The harder you work, the luckier you get." **McAlexander**

"We aim above the mark to hit the mark." **Ralph Waldo Emerson**

"Even if you are on the right track, you'll get run over if you just sit there." **Will Rogers**

"Success will not lower its standard to us. We must raise our standard to success."
Rev. Randall R. McBride, Jr.

"Creative minds have always been known to survive any kind of bad training."
Anna Freud

"Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it." **Samuel Johnson**

TABLE OF CONTENTS

Preface

This dissertation was prepared at the Department of Mechanical, Industrial and Manufacturing Engineering at Northeastern University between September 1998 and August 2002, under the supervision of Professor Emanuel Melachrinoudis. The other two members of my committee were Professor Marius Solomon and Professor Waleed Meleis.

The involvement in this area of research was motivated by several discussions with Professor Melachrinoudis who gave me the first flavor of the potential for research and the interesting applications in which this research can be applied. We decided to pursue this idea and this led to the work presented in this dissertation. During the last two years, Professor Solomon got actively involved in the research too, which provided us valuable support in many ways.

The dissertation consists of several self-contained independent parts. The main application of this research is presented in a paper that was published in the *International Journal of Transportation Research, Part A: Policy and Practice.* The other parts will be submitted soon but at the present their fate is unknown.

The most important model when dealing with resource allocation applications is by far the Knapsack Problem. As a result, most of the models developed for this type of problems depend heavily on the extensive theory that has been developed for the Knapsack Problem. This is also true for the present work. Many people working with Knapsack Problems have been fascinated by the work of the famous Dutch lithographic artist M. C. Escher. Martello and Toth, two of the pioneers in the field used the lithography “Relativity” as a cover of their book on Knapsack Problems. Pisinger, a more recent but definitely as brilliant a researcher, used the woodcut “Smaller and Smaller” to cover his dissertation. I chose the lithography “Ascending and Descending” (1960) by the same artist as a cover for mine, to reflect the fact that many times in Knapsack Problems an appropriate ordering of elements is necessary in order to address the problem under investigation. This has also been the case with the present work. The following is a small fragment that appears in the book “M. C. Escher – The Graphic Work” and refers to this lithography.

“The endless stairs which are the main motif of this picture were taken from an article by L.S. Roger Penrose in the February, 1958 issue of the British Journal of Psychology. A rectangular inner courtyard is bounded by a building that is roofed in by a never ending stairway. The inhabitants of these living-quarters would appear to be monks, adherents of some unknown sect. Perhaps it is their ritual duty to climb those stairs for a few hours each day. It would seem that when they get tired they are allowed to turn about and go downstairs instead of up. Yet both directions, though not without meaning, are totally useless. Two recalcitrant individuals refuse, for the time being, to take any part in this exercise. They have no use for it at all, but no doubt sooner or later they will be brought to see the error of their nonconformity.”

Boston, MA

August 2002

Acknowledgements

First of all, I would like to thank my dissertation advisor, Professor Melachrinoudis. He was the person who introduced me into this area of research and helped me pursue my ambitions. His ideas and suggestions improved tremendously the work presented in this dissertation. He always stood by me to help with all types of problems that came up. He motivated me and encouraged me in so many different ways and without him this work would never have been possible.

Next, I would like to thank my committee member Marius Solomon. Professor Solomon showed extensive interest in the research that we wanted to pursue from the very first moment that we presented it to him. His enormous experience improved this work in many different aspects. His help with the revision and the editing of the dissertation is priceless. I feel extremely lucky to have him as my committee member and I am confident that any other student in my position would feel the same.

Special thanks are also in order for a number of different people. Thanks to the third member of my committee, Professor Meleis for his help with the finalization and the approval of this dissertation. Professor Meleis kindly agreed to serve as a committee member although he was asked on a very short notice. I really appreciate his warm response. Thanks to all the Professors of the Department for contributing enormously to the enhancement of my knowledge in the fascinating field of Industrial Engineering. Thanks to the Department itself for providing me with all the valuable support (financial too) throughout these four years, which made possible the completion of this dissertation. Additionally, thanks to Northeastern University for making the last four years the most productive and creative years of my life so far. This period was an amazing experience for me and I will always feel grateful when looking back at it.

Finally, I would like to thank my family for giving me the most essential element for completing this work, their love. They gave meaning to what I was doing and helped me achieve my goals. I hope that some day I will be able to return a small part of what they have done for me. More specifically, I thank my parents Konstantinos and Theodora who were always next to me when I needed them. They supported me in so many different ways and were always willing to sacrifice everything if this were for my own good. I thank my sister Ioanna who showed me her love throughout my years here at Northeastern University. It was a tremendous relief knowing that she is always standing by to help me with all my concerns. Finally, I thank my nephew Tasos for his continuous love and support.

George Kozanidis

ABSTRACT

Optimal resource allocation has been a very active area of research in mathematical programming for many years. The problem arises in a large number of different situations, with many different forms. As a result, numerous papers have been published in the related literature, dealing with its various aspects.

The so-called Knapsack Problem is a big family of problems that have been formulated in order to address the numerous variations of the problem that arise in practice. The Knapsack Problem has been the basis for most of the resource allocation models that have been developed. Besides the tremendous theoretical interest that the problem enjoys, the practical applications in which it can be applied are countless. As a result, it continues to stimulate the interest of many researchers, in spite of the fact that it has been studied extensively in the past.

In this dissertation, several variations of this problem, with special constraints, are addressed. Both discrete and continuous decision variables are considered, depending on the nature of the activities that these variables represent. These activities (and therefore the variables representing them too) are partitioned into disjoint sets. A new special type of constraints called equity constraints is introduced that ensures a certain balance on the resource amounts allocated to different activity sets. Special constraints called multiple choice constraints are also included that handle the interactions that arise between the continuous activities of the problem.

The main application of the problems addressed in this dissertation is in transportation management for optimal allocation of funds to highway improvements. The decision variables represent highway improvements that can be applied to the

highways under consideration. These highways are partitioned into disjoint segments. Each of the disjoint variable sets corresponds to a set of improvements associated with a highway segment. The objective is to allocate an available budget in order to optimize some appropriate measure of effectiveness. The equity constraints are used to keep a certain balance on the budget amounts allocated to different highway segments.

To the best knowledge of the author, the problems addressed in this dissertation have not been studied in the past. For each of these problems, original theoretical groundwork and important properties are developed that provide valuable insight. Then, based on this theory, efficient algorithms are developed that can be used to obtain the optimal solution of each problem. Besides analyzing the complexity of each of these algorithms, computational results are presented that show their behavior and compare their performance with the performance of commercial software packages that can be used alternatively. The importance and the sensitivity of the various parameters of each of the problems addressed is also investigated thoroughly. This provides important insight that can be very useful in future research. The dissertation concludes with a discussion on the conclusions reached and on how this work can be extended in the future.

List of Figures

List of Tables

“Not everything that can be counted counts, and not everything that counts can be counted.” **Albert Einstein**

“The best way to escape from a problem is to solve it.” **Alan Saporta**

“Give me a lever long enough and a fulcrum on which to place it, and I shall move the world.” **Archimedes, Pappus of Alexandria**

“The brighter you are, the more you have to learn.” **Don Herold**

Chapter 1: Introduction

1.1 Optimal Resource Allocation

Optimal resource allocation has been a very active area of research in mathematical programming for decades. The objective is to distribute a number of limited resources among a set of activities in order to optimize some appropriate measure of effectiveness. It is not an overstatement to say that optimal resource allocation is one of the problems engineers are most often faced with. According to Hillier and Lieberman (2001), the field of Operations Research was born during World War II in an effort to allocate scarce resources to the various military operations and to the activities within each of these operations in an effective manner.

When we think of the term resource in its broad use, it becomes clear that the problem can arise in a very large number of different situations. In production analysis for example, the objective is to decide how to utilize the available production facilities in order to optimize some target objective while not violating the physical constraints imposed. Another example is encountered in investment analysis where the objective is to select from a set of proposed alternatives those that will maximize the return from the utilization of a limited budget. As a result of the large and diverse applicability of the problem, numerous papers have been published in the related literature, dealing with its various aspects. This comes also as a result of the vast theoretical interest that the problem stimulates.

1.2 The Knapsack Problem

The rapid progress in the area of optimal resource allocation is mostly due to the development and analysis of the so-called Knapsack Problem. The Knapsack Problem is a large collection of problems that effectively and lucidly have been used to formulate most of the resource allocation problems that arise in practice. In its simplest form, the problem is formulated as follows:

$$\text{Max } p_1x_1 + p_2x_2 + \ldots + p_nx_n$$

$$\text{s.t. } c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq b$$

$$x_1, x_2, \ldots, x_n \text{ binary}$$

The decision variables $x_1, x_2, \ldots, x_n$ represent the various activities considered for implementation. Parameter b represents the available amount of resource for the implementation of these activities. Coefficients p_i and c_i represent the profit and the cost, respectively, incurred from the implementation of activity x_i. The term "knapsack" originates from the fact that this can be visualized as the problem of trying to maximize the total value of a subset of n items inserted in a knapsack without exceeding the knapsack's weight capacity, b. In that context, p_i and c_i represent the value and the weight of item i, respectively, and x_i takes the value 1, if item i is inserted in the knapsack, and 0, if not.

The above representation is quite general. The decision variables may represent many different types of entities, such as people appointed to different departments of some organization, projects considered by some authority, or even production plans in some manufacturing facility. Similarly, the resource, b, may have numerous representations such as money, time, machines, etc. In this work, the words budget and resource are used interchangeably to denote this parameter.

The above formulation can be extended in numerous ways. When several resources b_1, b_2,…, b_k are considered instead of the single resource, b, the Multi-Dimensional Knapsack Problem arises. In the Bounded Knapsack Problem, each activity can be selected at most a certain number of times instead of only once. In the Multiple Choice Knapsack Problem, the activities are partitioned into disjoint sets and at most one activity from each set can be selected.

The Knapsack Problem is NP-hard (Garey and Johnson, 1979). This means that no polynomial algorithm has been developed for its solution so far and additionally, that it seems very unlikely that this will ever become possible. As a result, researchers have focused on developing efficient algorithms that take advantage of the special structure of the problem. The published results indicate that this attempt has been very successful and very promising for the future. This is one of the main reasons that the problem has attracted the interest of so many theoretical researchers. Three are the most popular techniques that have been used to find exact solutions to the problem:

- dynamic programming
- branch and bound
- cutting planes

A large number of other techniques have also been used to get approximate solutions to the problem, such as approximation algorithms, heuristics, etc.

One of the problems addressed in this work is formulated as a linear program. For this linear knapsack problem, a greedy solution procedure is developed having polynomial complexity. The remaining problems are NP-hard. The algorithms developed for those problems incorporate in some way a branch and bound procedure and their worst case performance is exponential.

1.3 Problem Formulation

In its general form, the formulation of the problem that includes all the individual subproblems considered in the various parts of this dissertation is as follows:

$$
\begin{aligned}
\text{Max} \quad & \sum_{k\in S} \sum_{i\in R_k} p_{ki}x_{ki} + \sum_{k\in S} \sum_{j\in D_k} q_{kj}y_{kj} && (1)\\
\text{s.t.} \quad & \sum_{k\in S} \sum_{i\in R_k} c_{ki}x_{ki} + \sum_{k\in S} \sum_{j\in D_k} d_{kj}y_{kj} \le b && (2)\\
& \sum_{i\in R_k} x_{ki} \le l_k \;, \forall\, k \in S && (3)\\
& L \le \sum_{i\in R_k} c_{ki}x_{ki} + \sum_{j\in D_k} d_{kj}y_{kj} \le U \;, \forall\, k \in S && (4)\\
& U - L \le f && (5)\\
& x_{ki} \ge 0, i \in R_k, \forall\, k \in S && (6)\\
& y_{kj} \in \{0,1\}, j \in D_k, \forall\, k \in S && (7)
\end{aligned}
$$

The decision variables x_{ki} and y_{kj} belong to disjoint sets. The specific set a variable belongs to is denoted by its first index, k. Set S contains the indexes of all these disjoint sets of decision variables. For a specific value of k in S, sets R_k and D_k contain the indexes of the continuous and binary variables, respectively, that belong to set k. Parameters p_{ki} and q_{kj} are called profit coefficients and parameters c_{ki} and d_{kj} are called cost coefficients. In the present work, all these coefficients are positive numbers.

Parameter b (budget or resource) is also a positive quantity, denoting the available amount of resource for the implementation of the activities represented by the variables x_{ki} and y_{kj}.

In the objective function of the problem, the total profit incurred from the selection of the decision variables is maximized. The first constraint is a typical resource constraint, stating that the total amount of resource that can be used for all activities cannot exceed the available quantity, b. The sum of all continuous variables within each set k is restricted to at most l_k by the set of constraints (3), which are called multiple choice constraints. The reason these constraints are introduced is illustrated in Chapter 2, where the specific area of application of the above model is presented. For a given k in S, $\sum_{i\in R_k} c_{ki}x_{ki} + \sum_{j\in D_k} d_{kj}y_{kj}$ is the total resource amount allocated to set k. This quantity is called the cost of set k. The equity constraints, defined by constraints (4) and (5), ensure that the cost of every set at the optimal solution of the problem belongs to an interval $[L,U]$ whose width cannot exceed a given value, f. The auxiliary decision variables L and U are used to denote the two endpoints of this interval. Finally, constraints (6) and (7) restrict variables x_{ki} to nonnegative values and variables y_{kj} to $\{0,1\}$ values, respectively.

1.4 Motivation of the Research

The research work presented in this dissertation was motivated by the need to address a number of interesting problems that arise in various application areas and haven't been studied in the past. Each of these problems and the context in which it is encountered are presented next. This presentation explains in detail how this work was stimulated.

As it will be explained in Chapter 2, the various problems addressed in this dissertation can be applied in transportation management for optimal allocation of funds to highway improvements. The number of different approaches that have appeared in the related literature for the treatment of this problem is limited. Moreover, only discrete improvements have been considered in previous papers. Besides these discrete improvements, however, there are also improvements that can be applied continuously over a section of a highway. Typical examples are pavement resurfacing or lighting. The modeling of such improvements cannot be accomplished with discrete variables. Continuous variables are needed instead.

The present research arose partly from the need for the development of a rational approach for the treatment of this problem. The incorporation of both continuous and discrete improvements into the model results in a mixed integer formulation, which is not very typical. As a result, a number of issues arise from this incorporation that require investigation. A typical example is the modeling of the interaction that arises between different improvements.

The present research was also motivated by the linear and integer optimization theory that has been developed for knapsack type problems. Although the research in this area is quite rich and continuous, most of it focuses on pure discrete applications of the problem. As a result, the great majority of the related papers deals with problems in which only discrete variables are considered. Problems with continuous variables are usually used as a linear relaxation to their discrete counterparts. In reality, however, there are a number of different occasions, where both continuous and discrete variables are necessary to represent the activities that are present. In that case, a mixed integer optimization problem arises. The research in this type of problems is quite limited.

In this work, a meaningful representation for the continuous decision variables of the problem is introduced. Then, the continuous and the discrete variables are combined in a meaningful mixed integer optimization problem. With the algorithms developed for these problems, it is also shown how efficient algorithms can be developed for similar mixed integer problems. The possibilities for research in this direction are numerous.

This work was also motivated by the need to address new optimization problems with practical applications that haven't been addressed before. As already mentioned, the problems addressed in this dissertation have not been investigated in the past. Besides introducing the general formulation and showing how each of these problems can be applied in practice, their important properties and efficient algorithms for their solution are also developed.

Finally, the present work was motivated by the need to address resource allocation related issues that have not been addressed in the past. One such issue is the need for equity when allocating resources among disjoint sets of activities. Decision makers for example, often seek a balance on the resource amounts allocated to different activity sets. These sets may be associated with different geographical regions, groups of people, or even departments within the same organization. The previous research on this subject is very limited. The incorporation of equity constraints raises a number of interesting theoretical questions such as what is the impact on the objective function value and on the behavior of the problem.

1.5 Research Summary

In the present dissertation several special subproblems that arise from the formulation that was introduced in Section 1.3 are considered. Among others, the corresponding application is presented and a number of important properties for each of these problems are developed. Besides focusing on each problem separately, the similarities and differences that arise between them are also investigated. This is a very important part of the research because it shows how the theory developed is modified when moving from one problem to another.

Next, a brief description of each of the problems addressed in this dissertation is presented. The algorithmic approach adopted is outlined and the results and the conclusions obtained from the study of each of them are presented. The first of these problems, introduced next, is very significant for our later algorithmic development. It provides the groundwork, on which a great part of the theory developed in this dissertation is based.

1.5.1 The Linear Multiple Choice Knapsack Problem

The problem arises when the binary variables and the equity constraints are ignored from the model introduced in Section 1.3. It consists of only continuous variables, partitioned into disjoint sets. The sum of the variables in each set is restricted to a maximum value, which is specific for each set.

The Linear Multiple Choice Knapsack Problem has been extensively studied in the past and numerous algorithms have been developed for its solution as reported in the related literature. As already pointed out, the main reason this problem is revisited is because its main structure provides important insight for the theory developed in later sections of this dissertation.

First, two important properties are presented, which have been the basis for the development of most algorithms that handle this problem. After presenting some theoretical results, a slight modification to an existing algorithm for the problem is presented. This modification was necessary in order to address the needs of the problems investigated in this dissertation. Some discussion on this modified algorithm follows and the differences that it has from the original algorithm are pointed out. Some new properties resulting from the study of this modified algorithm are also developed. These properties are very important for the theory developed in later chapters.

1.5.2 The 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice Constraints

This problem arises when the equity constraints are ignored from the model introduced in Section 1.3. It consists of both continuous and binary variables, with multiple choice constraints applied to the continuous variables. The problem involves both the Linear Multiple Choice Knapsack Problem and the 0-1 Mixed Integer Knapsack Problem.

Two important propositions are developed, which lead to the construction of an efficient branch and bound algorithm for this problem. The first one follows from the properties of its linear programming relaxation. It takes advantage of the special relationship that this problem has with the Linear Multiple Choice Knapsack Problem.

The second proposition explores the special relationship between the parent and children nodes of the branch and bound tree. This result is used to improve the performance of the algorithm.

Based on the theory developed for this problem, the significant advantages of the algorithm are presented. The superior performance of the algorithm is demonstrated by the computational results presented at the end of the corresponding chapter. Some discussion on the conclusions reached by the analysis of these results is also included. The chapter ends with a summary of the conclusions obtained. Some comments on possible extensions of this work are contained in Chapter 8.

1.5.3 The Linear Multiple Choice Knapsack Problem with Equity Constraints

This problem differs from the traditional Linear Multiple Choice Knapsack Problem in the equity constraints that ensure that the cost of each set belongs to an interval whose width does not exceed a certain value. In the past, the Linear Multiple Choice Knapsack Problem has mostly been used as a relaxation to the Binary Multiple Choice Knapsack Problem. In this work, a new important application in which this linear problem can be used is presented.

Important theory is developed for this problem and a number of interesting properties are proven. The similarities and the differences that arise from the incorporation of the equity constraints are pointed out. Then, based on this theory, an efficient greedy solution algorithm is developed.

The algorithm consists of two phases. In its first phase, the algorithm finds the optimal solution to the relaxed problem that results when the equity constraints are dropped. In the second phase, the equity constraints are incorporated. Then, starting from the current solution and while maintaining superoptimality, the algorithm works towards feasibility until the optimal solution is obtained. This is achieved by successively decreasing the width of the interval containing the costs of the disjoint variable sets in the least costly way. As soon as the equity constraints are satisfied, the solution at hand is optimal.

Besides presenting computational results demonstrating the efficiency of this algorithm, a number of interesting results obtained from the analysis of the experiments conducted are also presented. These results provide insight on the significance of the equity constraints. They show how the incorporation of these constraints affects the computational effort needed to solve the problem and additionally, what the resulting impact on the optimal objective function value of the problem is.

1.5.4 The 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice and Equity Constraints

This is exactly the problem introduced in Section 1.3. It includes all the individual subproblems that have been introduced so far, i.e., both binary and continuous variables, linear multiple choice and equity constraints. As before, the idea is that the decision variables are partitioned into disjoint sets and the maximum difference between the resource amounts allocated to any two sets should not exceed a given value.

Two different versions of this problem are considered. In the first one, the interval containing the optimal costs of the sets is known. In this case we only have to maximize the total profit, while making sure that the total resource amount used does not exceed the available budget, *b*, and that the cost of each set belongs to the given interval. The second version differs from the first one in that this interval is not known. This adds significant difficulty to the problem. This difficulty results from the fact that this interval now needs to be identified and becomes evident when we consider the infinite options that exist for this interval's exact location.

Based on the special structure of the problem, a branch and bound algorithm for the first version of the problem is developed. Then, this algorithm is extended to accommodate the second version of the problem too. Computational results for both of these algorithms are presented and their performance is compared with that of a commercial software package that can be used alternatively. The effect of the equity constraints is also investigated. Finally, the behavior of the two algorithms is studied and a discussion on how this behavior changes when the main parameters of the problem are modified is included.

1.6 Contributions of the Research

The present research provides significant contributions to various areas such as transportation, resource allocation, Knapsack Problems, linear and mixed integer programming.

The first contribution of this work is in transportation management and becomes evident with the details for the application of the problem, introduced in the next chapter. With the present work, a rational model is presented that can be used for addressing the problem of allocating funds to highway improvements. The formulation is quite general and can be extended easily. The model is very flexible, since both continuous and discrete improvements can be taken into account and different highway segments can have different choices of improvements. Additionally, a straightforward technique is presented that can handle the interactions that arise between the continuous improvements of the model. The introduction of the multiple choice constraints is a result of the application of this technique.

The second major contribution of this research is in the general area of Knapsack Problem applications. A specific application is presented in which a continuous variation of the problem can be used. Then, this is combined with a discrete version of the problem and it is shown how the resulting mixed integer problem can be used meaningfully. As it was already mentioned, the research and the published examples in this area are quite limited.

The third major contribution of this research is in linear and mixed integer programming and becomes evident from the theory and the solution algorithms developed. The efficiency of these algorithms is superior to the performance of common optimization software packages. Therefore, using these algorithms, problems of large size can be solved faster. Using the developed theory, the performance of these algorithms could probably be further improved in the future.

Another contribution of the present research is the insight gained from the analysis of the computational results. The special structure of the various problems is investigated thoroughly and their behavior is explained from the analysis of these results. Additionally, the significance of the various parameters of these problems becomes apparent. For example, the impact resulting from the incorporation of the equity constraints is investigated.

1.7 Outline of the Dissertation

The remainder of this dissertation is structured as follows. In Chapter 2 the transportation management application, in which the problems addressed can be utilized, is presented in detail. The model's importance becomes apparent from this presentation. The model's assumptions and requirements are also listed and a case study illustrating its application is presented.

Chapter 3 contains the literature review for all the various parts of this dissertation. Relevant work that has been done in the past and is used here is summarized. First, the work that has been published in the transportation literature for optimal highway improvement project selection is presented. Then, the attention turns to the theoretical aspect of the problem. A literature review of the work that has been done for the Knapsack Problem is provided. Then, focus shifts to the Linear Multiple Choice Knapsack Problem, which is used in many parts of this dissertation. The algorithms that have been developed both for the continuous and the discrete version of the problem are presented.

Chapter 4 is engaged with the Linear Multiple Choice Knapsack Problem. First the fundamental properties of the problem are presented and then an existing algorithm that can be used for its solution is slightly modified. This is necessary in order to accommodate the needs of the problems addressed later in this dissertation. Some discussion on the theoretical aspects of the algorithm follows and the differences that it has from the original algorithm are pointed out. The chapter concludes with some additional properties of the problem and a discussion of the related theory.

In Chapter 5 the 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice Constraints is treated. The development of some important propositions leads to the development of an efficient branch and bound solution algorithm for this problem. Several computational issues related to this algorithm are discussed and computational results that demonstrate its efficiency are presented. The chapter ends with the conclusions obtained from the analysis of these results.

In Chapter 6 the attention turns to the Linear Multiple Choice Knapsack Problem with Equity Constraints. After introducing the problem, the theory that has been developed for the traditional Linear Multiple Choice Knapsack Problem is extended. Some additional properties and an efficient greedy solution algorithm are developed. The chapter concludes with a discussion on the computational complexity of the algorithm and the presentation of computational results and the conclusions obtained from their analysis.

In Chapter 7 the 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice and Equity Constraints is treated. This is the aggregate problem introduced in Section 1.3. Two versions of the problem are considered and a branch and bound algorithm for each of them is designed, based on the important theory developed. As before, several computational issues are discussed and computational results obtained from computer experimentation are presented and analyzed. Then, the interpretation of these results and the conclusions obtained are presented.

In Chapter 8 the possible extensions of this research are presented. The various transportation models that arise from the models presented in this dissertation are discussed. These models incorporate additional features of the problem. Additional knapsack problems arise this way, which are natural extensions of the problems studied in this dissertation and provide a suitable area for future research. The approaches that can be taken for tackling these problems are mentioned briefly. Finally, in Chapter 9 the important results obtained from this work are outlined and the important merits gained and the interesting conclusions reached from this work are summarized.

“In the fields of observation chance favors only the prepared mind.” **Louis Pasteur**

“Education is not the filling of a pail, but the lighting of a fire.” **William Butler Yeats**

“Thinking: The talking of the soul with itself.” **Plato**

“He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.” **Chinese Proverb**

Chapter 2: The Transportation Management Application

2.1 Introduction

Highway improvement programming has been a major concern of transportation agencies for many years. The involvement of human lives gives to the related problem a tremendous significance that demands special attention. As a result, a great part of the federal funding in the United States is directed towards the maintenance and improvement of the condition of major highways, in an effort to reduce the potential loss and increase the level of public safety. A total of $100 billion of federal funds were spent to fix interstate highways and other major roads in the 1990s. As a percentage of the total federal transportation spending, spending to improve roads and bridges increased annually from 39% in 1990 to 49% in 1998 (STPP, 2000).

Optimal allocation of funds to highway improvements is a complicated and often tedious task. The large number of competing alternatives requires a methodology for prioritizing the recommended projects, so that the return from the utilization of a limited budget is maximized. Despite the numerous approaches that have appeared in the literature, the complexity of the problem in association with the difficulty in estimating some of its primary parameters often result in non-optimal decisions (Donaldson, 1988). A rational method for the treatment of the problem becomes increasingly necessary, as indicated by the rapid increase in the number of registered vehicles on the one hand and the unavoidable deterioration of the highways on the other.

Given a set of highways and a set of recommended improvements for each of them, the problem is to allocate an available budget among these highways, in order to optimize some appropriate measure of effectiveness. A commonly used measure of effectiveness is the reduction in the expected number of accidents, based on the fact that the frequency of accidents is directly related to the frequency of highway fatalities and injuries. Although transportation fatalities declined during the last decade, auto accidents remain the leading cause of accidental death among Americans and the leading cause of death overall for people ages 1 to 34 (STPP, 2000). The top goal of the U.S. Department of Transportation, which appropriates the largest share of the funding for highway improvements, remains the increase of transportation safety (U.S. DOT, 2001).

2.2 The Highway Improvement Model

All papers in the highway improvement programming literature thus far deal with discrete improvements. A discrete improvement is associated with a particular intervention at a specific point of the road, for which we have only two choices, implementation or not. The smoothing of a dangerous curve or the repair of a bridge are examples of discrete improvements. The mathematical representation of these improvements is thus accomplished by binary variables.

In Section 1.3 a new model was introduced that can be used for addressing the problem of allocating funds to highway improvements. In addition to the commonly used binary variables, this model includes continuous variables for improvements that can be implemented continuously over parts of the highways under consideration. For example, the quality of the pavement can be improved continuously in any subsection of a highway and therefore it can be categorized as a continuous improvement. The length of the highway over which a continuous improvement is carried out is represented by a continuous variable. Interactions that may arise from the implementation of more than one continuous improvements on the same part of a highway are explicitly considered as explained shortly.

For the needs of the model, every highway is divided into segments, each of which has uniform characteristics. This simply means that the profit and the cost of a continuous improvement does not change throughout the length of a highway segment. The cost of a continuous improvement is proportional to the length of the highway over which it is applied with a constant of proportionality being the unit cost that depends on the particular improvement. On the other hand, the cost of a discrete improvement is fixed. The same distinction is made for the profit of a continuous versus a discrete improvement. Thus, the formulation of Section 1.3 results, in which the following notation is used:

S = set of indexes of highway segments,

R_k = set of indexes of continuous improvements, considered in segment k,

D_k = set of indexes of discrete improvements, considered in segment k,

p_{ki} = unit profit of continuous improvement i of segment k,

c_{ki} = unit cost of continuous improvement i of segment k,

q_{kj} = profit of discrete improvement j of segment k,

d_{kj} = cost of discrete improvement j of segment k,

l_k = length of segment k,

b = available budget,

x_{ki} = length of highway segment k over which continuous improvement i is implemented,

y_{kj} = binary variable that takes the value 1, if discrete improvement j of segment k is implemented, and 0, if not.

The formulation of Section 1.3 is interpreted now in the context of highway programming. In the objective function (1) the total profit incurred from the implementation of the considered improvements is maximized. Constraint (2) ensures that the amount of money spent on all selected improvements will not exceed the available budget, *b*. Constraints (3) are called multiple choice constraints; They are introduced to handle the interactions that arise between continuous improvements as explained in the next section. Constraints (4) and (5) are the equity constraints. They ensure that the cost allocated to each highway segment belongs to an interval whose width is at most *f*. Constraints (6) restrict the lengths of continuous improvements to nonnegative values. Finally, constraints (7) restrict variables y_{kj} that represent the implementation or not of discrete improvements to {0,1} values.

The model can treat several variations of the problem. The user has the flexibility to choose his/her decision criterion based on both the priorities that he/she sets as well as the data that are available. Therefore, different factors reflecting the quality of transportation can be reflected in the objective function. For example, a surrogate objective of the quality of transportation from the point of view of the motorists could be the reduction in the total cost incurred to their vehicles through highway improvements. This cost reduction includes reduction in car wear and tear, reduction in repair expenses and increase in fuel economy. It is estimated that poor roads cost drivers in U.S. metropolitan areas alone $5.8 billion per year (STPP, 1998).

The same type of improvement may have different indexes for different highway segments. This representation is necessary because the same type of improvement may have different cost and/or profit coefficients for different highway segments. On the other hand, it allows different segments to have different choices of continuous improvements.

2.3 The Multiple Choice Constraints

The multiple choice constraints (3) were motivated by the need to model the interactions that arise between different continuous improvements within the same segment. In some cases, the cost needed to apply two or more distinct continuous improvements over the same part of a highway segment may be higher or lower than the sum of the costs of these improvements considered independently. For example, total cost can decrease when resources are being shared. In a similar manner, the actual profit from the application of two or more improvements over the same part of a highway segment may be lower or higher than the profit computed when these improvements are treated independently. In general, the combined return is expected to be lower except in the case of synergy. The procedure described next is used to handle the case that such interactions are present and motivates the introduction of the multiple choice constraints.

For each combination of continuous improvements within a segment, another variable is defined that represents the length of this segment over which all these improvements are applied. The profit and cost of this new variable is equal to the actual profit and cost incurred when all the improvements that are associated with this variable are implemented. Suppose that three continuous improvements can be implemented on highway segment *k*. Then, the following variables are defined: Variable x_{k1}, x_{k2} and x_{k3}, represents the length of highway segment *k* over which single improvement 1, 2 and 3 is implemented, respectively. Three combined variables are introduced representing segment lengths over which two improvements are implemented, i.e. 1 and 2 (x_{k4}), 1 and 3 (x_{k5}), and 2 and 3 (x_{k6}). Finally, a combined variable is introduced that represents the segment length over which all three improvements are implemented (x_{k7}). Thus, seven variables should be used in this case, of which three represent individual improvements and four represent combined improvements. The sum of all variables cannot exceed the length of highway segment k.

Conversely, any problem solution should be interpreted appropriately, according to the definition of the variables. If in the case described above for example, the length of segment k is 2.7 miles and $x_{k1} = 1.5$, $x_{k2} = 1.2$, $x_{k4} = 0$, then improvements 1 and 2 should not overlap on this segment, in order to ensure that the length of combined improvements 1 and 2 (x_{k4}) is 0.

The above described technique for the treatment of interactions among continuous improvements will increase the number of decision variables of the model. Nevertheless, in practical situations this increase remains manageable, since the number of individual continuous improvements considered is relatively small. In addition, the algorithms developed in this dissertation are very efficient in handling continuous variables.

If a segment has m variables representing individual improvements, then the number of combined variables to be introduced for that segment is

$$\sum_{i=2}^{m} \binom{m}{i} = 2^m - m - 1,$$

which is simply the total number of i-combinations of these m continuous improvements, over all the values of i between 2 and m. Thus, if there exist r segments on which $N_1,\ldots,N_r$

individual continuous improvements are considered, respectively, then the number of combined variables to be introduced is

$$\sum_{i=2}^{N_1}\binom{N_1}{i}+\ldots+\sum_{i=2}^{N_r}\binom{N_r}{i}=2^{N_1}+2^{N_2}+\ldots+2^{N_r}-N-r,$$

where $N_1 +\ldots+ N_r = N$. This brings the total number of continuous variables (individual and combined) to $2^{N_1}+\ldots+2^{N_r}-r$.

2.4 Model Assumptions

A key assumption of the model is that discrete improvements are associated with different points of a highway. Therefore, no interaction between two discrete improvements exists. In addition, it is assumed that no interactions between discrete and continuous improvements exist. This is a reasonable assumption for discrete improvements that are applied to "idealized" points of a highway. For example, when considering the improvement of the pavement of a highway segment (continuous improvement) and the repair of an overhead bridge on that same segment (discrete improvement), it is reasonable to assume that these two actions will be independent of each other.

Interactions between continuous improvements are explicitly considered in each highway segment. For segments in which no interactions exist, we can simply skip the introduction of combined variables and constrain each of the continuous variables to take a value that is not larger than the length of the associated segment. In that case, each of these variables satisfies alone a constraint of the form $x_{ki} \leq l_k$ and thus the model structure remains unchanged.

In the objective function (1), the profit incurred from the application of a continuous improvement is proportional to its length. This is a reasonable assumption that seems to approximate many real life situations. For example, someone would have a strong point in claiming that the return when two miles of a highway are resurfaced is twice as big as when one mile is resurfaced. This proportionality assumption seems also very reasonable for modeling the costs of the continuous improvements, when setup costs can be ignored. Transportation agencies very often express such costs for continuous improvements per mile of implementation. Furthermore, this assumption keeps the model simple, gives the user better understanding of the problem and makes the acquisition of the necessary data easier. Despite the above arguments, it should be pointed out that this proportionality assumption may not always be true. When it's not, nonlinear expressions for the anticipated profit and cost will have to be derived and additional parameters will have to be estimated.

2.5 Application of the Model

In this section the application of the proposed model is illustrated in a small case study. Let's assume that a budget of \$400,000 is available for the improvement of the Massachusetts Turnpike (I-90) part between the Allston and Weston exits in the westward direction. The total reduction in the expected number of accidents is used as the measure of anticipated profit, based on the fact that the frequency of accidents is directly related to the frequency of highway fatalities and injuries. The considered section is divided into three segments with homogeneous characteristics. The lengths and the average daily traffic volumes for each of these three segments are shown in Table 2.1 (data obtained from MHD, 1994 and MTA, 1998).

Table 2.1: Characteristics of highway segments of the example

Segment	Index	Length (miles)	Avg. Daily Traffic (vehicles)
Allston – Newton	1	3.2	89001
Newton-West Newton	2	2.7	99644
West Newton – Weston	3	1.9	97606

A maximum difference of $50,000 is allowed between the budget amounts allocated to any two of the considered highway segments. Three continuous improvements (resurfacing, lane widening and shoulder widening) are considered for each of the three segments. It is assumed that these improvements have the same unit costs and accident reduction factors in each of the three segments, although in general this doesn't need to be the case. The notation x_{ki} is used for the length (miles) over which these continuous improvements are implemented, where k is the index of the associated segment and i is the index defining the type of improvement, with $i = 1$ corresponding to resurfacing, $i = 2$ corresponding to lane widening and $i = 3$ corresponding to shoulder widening. The data for each individual improvement as well as for each combination of them are given in Table 2.2. These data show that the profit and cost for the implementation of any combination of individual improvements are smaller than the sum of profits and costs of individual improvements, respectively. The profit of a continuous improvement (accidents/mile) is obtained by multiplying its accident reduction factor (accidents/million vehicles, mile) by the average daily traffic volume (million vehicles).

Table 2.2: Data for the continuous improvements of the highway example

Improvement	Continuous Variable for segment $k = 1, 2, 3$	Unit Cost ($/mile)	Accident Reduction Factor (accidents/million vehicles, mile)
1	x_{k1}	105,000	0.027
2	x_{k2}	42,000	0.065
3	x_{k3}	24,000	0.015
1 and 2	x_{k4}	132,000	0.082
1 and 3	x_{k5}	116,000	0.035
2 and 3	x_{k6}	51,000	0.069
1, 2 and 3	x_{k7}	143,000	0.092

For each of the three segments the following three discrete interventions are also considered:

1. Smoothing of a steep curve.
2. Improvement at the exit (signing, lighting and lane corrections), and
3. Repair and maintenance of an overhead bridge.

These discrete improvements are represented by binary variables y_{kj}, where k denotes the index of the segment and j denotes the type of improvement, as before. Their costs and accident reduction factors are given in Table 2.3. The profit of a discrete improvement (accidents) is obtained by multiplying its accident reduction factor (accidents/million vehicles) by the average daily traffic volume (million vehicles).

Table 2.3: Data for the discrete improvements of the highway example

Binary variable	Description	Cost ($)	Accident Reduction Factor (accidents/million vehicles)
y_{11}	Curve Smoothing	48,000	0.036
y_{12}	Exit Improvement	9,000	0.007
y_{13}	Bridge Improvement	8,000	0.004
y_{21}	Curve Smoothing	32,000	0.011
y_{22}	Exit Improvement	7,000	0.009
y_{23}	Bridge Improvement	8,500	0.007
y_{31}	Curve Smoothing	19,000	0.015
y_{32}	Exit Improvement	8,000	0.003
y_{33}	Bridge Improvement	5,000	0.006

The data in Tables 2.2 and 2.3 are based on results presented by Sinha and Hu (1985) and Skinner (1985) and were adapted appropriately to reflect the highway system under consideration. After the necessary processing and scaling of the data, the problem is formulated as follows according to the formulation of Section 1.3. Note that the objective function is expressed in 10^{-3} accidents, the continuous variables in miles and the budget in units of $\$10^5$.

$$\begin{aligned}\text{Max }\ & 2.403x_{11} + 5.785x_{12} + 1.335x_{13} + 7.298x_{14} + 3.115x_{15} + 6.141x_{16} + 8.188x_{17} + \\ & 2.69x_{21} + 6.477x_{22} + 1.495x_{23} + 8.17x_{24} + 3.487x_{25} + 6.875x_{26} + 9.167x_{27} + \\ & 2.635x_{31} + 6.344x_{32} + 1.464x_{33} + 8.004x_{34} + 3.416x_{35} + 6.735x_{36} + 8.98x_{37} + \\ & 3.204y_{11} + 0.623y_{12} + 0.356y_{13} + 1.096y_{21} + 0.897y_{22} + 0.697y_{23} + 1.464y_{31} + \\ & 0.293y_{32} + 0.586y_{33}\end{aligned}$$

s.t. $1.05x_{11} + 0.42x_{12} + 0.24x_{13} + 1.32x_{14} + 1.16x_{15} + 0.51x_{16} + 1.43x_{17} +$

$1.05x_{21} + 0.42x_{22} + 0.24x_{23} + 1.32x_{24} + 1.16x_{25} + 0.51x_{26} + 1.43x_{27} +$

$1.05x_{31} + 0.42x_{32} + 0.24x_{33} + 1.32x_{34} + 1.16x_{35} + 0.51x_{36} + 1.43x_{37} +$

$0.48y_{11} + 0.09y_{12} + 0.08y_{13} + 0.32y_{21} + 0.07y_{22} + 0.085y_{23} + 0.19y_{31} +$

$0.08y_{32} + 0.05y_{33} \leq 4$

$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} \leq 3.2$

$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{26} + x_{27} \leq 2.7$

$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{36} + x_{37} \leq 1.9$

$L \leq 1.05x_{11} + 0.42x_{12} + 0.24x_{13} + 1.32x_{14} + 1.16x_{15} + 0.51x_{16} + 1.43x_{17} +$

$0.48y_{11} + 0.09y_{12} + 0.08y_{13} \leq U$

$L \leq 1.05x_{21} + 0.42x_{22} + 0.24x_{23} + 1.32x_{24} + 1.16x_{25} + 0.51x_{26} + 1.43x_{27} +$

$0.32y_{21} + 0.07y_{22} + 0.085y_{23} \leq U$

$L \leq 1.05x_{31} + 0.42x_{32} + 0.24x_{33} + 1.32x_{34} + 1.16x_{35} + 0.51x_{36} + 1.43x_{37} +$

$0.19y_{31} + 0.08y_{32} + 0.05y_{33} \leq U$

$U - L \leq 0.5$

$x_{ki} \geq 0,\ k = 1,2,3,\ i = 1,2,\ldots,7$

$y_{kj}\ 0\ \{0,1\},\ k = 1,2,3,\ j = 1,2,3$

The optimal solution of the above problem is: $x_{12} = 3.2$, $x_{22} = 0.933$, $x_{26} = 1.77$, $x_{32} = 1.9$, $y_{12} = 1$, $y_{13} = 1$, $y_{22} = 1$, $y_{23} = 1$, $y_{31} = 1$, $y_{33} = 1$, with all the other variables equal to 0. Thus, the optimal decision is to widen the lanes in segments 1, 2 and 3 and to widen the shoulder for 1.77 miles in segment 2. Additionally, the curve in segment 3, the exits in segments 1 and 2, and the bridges in segments 1, 2 and 3 should be improved. The optimal solution suggests that the recommended improvements will reduce the daily number of accidents in this section of the Massachusetts Turnpike by 0.05338. According to this solution, $151,400 should be spent on the first segment, $144,800 on the second and $103,800 on the third. The maximum difference between the budget amounts allocated to any two sets is 151,400 - 103,800 = $47,600 which is less than $50,000. In the following chapters the focus is on algorithmic procedures that can be used for solving problems like the one above.

"Success always occurs in private, and failure in full view." **Anon**

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world." **Albert Einstein**

"Minds are like parachutes; they work best when open." **Lord Thomas Dewar**

"Defeat is not the worst of failures. Not to have tried is the true failure."

George E. Woodberry

Chapter 3: Literature Review

3.1 Introduction

In this chapter, previous work pertaining to all topics addressed in this dissertation is summarized. First, the published work that deals with the problem of highway improvement programming and highway project fund allocation is reviewed. Then the attention turns to the algorithmic portion of the dissertation. A review of the extended work that has been done in the past for the Knapsack Problem is provided and then the focus shifts to the published work that deals with the Multiple Choice Knapsack Problem, which is used extensively in this work.

3.2 Highway Improvement Literature

Typical techniques for highway improvement projects' selection that have been reported in the transportation literature include dynamic programming (Brown, 1976 and 1980) and incremental benefit-cost analysis approaches (Farid et al., 1994). Sinha et al. (1981) suggested a procedure where the reduction in the expected number of accidents is used as the measure of effectiveness and binary variables are utilized to represent the various improvement alternatives. Following this procedure, the problem is formulated as a binary integer optimization model. Among others, the paper presents a multi-year model with the flexibility of carry-over of unspent funds from previous years. A stochastic version of the model that examines the case in which there are uncertainties in estimating model parameters is also introduced. Pal and Sinha (1998) extended this work by considering the effectiveness of the various projects in future years. Their model takes into account the expected growth in traffic resulting from the implementation of the recommended projects. As in the paper by Sinha et al. (1981), the problem is formulated as a binary integer optimization model where the total crash rate is minimized. Another binary integer optimization model was used by Melching and Liebman (1988) to address the problem of allocating railroad maintenance funds. The problem was modeled as a binary knapsack problem with precedence constraints and solved by an efficient heuristic after a suitable modification. A brief comparison on the various techniques that can be used for optimal highway safety funds allocation is presented by Brown et al. (1990).

A common measure of effectiveness, when evaluating highway improvement alternatives, is the total reduction in the expected number of accidents. Traffic volume data as well as accident reduction factors for the various types of highway improvements are necessary for the evaluation of such a measure of effectiveness.

Traffic volumes are usually available in the databases of most local highway authorities and project costs are more or less readily obtainable from construction companies. For the acquisition of the accident reduction factors however, more effort is needed. Someone has to combine personal experience and judgement with results from appropriate reports analyzing how these factors can be estimated. Sinha and Hu (1985) discuss an approach on how to evaluate safety impacts of highway projects. They derive a relationship between accident reduction rate and 14 major highway design elements for different classes of highways such as interstate and urban arterial. Additionally, they derive accident reduction factors for combined improvement or maintenance activities such as rehabilitation and pavement maintenance.

Skinner (1985) presents data that relate typical improvements, such as lane or shoulder widening, with different types of accidents, such as single vehicle run-off road and multi-vehicle opposite direction. He also reports the expected reduction in number of accidents when combinations of these improvements are implemented. Opiela (1985) outlines a methodology for developing a set of accident reduction factors. This methodology consists of several steps, such as establishment of integrated databases and categorization of situations. Numerous papers elaborate on the effect of single key highway features on safety. Barbaresso et al. (1982) present a procedure for identifying and evaluating highway safety projects. An extensive summary of recent research results on the relationship between accident rates and highway design elements can be found in FHWA (1997).

3.3 The Knapsack Problem

Most integer programming books contain a section on knapsack problems. See, for example, Hu (1969), Garfinkel and Nemhauser (1972), Salkin (1975), Taha (1975), Papadimitriou and Steiglitz (1982), Syslo et al. (1983), Schrijver (1986), Nemhauser and Wolsey (1988), Pisinger and Toth (1999) and Korte and Vygen (2000). A profound introduction to the subject is also given by Ibaraki (1987a and 1987b).

In the recent years a large amount of research on knapsack problems has been published in the literature. Reviews are presented in the following surveys: Salkin and De Kluyver (1975) present a number of industrial applications and results in transforming integer linear programs to knapsack problems. Martello and Toth (1979) consider exact algorithms for the zero-one knapsack problem and their average computational performance. This study was extended to the other linear knapsack problems and to approximate algorithms in Martello and Toth (1987). Dudzinski and Walukiewicz (1987) analyze dual methods for solving Lagrangian and linear programming relaxations.

Martello and Toth (1990) present an extensive survey on exact and approximate algorithms for a number of important integer linear problems. The results refer not only to “classical” knapsack problems, but also to less familiar problems and well-known problems that are not usually classified in the knapsack area. Lin (1998) presents a large survey on some well-known non-standard knapsack problems.

The first exact solution for the 0-1 knapsack problem is due to the dynamic programming theory developed by Bellman (1957). Dantzig (1957) gave an efficient method to determine the solution to the continuous relaxation of this problem. This also produced an upper bound on the objective value of the discrete problem that was used in most of the following studies of the problem. Gilmore and Gomory (1961, 1963, 1965 and 1966) investigated the dynamic programming approach to the knapsack problem and other knapsack-type problems. Kolesar (1967) experimented with the first branch and bound algorithm for the problem. The large computer memory and time requirements of the Kolesar algorithm were greatly reduced by the Greenberg and Hegerich approach (1970). This approach was further developed and another well-known algorithm of this period is due to Horowitz and Sahni (1974). Ingargiola and Korsh (1973) presented the first reduction procedure and Johnson (1974) gave the first polynomial time approximation scheme for the subset-sum problem. Sahni (1975) combined the greedy heuristic with enumeration to obtain a polynomial approximation scheme for the knapsack problem. Ibarra and Kim (1975) obtained the first fully polynomial time approximation scheme and Martello and Toth (1977) proposed the first upper bound dominating the value of the continuous relaxation. The running time of Ibarra and Kim’s heuristic was improved by Lawler (1979). A different fully polynomial approximation scheme for the 0-1 knapsack problem was given by Magazine and Oguz (1981). Ingargiola and Korsh (1977) presented a reduction algorithm for the bounded knapsack problem, related to the one in Ingargiola and Korsh (1973) and imbedded it into a branch-search algorithm related to the one in Greenberg and Hegerich (1970). A different branch and bound strategy has been proposed by Bulfin et al. (1979). Balas and Zemel (1980) presented a new approach to solve the problem by sorting, in many cases, only a small subset of the variables. Other algorithms have been derived by Guignard and Spielberg (1972), Barr and Ross (1975), Nauss (1976), Zoltners (1978), Lauriere (1978), Suhl (1978), Veliev and Mamedov (1981), Fayard and Plateau (1982), Martello and Toth (1988 and 1997), Pisinger (1995b, 1997) and Martello et al. (1999 and 2000). Fayard and Plateau (1975) and Aittoniemi (1982) give experimental comparisons of some of the above algorithms. The enumeration of an exponential number of search tree nodes by the branch and bound algorithms with linear programming relaxations for some families of knapsack problems is pointed out by Chvatal (1980).

3.4 The Multiple Choice Knapsack Problem

In the study of the multiple choice knapsack problem many algorithms have been developed specifically for the linear case. Witzgall (1980) discussed a solution approach to this problem with a minimization objective function and an equality resource constraint. For the same problem Glover and Klingman (1979) developed a dual Simplex method and Zemel (1980) proposed a two-phase solution procedure. Johnson and Padberg (1981) generalized Dantzig’s method (Dantzig, 1963) to solve the linear multiple choice knapsack problem with inequality resource constraints. Dyer (1984) developed a pairing algorithm for the solution to a linear multiple choice knapsack problem and Dudzinski and Walukiewicz (1984) questioned the appropriateness of neglecting the reduction phase in Dyer's solution approach. Zemel (1984) studied a generalized version of the multiple choice knapsack problem called *d*-dimensional. Gass and Shao (1985) considered the same type of linear multiple choice knapsack problem as Johnson and Padberg and reported a solution approach similar to that proposed by Dyer. Finally, Sarin and Karwan (1989) considered a linear multiple choice knapsack problem where the objective was minimized. Table 3.1 summarizes the algorithms that have been developed for the solution of the Linear Multiple Choice Knapsack Problem.

On the other hand, the Integer Multiple Choice Knapsack Problem is NP-hard, since the integer knapsack problem is NP-hard too (Garey and Johnson, 1979). Most of the algorithms that have been developed, utilize the branch and bound procedure to enforce its solution. Nauss (1978), proposed two Lagrangian relaxation algorithms to solve the multiple choice integer knapsack problem and incorporated them in the branch and bound procedure for general IP by Geoffrion and Marsten (1972). Sinha and Zoltners (1979) proposed an algorithm that starts with the elimination of redundant variables. A procedure similar to the one by Sinha and Zoltners was redesigned by Armstrong et al. (1983), specifically for large scale multiple choice knapsack problems. Another branch and bound procedure was proposed by Dyer et al. (1984) and the Lagrangian relaxation approach was revisited by Aggarwal et al. (1992). Pisinger (1995a) developed an algorithm that was based on the minimal core theory (Pisinger, 1997) and Dyer et al. (1995) developed a hybrid dynamic programming/branch and bound algorithm for the problem. A summary of the algorithms for the Integer Multiple Choice Knapsack Problem is presented in Table 3.2.

Finally, it should be noted that the multiple choice knapsack problem has been reported in numerous applications, such as capital budgeting (Lorie and Savage, 1955), line balancing (Kilbridge and Wester, 1962), linking system components (Kolesar, 1966), fixed charge problems (Jones and Soland, 1969), product pricing (Garfinkel and Nemhauser, 1972), menu planning (Balintfy et al.,1978), catalogue spacing (Johnson et al., 1979), sales resource allocation (Zoltners and Sinha, 1980; Sinha and Zoltners, 1982), multi-item scheduling (Sweeney and Murphy, 1981) and capacitated facility location problems (Klincewicz and Luss, 1986).

Table 3.1: Algorithms for the Linear Multiple Choice Knapsack Problem

Authors (year)	**Featured Technique**	**Computational Complexity**
Glover and Klingman (1979)	Specialized dual simplex method with sensitivity analysis	$O[(\Sigma_{k \varepsilon S}\, n_k) \log \Sigma_{k \varepsilon S}\, n_k]$
Witzgall (1980)	Improved dual simplex method	$O[(\Sigma_{k \varepsilon S}\, n_k) \log \Sigma_{k \varepsilon S}\, n_k]$ $+ O\{r[(\Sigma_{k \varepsilon S}\, n_k)-r]\}$
Zemel (1980)	Reduction and transformation	$O[(\Sigma_{k \varepsilon S}\, n_k) \log \max n_k]$ $+ O(n)$
Johnson and Padberg (1981)	Generalized Dantzig's method	$O(\Sigma_{k \varepsilon S}\, n_k^2)$
Dyer (1984)	Duality theory	$O(\Sigma_{k \varepsilon S}\, n_k)$
Dudzinski and Walukiewicz (1984)	Exploitation between the primal and dual feasible solutions	$O[(\Sigma_{k \varepsilon S}\, n_k) \log \max n_k]$ $+ O(r \log^2 (n/r)]$
Zemel (1984)	LP technique applied on dual problem	$O(\Sigma_{k \varepsilon S}\, n_k)$
Gass and Shao Jr. (1985)	Duplex method on the dual problem	$O(\Sigma_{k \varepsilon S}\, n_k)$
Sarin and Karwan (1989)	Dual gradient method using dual simplex pivoting	$O(\Sigma_{k \varepsilon S}\, n_k)$

n_k = number of variables in set k, k = 1,2,…,|S|, before the elimination of any dominated variables
n = number of variables after the elimination of dominated variables
r = number of multiple choice constraints (multiple choice sets)

Table 3.2: Algorithms for the Integer Multiple Choice Knapsack Problem

Authors (year)	**Featured Technique**
Nauss (1978)	Lagrangian relaxation branch and bound procedure
Sinha and Zoltners (1979)	Special ordered sets branch and bound procedure
Armstrong, Kung, Sinha and Zoltners (1983)	Branch and bound procedure for large scale MCKP
Dyer, Kayal and Walker (1984)	Dual simplex method branch and bound procedure
Aggarwal, Deo and Sarkar (1992)	Network duality and Lagrangian relaxation branch and bound procedure
Pisinger (1995a)	Minimum core algorithm using dynamic programming and breadth first search
Dyer, Riha and Walker (1995)	Hybrid dynamic programming/branch and bound procedure using Lagrangian duality for computation of bounds and reduction

“A person who aims at nothing is sure to hit it.” **Anon**

“When the only tool you have is a hammer, you tend to treat everything as if it were a nail.” **Abraham Harold Maslow**

“It's always too early to quit.” **Norman Vincent Peale**

“You cannot conceive the many without the one.” **Plato**

Chapter 4: The Linear Multiple Choice Knapsack Problem

4.1 Problem Formulation

After ignoring the binary variables and the equity constraints, the problem of Section 1.3 reduces to the well-known Linear Multiple Choice Knapsack (LMCK) Problem:

$$\begin{aligned}
\text{Max} \ & \sum_{k \in S} \ \sum_{i \in R_k} p_{ki} x_{ki} \\
\text{s.t.} \ & \sum_{k \in S} \ \sum_{i \in R_k} c_{ki} x_{ki} \leq b \\
& \sum_{i \in R_k} x_{ki} \leq l_k \ , \forall \ k \in S \\
& x_{ki} \geq 0, i \in R_k \, , \forall \ k \in S
\end{aligned}$$

When the decision variables are constrained to binary values and $l_k = 1$, for all k, LMCK reduces to the Binary Multiple Choice Knapsack Problem. The Binary Multiple Choice Knapsack Problem can be viewed as the problem of trying to maximize the total value of N objects inserted in a knapsack of capacity b with the additional constraint that the objects are partitioned into $|S|$ subsets and at most one object per subset can be selected. In our case all the profit and cost coefficients are positive numbers. Johnson and Padberg (1981) showed however, that any LMCK with arbitrarily signed coefficients can be brought to a standard form with positive ones.

So far, LMCK has been mostly used as a relaxation to the Binary Multiple Choice Knapsack Problem. Following the representation used in this dissertation, the problem can also be used for optimal fund allocation to disjoint sets of continuous highway improvements. The multiple choice constraints can be used to handle the interactions that arise between the continuous improvements of a highway segment as described in Chapter 2.

In the following sections an algorithm for LMCK is presented, which is a slight modification of an existing algorithm by Johnson and Padberg (1981). This modification was necessary in order to accommodate the needs of the problems addressed later in this dissertation. Although this algorithm does not have the best performance among all linear multiple choice knapsack algorithms (Lin, 1998), it is chosen because it can be further adapted to support the problems that result after the inclusion of the binary variables and the equity constraints.

4.2 LMCK Properties

Problem LMCK has been widely investigated (Lin, 1998). Most of the solution algorithms that have been proposed for the problem are based on two fundamental properties (for the proofs see Chandra et al., 1976, or Sinha and Zoltners, 1979):

Property 4.1: If for some k there are two distinct variables x_{kh} and x_{kg} such that $p_{kh}/c_{kh} \geq p_{kg}/c_{kg}$ and $p_{kh} \geq p_{kg}$, then there is an optimal solution to LMCK with $x_{kg} = 0$.

Property 4.2: If for some k there are three distinct variables x_{kf}, x_{kg} and x_{kh}, such that $p_{kf} \leq p_{kg} \leq p_{kh}$, $c_{kf} < c_{kg} < c_{kh}$ and $\dfrac{p_{kg} - p_{kf}}{c_{kg} - c_{kf}} \leq \dfrac{p_{kh} - p_{kg}}{c_{kh} - c_{kg}}$, then there is an optimal solution to LMCK with $x_{kg} = 0$.

The significance of these two properties lies in the fact that they enable us to eliminate in advance some of the decision variables and reduce the size of the problem. Variables eliminated by the conditions of Property 4.1 are termed integer dominated; variables eliminated by the conditions of Property 4.2 are termed convex (LP) dominated (Armstrong et al., 1983). The variables of a set k can be visualized graphically as points on a two-dimensional graph, where the x-axis represents the cost of a variable and the y-axis its profit, as illustrated in Figure 4.1 (see also Sinha and Zoltners, 1979). By connecting nondominated variables in this graph a piecewise linear concave function is defined ($N_{ki}N_{kj}N_{kl}N_{km}$). This function defines the upper left hull (up to the point with the highest profit) of all the variables in this set.

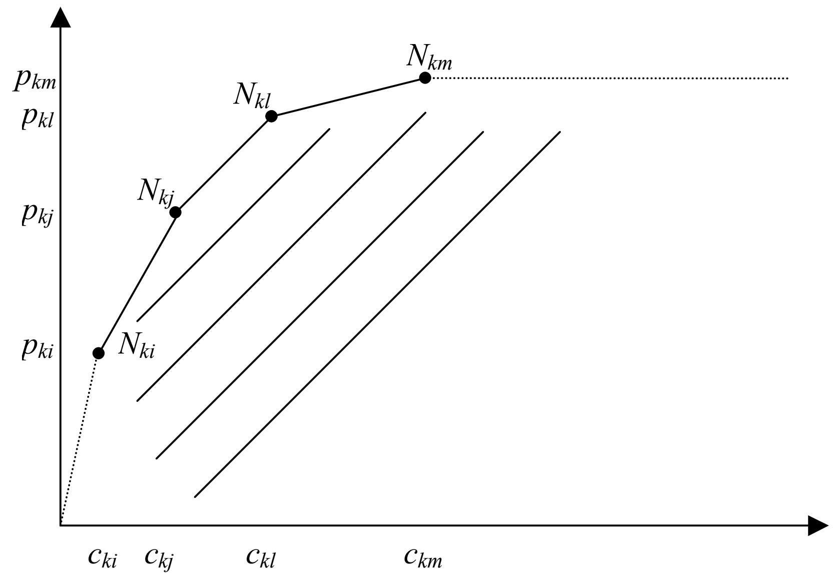

Figure 4.1: Coefficient space of decision variables within a multiple choice set *k*

Among others, Johnson and Padberg (1981) introduced an algorithm for LMCK, which can be viewed as a generalization of Dantzig's method (1963) for the linear knapsack problem with upper bounded variables. This algorithm was slightly modified to accommodate the more general structure of our formulation. This algorithm is presented in the next section followed by a discussion on its important properties.

4.3 Algorithm LMCK

The algorithm developed for LMCK consists of two steps. In the first step, each variable is associated with a ratio called the slope of the variable and these ratios are used to eliminate dominated variables. Next, one variable list L_k for each set k is constructed by arranging the variables of the set in non-increasing order of their associated slopes. Then, the variable lists of all sets are merged into a master list, *ML*. Variables in list *ML* are still arranged in non-increasing order of their associated slopes after this merging. In the second step of the algorithm, budget b is allocated successively to the decision variables (activities) according to the order they appear in the master list *ML*, until either the total budget is spent (scarce budget), or all decision variables (activities) in the master list are increased to their highest possible levels and there is still money left (surplus budget). The detailed steps of the algorithm follow along with a brief discussion on its properties.

Algorithm LMCK

Step 0: Preprocessing

For each multiple choice set k, construct a multiple choice list of variables L_k as follows: Order the nondominated variables in this set by increasing costs (list will be the same if variables are ordered by increasing profits). Define the slope of the first variable x_{ki} in each multiple choice list as the ratio p_{ki}/c_{ki}. For each subsequent variable x_{kj} in the list, define its slope as $\frac{p_{kj} - p_{ki}}{c_{kj} - c_{ki}}$, where x_{ki} is the variable immediately preceding x_{kj}. While maintaining all individual multiple choice lists, create a master list *ML* by merging the variables of all multiple choice lists in non-increasing order of their associated slopes. Initialize the objective function value u and the variables in all lists to 0. Let s be the index denoting the order in which variables appear in *ML*, $s = 1,..., |ML|$. Initialize $s = 0$.

Step 1: Iteration

Step 1.0 Set $s = s + 1$. If $s > |ML|$, STOP (there are no remaining variables for increase). If not, let the sth variable in *ML* be x_{kj}, and let b_{res} be the budget amount that has not been allocated yet (budget residual). If all variables in list k are 0, go to Step 1.1. If list k has exactly one variable x_{ki} with a positive value, go to Step 1.2.

Step 1.1 If $c_{kj}l_k < b_{res}$, set $x_{kj} = l_k$, $b_{res} = b_{\mathrm{res}} - c_{kj}l_k$, $u = u + p_{kj}l_k$ and go to Step 1.0.

If $c_{kj}l_k \geq b_{res}$, set $x_{kj} = b_{res}/c_{kj}$, $u = u + p_{kj}\, b_{res}/c_{kj}$, $b_{res} = 0$ and STOP.

Step 1.2 If $(c_{kj} - c_{ki})l_k < b_{res}$, set $x_{kj} = l_k$, $x_{ki} = 0$, $b_{res} = b_{\mathrm{res}} - (c_{kj} - c_{ki})l_k$, $u = u + (p_{kj} - p_{ki})l_k$ (variable x_{ki} is not eligible to be increased again). Go to Step 1.0.

If $(c_{kj} - c_{ki})l_k \geq b_{res}$, set $x_{kj} = b_{res}/(c_{kj} - c_{ki})$, $x_{ki} = l_k - x_{kj}$, $u = u + (p_{kj} - p_{ki})b_{res}/(c_{kj} - c_{ki})$, $b_{res} = 0$ and STOP. ▭

4.4 Properties of the Algorithm

The main difference between Algorithm LMCK presented here and the original algorithm, is the construction of the individual multiple choice lists and master list *ML*. These lists are utilized by the algorithms developed in later sections for the solution of different problems. The optimality of the above algorithm is further discussed in Johnson and Padberg (1981) and Pisinger (1995a). The algorithm can be viewed as a generalization of Dantzig's method for the linear knapsack problem with variables having upper bounds (Dantzig, 1963). In fact, if each multiple choice set is a singleton, it is precisely Dantzig's solution method to the problem without special ordered sets.

When the algorithm terminates in Step 1.0, each multiple choice set has exactly one positive-valued variable. When the algorithm terminates in Step 1.1, variable x_{kj} is called critical, i.e., x_{kj} is the rightmost variable in *ML* with a positive value. In this case, each multiple choice set has at most one positive-valued variable. Finally, when the algorithm terminates in Step 1.2, variable x_{kj} is also called critical. In the latter case, multiple choice set *k* has two positive-valued variables, including the critical, while any other multiple choice set has at most one positive-valued variable. Every positive-valued variable that belongs to a multiple choice set other than the one of the critical variable has a value which is equal to the right hand side of the associated multiple choice constraint. Note that, in Step 1.2 of the algorithm, if the budget residual is enough, variable x_{kj} will eventually get a value of l_k and variable x_{ki} will decrease to 0. As a result, at the time a variable is considered for increase, each multiple choice list has at most one positive-valued variable.

The list L_k and the budget amount b_k, allocated to set *k*, determine the optimal values of the variables x_{ki} in this set. Of course, the budget amount allocated to set *k* depends on the relative magnitude of the associated slopes of variables belonging to all sets in the master list *ML* and it is not known in advance. For the sake of illustration, let's assume that the budget amount allocated to set *k* at the optimal solution of the problem is b_k and that the variable list for set *k* is $L_k = \{x_{k2}, x_{k6}, x_{k7}\}$. For small values of b_k, i.e., $0 < b_k$

$\leq c_{k2}l_k$, only variable x_{k2} will have a positive value in set k because it has the highest slope among all nondominated variables in that set. For $b_k = c_{k2}l_k$, $x_{k2} = l_k$, and the profit from set k is $p_{k2}l_k$. When b_k increases beyond $c_{k2}l_k$, it becomes more profitable for x_{k6} to increase while x_{k2} has to decrease in order to ensure that the sum of the two variables does not exceed l_k. The reason x_{k6} gradually replaces x_{k2} is its higher profit (per unit length). It can be easily shown that the new profit from set k is $p_{k2}l_{k2} + \dfrac{p_{k6} - p_{k2}}{c_{k6} - c_{k2}}(b_k - c_{k2}l_k)$. Thus, the excess budget amount (b_k - $c_{k2}l_k$) is used for gaining additional profit. It is clear to see now that the associated slope of a variable (x_{k6}) in the list, represents the incremental profit when this variable is increased in place of the previous variable (x_{k2}). Eventually, when $b_k = c_{k6}l_k$, $x_{k6} = l_k$ and $x_{k2} = 0$. If b_k is increased further, x_{k7} will increase and x_{k6} will decrease. When $b_k = c_{k7}l_k$, there is only one positive-valued variable, $x_{k7} = l_k$. Of course, b_k cannot be larger than $c_{k7}l_k$ and the profit from set k cannot be larger than $p_{k7}l_k$. As a result, there are at most two positive-valued variables from a set in the optimal solution. These are (consecutive) variables from the associated set list and the higher the budget amount allocated to the set, the higher the order (towards the end) of these variables in the list.

Variables in each multiple choice list are arranged in non-increasing order of their associated slopes (see also Armstrong et al., 1983). The associated slope of each variable provides a measure of the incremental profit earned per unit of budget spent, when this variable is increased. At each iteration, the algorithm selects the variable with the highest such slope among all variables that haven't been increased yet. For a multiple choice set with all variables equal to 0, this is the variable x_{ki} appearing first in the associated multiple choice list, because that variable has the highest p_{ki}/c_{ki} ratio. For a multiple choice set with exactly one positive-valued variable x_{ki}, this is the variable x_{kj} immediately succeeding x_{ki} in the associated multiple choice list. At first glance, it may seem counter-intuitive that a variable with an inferior ratio p_{kj}/c_{kj} replaces a variable with a superior ratio p_{ki}/c_{ki}. This is justified however by the following reasoning. When additional funds are available, they can be used to "buy" additional profit, although at a higher price. An implication of this is that, after the increase of a variable, the next variable considered for increase from the same set always has a higher profit. Ties in any part of the algorithm can be broken arbitrarily.

The following proposition provides some insight into Algorithm LMCK. It will be utilized in subsequent parts of the dissertation.

Proposition 4.1: Increase of any variable in Step 1 of Algorithm LMCK results in a concurrent increase in both total cost and total profit.

Proof: The validity of the proposition is clear in the case that the variable selected for increase belongs to a multiple choice set with all variables equal to 0. Let's consider the case when the variable x_{kj} selected for increase belongs to a multiple choice set with one positive-valued variable x_{ki}. Let Δl be the amount by which x_{ki} is decreased and x_{kj} is increased. The net change in total profit and total cost is $(p_{kj} - p_{ki})\ \Delta l$ and $(c_{kj} - c_{ki})\ \Delta l$, respectively. Both changes are positive, since variables in a multiple choice list are arranged in increasing order of both their profits and costs. ▭

"Knowing is not enough; we must apply. Willing is not enough; we must do."

Johann von Goethe

"Our greatest glory is not in never falling, but in rising every time we fall." **Confucius**

"The foundation of every state is the education of its youth." **Diogenes**

"Cultivation to the mind is as necessary as food to the body." **Marcus Tullius Cicero**

Chapter 5: The 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice Constraints

5.1 Introduction

In this chapter, the problem that results when the equity constraints are dropped from the model of Section 1.3 is considered. The formulation of this problem is as follows:

$$\begin{aligned}
\text{Max} \ & \sum_{k\in S} \sum_{i\in R_k} p_{ki}x_{ki} + \sum_{k\in S} \sum_{j\in D_k} q_{kj}y_{kj} \\
\text{s.t.} \ & \sum_{k\in S} \sum_{i\in R_k} c_{ki}x_{ki} + \sum_{k\in S} \sum_{j\in R_k} d_{kj}y_{kj} \le b \\
& \sum_{i\in R_k} x_{ki} \le l_k \ , \forall\, k \in S \\
& x_{ki} \ge 0, i\in R_k, \forall\, k\in S \\
& y_{kj} \in \{0,1\}, j\in D_k, \forall\, k\in S
\end{aligned}$$

Conforming to the knapsack problem terminology and trying to avoid confusing notation, this problem is abbreviated as MIMCK, standing for 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice Constraints. The problem involves both the general mixed integer knapsack and the linear multiple choice knapsack problems. To the best knowledge of the author, an algorithm for this special problem has not appeared in the literature. In the next sections, two important propositions are developed, which lead to the construction of an efficient branch and bound algorithm for Problem MIMCK.

5.2 Solution Procedure

Problem MIMCK is related to Problem LMCK in the following way. When the binary constraints are relaxed (i.e. each binary variable can take any value between 0 and 1), MIMCK reduces to the LMCK Problem. This is because each originally binary variable y_{kj} forms a multiple choice set of cardinality 1 with associated multiple choice constraint $y_{kj} \leq 1$. Each binary variable, therefore, occupies its own list, having no other variables to compete with for dominance. The domination criteria, discussed above for LMCK, also apply to Problem MIMCK and the procedure for eliminating dominated continuous variables remains the same. The master list now contains both types of variables, x_{ki}'s and y_{kj}'s.

The first proposition developed follows from the theory that was discussed in the previous chapter for Problem LMCK and deals with the linear programming (LP) relaxation of MIMCK. The LP relaxation of MIMCK is formed by replacing the binary constraints of the problem with bound constraints, $0 \leq y_{kj} \leq 1$. If B is the total number of binary variables, $B = \Sigma_{k \varepsilon S} |D_k|$, the resulting problem is a linear multiple choice knapsack problem with B additional multiple choice constraints. Each of the additional multiple choice sets is a singleton. As a result, the following important property holds.

Proposition 5.1: The optimal solution to the LP relaxation of MIMCK, contains at most one binary variable with a fractional value.

Proof: The optimal solution to Problem LMCK has at most two fractional variables (a continuous variable x_{ki} is considered fractional if $0 < x_{ki} < l_k$). If however this solution has indeed two fractional variables, these variables belong to the same multiple choice set. Since each originally binary variable belongs to a multiple choice set with cardinality 1, the optimal solution to the LP relaxation of MIMCK contains at most one binary variable with a fractional value. $\square$

The result stated by Proposition 5.1 is important because it ensures that out of the B binary variables, no more than one will have a fractional value at the optimal solution to the LP relaxation of MIMCK. This simplifies the treatment of the problem since there is at most one variable violating the integrality constraints. The power of this result becomes clear, when we consider that the optimal solution to the LP relaxation of a mixed integer problem usually has many fractional-valued binary variables.

At the first iteration of the proposed branch and bound (B&B) procedure, the solution to the LP relaxation of the original problem is checked for feasibility. If it is feasible with respect to the integrality constraints, the algorithm terminates since this is the optimal solution of the problem. If not, this solution contains one fractional-valued binary variable, y_{kj}, according to Proposition 5.1. This is the critical variable, as defined in the previous chapter. It is used for branching, generating two subproblems (descendants), each of which has the additional constraint $y_{kj} = 0$ and $y_{kj} = 1$, respectively. Clearly, Proposition 5.1 also applies to each of these subproblems, since each of them is a new MIMCK Problem. The LP relaxation of each of these new subproblems is solved and the objective value is stored as an upper bound to any feasible solution that may be obtained in the associated subtree. At the next iteration, the subproblem with the maximum upper bound is selected for exploration. The iterations continue by branching on the fractional binary variables until an optimal solution to the original problem is reached.

The subproblems of the B&B tree differ from each other only in a small number of constraints, i.e., in the additional constraints that set some of the binary variables to 0 or 1. The following proposition is used to reduce significantly the time needed to solve the LP relaxation of these subproblems.

Proposition 5.2: With the exception of the critical variable, the order of increasing variables when Algorithm LMCK is applied to two descendant subproblems generated by the B&B procedure is the same as the order of increasing variables when Algorithm LMCK is applied to the parent problem.

Proof: The validity of this proposition results from the fact that the only difference between the LP relaxation of the parent problem and the LP relaxation of its children problems, is that the critical variable is constrained to have a specific value (0 or 1) in each of the two children problems. The exclusion of the critical variable does not affect in any way the order of the other variables in *ML*, since this variable is a multiple choice set by itself. Therefore, the master list *ML* of increasing variables for the children nodes is identical to that of the parent node with the only exception that the critical variable is excluded. ▭

The important implication of this proposition is that it is not necessary to solve from scratch the LP relaxations of the various subproblems, since the solution to the LP relaxation of their parent problem is available. Thus, the solution of the two descendant subproblems can be found, starting from the solution of their parent problem, by utilizing properly Algorithm LMCK. The only additional information needed for this procedure is the order in which the remaining variables, besides the critical variable, should be increased. This information is provided by the list *ML* which was already created in Step 0 of Algorithm LMCK.

A more detailed outline of the algorithm, called Algorithm MIMCK, follows. For reasons of clarity and simplicity, the variables are indexed sequentially using a single index. The following additional notation is used:

t = index for labeling the nodes of the B&B tree,

A = set containing the active nodes of the B&B tree, i.e., the nodes yet to be explored,

$I_0(t)$ = set containing the indexes of the binary variables set to 0 at node t,

$I_1(t)$ = set containing the indexes of the binary variables set to 1 at node t,

u_t = upper bound on the optimal objective value in the subtree with root t,

e_t = indicator that takes the value 1 if the critical variable of subproblem t is binary (infeasible solution) and 0 if it is continuous (feasible solution),

f_t = index of critical variable of subproblem t,

v_t = value of critical variable of subproblem t, and

b^t_{res} = budget residual of the solution to the LP relaxation of subproblem t.

Algorithm MIMCK

Step 1: Initialization

Initialize the index of the root of the B&B tree to 1. Set $A = \{1\}$, $I_0(1) = \varnothing$ and $I_1(1) = \varnothing$.

Step 2: LP relaxation of original problem

Solve the LP relaxation of the original problem using Algorithm LMCK. Store in u_1 the objective value for this solution and in b^1_{res} the associated budget residual.

If Algorithm LMCK terminates at Step 1.0, set $f_1 = v_1 = e_1 = 0$ and STOP.

If Algorithm LMCK terminates at Step 1.1 or 1.2, store in f_1 and v_1 the index and the value of the critical variable for this solution, respectively. If this critical variable is binary, set $e_1 = 1$ and go to Step 3. Otherwise, set $e_1 = 0$ and STOP; the current solution is optimal for the original problem.

Step 3: Branching

Let t_0 be the smallest unused index for labeling nodes and $t^* = \underset{t \in A}{arg\ max}\, u_t$.

If $e_{t^*} = 0$, STOP; the solution at node t^* is optimal for the original problem.

Else, branch on variable y_r, where $r = f_{t^*}$, generating two new subproblems and their associated nodes. Label the nodes t_0 and $t_0 + 1$, corresponding to subproblem with $y_r = 0$ and $y_r = 1$, respectively. Set $I_0(t_0) = I_0(t^*)\ \chi \{r\}$, $I_1(t_0) = I_1(t^*)$, $I_0(t_0 + 1) = I_0(t^*)$, $I_1(t_0 + 1) =$

$I_1(t^*)\ \chi \{r\}$ and $A = A - \{t^*\} \chi \{t_0\} \chi \{t_0 + 1\}$.

Step 4: LP relaxation of subproblem t_0

Set $b^{t_0}_{res} = d_r\, v_{t^*}$ and $u_{t0} = u_{t^*} - q_r\, v_{t^*}$ (results obtained after setting y_r to 0). Let s be the index denoting the order of y_r in list $ML - \{y_j \,/\, j0\, I_0(t^*)$ or $j0\, I_1(t^*)\}$. Using this list in place of ML, resume Step 1 of Algorithm LMCK, updating $b^{t_0}_{res}$ and u_{t0} in place of b_{res} and u, respectively.

If Algorithm LMCK terminates at Step 1.0, set $f_{t0} = v_{t0} = e_{t0} = 0$.

If Algorithm LMCK terminates at Step 1.1 or 1.2, store in f_{t0} and v_{t0} the index and the value of the critical variable of that solution, respectively. If this variable is binary, set $e_{t0} = 1$; otherwise, set $e_{t0} = 0$.

Step 5: LP relaxation of subproblem t_0 + 1

Step 5.0 Set $b^{t_0+1}_{res} = -\, d_r\,(1 - v_{t^*})$ and $u_{t0+1} = u_{t^*} + q_r\,(1 - v_{t^*})$ (results obtained after setting y_r to 1). Let s be the index denoting the order of y_r in list $ML - \{y_j \,/\, j0\, I_0(t^*)$ or $j0\, I_1(t^*)\}$.

Step 5.1 Set $s = s - 1$.

If $s = 0$, fathom the current node by removing its index from A, since the budget residual is negative (infeasible subproblem) and go to Step 3.

Otherwise, select the s^{th} variable for decrease from list $ML - \{y_j \,/\, j\in I_0(t^*) \text{ or } j\in I_1(t^*)\}$. Let this variable belong to a multiple choice list k. If this is the first variable in this multiple choice list, go to Step 5.2. If not, this is a continuous variable, x_j, that originally replaced another variable, x_i, in list k; go to Step 5.3.

Step 5.2 Use in this step d_j, q_j, y_j and 1 in place of c_j, p_j, x_j and l_k, respectively, if the selected variable is binary (y_j) instead of continuous (x_j).

If $c_j\, l_k < |b_{res}^{t_0+1}|$, set $x_j = 0$, $b_{res}^{t_0+1} = b_{res}^{t_0+1} + c_j\, l_k$, $u_{t0+1} = u_{t0+1} - p_j\, l_k$ and go to Step 5.1.

If $c_j\, l_k \geq |b_{res}^{t_0+1}|$, set $x_j = l_k - |b_{res}^{t_0+1}|/c_j$, $u_{t0+1} = u_{t0+1} - p_j|b_{res}^{t_0+1}|/c_j$, $b_{res}^{t_0+1} = 0$ (x_j is the critical variable) and go to Step 5.4.

Step 5.3 If $(c_j - c_i)l_k < |b_{res}^{t_0+1}|$, set $x_j = 0$, $x_i = l_k$, $b_{res}^{t_0+1} = b_{res}^{t_0+1} + (c_j - c_i)l_k$, $u_{t0+1} = u_{t0+1} - (p_j - p_i)l_k$ and go to Step 5.1.

If $(c_j - c_i)l_k \geq |b_{res}^{t_0+1}|$, set $x_j = l_k - |b_{res}^{t_0+1}|/(c_j - c_i)$, $x_i = l_k - x_j$, $u_{t0+1} = u_{t0+1} - (p_j - p_i)|b_{res}^{t_0+1}|/(c_j - c_i)$, $b_{res}^{t_0+1} = 0$ (x_j is the critical variable) and go to Step 5.4.

Step 5.4 Store in f_{t0+1} and v_{t0+1} the index and the value of the critical variable of this solution, respectively. If this variable is binary, set $e_{t0+1} =1$; otherwise, set $e_{t0+1} = 0$. Go to Step 3. $\square$

The following remarks should be made about the algorithm. Step 5 of Algorithm MIMCK is essentially a backward move of Algorithm LMCK. Starting from a solution with negative budget residual, variables are successively decreased in the exact reverse order in which they were increased originally, until the budget residual becomes 0.

Proposition 4.1 guarantees that, during this move, the total cost will monotonically decrease and for this reason at some point (if the current subproblem is feasible) it will become equal to budget *b*. This new solution coincides with the one that would have been obtained if Algorithm LMCK had been applied from scratch. Note that, each time branching on a binary variable takes place, the budget residual of the current subproblem is equal to 0, otherwise the increase of this binary variable wouldn't have been interrupted.

A tree node is fathomed from the B&B tree if the associated subproblem is infeasible. This happens when the binary variables that have been set to 1 result in a total cost that exceeds the available budget, *b*. A check for this is done at the beginning of Step 5.1. If the backward scanning of list *ML* ends with a negative budget residual then the current tree node is removed from the set of active nodes.

If no binary variables are present, the above algorithm can be used to solve the linear multiple choice knapsack problem. Similarly, if no multiple choice constraints are present, the algorithm can be used to solve the general mixed integer knapsack problem. Finally, if no continuous variables are present, the algorithm can be used to solve the general binary knapsack problem.

The advantages of the algorithm become immediately clear. The solution to the LP relaxation of the subproblems generated in subsequent steps of the algorithm is obtained in most cases in a small number of iterations, since the relationship between the optimal solutions of parent and children nodes is cleverly utilized. Additionally, the memory space usage is low, since only a small number of values need to be stored for each of the nodes of the B&B tree. These are the values of u_t, f_t, v_t, e_t, b^t_{res} and the elements of sets $I_0(t)$ and $I_1(t)$. Decision variable values are not stored explicitly.

To obtain the values of the decision variables at the termination of Algorithm MIMCK, the optimal node g and its associated critical variable are needed. If the optimal node does not have a critical variable (termination at Step 1.0 of Algorithm LMCK), a fictitious variable is appended as critical at the end of list *ML* with value 0. The values of the decision variables are obtained as follows. All binary variables appearing in *ML* to the left of the critical variable have a value of 1, unless their index is in set $I_0(g)$. Similarly, all binary variables appearing in *ML* to the right of the critical variable, have a value of 0, unless their index is in set $I_1(g)$.

The values of the continuous variables are determined as follows: Let k be the multiple choice set containing the critical variable. If the critical variable is the first variable in multiple choice list k then no other variable from k will have a positive value. If not, the only other continuous variable with a positive value from multiple choice set k will be the one appearing in *ML* to the left and closest to the critical variable. The value of that variable is equal to the right hand side of the associated multiple choice constraint minus the value of the critical variable.

If a multiple choice set has no continuous variables appearing in *ML* to the left of the critical variable then all its variables are equal to 0. If it has at least one variable appearing in *ML* to the left of the critical variable, exactly one variable in this set has a positive value and all the remaining variables are equal to 0. The variable with the positive value from this multiple choice set is the one appearing in *ML* to the left and closest to the critical variable. The value of this variable is equal to the right hand side of the associated multiple choice constraint.

5.3 Application of the Algorithm

In this section the application of the proposed algorithm is illustrated in a small numerical example. Consider the following MIMCK Problem having 3 disjoint variable sets ($|S| = 3$), each of which contains 7 continuous and 3 binary variables ($|R_k| = 7$, $|D_k| = 3$, $k = 1,2,3$):

$$\begin{aligned}
\text{Max}\ \ & 3x_{11} + 5x_{12} + 8x_{13} + 6x_{14} + 5x_{15} + 6x_{16} + 7x_{17} + \\
& 2x_{21} + 6x_{22} + 4x_{23} + 6x_{24} + 3x_{25} + 6x_{26} + 7x_{27} + \\
& 2x_{31} + 4x_{32} + 4x_{33} + 8x_{34} + 3x_{35} + 5x_{36} + 9x_{37} + \\
& 3y_{11} + 0.6y_{12} + 3y_{13} + y_{21} + 7y_{22} + 6y_{23} + 4y_{31} + \\
& 3y_{32} + 5y_{33} \\
\text{s.t.}\ \ & x_{11} + 2x_{12} + 4x_{13} + 3x_{14} + 6x_{15} + 5x_{16} + 4x_{17} + \\
& 5x_{21} + 2x_{22} + x_{23} + 3x_{24} + 6x_{25} + 5x_{26} + 3x_{27} + \\
& 5x_{31} + 2x_{32} + 4x_{33} + 2x_{34} + 6x_{35} + x_{36} + 3x_{37} + \\
& 7y_{11} + 5y_{12} + 4y_{13} + 2y_{21} + 6y_{22} + 4y_{23} + 7y_{31} + 6y_{32} + y_{33} \leq 40 \\
& x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} \leq 1 \\
& x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{26} + x_{27} \leq 1 \\
& x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{36} + x_{37} \leq 1 \\
& x_{ki} \geq 0,\ k = 1,2,3,\ i = 1,2,\ldots,7 \\
& y_{kj}\ 0\ \{0,1\},\ k = 1,2,3,\ j = 1,2,3
\end{aligned}$$

The application of Algorithm MIMCK is illustrated below.

Step 1: Initialize the index of the first B&B tree node to 1. Set $A = \{1\}$ and initialize $I_0(1) = \varnothing$ and $I_1(1) = \varnothing$.

Step 2: The complete list of increasing variables for this problem is: $ML = \{x_{36}, y_{33}, x_{23}, x_{11}, x_{34}, x_{12}, x_{22}, y_{23}, x_{13}, y_{12}, y_{22}, x_{27}, x_{37}, y_{13}, y_{31}, y_{32}, y_{21}, y_{11}\}$. This list was created by merging multiple choice lists $L_1 = \{x_{11}, x_{12}, x_{13}\}$, $L_2 = \{x_{23}, x_{22}, x_{27}\}$, $L_3 = \{x_{36}, x_{34}, x_{37}\}$ and the 9 singletons containing the binary variables. The remaining 12 continuous variables are not included in ML because they are dominated under Properties 4.1 and 4.2. To understand how these list are created, it is illustrated next how multiple choice list L_1 and the associated slopes of the variables in this set are found. As already mentioned in Chapter 4, the nondominated variables form the function that defines the upper left hull of the variables in this set. Therefore, in order to identify them, any algorithm for finding the convex hull of n points in 2 dimensions can be used. The nondominated variables for the first set are x_{11}, x_{12} and x_{13}. When these variables are ordered by increasing costs, the following list results for set 1: $L_1 = \{x_{11}, x_{12}, x_{13}\}$. The associated slopes of these 3 variables are 3, 2 and 1.5, respectively. The associated slope for variable x_{11} is computed as $p_{11}/c_{11} = 3/1$ and the associated slopes for variables x_{12} and x_{13} are computed as $\frac{p_{12}-p_{11}}{c_{12}-c_{11}} = \frac{5-3}{2-1}$ and $\frac{p_{13}-p_{12}}{c_{13}-c_{12}} = \frac{8-5}{4-2}$, respectively. The multiple choice lists of the other two sets of continuous variables are constructed similarly. Each binary variable forms alone a multiple choice list, with associated slope its ratio q_{kj}/d_{kj}. After construction of the individual lists, master list ML is constructed by merging the variables from all these lists in non-increasing order of their associated slopes. For example, variable x_{36} appears first in *ML* because it has the highest associated slope (5).

Algorithm LMCK, applied to the LP relaxation of the original problem, terminates with critical variable $y_{32} = 0.5$, $b^1_{res} = 0$ and $u_1 = 56.5$. Therefore, $f_1 = 32$ and $v_1 = 0.5$. Set $e_1 = 1$ and go to Step 3.

Step 3: The only index in A is 1, therefore node 1 is selected. Since $e_1 = 1$ and $f_1 = 32$, branch on variable y_{32}, generating two new nodes, node 2 and node 3. Node 2 has the additional constraint $y_{32} = 0$ and node 3 has the additional constraint $y_{32} = 1$. Set $I_0(2) = I_0(1) \chi \{32\}$, $I_1(2) = I_1(1)$, $I_0(3) = I_0(1)$, $I_1(3) = I_1(1) \chi \{32\}$ and $A = A - \{1\} + \{2,3\} = \{2,3\}$.

Step 4: After setting $y_{32} = 0$, we get $b^2_{res} = d_{32}v_1 = 6(0.5) = 3$ and $u_2 = u_1 - q_{32}v_1 = 56.5 - 3(0.5) = 55$. Then, after Algorithm LMCK is resumed, it terminates with critical variable $y_{11} = 0.143$, $b^2_{res} = 0$ and $u_2 = 56.428$. Set $f_2 = 11$, $v_2 = 0.143$ and $e_2 = 1$, since the critical variable of this solution is binary.

Step 5: After setting $y_{32} = 1$, we get $b^3_{res} = - d_{32}(1 - v_1) = 0 - 6(1 - 0.5) = -3$ and $u_3 = u_1 + q_{32}(1 - v_1) = 56.5 + 3(1 - 0.5) = 58$. The backward move of the algorithm terminates with critical variable $y_{31} = 0.57$, $b^3_{res} = 0$ and $u_3 = 56.28$. Set $e_3 = 1$, $f_3 = 31$ and $v_3 = 0.57$.

The algorithm proceeds similarly and the B&B tree, shown in Figure 5.1, summarizes all iterations. Next to each node there are two entries. The first entry is the upper bound on the optimal objective value of the associated subtree. The second entry is the value of the critical variable of this node. Only 7 nodes were generated by the algorithm and the subproblem solutions were easily found as an implication of Proposition 5.2.

The optimal solution of the problem is obtained at node 4. The optimal values of the decision variables are found as follows: No critical variable exists for this solution, since list *ML* was scanned completely (surplus budget). Thus, a fictitious variable is appended at the end of list *ML*. Each of the three multiple choice sets has at least one continuous variable to the left of the critical variable in *ML*. Therefore, for each set, the only positive continuous variable will be the one to the left and closest to the critical variable with value equal to the right hand side of the associated multiple choice constraint. Therefore, the only positive continuous variables are $x_{13} = 1$, $x_{27} = 1$, $x_{37} = 1$. For node 4 we have: $I_0(4) = \{32,11\}$ and $I_1(4) = \varnothing$. Thus, all binary variables but y_{11} and y_{32} have a value of 1.

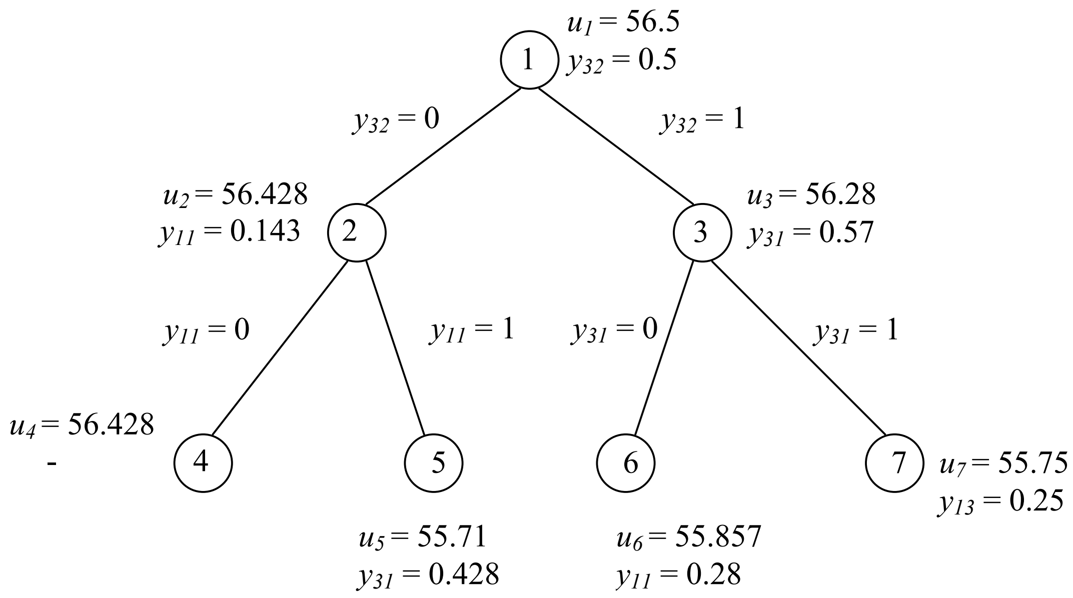

Figure 5.1: Branch and bound tree for the MIMCK numerical example

5.4 Computational Complexity and Results

In this section, the complexity of Algorithm MIMCK is analyzed and some computational results obtained from computer experimentation are presented. The special case of MIMCK without continuous variables is the general binary knapsack problem which is NP-hard. Therefore, Problem MIMCK is NP-hard too.

Let N_k be the number of (continuous) variables in multiple choice set k, $N = \sum_{k\in S} N_k$, $N_{max} = \max_{k\in S} N_k$ and $r = |S|$. The time needed for the construction of the multiple choice lists in Step 0 of Algorithm LMCK is $\sum_{k\in S} \mathrm{O}(N_k \log N_k) =$ $\mathrm{O}(\sum_{k\in S} N_k \log N_k) = \mathrm{O}(N \log N_{max})$. The time needed for merging these lists to obtain the master list *ML* is O(N log r) (see Cormen et al., 2001). The work needed in Step 0 of Algorithm LMCK dominates the work needed in Step 1, which is linear in the total number of variables.

As a B&B algorithm, MIMCK has an exponential worst case complexity. Step 1 requires constant time. If B is the number of binary variables, they can be ordered in time O(B log B). The remaining r multiple choice lists of continuous variables can be constructed in time $\mathrm{O}(N \log N_{max})$. Then, these $r + 1$ lists can be merged into a single list in time O((N+B) log (r+1)). Therefore, Step 2 requires time O(B log B) $+\mathrm{O}(N \log N_{max})$ + O((N+B) log (r+1)) = O((N+B) m, where m = max (log B, log N_{max}, log r). Steps 4 and 5 require time which is linear in the total number of variables and on the average linear on only a subset of them. This is because, in most situations, only a subset of the variables in master list *ML* will be scanned. Together with Step 3 however, Steps 4 and 5 are executed in the worst case exponentially many times, since the maximum number of subproblems generated is an exponential function of the total number of binary variables.

Algorithm MIMCK was coded in C and tested on a Pentium IV/1.8 GHz processor. As shown in the results presented in Table 5.1, the number of sets as well as the number of continuous variables within each set were varied from 100 to 400 in steps of 100. For all problems, the total number of binary variables was equal to the total number of continuous variables. Thus, the smallest size problems have 10,000 continuous and 10,000 binary variables, while the largest size problems have 160,000 continuous and 160,000 binary variables. Parameters p_{ki}, c_{ki}, q_{kj} and d_{kj} were uniformly distributed between 0 and N_k. While Table 5.1 shows the initial mix of binary and continuous variables, the problems that result after elimination of the dominated variables, contain significantly fewer continuous variables. This is because the expected percentage of integer dominated variables within a multiple choice set increases from 70% when N_k = 10, to 96% when N_k = 150 (Sinha and Zoltners, 1979). This reduction in the number of continuous variables increases the difficulty of the problem as explained shortly. Parameter l_k was set equal to 1 for all multiple choice sets. Finally, the budget b was set at $\frac{1}{2}\sum_{k\in S}(\min_{i\in R_k} c_{ki} + \max_{i\in R_k} c_{ki}) + 0.25\,BN_k$. Thus, the average budget amount available for allocation to each multiple choice set and each binary variable was 0.5 N_k and 0.25 N_k, respectively. This decision for the average budget amount allocated to each binary variable ensures a tight budget constraint, i.e., a scarce budget. Computational experience indicates that the problems generated this way show more computational interest.

The above procedure for generating random problems is based on the procedure used by Sinha and Zoltners (1979) to test the multiple choice knapsack problem. Later papers dealing with the same problem used a similar procedure for generating random problem instances. Of course, this procedure was properly adapted to suit the structure of the problem considered in this chapter.

For each problem size, 10 different random instances were solved. The column labeled “LP” shows the average time needed for solving the LP relaxation of the problem (Steps 1 and 2 of Algorithm MIMCK). Only the average time is reported in Table 5.1, since no significant variation was observed for different instances of the same problem size. The results for the total time needed and the number of nodes of the B&B tree generated until the optimal solution of the problem is found, are shown in the columns under the label “Time” and “Nodes”, respectively. The minimum, maximum and average for each of these parameters are reported. The last column of Table 5.1 shows the percentage of the continuous variables that are dominated and therefore eliminated as an implication of Properties 4.1 and 4.2. Execution times are reported in seconds.

Table 5.1: Computational results for MIMCK (time in seconds)

r	N_k	B	LP		Time			Nodes		dom.
			Avg	Min	Avg	Max	Min	Avg	Max	%
100	100	10000	0.171	0.180	0.702	5.008	45	1034	8415	96.47
100	200	20000	0.363	0.371	0.733	1.362	59	689	1855	97.94
100	300	30000	0.549	0.661	3.336	12.618	101	2429	6785	98.49
100	400	40000	0.812	1.022	5.342	33.589	195	2654	16573	98.82
200	100	20000	0.361	0.410	0.689	1.192	39	594	1463	96.31
200	200	40000	0.794	0.871	1.358	1.963	67	572	1185	97.90
200	300	60000	1.267	1.372	4.267	16.524	117	1856	8531	98.50
200	400	80000	1.767	2.684	6.384	10.615	483	2160	4247	98.88
300	100	30000	0.603	0.611	1.152	3.575	21	695	3647	96.38
300	200	60000	1.297	1.332	3.829	15.272	31	1663	8739	97.94
300	300	90000	2.071	2.514	12.085	48.88	277	3891	15919	98.56
300	400	120000	2.799	3.024	9.270	25.587	101	1937	6069	98.87
400	100	40000	0.815	0.891	1.672	4.827	69	870	3879	96.31
400	200	80000	1.798	1.883	4.543	11.557	3	1403	4957	97.99
400	300	120000	2.711	2.864	11.387	28.861	77	2772	9137	98.55
400	400	160000	3.883	4.086	13.901	28.301	7	2660	6525	98.85

The efficiency of the algorithm is demonstrated by the results of Table 5.1. The average number of tree nodes is a small percentage of the total number of binary variables for all problem sizes. The average and the maximum number of nodes seem to increase very slowly as the number of binary variables increases. Similar observations can be made for the total time needed to obtain the optimal solution of the problem. It is also very interesting to note that, as shown in the column for the minimum number of nodes, in some instances the algorithm terminated with a negligible number of tree nodes.

For most problem sizes, both the total time and the size of the B&B tree vary significantly, as indicated by the high values of the ranges (Max-Min). This is due to the fact that occasionally a problem requires a large B&B tree, while most problems require a relatively small B&B tree. This behavior is not surprising, since the performance of most B&B algorithms depends on the specific instance of the problem.

The time needed for solving a given problem is substantially smaller than the time needed to obtain the solution using a commercial package for mixed integer programming (e.g. LINGO, 2001). The savings in computational effort become more significant as the size of the problem increases. For example, in the problems with $r = 100$, $N_k = 100$, $B = 10000$, LINGO needs on the average 6 seconds to find the optimal solution. For the problems with $r = 200$, $N_k = 200$, $B = 40000$, this time increases to 30 seconds and for the problems with $r = 300$, $N_k = 300$, $B = 90000$, it increases to 70 seconds. The machine runs out of memory for the problems with $r = 400$, $N_k = 400$, $B = 160000$.

A significant amount of time is spent for solving the LP relaxation of the original problem in Step 2 of Algorithm MIMCK, due to the fact that the nondominated variables have to be identified in order to construct the multiple choice lists. The time consumed on branching and solving subproblems of the B&B tree is relatively small. This is mainly due to the utilization of Proposition 5.2. As a percentage of the average total time, the time, t, needed to obtain the solution to the LP relaxation of the problem ranges between 15% and nearly 60%. When r is fixed, a power trendline with equation $t = aN_k^b$ provides very good fit. The values of the parameters (a,b) are (0.001,1.1071) when r = 100, (0.0018,1.1453) when r = 200, (0.0036,1.1124) when r = 300 and (0.0048,1.1152) when r = 400. A similar trend is observed when N_k is fixed, with equation $t = ar^b$. The values of the parameters (a,b) now are (0.0009,1.1359) when N_k = 100, (0.0017,1.1573) when N_k = 200, (0.0026,1.1659) when N_k = 300 and (0.0045,1.1285) when N_k = 400. Thus, when fixing one of the parameters r or N_k, the average time for solving the LP relaxation increases almost linearly with respect to the other parameter.

A decrease in the number of continuous variables of the problem reduces the effort needed for the construction of the multiple choice lists in Step 2 of Algorithm MIMCK. On the other hand, however, this decrease results in large size B&B trees. This happens because, as the multiple choice lists (and therefore master list *ML* too) become more dense in continuous variables, it is more likely that the critical variable of any subproblem will be continuous. This implies that it is also more likely that a subproblem has a feasible mixed integer solution. As a result, the associated B&B tree does not extend to many levels. Therefore, for a fixed number of discrete variables, a decrease in the number of continuous variables is expected to increase the execution time of MIMCK Algorithm. This is in contrast with the typical B&B algorithm that spends most of its time in solving LP relaxations by simplex.

The number of multiple choice sets also has a significant impact on the total computation time. A higher number of multiple choice sets results in more nondominated continuous variables. This increases the density of the problem in continuous variables, which affects the total computational effort as explained above. Therefore, as the same total number of continuous variables is distributed into more multiple choice sets, the total computational effort is expected to decrease. For example, consider the problems with r = 100, N_k = 400, B = 40000, from Table 5.1, with a total of 40,000 continuous variables. In these problems, the number of nondominated continuous variables is 40,000(1-0.9882) = 472. On the other hand, if the same 40,000 variables are spread evenly into 400 multiple choice sets, 40,000(1-0.9631) = 1476 are nondominated, as shown in Table 5.1 for the problems with r = 400, N_k = 100, B = 40,000. This increases the density of the master list *ML* in continuous variables that in turn decreases the expected number of tree nodes generated. With very few exceptions, the results of Table 5.1 are quite in agreement with this behavior.

"You may be disappointed if you fail, but you are doomed if you don't try." **Beverly Sills**

"Only those who dare to fail greatly can ever achieve greatly." **Robert Francis Kennedy**

"I shall take nothing for granted until I have the opportunity of looking personally into it." **Sherlock Holmes**

"Every great advance in science has issued from a new audacity of imagination."

John Dewey

Chapter 6: The Linear Multiple Choice Knapsack Problem with Equity Constraints

6.1 Introduction

In this chapter, the problem that results when the binary variables are dropped from the model of Section 1.3 is treated. The problem is abbreviated as LMCKE, standing for Linear Multiple Choice Knapsack Problem with Equity Constraints and is formulated as follows:

$$
\begin{aligned}
\text{Max} \; & \sum_{k\in S}\sum_{i\in R_k} p_{ki}x_{ki} \\
\text{s.t.} \; & \sum_{k\in S}\sum_{i\in R_k} c_{ki}x_{ki} \le b \\
& \sum_{i\in R_k} x_{ki} \le l_k \,, \forall\, k \in S \\
& L \le \sum_{i\in R_k} c_{ki}x_{ki} \le U \,, \forall\, k \in S \\
& U - L \le f \\
& x_{ki} \ge 0, i \in R_k \,, \forall\, k \in S
\end{aligned}
$$

The auxiliary decision variables *L* and *U* are defined so that the total resource amount allocated to each set *k*, $\sum_{i\in R_k} c_{ki}x_{ki}$, belongs to the interval [*L*,*U*]. The width of this interval is restricted to at most *f*. The above problem is obtained by adding these equity constraints to the LMCK Problem introduced in Chapter 4. It can be used for optimal fund allocation to disjoint sets of continuous highway improvements.

As it was seen in the previous chapters, many different solution techniques have been developed for the LMCK Problem. To the best knowledge of the author, however, the aggregated problem that results from the inclusion of the equity constraints has not been studied before. The contribution of the present chapter is the introduction of a new significant problem, and the methodological and computational advances made for it.

Specifically, a mathematical formulation of the problem is developed of which several important properties are identified. These are utilized to design an optimal two-phased greedy procedure for its solution. Then, the computational complexity of this algorithm is examined and the results of computational experiments illustrating its efficiency are presented.

6.2 Solution Methodology

The algorithm proposed for the solution of LMCKE utilizes the MC variable lists constructed in Step 0 of Algorithm LMCK. It can be divided into two phases. In the first phase, the equity constraints are relaxed and an optimal solution to Problem LMCK is obtained. Starting from this solution, in the second phase, the equity constraints are enforced and an optimal LMCKE solution is obtained. The modified algorithm by Johnson and Padberg (1981), described in Chapter 4, is utilized for the solution of LMCK in Phase I, because it provides valuable information for obtaining the overall optimal solution in Phase II.

6.2.1 Properties of LMCKE

While Problems LMCK and LMCKE exhibit significant differences, they also share common properties allowing Algorithm LMCK to be utilized as a building block of an algorithm for the LMCKE. Next, some useful terminology is introduced and then these important properties are developed.

Consider any feasible solution to the LMCK Problem that possibly violates the equity constraints of LMCKE. The resource amount allocated to a multiple choice set according to this solution is called the *cost* of this set. The terms *maxcost* and *mincost* are used to denote the maximum and minimum resource amount, respectively, that is allocated to a single multiple choice set, according to this solution. Any multiple choice set whose cost differs from maxcost and mincost is called an *internal* set. A multiple choice set whose cost is equal to maxcost or mincost is called an *upper* or a *lower* set, respectively.

Let's focus now on one multiple choice set, say k. Parameter l_k is called the length of set k. The values of the decision variables in set k are called the *partial* solution of set k. If this partial solution contains exactly one positive decision variable with value equal to the length of this set, it is called a *rounded* partial solution. If it contains one or two positive decision variables with fractional values, it is called a *fractional* partial solution.

Finally, the terms increasing and decreasing are used to characterize both multiple choice sets as well as their associated slopes. The *increasing* and *decreasing slopes* of a multiple choice set k are defined as follows: If the partial solution of this set is rounded with $x_{km} = l_k$, then the increasing slope of this set, $\frac{p_{kn} - p_{km}}{c_{kn} - c_{km}}$, is the associated slope of variable x_{kn}, immediately succeeding x_{km} in multiple choice list k. If no variable follows x_{km} in the associated list, then the increasing slope of this set does not exist. If the partial solution of this set is fractional, then the increasing slope of this set is the associated slope of this fractional decision variable (if there are two, consider the one that succeeds the other in the associated list k). The decreasing slope of a set is always equal to the associated slope of its positive decision variable (as before, consider the second variable if two positive variables exist). The increasing slope of a set represents the incremental increase in total profit per unit of resource that is additionally allocated to this set. Similarly, the decreasing slope of a set represents the incremental decrease in total profit per unit of resource that is removed from this set.

In summary, the decreasing slope of a set always exists, assuming that a positive resource amount has been allocated to that set. However, the increasing slope may not exist. This happens for a multiple choice set whose partial solution contains one rounded positive variable and that variable happens to be the last in the associated multiple choice list. Clearly, in this case we cannot allocate additional resource units to this set, since there is no other variable that can be increased. As will be explained later, both the increasing and the decreasing slopes of a set can have positive or negative values.

The terms increasing and decreasing will also be used to characterize sets. If we decide to allocate additional resource units to a set, then this set is called an *increasing set* because the resource amount allocated to it increases. Similarly, if we decide to decrease the resource amount allocated to it, it is called a *decreasing set*.

An important difference between the LMCKE and the LMCK Problems is that variables which qualify for elimination according to Properties 1 or 2 but belong to the right upper hull of the associated multiple choice set may appear in an optimal LMCKE solution. To illustrate why this can happen, consider the following Linear Multiple Choice Knapsack example:

$$\text{Max } 5x_{11} + 3x_{12} + 10x_{21}$$

$$\text{s.t. } 2x_{11} + 4x_{12} + 5x_{21} \leq 9$$

$$x_{11} + x_{12} \leq 1$$

$$x_{21} \leq 1$$

$$x_{11}, x_{12}, x_{21} \geq 0$$

The problem has two multiple choice sets. The first consists of the variables x_{11} and x_{12} and the second of the variable x_{21}. The optimal LMCK solution is $x_{11} = 1$, $x_{12} = 0$ and $x_{21} = 1$. If, however, a maximum difference of one unit is allowed between the resource amounts allocated to the two multiple choice sets, the optimal solution when the equity constraints are included is $x_{11} = 0$, $x_{12} = 1$ and $x_{21} = 1$. Note that, even though variable x_{12} is integer dominated by variable x_{11} ($p_{12}/c_{12} < p_{11}/c_{11}$ and $p_{12} < p_{11}$), it has a positive optimal value. The reason is because x_{12} enlarges the cost domain of the first multiple choice set. The situation is illustrated in Figure 6.1. If only x_{11} is considered in multiple choice set one and x_{12} is eliminated, then at most three units of resource can be allocated to multiple choice set two. This is because the maximum resource amount that can be allocated to multiple choice set one is equal to two, the cost of x_{11}, and additionally, the resource difference between the two sets cannot exceed one unit. On the other hand, when x_{12} is not eliminated, as many as five units can be allocated to multiple choice set two, since $c_{12} = 4$. Hence, although increasing variable x_{12} after the increase of x_{11} is not a profitable action within the first multiple choice set, since there is a negative profit increase, it is an overall optimal action because the profit gained by the further increase of x_{21} is greater.

Based on the above analysis, variables that enlarge the cost domain of the associated multiple choice set should not be eliminated for Problem LMCKE, even if they are dominated according to Properties 1 and 2. The above discussion implies the following result:

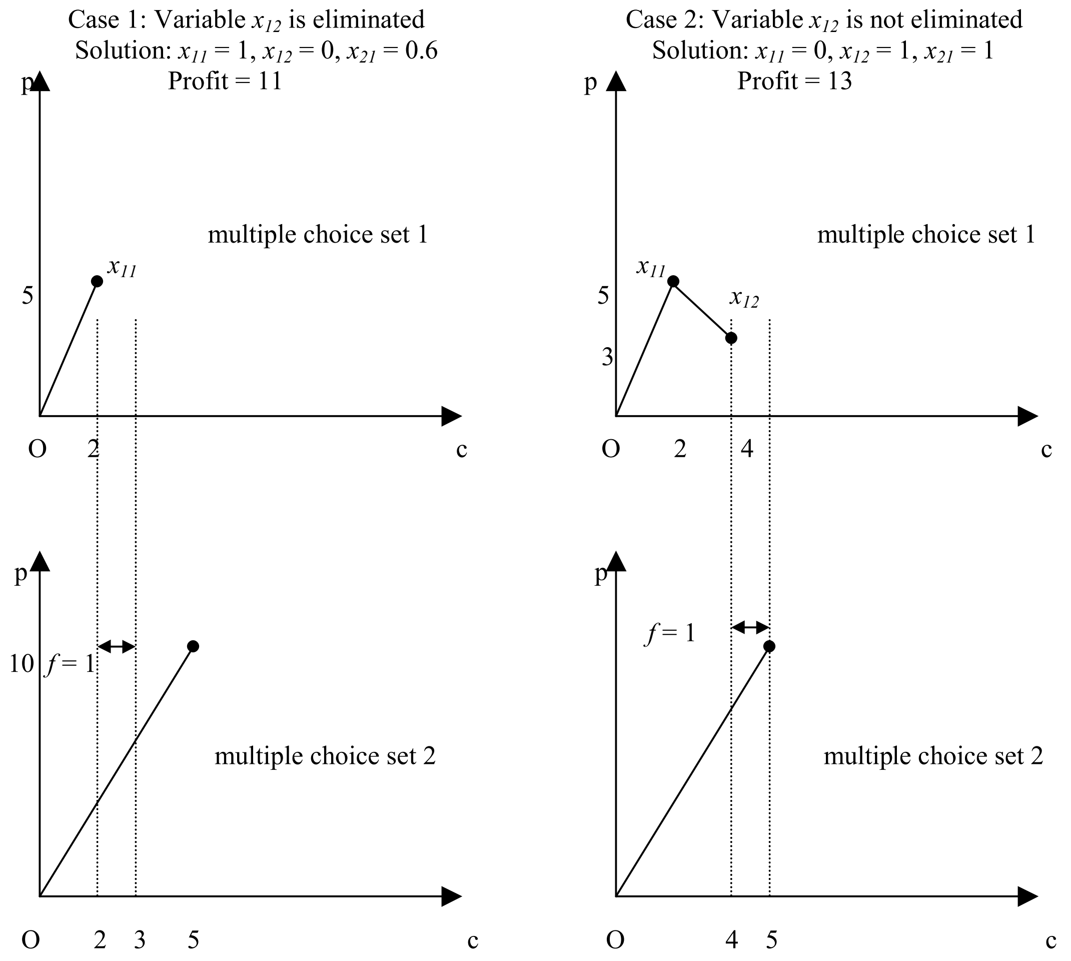

Figure 6.1: An example where an integer dominated variable appears at the optimal solution of LMCKE

Proposition 6.1: Variables that do not belong to the upper hull of the associated multiple choice coefficient space are dominated and can be eliminated from further consideration for LMCKE.

Proof: Consider the variables that define the complete upper hull of a multiple choice set, as shown in Figure 6.2 and assume that the optimal solution of LMCKE contains a variable, x_{kg}, that does not belong to this upper hull. Let x_{kf} and x_{kh} be two adjacent variables from the upper hull such that $c_{kf} \leq c_{kg} \leq c_{kh}$. Note that these variables are always defined uniquely, although x_{kf} may need to be defined as the origin with $c_{kf} = p_{kf} = 0$. Consider the system of equations $c_{kf} z + c_{kh} w = c_{kg} x_{kg}$ and $z + w = x_{kg}$, in the unknowns (z,w). Note that this system always has a unique solution if $c_{kf} \neq c_{kh}$. Consider the alternative solution that results when $x_{kf}' = x_{kf} + z$, $x_{kh}' = x_{kh} + w$ and $x_{kg}' = 0$, while the values of all the other variables remain the same. By the way z and w are defined, the budget, equity and multiple choice constraints are still satisfied. From the values of z and w obtained from the solution of the system we get after some basic manipulation: $p_{kf} z + p_{kh} w > p_{kg} x_{kg}$ <=> ... <=> $\frac{c_{kg} - c_{kh}}{c_{kf} - c_{kh}}(p_{kf} - p_{kh}) > (p_{kg} - p_{kh})$ or $\frac{c_{kg} - c_{kf}}{c_{kh} - c_{kf}}(p_{kh} - p_{kf}) > (p_{kg} - p_{kf})$. The last two expressions are equivalent to $\frac{p_{kf} - p_{kh}}{c_{kf} - c_{kh}} < \frac{p_{kg} - p_{kh}}{c_{kg} - c_{kh}}$ and $\frac{p_{kh} - p_{kf}}{c_{kh} - c_{kf}} > \frac{p_{kg} - p_{kf}}{c_{kg} - c_{kf}}$, respectively, and they are always true by the way x_{kf}, x_{kg} and x_{kh} and their slopes are defined (see Figure 6.2). This contradicts the fact that the current solution is optimal and thus x_{kg} must have a value of 0 at the optimal solution of the problem. Note that, if $c_{kg} = c_{kf}$ or $c_{kg} = c_{kh}$, then the current objective can be improved by substituting directly variable x_{kg} with the upper hull variable at the same cost (x_{kf} or x_{kh}). Therefore, variables that are enclosed in the (complete) upper hull of the associated multiple choice set can always be eliminated for LMCKE. $\square$

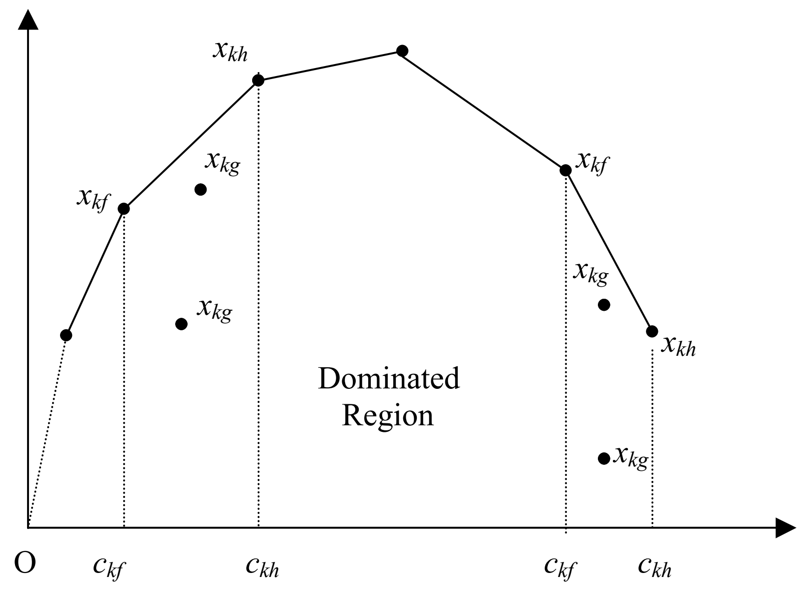

Figure 6.2: Eliminating dominated variables within a MC set for LMCKE

The variables within each list are ordered in increasing order of their costs. On the other hand, the multiple choice constraints restrict the sum of all variables of each set to at most l_k. Therefore, the maximum budget amount, MC_k, that can be allocated to set k is $MC_k = c_{kq}l_k$, where x_{kq} is the variable with the largest cost among all nondominated variables of set k. Let $MC_{min} = \min_{k\in S} MC_k$. By the way the equity constraints are defined, it is clear that the following holds:

Corollary 6.1: The cost of any single set at the optimal solution of LMCKE is at most $MC_{min} + f$.

Let $r = |S|$ be the total number of multiple choice sets. Another upper bound on the cost of any set at the optimal solution of LMCKE is given by the following proposition:

Proposition 6.2: The cost of any single set at the optimal solution of LMCKE cannot be larger than $\frac{b}{r} + \frac{r-1}{r} f$.

Proof: Let g denote the maximum value by which the cost of any single set can exceed the average amount b/r, at the optimal solution of LMCKE. Then, in order for the equity constraints to be satisfied, the cost of any other set must be at least $b/r + g - f$. Therefore, in the extreme case that every other set has a cost of $b/r + g - f$, we must have $(b/r + g) + (r\text{-}1)(b/r + g - f) = b \Rightarrow b + rg - (r\text{-}1)f = b \Rightarrow g = f(r\text{-}1)/r$. If the cost of a set increases beyond $b/r + f(r\text{-}1)/r$, the problem will be infeasible because even if the cost of every other set is at its lowest possible value, the total budget amount required for this solution will be more than b. Thus, the maximum value that the cost of a single variable set can have at the optimal solution of LMCKE is $b/r + g = b/r + f(r\text{-}1)/r$. $\square$

Under the assumption that the total available budget amount is used at the optimal solution of the problem, we can also prove similarly that $b/r - f(r\text{-}1)/r$ is a lower bound of the optimal cost of each variable set. Hence, the following additional result holds:

Corollary 6.2: If the unused budget at the optimal solution of LMCKE is equal to 0, the cost of any single variable set lies in an interval centered at b/r whose width is less than but tends to $2f$ as r goes to ∞.

Corollary 6.2 suggests that, under the assumption that the total available budget is used at the optimal solution of the problem, the interval of width f containing the costs of the variable sets lies entirely in the interval $[b/r - f(r\text{-}1)/r,\ b/r + f(r\text{-}1)/r]$. Of course, it is not possible to know in advance whether the budget residual will be 0 at the optimal solution of the problem. Therefore, in the general case, the lower bound $b/r - f(r-1)/r$ on the optimal cost of each set cannot be used.

Combining Corollary 6.1 and Proposition 6.2, the following result is obtained:

Corollary 6.3: The minimum of $MC_{min} + f$ and $b/r + f(r-1)/r$ is a valid upper bound, UB, on the cost of any single variable set at the optimal solution of LMCKE.

The construction of the MC variable lists of Algorithm LMCK has to be modified slightly to take into account Proposition 6.1. Thus, the variables that belong to the right upper hull have to be added to the set of nondominated variables obtained by Algorithm LMCK. This means than not only the left upper hull up to the point with the highest profit should be considered, but the complete upper hull up to the variable with the highest cost (this variable will always be nondominated). In this way, variables which are dominated according to Properties 1 and 2 and may appear at the optimal solution of the problem will not be eliminated. The situation is illustrated in Figure 6.3. For Problem LMCK, only variables x_{ki}, x_{kj}, x_{kl} and x_{km} are nondominated. For Problem LMCKE, besides these variables, x_{kp} and x_{kq} are nondominated too. This also explains why the increasing or the decreasing slope of a multiple choice set can also take negative values. For example, if the cost allocated to set k, shown in Figure 6.3, is more than $c_{km}l_k$, then both the increasing and the decreasing slope of that set are negative.

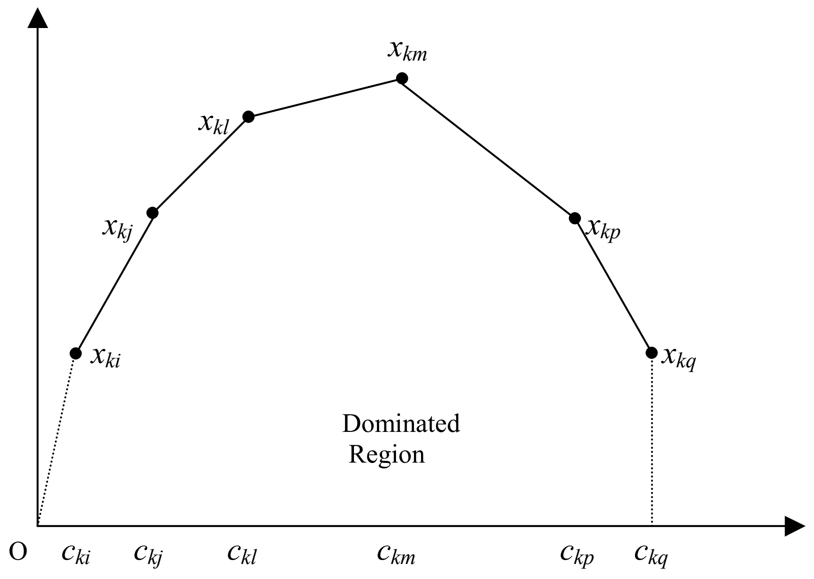

Figure 6.3: Nondominated variables within a multiple choice set for LMCKE

The multiple choice lists show the order in which variables within a multiple choice set should be increased (decreased), when more (fewer) resource units are allocated to that set. Any partial solution of a multiple choice set corresponds to a point that belongs to the upper hull of the variables in this set. By allocating more or fewer resource units to this set, we are simply moving to adjacent points while always staying on this boundary. When the cost of a set increases, we can find the new partial solution of this set by using the order of the variables in the associated multiple choice list. This amounts to performing enough iterations of the Algorithm LMCK to use the additional resource amount. Similarly, when the resource amount allocated to one set decreases, we can find the new partial solution by following the reverse order of the variables in this list. The algorithm is run until the resource amount allocated to this set drops to the desired value.

To illustrate, let's assume that, in Figure 6.3, $p_{ki} = 2$, $p_{kj} = 3$, $p_{kl} = 4$, $c_{ki} = 1$, $c_{kj} = 2$ and $c_{kl} = 4$. Let's also assume that the length of segment k is equal to 1 and its current cost, b_k, is equal to 1.5. The partial solution is: $x_{ki} = 0.5$ and $x_{kj} = 0.5$ with an associated profit of 2.5. This point belongs to the line segment that connects the points corresponding to variables x_{ki} and x_{kj}. If one additional resource unit is allocated to this set, to find the resulting solution we iterate as follows: The current solution was obtained by increasing x_{kj} and decreasing x_{ki} (x_{kj} was replacing x_{ki}) in Step 1.2 of Algorithm LMCK. Since x_{kj} currently has a value of 0.5, $c_{kj} - c_{ki} = 1$ and $l_k = 1$, we need 0.5 resource units so that the replacement of x_{ki} by x_{kj} is complete. At that point we have $x_{ki} = 0$, $x_{kj} = 1$ and $b_{res} = 1 - 0.5 = 0.5$. This solution is represented by the point corresponding to variable x_{kj} in Figure 6.3. The next variable considered for increase is x_{kl}. Since $(c_{kl} - c_{kj})\, l_k = (4-2)1 = 2 > b_{res}$, we set $x_{kl} = b_{res} / (c_{kl} - c_{kj}) = 0.25$, $x_{kj} = l_k - x_{kl} = 0.75$ and we stop because this is the new partial solution for this set. This is exactly one iteration according to Step 1 of Algorithm LMCK. The new solution is represented by a point on the line segment connecting the points corresponding to x_{kj} and x_{kl}. The new contribution of this set to the total profit is $p_{kj} + \dfrac{p_{kl} - p_{kj}}{c_{kl} - c_{kj}}(b_k - c_{kj}) = 3 + \dfrac{4-3}{4-2}(2.5 - 2) = 3.25$. Similarly, if the resource amount allocated to this set decreases by one unit, then we first get the intermediate solution $x_{ki} = 1$, $x_{kj} = 0$, $x_{kl} = 0$ and finally the solution $x_{ki} = 0.5$, $x_{kj} = 0$, $x_{kl} = 0$ with associated contributed profit $p_{ki}x_{ki} = 1$. In this case, variables are decreased in the reverse order in which they appear in the associated multiple choice list and as a result, the total cost and profit also decrease. The above analysis suggests that when the resource amount allocated to some set changes, we can find the new partial solution just by moving on the upper hull boundary of this set, i.e., by increasing or decreasing variables in the order they appear in the associated multiple choice list.

6.2.2 A Greedy Algorithm for LMCKE

In the first phase of the algorithm proposed for Problem LMCKE, the equity constraints are relaxed and the resulting LMCK Problem is solved using Algorithm LMCK. The second phase starts with this superoptimal solution that only violates the equity constraints and tries to reach feasibility while maintaining superoptimality. The algorithm terminates as soon as the equity constraints are satisfied.

Phase II of the algorithm works as follows: The costs of the multiple choice sets in the solution obtained from Phase I are arranged in increasing order. The difference between the smallest and the largest of these sets is denoted by f_a and represents the actual maximum resource difference between any two sets. If this difference is less than or equal to the desired value, f, then this solution is optimal for LMCKE. Otherwise, the differences in the costs of sets exceeding this value are iteratively narrowed until the equity constraints are satisfied. This is done in such a way that superoptimality is maintained.

At each iteration there are five distinct options for decreasing the value of f_a. Out of these options, the optimal is selected and carried out. The criterion used for the selection of the next option is the incremental loss in total profit per unit decrease in the maximum resource amount difference. In other words, if ΔP and Δf_a are the resulting differences in total profit and in f_a, respectively, then at each iteration the option that yields the minimum loss in total profit per unit decrease in f_a is selected. These actions are illustrated in Figure 6.4. The computation of $\Delta P/\Delta f_a$ for each of them is described next. The following additional notation is used:

m = number of upper sets,

n = number of lower sets,

a = sum of decreasing slopes of upper sets,

e = sum of increasing slopes of lower sets,

u = decreasing slope of decreasing set,

o = increasing slope of increasing set.

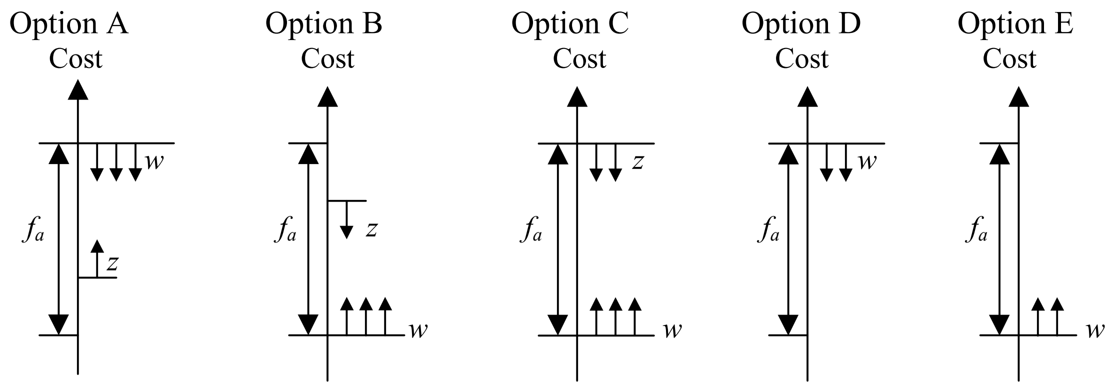

Figure 6.4: The five best options for decreasing the value of f_a

Option A: Decrease the resource amount allocated to all the upper sets and reallocate this amount to the internal or lower set with the maximum increasing slope (increasing set).

Let w be the amount by which the resource of the upper sets will be decreased and z the amount by which the resource of the increasing set will be increased. Then, we have: $mw = z$. The amount Δf_a by which f_a will be decreased is equal to w. Therefore, the value of ΔP is $oz - aw = omw - aw = (om-a)\Delta f_a$.

Option B: Decrease the resource amount allocated to the internal or upper set with the minimum decreasing slope (decreasing set) and reallocate this amount to the lower sets.

Let w be the amount by which the resource of the lower sets will be increased and z the amount by which the resource of the decreasing set will be decreased. Then, we have: $nw = z$. The amount Δf_a by which f_a will be decreased is equal to w. Therefore, the value of ΔP is $ew\text{-}uz = ew\text{-}unw = (e\text{-}un)\Delta f_a$.

Option C: Decrease the resource amount allocated to all the upper sets and reallocate this amount to all the lower sets.

Let w be the amount by which the resource of the lower sets will be increased and z the amount by which the resource of the upper sets will be decreased. Then, the amount Δf_a by which f_a will be decreased is equal to $z + w$. We also have $mz = nw$. Therefore, the value of ΔP for this option is $ew\text{-}az = \dfrac{me - na}{m + n}\,\Delta f_a$.

Option D: Decrease the resource amount allocated to all the upper sets.

Let w be the amount by which the cost of the upper sets will be decreased. Then, the value of ΔP for this case is $–aw = \text{-}a\Delta f_a$. At the same time, the resource residual increases by $mw = m\Delta f_a$.

Option E: Increase the resource amount allocated to all the lower sets.

Let w be the amount by which the cost of the lower sets will be increased. Then, the value of ΔP for this case is $ew = e\Delta f_a$. At the same time, the resource residual decreases by $nw = n\Delta f_a$.

It should be noted that the lower (upper) set was assumed to be non unique in Option A (Option B). Although Option C is really a special case of both Options A and

B, it is treated separately. Option A reduces to Option C if the increasing set is the unique lower set. Similarly, Option B reduces to Option C if the decreasing set is the unique upper set. Option D is a special case in which the resource amount removed from a set is not allocated anywhere else but is kept for future use. A case where it is better to keep rather than reallocate the resource amount recovered from a set is when the increasing slopes of all the lower and internal sets are negative. Similarly, Option E is a special option that can only be selected if at the time that it is considered there is a positive resource residual.

Once a decision is made and one of the above five options is selected, a stopping criterion determines how far this iteration is carried out. In general, we can say that the algorithm stops as soon as either some of the ratios $\Delta P/\Delta f_a$ change, or f_a becomes equal to f. In the latter case, the algorithm stops simply because the equity constraints are satisfied and therefore the solution at hand is optimal. In the former case, the algorithm stops because new ratios $\Delta P/\Delta f_a$ have to be computed at that point and compared for the five options. Depending on which of the five options is chosen the stopping conditions are:

Option A: There are five stopping conditions in this case. The first is when the decreasing slope of one of the decreasing upper sets changes. The second comes about when the increasing slope of the increasing set changes. In both of these cases, the ratio $\Delta P/\Delta f_a$ of some of the five options changes and has to be recomputed. The third condition is when the cost of the decreasing upper sets becomes equal to the cost of the increasing set. The fourth arises when the cost of the decreasing upper sets becomes equal to the cost of one of the internal sets. In either of these cases, if we continue this iteration, the value of f_a will not decrease further from its present value. The final condition appears when the value of f_a becomes equal to f.

Option B: There are also five stopping conditions for this option and they are symmetric to the ones for Option A. Specifically, the first: when the increasing slope of one of the increasing lower sets changes. The second: when the decreasing slope of the decreasing set changes. The third: when the cost of the increasing lower sets becomes equal to the cost of the decreasing set. The fourth: when the cost of the increasing lower sets becomes equal to the cost of one of the internal sets. The final condition: when the value of f_a becomes equal to f.

Option C: The five stopping conditions for this option are similar to the conditions for options A and B. The first: when the increasing slope of one of the increasing lower sets changes. The second: when the decreasing slope of one of the decreasing upper sets changes. The third: when the cost of the increasing lower sets becomes equal to the cost of one of the internal sets. The fourth: when the cost of the decreasing upper sets becomes equal to the cost of one of the internal sets. The final condition: when the value of f_a becomes equal to f.

Option D: In this case there are only three stopping conditions. The first arises when the decreasing slope of one of the decreasing upper sets changes. The second is met when the cost of the decreasing upper sets becomes equal to the cost of one of the internal sets. The third condition is met when the value of f_a becomes equal to f.

Option E: In this final case, there are four stopping conditions. As mentioned before, when this action is selected, there is a positive resource residual that can be used for allocation. The first condition appears when the increasing slope of one of the increasing lower sets changes. The second comes about when the cost of the increasing lower sets becomes equal to the cost of an internal set. The third condition is met when the resource residual decreases to zero while the fourth when the condition on f is satisfied.

Using the above stopping criteria we are able to determine the extent of the current iteration. Each of the above conditions defines some value for Δf_a. The minimum of these values for the conditions that apply is the value by which f_a will decrease in this iteration. If the equity constraints are still not satisfied, the new ratios for the five options are recomputed and compared and a new option is selected. It is illustrated next how Δf_a is computed for Option A. For each of the other options, the calculation is similar.

For each of the upper sets whose cost is decreased, the first condition is met when the variable immediately preceding the variable that was last increased at the partial solution of that set gets a value equal to the length of that set. Supposing that Figure 6.3 refers to an upper set and the starting partial solution of this set is represented by any of the points on the line connecting x_{ki} and x_{kj} (excluding the point representing x_{ki}), this corresponds to the point representing x_{ki}. If b_k is the cost of this set at the current solution and c_{ki} the cost of that variable, then the value of Δf_a for this set is equal to $b_k - c_{ki}l_k$. Of course, if no such variable exists, then only the first variable of this list has been increased and therefore the first condition is met when the cost of this set drops to 0. The first stopping condition is defined by the minimum of the Δf_a values computed this way for each of the upper sets. The value of Δf_a for the second stopping condition is defined once the variable that is currently increased in the increasing set gets a value equal to the length of that set. This solution corresponds to a point such as the one representing x_{kj} in Figure 6.3. If b_k is the cost of this set in the current solution and c_{kj} the cost of that variable, then the value of Δf_a defined from the second condition is equal to $(c_{kj}l_k - b_k)/m$. For the third condition, let b_u and b_i be the cost of the upper sets and the increasing set, respectively, for the current solution of the problem. The third condition is met when $b_u - w = b_i + z$ or $b_u - \Delta f_a = b_i + m\Delta f_a \Rightarrow \Delta f_a = (b_u - b_i)/(m + 1)$. Similarly, if b_o is the maximum cost spent on an internal set, then the value of Δf_a defined by the fourth stopping condition is $b_u - b_o$. Finally, the value of Δf_a defined by the fifth condition is $f_a - f$.

At each iteration, after the best option is selected, the value of Δf_a is computed using the above procedure and the new solution is found. The algorithm terminates with an optimal solution when f_a becomes equal to the value of f. Next, the algorithm developed for Problem LMCKE is formally introduced.

Algorithm LMCKE

Phase I (Optimal solution of relaxed problem)

Step 0 (Construction of multiple choice lists)

Using the input data compute the value of *UB*.

For each multiple choice set k, construct a multiple choice list L_k of variables as follows: Identify the nondominated variables in this set and order them by increasing costs. Stop adding variables to list k as soon as the first variable with cost $\geq UB/l_k$ is appended. Variables succeeding this variable (i.e., variables with greater cost) can be eliminated. Define the associated slope of the first variable x_{ki} in each multiple choice list k as the ratio p_{ki}/c_{ki}. For each subsequent variable x_{kj} in the list, define its associated slope as $(p_{kj} - p_{ki})/(c_{kj} - c_{ki})$, where x_{ki} is the variable immediately preceding x_{kj} in L_k. While maintaining all individual multiple choice lists, create a master list, *ML*, by merging the variables of all multiple choice lists in nonincreasing order of their associated slopes. Initialize the objective function value to 0 and the budget residual, b_{res}, to b.

Step 1 (Iteration)

```
while ($b_{res} > 0$ & the next variable from $ML$ exists & the associated slope
of this variable is > 0) do{
    if the cost of the set containing this variable is less than $UB$
        iterate as in Step 1 of Algorithm LMCK
    else
        go to the next variable from $ML$
}end while
```

Phase II (Optimal solution of LMCKE)

```
while ($f_a > f$) do{
    select option with minimum loss in total profit per unit decrease in $f_a$,
    find stopping condition and compute optimal $\Delta f_a$,
    iterate, update solution and find new value of $f_a$
}end while ▭
```

When the first variable with non-positive associated slope is encountered, Step 1 of Phase I of Algorithm LMCKE terminates because iterating further will decrease the total profit. In the same step, a variable of a set is only increased if the cost of this set is less than *UB*, in accordance with Corollary 6.3. Thereafter, such a set is skipped rather than considered for further resource allocation. The validity of the above algorithm is summarized in the following result:

Proposition 6.3: Algorithm LMCKE correctly produces an optimal solution to Problem LMCKE.

Proof: The above modification of Algorithm LMCK terminates with a valid optimal solution to Problem LMCK in Phase I of Algorithm LMCKE, when an upper bound *UB* to the optimal cost of each set is imposed. The five options considered at each iteration in Phase II of Algorithm LMCKE are the best choices for decreasing the value of f_a. The option selected at each iteration is the one that results in the smallest decrease in the total profit per unit decrease in the value of f_a. During any iteration, the ratios $\Delta P/\Delta f_a$ of the five options do not change. Thus, after an iteration terminates, the solution obtained is optimal to the special LMCKE Problem that has the current value of f_a as value of f. Therefore, as soon as the value of f_a becomes equal to f, the solution at hand is optimal for LMCKE. ▭

Another interesting result can be obtained by further analyzing Algorithm LMCKE:

Corollary 6.4: In the optimal solution of LMCKE, each of the multiple choice sets has at most two positive decision variables. If a set has exactly two positive variables then these are adjacent in the associated list and they both have fractional values. Moreover, their sum is equal to the length of that set.

6.2.3 Illustration of the Algorithm

Having introduced Algorithm LMCKE and proved its validity, its application is illustrated next on a small numerical example. Consider the problem with eight multiple choice sets and the following data:

Table 6.1: Data for the LMCKE numerical example

Variable	Profit	Cost	Variable	Profit	Cost	Variable	Profit	Cost
x_{11}	7	3	x_{33}	7	9	x_{62}	5	10
x_{12}	8	4	x_{34}	1	1	x_{63}	7	5
x_{13}	5	2	x_{41}	4	1	x_{64}	8	8
x_{14}	7	15	x_{42}	6	4	x_{71}	3	2
x_{21}	2	1	x_{43}	7	7	x_{72}	4	3
x_{22}	6	6	x_{51}	6	3	x_{73}	6	7
x_{23}	4	3	x_{52}	8	10	x_{81}	3	7
x_{24}	7	12	x_{53}	9	9	x_{82}	6	10
x_{31}	2	7	x_{54}	7	4	x_{83}	4	5
x_{32}	5	6	x_{61}	6	4	x_{84}	3	3

We also assume that $b = 25$, $l_k = 1$ for all k and $f = 2$.

Phase I:

The eight multiple choice lists are: $L_1 = \{x_{13}, x_{11}, x_{12}, x_{14}\}$, $L_2 = \{x_{21}, x_{23}, x_{22}, x_{24}\}$, $L_3 = \{x_{34}, x_{32}, x_{33},\}$, $L_4 = \{x_{41}, x_{42}, x_{43}\}$, $L_5 = \{x_{51}, x_{54}, x_{53}, x_{52}\}$, $L_6 = \{x_{61}, x_{63}, x_{64}, x_{62}\}$, $L_7 = \{x_{71}, x_{72}, x_{73},\}$ and $L_8 = \{x_{84}, x_{83}, x_{82}\}$. Variables x_{31} and x_{81} do not appear in the associated lists because they are dominated. As an example, multiple choice list 1 was constructed as follows: It contains four nondominated variables. These are arranged in increasing order of their costs. The associated slope of variable x_{13} is $p_{13}/c_{13} = 2.5$, while the associated slopes of x_{11}, x_{12} and x_{14} are $\dfrac{p_{11} - p_{13}}{c_{11} - c_{13}} = 2$, $\dfrac{p_{12} - p_{11}}{c_{12} - c_{11}} = 1$ and $\dfrac{p_{14} - p_{12}}{c_{14} - c_{12}} =$ -0.09, respectively. Note that, by construction, variables appear in this list in decreasing order of their associated slopes. The multiple choice lists of the remaining seven sets are constructed similarly.

Based on these lists of variables and their associated slopes, the master list of increasing variables is $ML = \{x_{41}, x_{13}, x_{11}, x_{21}, x_{51}, x_{61}, x_{71}, x_{12}, x_{23}, x_{34}, x_{54}, x_{63}, x_{72}, x_{84}, x_{32},$

x_{22}, x_{33}, x_{42}, x_{73}, x_{83}, x_{53}, x_{82}, x_{43}, x_{64}, x_{24}, x_{14}, x_{52}, $x_{62}\}$. When there is a tie between variables to append to *ML* next, the decision is made arbitrarily. The value of MC_{min} is 7 (for $k = 4$ and $k = 7$). Therefore, the first upper bound on the cost of each set is $MC_{min} + f = 7 + 2 = 9$. The second upper bound on the cost of a set as determined by Proposition 6.2 is $b/r + f(r - 1)/r = 25/8 + 2(8-1)/(8) = 4.875$. Hence, the overall upper bound, *UB*, on the optimal cost of each set is equal to min(9, 4.875) = 4.875. The solution obtained from Phase I of Algorithm LMCKE is the following (only nonzero variables are shown):

MC set 1 (internal set): $x_{12} = 1$ with cost 4, profit 8, increasing slope -0.09 and decreasing slope 1.

MC set 2 (internal set): $x_{23} = 1$ with cost 3, profit 4, increasing slope 0.67 and decreasing slope 1.

MC set 3 (internal set): $x_{34} = 0.775$, $x_{32} = 0.225$ with cost 2.125, profit 1.9, increasing slope 0.8 and decreasing slope 0.8.

MC set 4 (lower set): $x_{41} = 1$ with cost 1, profit 4, increasing slope 0.67 and decreasing slope 4.

MC set 5 (internal set): $x_{54} = 1$ with cost 4, profit 7, increasing slope 0.4 and decreasing slope 1.

MC set 6 (upper set): $x_{61} = 0.125$, $x_{63} = 0.875$ with cost 4.875, profit 6.875, increasing slope 1 and decreasing slope 1.

MC set 7 (internal set): $x_{72} = 1$ with cost 3, profit 4, increasing slope 0.5 and decreasing slope 1.

MC set 8 (internal set): $x_{84} = 1$ with cost 3, profit 3, increasing slope 0.5 and decreasing slope 1.

The total profit for this solution is 38.775 and the value of f_a is 3.875. The total available budget amount is used for this solution.

Phase II:

Since $f_a > f$, we compute the ratio $\Delta P/\Delta f_a$ for each of the five options. The internal or lower set with the largest increasing slope is set 3. If the cost allocated to multiple choice set 6 is decreased by Δf_a and the cost allocated to multiple choice set 3 is increased by the same amount, the net change in profit is $(0.8 - 1)\Delta f_a$. Therefore, the ratio $\Delta P/\Delta f_a$ for option A is -0.2. The internal or upper set with the smallest decreasing slope is multiple choice set 3. If the cost allocated to multiple choice set 4 is increased by Δf_a and the cost allocated to multiple choice set 3 is decreased by the same amount, the net change in profit is $(0.67 - 0.8)\Delta f_a$. Therefore, the ratio $\Delta P/\Delta f_a$ for option B is -0.130. If the cost allocated to multiple choice set 4 is increased by $\Delta f_a/2$ and the cost allocated to multiple choice set 6 is decreased by the same amount (so that the total decrease of f_a is Δf_a), the net change in profit is $(0.67-1)\Delta f_a/2$. Therefore, the ratio $\Delta P/\Delta f_a$ for option C is -0.165. If the cost allocated to multiple choice set 6 is decreased by Δf_a, then the net change in profit is $-1\Delta f_a$. Therefore, the ratio $\Delta P/\Delta f_a$ for option D is -1. Finally, the ratio $\Delta P/\Delta f_a$ for option E is not defined, since the budget residual for the present solution is not positive. Comparing the ratios $\Delta P/\Delta f_a$, the best option is B.

Next, the value of Δf_a defined by the stopping conditions for this iteration must be determined. The increasing slope for multiple choice set 4 changes when its cost becomes equal to 4. Since the current cost of this set is 1, the first stopping condition is met for $\Delta f_a = 3$. The decreasing slope for multiple choice set 3 changes when its cost drops to 1. Since the current cost of this set is 2.125, the second stopping condition is met for $\Delta f_a =$

1.125. If $\Delta f_a = 0.5625$, the costs of multiple choice set 4 and multiple choice set 3 become equal. Therefore, the third stopping condition is met for $\Delta f_a = 0.5625$. The fourth stopping is met at the time that the cost of multiple choice set 4 becomes equal to 3. Therefore, the value of Δf_a defined from this condition is 2. Finally, the value of Δf_a defined by the fifth condition is 1.875, since $f = 2$ and $f_a = 3.875$. Comparing the value of Δf_a determined by the various stopping conditions, we conclude that the iteration should be terminated for $\Delta f_a = 0.5625$, since this is the minimum among them.

Next, we perform the iteration and update the solution that we have. During this iteration, only the partial solutions for multiple choice sets 3 and 4 change. The new solutions for these two sets are:

MC set 3 (lower set): $x_{34} = 0.8875$, $x_{32} = 0.1125$ with cost 1.5625, profit 1.45, increasing slope 0.8 and decreasing slope 0.8.

MC set 4 (lower set): $x_{41} = 0.8125$, $x_{42} = 0.1875$ with cost 1.5625, profit 4.375, increasing slope 0.67 and decreasing slope 0.67.

The total profit for this new solution is 38.7, the budget residual is still 0 and the value of f_a decreased to 3.3125. The algorithm continues in the same way and the optimal solution to the problem is: $x_{12} = 1$, $x_{23} = 1$, $x_{34} = 0.8$, $x_{32} = 0.2$, $x_{41} = 0.667$, $x_{42} = 0.333$, $x_{54} = 1$, $x_{61} = 1$, $x_{72} = 1$ and $x_{84} = 1$ with all the other variables equal to 0. The objective function value is 38.467 and of course we have $f_a = f = 2$.

6.3 Computational Implementation

In this section, the complexity of Algorithm LMCKE is discussed and then the results of the computational experiments conducted are presented. The conclusions reached from the analysis of these results are also summarized.

6.3.1 Computational Complexity

To analyze the complexity of Algorithm LMCKE, each of its two phases is considered separately. The time needed for the construction of the multiple choice lists in Step 0 of Algorithm LMCKE is $\text{O}(\sum_{k\in S} N_k \, log \, N_k) = \text{O}(N \, log \, N_{max})$, where N_k is the number of variables in MC set k, $N = \sum_{k\in S} N_k$ and $N_{max} = \max_{k\in S} N_k$. As already mentioned, the nondominated variables form the upper hull of all the variables in the same set. Therefore, to identify them, any O(n log n) algorithm for finding the convex hull of n points in two dimensions can be used. The time needed for merging the individual multiple choice lists to obtain master list ML is O(N log r) (see also Cormen et al., 2001). The work needed in Step 0 of Phase I Algorithm LMCKE dominates the work needed for the increase of the decision variables in Step 1, which is linear in the total number of variables. Therefore, the worst case complexity of Phase I is O(N log N_{max}) + O(N log r) = O(N log m), where m = max(N_{max}, r).

Consider now Phase II of the algorithm. The work needed at each iteration to determine the upper, lower and internal sets is bounded above by O(r). Once this is done, the work needed to find the ratios $\Delta P/\Delta f_a$ for each of the five options, to find the optimal value of Δf_a and to update in order to obtain the new solution is also bounded above by

O(r). The number of Phase II iterations is affected by the number of times each of the different stopping conditions applies which is not known in advance. Therefore, in what follows, the average-case performance of Algorithm LMCKE is examined by analyzing the computational experiments that were conducted.

6.3.2 Computational Experiments

Algorithm LMCKE was coded in C and tested on a Pentium III/600 MHz processor. The results obtained are presented in Tables 6.2-6.5. In these tables, r denotes the number of multiple choice sets and N_k the number of variables in set k, as before, which was assumed to be the same for all k.

Two types of problems were tested. The first considered both dominated and nondominated variables among the initial variables of the problem. Problems of this type resemble the problems that arise in real world applications. In the second, the variable coefficients were randomly generated in such a way that no variable was dominated. Problems of this type provide good insight into the performance of the algorithm, yet would rarely arise in practice.

For problems with dominated variables, parameters p_{ki} and c_{ki} were uniformly distributed between 0 and N_k. The budget amount b was set equal to $\frac{1}{2}\sum_{k\in S}(\min_{i\in R_k} c_{ki} + \max_{i\in R_k} c_{ki})$, which is on the average equal to 0.5(rN_k). The value of f was set to 0.001b, i.e., 0.0005(rN_k) on the average. The value of l_k was set to 1 for all k.

For this type of problems, computational time showed significant variance. As a result, 30 random instances were solved for each problem size. The results reported in columns 3-6 of Table 6.2 are the average and the maximum time in seconds needed for Phase I and for total execution of the algorithm, respectively. The last column in this table presents the percentage of total variables that were eliminated, either because they were enclosed in the upper hull of the associated set, or because they were eliminated as a consequence of Corollary 6.3. It is clear from this column that the vast majority of the initial variables are dominated. The average time needed for Phase II when r or N_k are fixed is also depicted in Figures 6.5 and 6.6, respectively.

Tables 6.3 and 6.4 present results indicating the sensitivity of the total computational time and of the objective function value to changes in f, for problems with dominated variables. For each problem size, the results for a single instance are shown to ensure that the averages over all instances do not blur the analysis.

The third column of Table 6.3 shows, for each problem size, the total computational time, required for Phase I. The same instance was then solved successively with smaller values of f. In the three middle columns, the changes in total computational time are shown when the value of f decreases by the same quantity (25% of its original value each time), up to the point where it becomes four times smaller than its initial value. In the last three columns, computational time results are reported when the value of f decreases by the same percentage (90% each time), up to the point where it becomes 1000 times smaller than its initial value. In Table 6.4, the changes in the objective function value are reported for the same changes in f as above.

Table 6.5 shows computational time results for the scenario involving problems with nondominated variables only. In this case, the variance exhibited among problems of the same size was not significant and as a result, only 10 instances were solved for each problem size. The variable coefficients were generated differently than for the previous scenario to ensure that no dominated variables were among the initial variables of the problem. More specifically, these variables were constructed in such a way that they all belonged to the upper hull of the associated set. Using random numbers, it was possible to generate the cost (for variables in the left upper hull) or the profit (for variables in the right upper hull) and the associated slope of each of these variables. Once these two parameters for each variable were known, the third (profit or cost) was uniquely defined. All distributions used in this procedure were uniform. The intention was to have similar distributions as in the case with dominated variables. Depending on the random number sequence however, the cost or the profit of a variable could turn out to be more than N_k. The budget amount b was set equal to $rc_u/2$, where c_u was the variable with the smallest cost among all variables that appeared last in their associated MC list. The length of each set was set equal to one as before. The value of f was set equal to $0.001f_a$. The reason a different value of f than before was used is because the previous one was no longer very restricting in this scenario, thus leading to easier problems. The average total computational time to solve these problems when r or N_k are fixed is also depicted in Figures 6.7 and 6.8, respectively.

Sensitivity analysis results for the problems with only nondominated variables are not reported. For these problems, the initial value of f_a is small - less than 10 for all problem sizes. As a result, the computational effort needed to find an optimal solution is not very large, as shown in Table 6.5. Additionally, the changes in the objective function value are negligible, even when the value of f is 1000 times smaller than the value of f_a.

Table 6.2: Computational results for problems with dominated variables (in seconds)

		Phase I		Total time		%
r	N_k	Avg	Max	Avg	Max	domin.
100	100	0.053	0.057	0.098	0.156	95.74
100	200	0.097	0.105	0.142	0.189	97.67
100	300	0.137	0.139	0.176	0.251	98.33
100	400	0.187	0.191	0.227	0.260	98.68
100	500	0.238	0.239	0.275	0.341	98.94
200	100	0.096	0.101	0.421	0.861	95.67
200	200	0.177	0.181	0.558	0.997	97.59
200	300	0.257	0.261	0.564	0.972	98.30
200	400	0.341	0.349	0.643	1.082	98.67
200	500	0.420	0.423	0.752	1.082	98.91
300	100	0.139	0.140	1.232	2.433	95.58
300	200	0.253	0.270	1.237	2.464	97.54
300	300	0.368	0.371	1.342	2.934	98.26
300	400	0.475	0.481	1.518	2.864	98.64
300	500	0.593	0.601	1.558	3.204	98.88
400	100	0.181	0.190	2.020	3.946	95.43
400	200	0.330	0.339	2.235	3.805	97.47
400	300	0.472	0.481	2.457	4.346	98.22
400	400	0.642	0.651	2.733	5.317	98.63
400	500	0.772	0.781	2.823	5.618	98.87
500	100	0.224	0.240	2.749	5.788	95.32
500	200	0.409	0.413	3.185	7.001	97.41
500	300	0.583	0.591	3.293	8.452	98.19
500	400	0.769	0.776	3.953	8.763	98.59
500	500	0.956	0.962	4.027	8.652	98.84

Table 6.3: Sensitivity analysis of computational time (in seconds) with respect to changes in f for problems with dominated variables

r	N_k	Initial	$0.75f$	$0.50f$	$0.25f$	$0.1f$	$0.01f$	$0.001f$
100	100	0.051	0.065	0.079	0.092	0.114	0.165	0.166
100	200	0.099	0.112	0.126	0.147	0.173	0.185	0.186
100	300	0.137	0.149	0.153	0.165	0.178	0.192	0.193
100	400	0.186	0.201	0.205	0.213	0.225	0.237	0.238
100	500	0.239	0.252	0.256	0.265	0.271	0.278	0.281
200	100	0.097	0.121	0.157	0.254	0.315	0.356	0.357
200	200	0.175	0.198	0.244	0.349	0.592	0.777	0.823
200	300	0.252	0.275	0.308	0.365	0.423	0.446	0.456
200	400	0.341	0.364	0.423	0.669	0.916	0.985	1.008
200	500	0.421	0.467	0.502	0.560	0.636	0.659	0.670
300	100	0.138	0.184	0.299	0.472	0.587	0.725	0.748
300	200	0.251	0.319	0.420	0.625	0.739	0.852	0.864
300	300	0.370	0.426	0.514	0.716	1.131	1.332	1.353
300	400	0.473	0.528	0.637	0.807	0.936	1.022	1.044
300	500	0.591	0.700	0.842	1.180	1.397	1.645	1.657
400	100	0.181	0.259	0.506	0.899	1.235	1.484	1.495
400	200	0.335	0.468	0.779	1.881	2.273	2.506	2.551
400	300	0.474	0.549	0.742	1.280	1.510	1.705	1.737
400	400	0.643	0.741	0.915	1.329	1.983	2.220	2.230
400	500	0.778	0.897	1.555	2.138	2.442	2.646	2.667
500	100	0.223	0.391	1.348	2.901	3.997	4.566	4.588
500	200	0.406	0.555	0.951	2.473	3.429	4.543	4.618
500	300	0.583	0.760	1.742	3.365	4.479	5.009	5.042
500	400	0.771	0.930	1.251	2.001	4.096	5.823	5.834
500	500	0.955	1.156	2.059	4.063	4.895	5.301	5.332

Table 6.4: Sensitivity analysis of objective function value with respect to changes in f for problems with dominated variables

r	N_k	Initial	$0.75f$	$0.50f$	$0.25f$	$0.1f$	$0.01f$	$0.001f$
100	100	9917	9911	9897	9873	9851	9834	9832
100	200	19903	19898	19888	19867	19847	19831	19829
100	300	29883	29880	29873	29852	29830	29811	29809
100	400	39902	39899	39887	39866	39846	39830	39828
100	500	49915	49912	49903	49884	49864	49846	49845
200	100	19811	19801	19775	19723	19669	19628	19623
200	200	39799	39792	39771	39725	39681	39647	39643
200	300	59761	59753	59730	59687	59646	59614	59610
200	400	79778	79770	79747	79698	79654	79618	79614
200	500	99813	99806	99780	99735	99692	99659	99655
300	100	29725	29714	29678	29604	29534	29481	29475
300	200	59695	59681	59645	59570	59501	59448	59442
300	300	89711	89702	89676	89609	89536	89479	89473
300	400	119704	119691	119654	119583	119514	119461	119456
300	500	149696	149686	149650	149581	149517	149464	149458
400	100	39605	39593	39551	39463	39378	39311	39304
400	200	79604	79586	79536	79431	79324	79243	79234
400	300	119630	119617	119572	119477	119377	119306	119298
400	400	159587	159574	159534	159444	159352	159283	159275
400	500	199603	199589	199542	199450	199322	199248	199240
500	100	49429	49412	49359	49247	49167	49071	49061
500	200	99483	99465	99402	99280	99180	99095	99085
500	300	149518	149502	149446	149332	149148	149050	149039
500	400	199504	199486	199430	199307	199230	199143	199134
500	500	249511	249493	249430	249291	249180	249088	249077

Table 6.5: Computational time results (in seconds) for problems with nondominated variables only

		Phase I		Total time		%
r	N_k	Avg	Max	Avg	Max	domin.
100	100	0.138	0.141	0.170	0.172	48.29
100	150	0.239	0.241	0.267	0.271	48.59
100	200	0.361	0.362	0.396	0.399	48.85
100	250	0.519	0.521	0.555	0.559	48.95
100	300	0.696	0.701	0.731	0.734	49.10
150	100	0.217	0.221	0.302	0.309	48.37
150	150	0.374	0.381	0.458	0.467	48.65
150	200	0.566	0.571	0.654	0.661	48.89
150	250	0.802	0.811	0.893	0.902	49.09
150	300	1.076	1.082	1.168	1.173	49.11
200	100	0.294	0.301	0.466	0.471	48.36
200	150	0.505	0.511	0.680	0.684	48.72
200	200	0.764	0.772	0.944	0.952	48.99
200	250	1.082	1.084	1.267	1.273	48.89
200	300	1.447	1.452	1.636	1.643	49.14
250	100	0.371	0.372	0.680	0.691	48.34
250	150	0.632	0.641	0.947	0.952	48.79
250	200	0.961	0.962	1.279	1.282	48.95
250	250	1.370	1.432	1.698	1.763	49.08
250	300	1.816	1.823	2.144	2.154	49.14
300	100	0.465	0.471	0.970	0.982	48.37
300	150	0.786	0.791	1.300	1.305	48.74
300	200	1.190	1.192	1.713	1.721	48.97
300	250	1.674	1.682	2.201	2.213	49.09
300	300	2.239	2.253	2.800	3.144	49.20

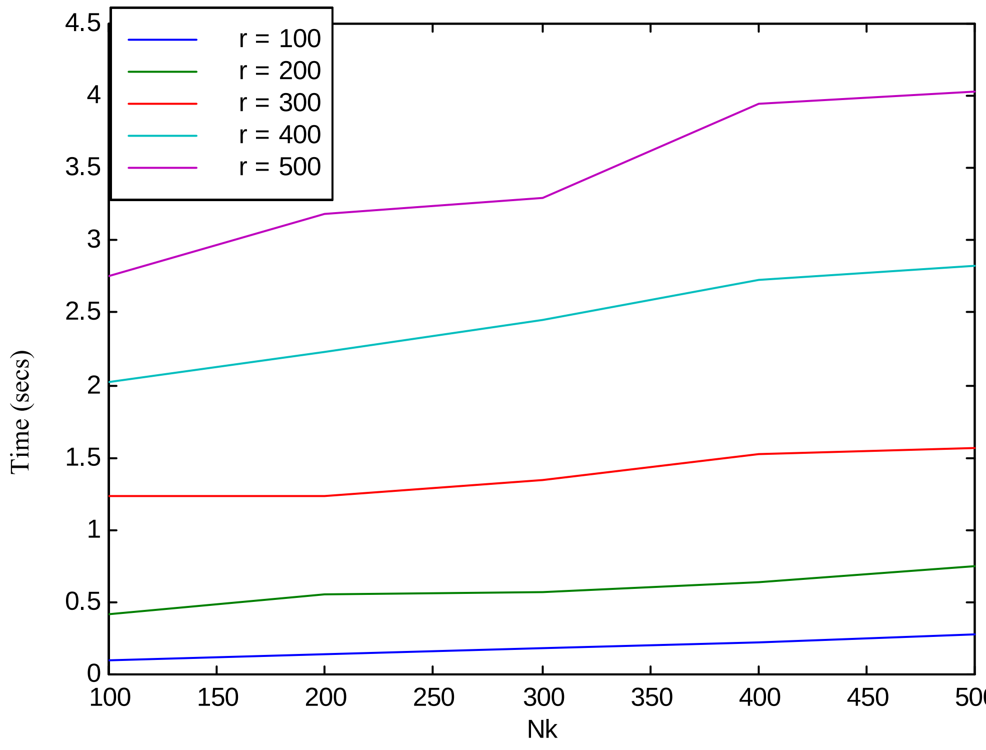

Figure 6.5: Graph of results for problems with dominated variables (fixed *r*)

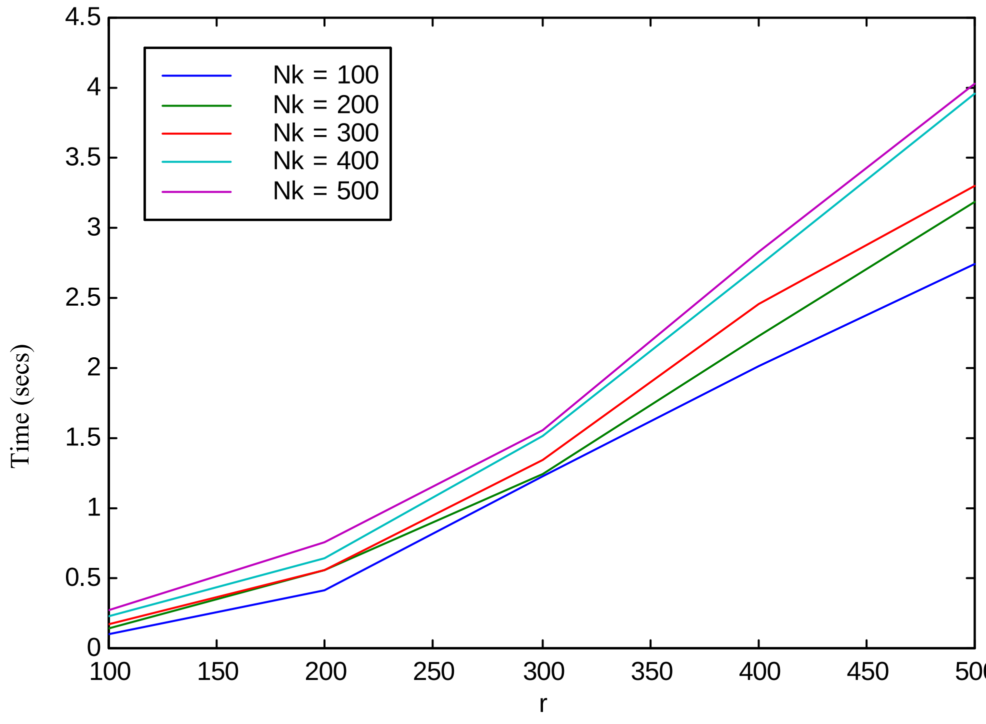

Figure 6.6: Graph of results for problems with dominated variables (fixed N_k)

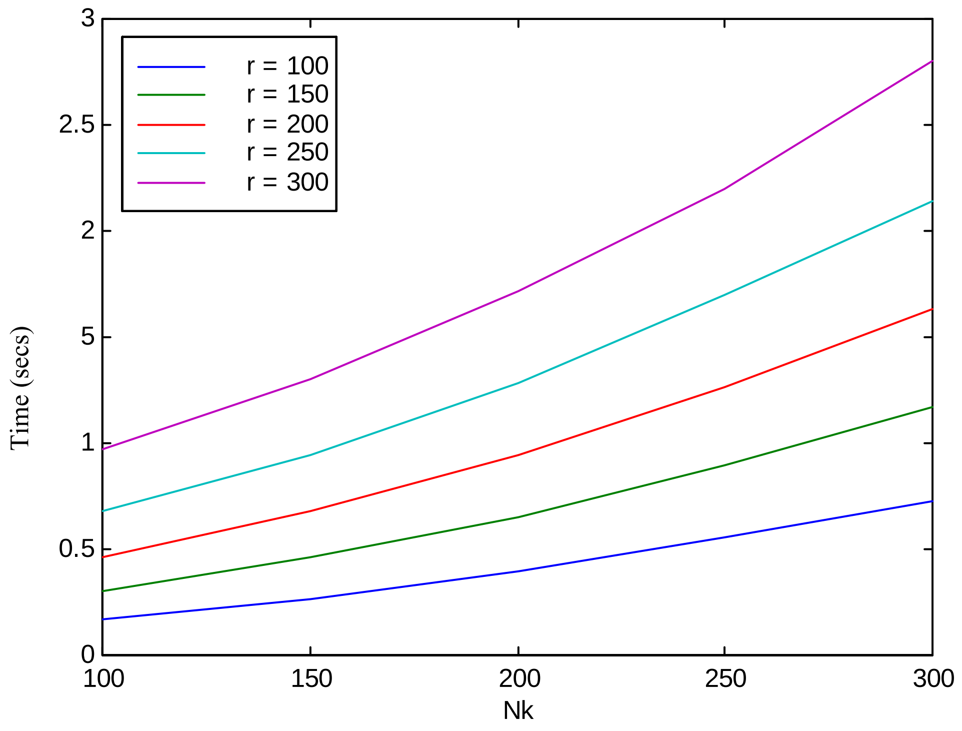

Figure 6.7: Graph of results for problems with nondominated variables only (fixed *r*)

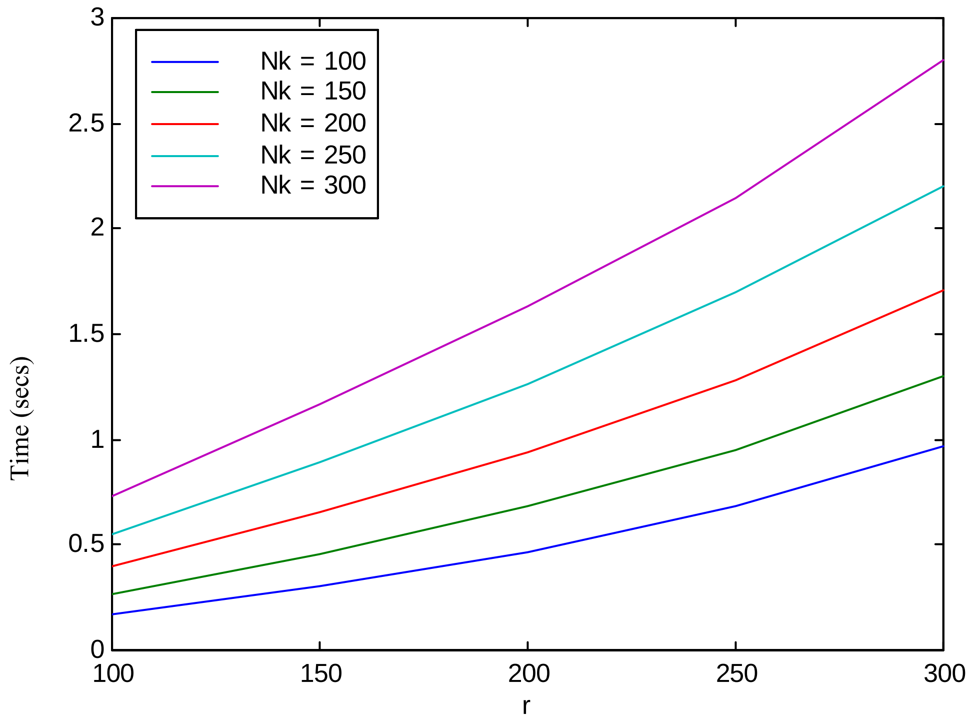

Figure 6.8: Graph of results for problems with nondominated variables only (fixed N_k)

This behavior can be explained by the fact that since each variable set has a large number of nondominated variables this results in a large number of options for each set. Consequently, the sets look similar and the maximum resource difference observed even when the equity constraints are relaxed is relatively small.

6.3.3 Discussion of Results

The algorithm was compared with the commercial software package for linear programming, LINGO, 2001. For all problems tested, the time it took Algorithm LMCKE to solve a problem was much smaller than the time required by LINGO. These time savings increase as the problem size increases. Therefore, the present algorithm can be very useful to practitioners for solving real life applications with a large number of decision variables.

For the problems with dominated variables and $r > 100$, the percentage of total time devoted to Phase II was larger than the percentage of time devoted to Phase I. As the number of sets increased, the percentage of total execution time devoted to Phase II increased. On the other hand, when the number of variables in each set increased, the percentage of total execution time devoted to Phase I increased. The same behavior was observed for the problems with only nondominated variables.

The algorithm exhibited high variability in the total computational time for the problems with dominated variables. The time consumed in Phase I did not vary drastically for different instances. Phase II, however, required time that significantly varied across instances. Therefore, it was analyzed further how the total computational effort for Phase II depends on the specific input for a given problem size. When the number of sets was fixed, the difference in total computational time, recorded when the number of variables varied, was not significant. This can be explained by the fact that, for a fixed value of *r*, the number of variables remaining after the elimination of the dominated variables does not differ significantly, even when the value of N_k changes from 100 to 500. The number of sets clearly plays a more important role in the computational effort needed in order to obtain an optimal solution than the number of variables in each set.

Another interesting observation has to do with changes in the total computational time when the same number of decision variables is distributed across more multiple choice sets. For problems with the same total number of decision variables, Phase II seems to perform better as the number of variables within each multiple choice set increases and the number of multiple choice sets decreases. This can be explained by the fact that a large percentage of the initial variables in each multiple choice set is expected to be dominated (see also Sinha and Zoltners, 1979). Therefore, for the same total number of variables, the more multiple choice sets there are, the fewer the number of dominated variables.

As Phase II proceeds, the algorithm sweeps an increasing number of segment costs in an effort to bring the endpoints of the interval containing the set costs closer. This was confirmed empirically as the cumulative number of upper and lower sets was generally increasing as the number of iterations increased, although not necessarily between any two consecutive iterations. For example, in a sample problem with 50 sets, the cumulative number of upper and lower sets was two after termination of Phase I, 22 after 30 Phase II iterations, 37 after 60 iterations, 43 after 90 iterations and 49 after 106 Phase II iterations.

As shown in Table 6.2, for the problems with dominated variables the time needed for the execution of Phase II was significantly smaller for certain "easy" problems than for other "hard" problems, even when the problem size was the same. To investigate what makes a problem hard, some special problem instances were constructed. It was observed from these experiments that the distribution of set costs after Phase I was terminated, strongly affects the number of Phase II iterations and as a result, the total computational effort. For a specific problem size with 100 sets for example, four different cases were considered. In the first, the set costs after termination of Phase I were uniformly distributed between 0 and 100. In the second, half of them were uniformly distributed between 0 and 20 and the other half were uniformly distributed between 80 and 100. In the third, 40 costs were uniformly distributed between 0 and 20, 40 between 80 and 100 and 20 between 40 and 60. In the last case, all the costs were uniformly distributed between 30 and 70. As expected, these last type problems were the easiest. Their average total computational time was about half of that for the problems in the second case which were the hardest. The problems in the first case were slightly harder with a computational time increase of about 33% and the problems in the third case were even harder with a computational time increase of about 27 % with respect to the problems in the first case. A similar behavior was observed with respect to the number of Phase II iterations. This analysis suggests that Phase II performs worse when the segment costs are isolated into groups close to the endpoints of the interval containing the set costs. As these costs are distributed more uniformly and closer to the midpoint of the containing interval, the performance of Phase II improves and the number of iterations decreases.

With respect to the sensitivity analysis results, for all problem sizes, the initial value of f_a is close to N_k which is also an upper bound for f_a. The objective function value doesn't appear to be very sensitive as the value of f decreases. The largest decrease in total profit was less than 1% even when the value of f was 1000 times smaller than f_a. Almost all this decrease is experienced up to the point where the value of f becomes equal to 1% of its initial value. The further decrease in profit, observed when f decreases to 0.1% of its initial value, f_a, was negligible. The same behavior was observed for the sensitivity of the total computational time. A large percentage of the total effort was devoted to bringing the value of f within 1% of its original value. When f was further decreased, the additional effort needed was negligible. In contrast to the objective function value, total computational time was strongly dependent on changes in f.

This behavior is better explained when we consider the order of magnitude of the variable coefficients and f. When the value of f needs to be decreased to 100 times its initial value, the variable coefficients are on the same order of magnitude with the desired incremental decrease of the value of f. Therefore, this decrease cannot be easily accomplished. On the other hand, when f must be decreased further, the absolute value of this decrease is relatively small with respect to the coefficients of the variables and therefore can be accomplished faster. For the same reason, the changes in the objective function value become negligible as the desired value of f decreases further. This also suggests that the absolute difference in f from its initial value is crucial for the computational effort needed to achieve this decrease and not the value of this decrease expressed as a percentage.

Figure 6.9 presents a potential graph of the change in total profit (y-axis) with respect to changes in f (x-axis). As shown, the function intercepts the y-axis at a point which is equal to the total profit incurred when all the sets are allocated the same cost, b/r. As the value of f increases, the total profit increases also, up to the point where the value of f is equal to the value, f_a, observed after the termination of Phase I. As expected, the total profit remains constant when the value of f increases beyond this value.

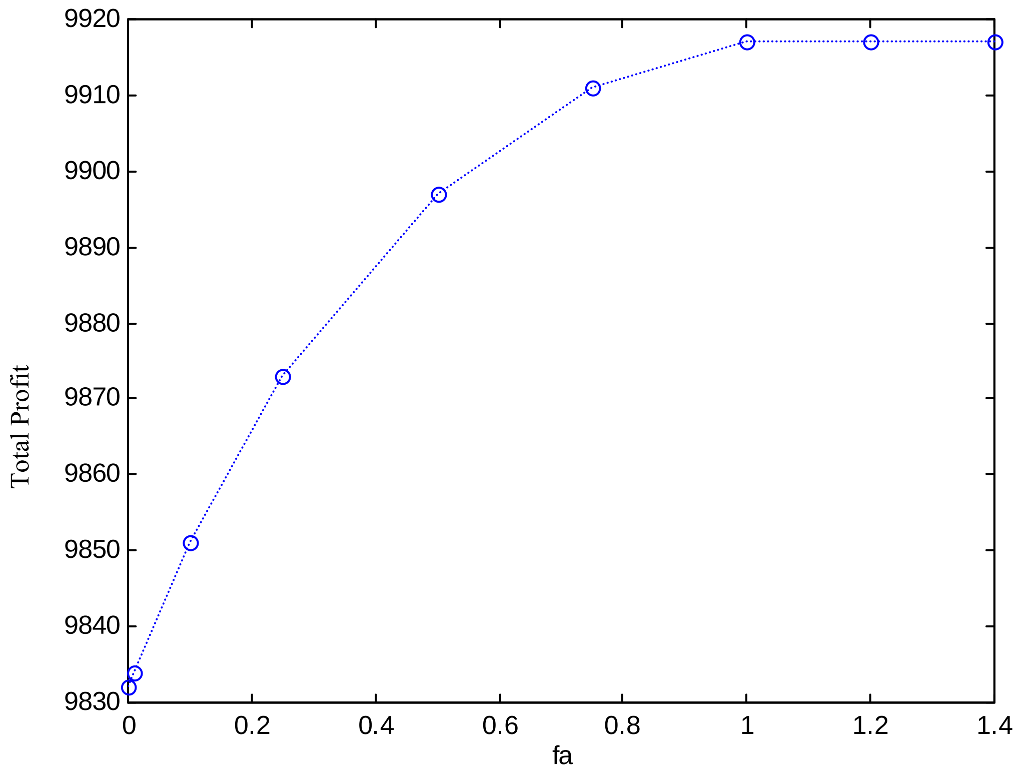

Figure 6.9: Optimal objective function value for different values of the parameter f, expressed as a percentage of f_a

"What we see depends mainly on what we look for." **John Lubbock**

"C makes it easy to shoot yourself in the foot. C++ makes it harder, but when you do, it blows away your whole leg." **Bjarne Stroustrup**

"Avoid those conclusions that are not confirmed by experience." **Leonardo da Vinci**

"Obstacles are those frightful things you see when you take your eyes off your goal."

Henry Ford

Chapter 7: The 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice and Equity Constraints

7.1 Introduction

In this chapter, the problem of Section 1.3 is studied and an important methodology is presented that can be used to obtain its solution. Fundamental insights into its properties and significant contributions are provided. One such contribution is the development of two optimization algorithms that exploit the special structure of the problem, as well as an efficient optimization-based heuristic that can provide excellent approximate results. Additionally, computational experiments are conducted, to investigate how these algorithms and a commercial software package compare and how they are affected by the main parameters of the model. Finally, the impact of the equity constraints on the optimal objective function value is explored.

7.2 Problem Interpretation

In the formulation of Section 1.3, each variable set contains both discrete and continuous variables. The continuous variables of each variable set, *k*, form the associated multiple choice set, *k*. The quantity $\sum_{i\in R_k} c_{ki}x_{ki} + \sum_{j\in D_k} d_{kj}y_{kj}$ is the total resource amount allocated to set *k*, which is also called the cost of set *k*. As was the case with Problem LMCKE, equity constraints are used to ensure a balance between the resource amounts allocated to different variable sets. In order to reach the desired level of equity, the resource amount allocated to each variable set must belong to an interval of certain width. In this way, the maximum difference between the resource amounts allocated to any two variable sets is bounded above by the width of this interval. The auxiliary decision variables L and U are used to denote the two endpoints of this interval which is called the interval of uncertainty.

Two different versions of the problem are considered. In the first, both the width and the exact location of the interval of uncertainty are known. In this case, the values of L and U are known and the constraint $U - L \leq f$ is dropped. This problem is abbreviated as MIMCKFE, standing for 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice and Fixed Equity Constraints. In the second version of the problem, only the width of this interval is known, while its exact location is not. In this case, the constraint $U - L \leq f$ is used to restrict the width of this interval to a maximum value, f. This problem is abbreviated as MIMCKE, standing for 0-1 Mixed Integer Knapsack Problem with Linear Multiple Choice and Equity Constraints. The latter scenario adds a great deal of difficulty to the problem and makes the development of an efficient procedure for solving it significantly harder. In what follows, both variants of the problem are investigated thoroughly.

7.3 The Problem with Fixed Interval of Uncertainty

A number of interesting properties of the MIMCKFE Problem are developed next. Then, a discussion on their significance is provided. These properties are utilized later in the development of a branch and bound solution algorithm.

7.3.1 The Structure of Optimal Solutions

Consider the linear relaxation of MIMCKFE that results when the integrality constraints are replaced by bound constraints $0 \leq y_{kj} \leq 1$. Then, each binary variable becomes a multiple choice set of size 1. Therefore, if the equity constraints are ignored, the resulting is a LMCK Problem. As illustrated in Figure 4.1 for LMCK, continuous variables that do not belong to the left upper hull of the associated coefficient space are dominated and should be eliminated from further consideration. This is because a suitable combination of variables from the left upper hull will always provide the maximum possible profit for a given value of the budget amount allocated to this set. This is due to the linear multiple choice constraints that restrict the sum of all continuous variables within this set to be at most l_k.

Part of the theory for variable elimination that was developed for LMCK also applies to MIMCKFE. Similarly with LMCKE, however, MIMCKFE differs from LMCK in the following sense: Continuous variables which belong to the right upper hull of the associated multiple choice coefficient space may appear in an optimal solution. In other words, the set of nondominated continuous variables for MIMCKFE consists of all the variables that belong to the complete upper hull and not only its left side.

The reason that the continuous variables of the right upper hull of the associated multiple choice set cannot be eliminated, is because they enlarge the cost domain of that variable set. This can be very useful in the following case. With reference to Figure 4.1, this situation arises when the cost of set k is still less than L even though x_{km} (the variable with the highest profit among all continuous variables of set k) is equal to l_k and all the binary variables of set k are equal to 1. In this case, the cost of set k cannot belong to the interval $[L,U]$ unless some of the continuous variables in the right upper hull take a positive value. In other words, feasibility can only be reached by increasing at least one variable from the right upper hull. Since at least one such variable will appear with a positive value in the optimal solution, $x_{km} < l_k$, and the maximum profit for this set will not be reached. Therefore, some profit has to be sacrificed to provide for an equitable resource distribution. To illustrate this, consider the following example:

$$\begin{aligned}
&\text{Max } 5x_{11} + 3x_{12} + 10\,x_{21} + 2\,y_{11} + 10y_{21}\\
&\text{s.t. } 2x_{11} + 4x_{12} + 5\,x_{21} + 2\,y_{11} + 2y_{21} \leq 13\\
&\quad x_{11} + x_{12} \leq 1\\
&\quad x_{21} \leq 1\\
&\quad x_{11}, x_{12}, x_{21} \geq 0\\
&\quad y_{11}, y_{21} \in \{0,1\}
\end{aligned}$$

The problem has two disjoint variable sets. The first contains the continuous variables x_{11} and x_{12} and the binary variable y_{11}. The second contains the continuous variable x_{21} and the binary variable y_{21}. The optimal solution is $x_{11} = 1$, $x_{12} = 0$, $x_{21} = 1$, $y_{11} = 1$, $y_{21} = 1$, with a corresponding objective function value of 27. Let's assume now that an equitable allocation involves a budget belonging to the interval [6,7] for each of these two sets. The optimal solution when the equity constraints are included is $x_{11} = 0$, $x_{12} = 1$, $x_{21} = 1$, $y_{11} = 1$, $y_{21} = 1$, and the objective function value is 25. Note however that, within set 1, variable x_{12} is integer dominated by variable x_{11}, since $p_{12}/c_{12} < p_{11}/c_{11}$ and $p_{12} < p_{11}$. The reason that x_{12} is positive in the optimal solution is because it enlarges the cost domain of the first set.

If only x_{11} and y_{11} are considered in set 1 and x_{12} is eliminated from multiple choice set 1, at most 4 units of resource can be allocated to set 1. Therefore, the problem is infeasible. On the other hand, if x_{12} is increased replacing x_{11}, as many as 6 resource units can be allocated to set 1, making the problem feasible. Although increasing x_{12} after the increase of x_{11} is not a profitable action when the equity constraints are not present, the substitution is necessary in order to reach feasible equitable allocation. Based on the above analysis, continuous variables that enlarge the cost domain of the associated set should not be eliminated for MIMCKFE, even if they are dominated. On the other hand, if a continuous variable is not on the boundary of the upper hull of the associated multiple choice coefficient space, the variables that belong to this hull ensure that the cost domain of this set will not be reduced when this variable is eliminated. Hence, continuous variables not on the complete upper hull can always be eliminated. The above discussion leads to the following result:

Proposition 7.1: Continuous variables that do not belong to the upper hull of the associated multiple choice coefficient space are dominated and can be eliminated from further consideration for MIMCKFE.

Proof: Similar to the proof of Proposition 6.1. $\square$

It follows from Proposition 7.1 that Figure 6.3 presents the nondominated continuous variables within a multiple choice set for MIMCKFE too.

For each variable set, a list of variables, L_k, can be constructed as follows. First construct the individual list for the continuous variables and define their associated slopes using the same procedure as the one for Problem LMCKE. Next, define the associated slope of each binary variable y_{kj} as its ratio q_{kj}/d_{kj}, and then merge all the variables (both binary and continuous) of this set in a single list L_k in non-increasing order of their associated slopes.

Assuming that the left upper hull of the associated multiple choice coefficient space coincides with the complete upper hull, the situation is illustrated in Figure 7.1.

This is the multiple choice set k comprising only continuous variables of Figure 4.1, augmented with the binary variables y_{kr} and y_{kt} that belong to variable set k. Since the slope of a binary variable is always positive, these will always appear on the left upper hull of this graph. Let b_k be the resource amount allocated to set k for the linear relaxation of the problem. Based on this graph, a one to one relationship is defined between b_k and the profit contribution of set k. If the variables of set k are increased in the same order that they appear in L_k, then the profit derived from this set will always be the largest possible for a specific value of b_k. Additionally, the optimal values of the variables in this set can be found using this graph. Of course, b_k depends on the relative magnitude of the associated slopes of variables belonging to all sets and is not known in advance.

For small values of b_k, i.e., $0 < b_k \leq c_{ki}l_k$, variable x_{ki} will have a value equal to b_k/c_{ki} and will be the only positive variable in set k. This is because it has the highest profit/cost ratio among all variables in this set. For $b_k = c_{ki}l_k$, $x_{ki} = l_k$ and the profit from set k is $p_{ki}l_k$. If b_k is increased further, the next variable considered for increase is y_{kr}. When $b_k = c_{ki}l_k + d_{kr}$, $x_{ki} = l_k$ and $y_{kr} = 1$. When b_k increases beyond $c_{ki}l_k + d_{kr}$, it becomes more profitable for x_{kj} to increase while x_{ki} has to decrease in order to ensure that the sum of the two variables does not exceed l_k. The reason x_{kj} gradually replaces x_{ki} is its higher profit (per unit). It can be easily shown that the new profit from set k is $q_{kr} + p_{ki}l_k + \dfrac{p_{kj} - p_{ki}}{c_{kj} - c_{ki}}(b_k - c_{ki}l_k - d_{kr})$. Thus, the excess budget amount $(b_k - c_{ki}l_k - d_{kr})$ is used for gaining additional profit.

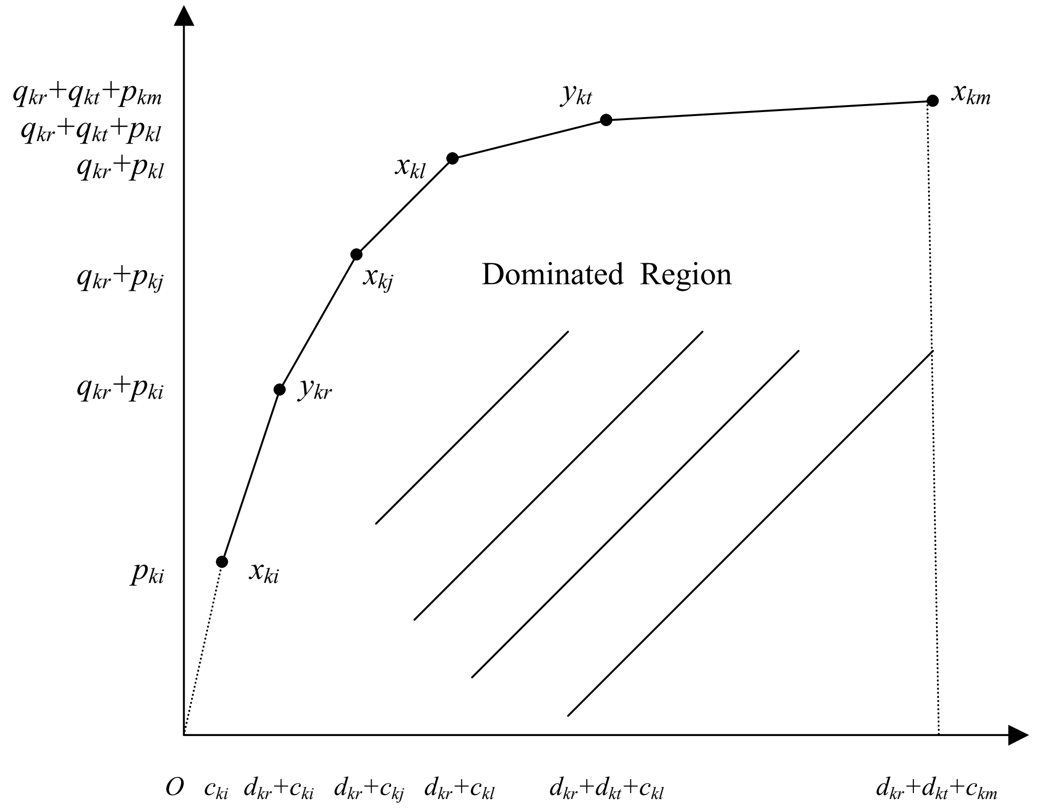

Figure 7.1: Arrangement of the variables in a disjoint variable set for MIMCKFE

It is clear now that the associated slope of a variable provides a measure of the incremental profit earned per unit of budget additionally used when this variable is increased. Eventually, when $b_k = c_{kj}l_k + d_{kr}$, $x_{kj} = l_k$, $y_{kr} = 1$ and $x_{ki} = 0$. If b_k is increased further, x_{kl} will increase and x_{kj} will decrease. The maximum value that b_k can take is $c_{km}l_k + d_{kr} + d_{kt}$, with a corresponding profit of $p_{km}l_k + q_{kr} + q_{kt}$. As a result, at any point there are at most two positive-valued continuous variables from a set. If the binary variables are ignored, these are consecutive variables from the associated variable list and the higher the value of b_k, the higher their rank order in the list.

The procedure described above is called a forward move applied to set *k*. It can be used to find the partial solution corresponding to this set for the linear relaxation of the problem if the value of b_k for this solution is known. During this procedure, the profit of this set and the total profit are updated accordingly. The above analysis implies the following result:

Proposition 7.2: The optimal solution to the linear relaxation of MIMCKFE contains at most one fractional-valued binary variable from each disjoint variable set.

Proof: Assuming that b_k is known, the values of the decision variables of set *k* in the optimal solution to the linear relaxation of MIMCKFE can be found by solving the following problem:

$$\begin{aligned}
\text{Max } & \sum_{i\in R_k} p_{ki}x_{ki} + \sum_{j\in D_k} q_{kj}y_{kj} \\
\text{s.t. } & \sum_{i\in R_k} c_{ki}x_{ki} + \sum_{j\in D_k} d_{kj}y_{kj} = b_k \\
& \sum_{i\in R_k} x_{ki} \le l_k \\
& x_{ki} \ge 0, \forall\, i \in R_k \\
& 0 \le y_{kj} \le 1, \forall\, j \in D_k
\end{aligned}$$

This is a LMCK problem. There is one multiple choice constraint for each of the variables y_{kj} and one multiple choice set that contains all continuous variables x_{ki}. Variables x_{ki} and y_{kj} are considered fractional if $0 < x_{ki} < l_k$ and $0 < y_{kj} < 1$, respectively. Moreover, the optimal solution to the LMCK contains at most two variables with a fractional value (see Pisinger, 1995a). If however the LMCK has two fractional-valued variables they must belong to the same multiple choice set (see Pisinger, 1995a). Since each variable y_{kj} forms its own multiple choice set of cardinality 1, the above result follows. ▭

A similar procedure can be used to find the solution that results when the cost of a set is decreased from some initial positive value. Variables from this set are now decreased in the exact reverse order in which they were originally increased. Consider the variable set shown in Figure 7.1 and assume that the current solution is $x_{km} = l_k$, $y_{kr} = 1$ and $y_{kt} = 1$, with $b_k = c_{km}l_k + d_{kr} + d_{kt}$. If b_k is decreased, initially x_{km} will be decreased. At the same time, x_{kl} is increased, since x_{km} replaced x_{kl} when it was originally increased. When $b_k = c_{kl}l_k + d_{kr} + d_{kt}$, we have $x_{kl} = l_k$, $y_{kr} = 1$ and $y_{kt} = 1$. If b_k is decreased further, y_{kt} has to be decreased. Variable y_{kt} decreases to 0 as soon as b_k drops to $c_{kl}l_k + d_{kr}$. To distinguish it from the previous move, we call this a backward move applied to set k.

Let's assume now that we wish to find the optimal solution to the linear relaxation of MIMCKFE when the equity constraints are ignored. To optimally allocate the total available budget b to all sets, a similar procedure as above can be applied. The only difference is that the next variable selected for increase is always the one with the highest slope among all variables that haven't been increased to their highest value yet, 1 for a binary variable and l_k for a continuous variable. Once a variable is increased to its highest value, it is not eligible to be increased again. During this procedure, besides updating the budget residual (the remaining budget that has not yet been allocated) and the total profit, the cost of each variable set should also be updated accordingly. An iteration is carried out exactly as in the case when a forward move is applied to a set.

To speed up the above procedure, a master list is constructed that contains the variables of all sets arranged in non-increasing order of their associated slopes. This list can be easily constructed by merging all the individual lists of sets, since the variables of each set are already increased in non-increasing order of their slopes in the associated list. The information regarding the next variable from any set that should be increased after each iteration is now directly obtained from this master list.

To find the new solution that results when the total available budget b decreases from some initial value, a backward move can similarly be applied to the whole problem. The variable selected for decrease is the one with the lowest slope among all variables that have a positive value. An iteration in this case is carried out exactly as in the case when a backward move is applied to a set. During this procedure, besides updating the budget residual and the total profit, the cost of each variable set should also be updated accordingly. As before, the procedure can be simplified using the master list of variables that is constructed as explained above.

The above discussion suggests that, starting from a partial solution of a set k with initial cost b_k, it is possible to find the new solution of this set for a different value of b_k, without having to apply the procedure for increasing variables from scratch. Similarly, starting from a solution to the whole problem with initial total budget b, it is possible to find the new solution for a different value of b, without having to apply the procedure for increasing variables from scratch. The new solution can always be obtained by performing forward or backward moves, utilizing the variable lists of the problem. The resulting savings in computational effort can be very substantial, especially for problems with a large number of decision variables.

When the equity constraints are included, the above procedure can be modified as follows in order to find the optimal solution to the linear relaxation of MIMCKFE. A forward move is applied to each set to bring its cost up to L. Then, the remaining budget residual is allocated to all sets by applying a forward move to the whole problem and ensuring that the cost of a set does not exceed the upper limit, U.

During a forward move applied to the whole problem, the budget allocation to a set should terminate when one of the two following stopping conditions is met. The first condition is when the cost of this set becomes equal to U and the second when the budget residual decreases to 0. In the first case, no additional budget amount can be allocated to this set, since this will make the problem infeasible. Therefore, the procedure switches to another set and the forward move continues with another variable from that different set. In the second case, the move terminates because the optimal solution for the linear relaxation of the problem has been reached. The situation is similar when a backward move is applied to a set. Unless the stopping condition is met at the same time that the variable being modified gets a rounded value (i.e., equal to the length of the set if it is continuous, or equal to 1 if it is binary), the iteration will not be complete. As a result, this variable will have a fractional value. We call this fractional-valued variable that was modified last from a set, the critical variable of this set.

In the case of a binary fractional-valued variable, the critical variable is uniquely defined. On the other hand, when the value of a continuous variable changes and this is not the first continuous variable from the associated list L_k, then the value of the previous continuous variable in this list is also changing at the same time. Ignoring the binary variables of this set, these two continuous variables are adjacent in L_k. The critical variable is always the rightmost of these two variables (the one with the smallest associated slope). For example, in Figure 7.1, x_{kj} will always be the critical variable of set

k if the move terminates while x_{kj} is increased replacing x_{ki} or decreased while x_{ki} is increased. Of course, the critical variable of a set can also have a rounded value. This happens when the variable being modified takes a rounded value simultaneously with the time that the stopping condition is met.

7.3.2 A Branch and Bound Algorithm

The algorithm developed for MIMCKFE is a branch and bound solution procedure that utilizes the variable lists introduced above. It works by relaxing and enforcing successively the integrality constraints of the problem. The solution to the linear relaxation of the subproblems that arise is obtained by using these variable lists to perform forward and backward moves as explained above. One of the fractional-valued critical binary variables is always the variable selected for branching.

Initially, the algorithm eliminates the dominated continuous variables and constructs the individual list of variables for each set, from which the master list is constructed. Then, using these lists, it finds the optimal solution that results when each set is allocated a resource amount which is equal to the lower limit, L. This amounts to finding the critical variable of each set when its cost is equal to L. At this solution, the budget residual is equal to $b - rL$, where $r = |S|$. Now, the cost of each set belongs to the interval of uncertainty. Thus, the allocation of the remaining budget can continue by next increasing the variable with the highest slope that hasn't been yet increased to its highest value. The cost of a set should not exceed the upper limit U, to ensure that the equity constraints are not violated. Therefore, as soon as the cost of a set becomes equal to U, no more resource units can be allocated to this set. The forward move terminates when the budget residual decreases to 0, or when the costs of all the sets become equal to U, or when there are no remaining variables with a positive associated slope that can be increased.

The solution derived by using the above procedure is optimal for the linear relaxation of the MIMCKFE. If the critical variable of every set is continuous, then this solution is also optimal for the MIMCKFE. If not, we branch on one of the fractional valued binary variables of this solution. In order to find the solution to the linear relaxation of the two subproblems that arise by setting this variable to 0 or 1, we use the following result:

Proposition 7.3: When a binary variable is set to 0 or 1 and therefore excluded from Problem MIMCKFE, the order of the remaining variables in each variable list remains unchanged.

Proof: The exclusion of a binary variable does not affect in any way the associated slope of any of the remaining variables, since this variable is a multiple choice set by itself. Therefore, when this variable is excluded, the order of the other variables in the lists remains unchanged, since this order depends solely on each variable's associated slope. ▭

When we branch on one of the fractional-valued binary variables by setting it to 0 or 1, we can use Proposition 7.3 to find the solution to the linear relaxation of the two subproblems that arise as follows. For the left subproblem in which this variable is set to 0, initially the cost of the set containing this variable and the total profit decrease while the budget residual increases. A check on whether the new cost of this set violates the equity constraints is made first. If this cost does not decline below L, a forward move does not need to be applied to this set. If is does, it must be brought up to L again. This is done by applying a forward move to this set starting with the current solution, until its cost becomes equal to L. The variable that was just set to 0 is now excluded from the associated list and the variable immediately succeeding it is next considered for increase. The new critical variable is stored after this forward move terminates.

After the cost of this set is brought within the limits L and U again, and assuming that there is still a positive budget residual, this can be used to improve the current objective function value. Therefore, using the master list, a forward move is applied to the whole problem. The critical variable with the highest slope among all sets is the next variable considered for increase. This move continues until either the budget residual declines to 0, or the cost of all sets becomes equal to U, or no variable with a positive associate slope that can be increased exists.

When a binary variable is set to 1, a similar procedure is followed. In this case, the cost of the set containing it and the total profit increase, while the budget residual decreases. A check is made on whether the cost of this set has exceeded the upper limit, U. If it has, a backward move is applied to this set until its cost equals U again. To ensure that the new solution is optimal, variables from this set are decreased in the exact reverse order in which they were originally increased. If the budget residual is negative after this move, a backward move to the whole problem is applied in order to make the problem feasible with respect to the budget constraint. The variable that should be decreased is the one with the worst associated slope among all critical variables. This variable is obtained efficiently using the master list. During this procedure the cost of a set is not allowed to drop below L, in order to ensure that the problem remains feasible.

If after setting a binary fractional variable to 0 or 1, the cost of this set in the associated subproblem cannot be brought within the limits L and U by increasing or decreasing variables from this set, then this subproblem is infeasible and should be fathomed. This happens when the constraints that have been added as a result of branching eliminate all the solutions for which the cost of this set belongs to the interval of uncertainty.

At each iteration, the algorithm selects for exploration the tree node with the maximum upper bound on the objective function value of the original problem. This strategy is used in order to keep the size of the tree as small as possible. This upper bound is known once the solution to the linear relaxation of this node has been obtained. If the solution at this node satisfies the integrality solutions, then this solution is also optimal to the original problem. If not, one of the fractional-valued binary variables is chosen for branching. The two subproblems that arise from setting this variable to 0 or 1 are added to the tree node and the solution to their linear relaxation is found using the same procedure. By using the special relationship between parent and children subproblems the optimal solution to the linear relaxation of new subproblems is obtained fast. Additionally, the memory usage is low, since the decision variable values are not stored explicitly. This is a consequence of the special structure of the problem.

The performance of the algorithm is strongly affected by the strategy used for branching. Several experiments were performed to identify the best rule to use. The criteria used were based on characteristics of the binary variables such as their cost, their profit, their associated slope, their fractional value, and their relative order in the variable lists of the problem. It was found that the best performing rule was to branch on the variable with the largest cost among all fractional-valued binary variables of the current solution. Although this is not very intuitive, a possible explanation is that this rule allows the algorithm to move away from local optima and avoid wasting time exploring localized regions.

For each subproblem, there exists an upper bound which is tighter than the objective function value of the associated linear relaxation solution. This bound is obtained by imposing integrality on each of the critical binary variables of this linear relaxation solution and is an extension of the upper bound developed for the 0-1 Knapsack Problem by Martello and Toth (1977). The existence of two bounds *B1* and *B2* is proven next and then this tighter bound is derived as the maximum between *B1* and *B2*.

Consider the solution to the linear relaxation of a subproblem of the branch and bound tree. If this solution is not feasible, then it contains one or more binary critical variables with a fractional value; let y_{kj} be one of them that belongs to set k and let b_k be the cost of this set in this solution. Let also u_{LP} denote the objective function value of this solution.

Since y_{kj} is a binary variable, it has to be set to 0 or 1. Consider the case when y_{kj} is set to 0. Let s_b be the highest associated slope of any variable besides y_{kj} that belongs to a set with cost less than U and hasn't yet been increased to its highest level (if $s_b < 0$, let $s_b = 0$). This is the variable that would be increased next if additional resource units were available. If $b_k - d_{kj}y_{kj} < L$, set $B1 = u_{LP} - q_{kj}\, y_{kj} + (L - b_k + d_{kj}y_{kj})s_{kn} + (b_k - L)s_b$, where s_{kn} is the associated slope of the variable that immediately succeeds y_{kj} in list L_k. Otherwise, set $B1 = u_{LP} - q_{kj}y_{kj} + (d_{kj}y_{kj})s_b$.

Consider now the case when y_{kj} is set to 1. Let s_w be the lowest associated slope of any variable besides y_{kj} that belongs to a set with cost greater than L and has a positive value. This is the variable which would be decreased next if the budget residual of the current solution was negative. If $b_k + d_{kj}(1 - y_{kj}) > U$, set $B2 = u_{LP} + q_{kj}(1 - y_{kj}) - (b_k + d_{kj}(1 - y_{kj}) - U)s_{kp} - (U - b_k)s_w$, where s_{kp} is the associated slope of the variable immediately preceding y_{kj} in list L_k. Otherwise, set $B2 = u_{LP} + q_{kj}(1 - y_{kj}) - d_{kj}(1 - y_{kj})s_w$.

Proposition 7.4: $B = \max(B1, B2)$ is a valid upper bound to the optimal objective function value of Problem MIMCKFE.

Proof: Consider the two subproblems that arise when we choose to branch on y_{kj}. If the budget residual of the current linear relaxation solution is positive then $b_k = U$ (otherwise y_{kj} would be increased further) and there are no variables with a positive slope that can be increased from another set. For the left subproblem in which y_{kj} is set to 0, the total profit decreases initially by $q_{kj}y_{kj}$ and the budget residual increases by $d_{kj}y_{kj}$. If $b_k - d_{kj}y_{kj} < L$, then from the budget recovered an amount equal to $(L - b_k + d_{kj}y_{kj})$ must be allocated to the same set in order to bring its cost up to L again. Thus, in the best case, the total profit will increase by $(L - b_k + d_{kj}y_{kj})s_{kn}$, since s_{kn} is the highest slope of any variable from this set besides y_{kj} that hasn't been increased to its highest value (if no such variable exists, then this subproblem is infeasible). After this, a budget amount equal to at most $b_k - L$ can be used to improve the current objective. Since s_b is the highest slope of any variable besides y_{kj} that can be increased next, the result for $B1$ follows. If $b_k - d_{kj}y_{kj} \geq L$, the derivation for $B1$ remains the same but the step for bringing b_k up to L must be skipped. For the subproblem in which y_{kj} is set to 1, the total profit increases initially by $q_{kj}(1 - y_{kj})$ and the budget residual decreases by $d_{kj}(1 - y_{kj})$. If $b_k + d_{kj}(1 - y_{kj}) > U$, the budget amount that has to be deallocated from set k in order to bring its cost down to U again is $b_k + d_{kj}(1 - y_{kj}) - U$. Thus, in the best case, the total profit will decrease by $(b_k + d_{kj}(1 - y_{kj}) - U)s_{kp}$, since s_{kp} is the lowest slope of any variable from this set besides y_{kj} that can be decreased (if no such variable exists, then this subproblem is infeasible). After this, a budget amount of $U - b_k$ still needs to be recovered. Since s_w is the lowest slope of any variable with a positive value besides y_{kj} that can be decreased, the result for $B2$ follows as well. If $b_k + d_{kj}(1 - y_{kj}) \leq U$ the derivation for $B2$ remains the same but the step for bringing b_k down to U must be skipped. Since y_{kj} has to be set to 0 or 1, $B = \max(B1,B2)$ is a valid upper bound on the optimal solution of the current subproblem. ▭

Proposition 7.5: $B \leq u_{LP}$

Proof: To prove that this inequality holds, it is sufficient to show that $s_{kn} \leq s_{kj} \leq s_{kp}$, and if $L < b_k < U$, that $s_b \leq s_{kj} \leq s_w$, where $s_{kj} = q_{kj}/d_{kj}$ is the associated slope of y_{kj}. This is because $B1 \leq u_{LP}$ is equivalent to $q_{kj}y_{kj} \geq (L - b_k + d_{kj}y_{kj})s_{kn} + (b_k - L)s_b$ when $b_k - d_{kj}y_{kj} < L$, and equivalent to $q_{kj}y_{kj} \geq (d_{kj}y_{kj})s_b$ when $b_k - d_{kj}y_{kj} \geq L$. Therefore, if the above expressions for the variable slopes hold, we have $(L - b_k + d_{kj}y_{kj})s_{kn} + (b_k - L)s_b \leq (L - b_k + d_{kj}y_{kj})s_{kj} + (b_k - L)s_{kj} = (d_{kj}y_{kj})s_{kj} = q_{kj}y_{kj} \Rightarrow B1 \leq u_{LP}$. Similarly, $B2 \leq u_{LP}$ is equivalent to $(b_k + d_{kj}(1 - y_{kj}) - U)s_{kp} + (U - b_k)s_w \geq q_{kj}(1 - y_{kj})$ when $b_k + d_{kj}(1 - y_{kj}) > U$, and equivalent to $d_{kj}(1 - y_{kj})s_w \geq q_{kj}(1 - y_{kj})$ when $b_k + d_{kj}(1 - y_{kj}) \leq U$. Therefore, if the above expressions for the variable slopes hold, we have $(b_k + d_{kj}(1 - y_{kj}) - U)s_{kp} + (U - b_k)s_w \geq (b_k + d_{kj}(1 - y_{kj}) - U)s_{kj} + (U - b_k)s_{kj} = d_{kj}(1 - y_{kj})s_{kj} = q_{kj}(1 - y_{kj}) \Rightarrow B2 \leq u_{LP}$. The inequalities $s_{kn} \leq s_{kj} \leq s_{kp}$ are clearly true, since the variables in L_k are arranged in non-increasing order of their associated slopes. To prove that the inequalities $s_b \leq s_{kj} \leq s_w$ are true when $L < b_k < U$, note that s_b is the slope of the variable that can be increased next. If

$s_b > s_{kj}$, then this variable would have been increased to its highest value before y_{kj} could be increased. Therefore, we have $s_b \le s_{kj}$. Similarly, s_w is the slope of the variable with a positive value that should be decreased next. If $s_w < s_{kj}$, then y_{kj} would have been increased to its highest value before this variable could be increased. Therefore, we have $s_w \ge s_{kj}$. □

By following the same procedure as above an upper bound for each fractional-valued binary critical variable of the current linear relaxation solution can be obtained. Clearly, the lowest of these bounds provides the tightest bound on the objective function value of the current subproblem. This is the upper bound used in Algorithm MIMCKFE. To introduce this algorithm, the following additional notation is needed:

b^i_k = cost of set k in subproblem i,

b^i_{res} = budget residual of subproblem i,

u^i = objective function value of the solution to the linear relaxation of subproblem i,

t = index for labeling the nodes of the B&B tree,

A = set containing the active nodes of the B&B tree, i.e., the nodes yet to be explored,

$I_0(t)$ = set containing the indexes of the binary variables set to 0 at node t,

$I_1(t)$= set containing the indexes of the binary variables set to 1 at node t.

Algorithm MIMCKFE

Step 0 (Preprocessing)

For each variable set k, construct the list L_k containing all the nondominated continuous variables and the binary variables of this set and define their associated slopes. While maintaining all individual variable lists, create a master list ML by merging the variables of all these lists in non-increasing order of their associated slopes. Initialize the objective function value u^1 and the values of all variables to 0.

Step 1 (Allocation of L to each set)

Apply a forward move to each set until the cost of all sets becomes equal to L. If $b < rL$ or the variable list of a set is scanned completely before its cost becomes equal to L, the problem is infeasible. Otherwise, store the critical variable of each set, initialize $b^1_k = L$, for all k in S, $b^1_{res} = b - rL$ and store the objective function value of this solution in u^1.

Step 2 (Linear relaxation of original problem)

Starting with the solution from Step 1 above, perform a forward move to the whole problem until either b^1_{res} drops to 0 or the cost of all sets becomes equal to U, or no variable with a positive slope that can be increased exists. The cost of a set should not exceed U during this procedure. Update the critical variables and the costs of the sets and the values of b^1_{res} and u^1 accordingly.

Step 3 (Initialization of the B&B tree)

Initialize the index of the root of the B&B tree to 1. Set $A = \{1\}$, $I_0(1) = \varnothing$ and $I_1(1) = \varnothing$.

Step 4 (Branching)

Let t_0 be the smallest unused index for labeling tree nodes and t^* be the tree node in A with the maximum upper bound to the optimal objective function value of the problem.

If this node has no fractional-valued binary variables, STOP; the solution at node t^* is optimal for the original problem. Otherwise, branch on a fractional-valued binary variable y_{mr}, generating two new subproblems and their associated nodes. Label the nodes t_0 and t_0+1, corresponding to subproblem with $y_{mr} = 0$ and $y_{mr} = 1$, respectively. Set $I_0(t_0) = I_0(t^*) \cup \{mr\}$, $I_1(t_0) = I_1(t^*)$, $I_0(t_0+1) = I_0(t^*)$, $I_1(t_0+1) = I_1(t^*) \cup \{mr\}$and $A = A - \{t^*\} \cup$

$\{t_0\}$U $\{t_0+1\}$. Variable y_{mr} is excluded from the variable lists of all the subproblems of the current subtree.

Step 5 (Linear relaxation of subproblem t_0)

Set $b^{t_0}{}_{res} = b^{t^*}{}_{res} + d_{mr}y_{mr}$ and $u^{t_0} = u^{t^*} - q_{mr}y_{mr}$ (results obtained after setting y_{mr} to 0). Set $b^{t_0}{}_k = b^{t^*}{}_k$, for all k in S, $k \neq m$ and $b^{t_0}{}_m = b^{t^*}{}_m - d_{mr}y_{mr}$. If $b^{t_0}{}_m < L$, perform a forward move on set m to bring its cost up to L. If the list of this set is scanned completely before its cost becomes equal to L, the node is infeasible; fathom it by removing its index from the set of active nodes and go to Step 4. Otherwise, perform a forward move on the whole problem until either $b^{t_0}{}_{res}$ decreases to 0 or the cost of all sets becomes equal to U, or no variable with a positive slope that can be increased exists. Update the critical variables and the costs of the sets and the values of $b^{t_0}{}_{res}$ and u^{t_0} for the new solution found.

Step 6 (Linear relaxation of subproblem t_0+1)

Set $b^{t_0+1}{}_{res} = b^{t^*}{}_{res}$ - $d_{mr}(1 - y_{mr})$ and $u^{t_0+1} = u^{t^*} + q_{mr}(1 - y_{mr})$ (results obtained after setting y_{mr} to 1). Set $b^{t_0+1}{}_k = b^{t^*}{}_k$ for all k in S, $k \neq m$ and $b^{t_0+1}{}_m = b^{t^*}{}_m + d_{mr}(1 - y_{mr})$. If $b^{t_0+1}{}_m > U$, perform a backward move on set m to bring its cost down to U. If the list of this set is scanned completely before its cost becomes equal to U, the node is infeasible; fathom it by removing it from the set of active nodes and go to Step 4. Otherwise, perform a backward move on the whole problem until either $b^{t_0+1}{}_{res}$ becomes non-negative or no variable with a positive value that can be decreased exists. If $b^{t_0+1}{}_{res}$ is still negative after this move, then the current node is infeasible; fathom it by removing its index from the active nodes and go to Step 4. Otherwise, update the critical variables and the costs of the sets and the values of $b^{t_0+1}{}_{res}$ and u^{t_0+1} for the new solution found and go to Step 4. □

To obtain the values of the decision variables at the termination of the Algorithm MIMCKFE, the optimal node *g* and the values of the critical variables of this solution are needed (note that the critical variables are continuous, otherwise the solution would be infeasible). If a list L_k doesn't have a critical variable (i.e., if L_k has been scanned completely), an artificial variable is appended as critical at the end of the list with a value equal to 0. The values of the decision variables are obtained as follows. All binary variables appearing in L_k to the left of the critical variable have a value of 1, unless their index is in the set $I_0(g)$. Similarly, all binary variables in L_k appearing to the right of the critical variable, have a value of 0, unless their index is in set $I_1(g)$.

The values of the continuous variables for set *k* are determined as follows: If the critical variable is the first continuous variable in L_k, then no other continuous variable from *k* will have a positive value. If not, the only other continuous variable with a positive value will be the one appearing to the left and closest to the critical variable. The value of that variable is equal to the right hand side of the associated multiple choice constraint minus the value of the critical variable.

7.4 A Heuristic for MIMCKFE

In this section an efficient heuristic that was developed for the MIMCKFE is presented. This heuristic is a natural extension of a primal heuristic for the 0-1 Knapsack Problem (see Nemhauser and Wolsey, 1988).

Similarly with the Algorithm MIMCKFE, the heuristic finds initially the linear relaxation solution that results when a budget amount equal to *L* is allocated to each set. Next, the heuristic increases all the fractional-valued binary variables to 1 and updates accordingly the total profit and the budget residual. Note that this could result in a set whose cost exceeds U or in a negative budget residual. If the cost of all sets is less than U and if the budget residual is non-negative, then the current solution is feasible for the original mixed integer optimization problem. Next, the heuristic allocates the remaining budget residual (if positive) by increasing variables that have not been increased to their highest value in the order they appear in the master list. While continuous variables are increased as usually, a binary variable that cannot be increased to its highest value without violating the equity constraints, is skipped. The procedure continues until either the budget residual drops to 0 or the cost of all sets becomes equal to U, or no variable with a positive slope that can be increased exists.

After termination of this forward move, the heuristic does a backward pass in order to reduce the cost of sets whose cost is possibly more than U, or to recover a missing budget amount in case that the current budget residual is negative. If the budget residual is negative, the master list is scanned backwards and variables are decreased until it becomes non-negative again. If the cost of a set is more than U, then variables from this set are decreased until its cost lies within the interval of uncertainty again. During any of these two moves, a binary variable which, when decreased to 0, reduces the cost of the associated set to less than L is skipped.

If after termination of the heuristic the budget residual is still negative or the cost of one or more sets does not lie in the interval of uncertainty, then the current solution is infeasible. The likelihood that such a case will arise depends strongly on the input parameters of the problem. It is possible to develop a more sophisticated heuristic that would reduce this likelihood. In this work this was not pursued because the information obtained from such a heuristic solution cannot be used in the branch and bound algorithm.

7.5 The Problem with Unknown Interval of Uncertainty

In this section the second variant of the problem (Problem MIMCKE) is considered, where the interval of uncertainty is unknown. This is a much harder problem to solve than the previous version, because this interval has to be identified in order to find the optimal solution and there is an infinite number of choices for its exact location.

Clearly, Propositions 7.1, 7.2 and 7.3 carry over to MIMCKE, since their validity does not depend on whether the interval of uncertainty is known or not. Some additional important properties for Problem MIMCKE are developed next.

The maximum resource amount, MC_k, allocated to a set k is when all the binary variables of set k take a value of 1 and, additionally, the continuous variable, x_{kq}, with the largest cost takes a value which is equal to the length of this set. Thus, we have: $MC_k = c_{kq}l_k + \sum_{j\in D_k} d_{kj}$. As with Problem LMCKE, let again $MC_{min} = \min_{k\in S} MC_k$ .With these new definitions, Corollaries 6.1, 6.2 and 6.3 and Proposition 6.2 carry over to Problem MIMCKE:

Corollary 7.1: The cost of any single set at the optimal solution of MIMCKE is at most $MC_{min} + f$.

Proposition 7.6: The cost of any single set at the optimal solution of MIMCKE cannot be larger than $\frac{b}{r} + \frac{r-1}{r} f$.

Proof: Same as the proof of Proposition 6.2. ▭

Corollary 7.2: Assuming that the unused budget at the optimal solution of MIMCKE is equal to 0, the cost of any single variable set lies in an interval centered at b/r whose width is less than but tends to $2f$ as r goes to ∞.

Corollary 7.3: The minimum of $MC_{min} + f$ and $b/r + f(r-1)/r$ is a valid upper bound, UB, on the cost of any single variable set at the optimal solution of MIMCKE.

To develop an algorithm for the MIMCKE, the algorithm presented above for the MIMCKFE was embedded in a binary search procedure. This tries to identify the interval of uncertainty $[L,U]$ by identifying its midpoint. It is clear that the midpoint of the interval of uncertainty lies in the interval $[f/2, UB-f/2]$. The algorithm performs a binary search for the midpoint in this interval.

During this procedure, new subproblems are generated and stored in a binary search tree when bound constraints are imposed on the value of this midpoint. In turn, these constraints impose specific bounds on the optimal cost of each variable set. Therefore, the resulting subproblems can be solved using the algorithm discussed above for the MIMCKFE. If the maximum difference between the costs of any two sets at a node of the binary search tree is no larger than f, then this node is feasible with respect to the equity constraints. Therefore, this solution can be compared with the best feasible solution that has been found so far. If it is better, it becomes the new best solution, substituting the previous one. If not, it is eliminated from further consideration. On the other hand, if this solution is not feasible with respect to the equity constraints, further exploration is needed. Therefore, starting from this node, new subproblems have to be generated with narrower widths for the unknown interval.

To illustrate this, suppose we want to check whether at the optimum the midpoint belongs to the interval $[f/2,v]$ or to $[v,UB-f/2]$, where v is some known value. Depending on the result of this check, we will be able to eliminate out of the two above intervals, the one that doesn't contain this midpoint. To do this, we relax the constraint $U - L \leq f$ from the original problem and we replace the values of L and U with the values 0 and $v + f/2$, respectively. Clearly, this is equivalent to restricting the midpoint of the interval of uncertainty to belong to the interval $[f/2,v]$. The resulting problem (left child) is a MIMCKFE problem and can be solved using the algorithm introduced above. Although the width of $[0,v+f/2]$ may be more than f, the solution to this problem is an upper bound to any feasible solution in which the midpoint is contained in $[f/2,v]$. Therefore, if this upper bound is not better than the objective function of the best feasible solution found so far, this node can be fathomed. If not, this node is added to the equity tree. For the second problem (right child), the values of L and U are set to $v - f/2$ and UB, respectively and the procedure is similar.

In the situation described above, neither of the two subproblems added to the equity tree are guaranteed to give a feasible solution. This is because the width of the intervals $[0,v+f/2]$ and $[v-f/2,UB]$ will not necessarily be less than f. Therefore, in certain cases we also have to solve the problem in which the midpoint is equal to v, i.e., the problem in which the values of L and U are set to $v-f/2$ and $v+f/2$, respectively. Such problems can be used to update the current best feasible solution because their solution satisfies the equity constraints by default.

From the above analysis it follows that, each time we pivot on a value v, there are three candidate nodes that may be added to the equity tree. The first (left child) is the one for which the midpoint is less than *v*. The second one (middle child) is the one for which the midpoint is equal to *v*. The third (right child) is the one for which the midpoint is greater than *v*. As discussed above, not all three nodes have to be solved at each iteration.

The binary search procedure would normally start by checking the value of the unknown midpoint versus the value $(UB\text{-}f)/2 + f/2$, since this is the value that divides the width of the initial interval in half. Computational experience has indicated that most of the times the optimal value of this midpoint is relatively close to the average b/r. Therefore, the performance of the algorithm is improved when this is the first value used to divide the initial interval by half. The size of the binary search tree (equity tree) is limited as a result of this decision. Of course, this is only done if $b/r < UB\text{-}f/2$. If not, then the first value used is $(UB\text{-}f)/2 + f/2$.

The algorithm uses intelligently the special structure of the problem. More specifically, the following rules provided significant time savings and considerably improved the performance of the algorithm.

1. If the solution to a parent node of the equity tree is feasible for one of its children nodes, then it is also the optimal solution to this child node. This is because the set of feasible solutions of a child node is a subset of the set of feasible solutions of its parent node.

2. If the solution to a left or to a right subproblem of the equity tree is feasible to the middle subproblem then it is also the optimal solution for this middle subproblem. This is because the set of feasible solutions of a middle subproblem is a subset of the set of feasible solutions of both its sibling subproblems.

3. For the same reason as above, a middle node is added to the equity tree and solved only if none of its two sibling subproblems was fathomed before.

4. The branch and bound search procedure at a node of the equity tree is stopped if an upper bound for this solution is found which is smaller than the objective value of the best solution that has been found so far.

As the algorithm proceeds, the interval containing the unknown midpoint becomes smaller and smaller. Similarly, the objective function value of the best feasible solution that has been found so far gets closer to the optimal objective function value of the problem. Of course, the exact value of this midpoint will not be found unless at some point we use exactly this value to pivot and divide the search interval. Therefore, the binary search concludes as soon as a preset relative accuracy on the optimal objective function value has been achieved. Note that, although the search is performed based on the midpoint of the interval of uncertainty, the desired accuracy refers to the optimal objective function value.

Based on the above analysis, a brief outline of the Algorithm MIMCKE that was developed for the problem is presented next. Note that the variable lists constructed in Step 0 of Algorithm MIMCKFE need only to be constructed once.

Algorithm MIMCKE

Step 0 (Preprocessing)

Compute the upper bound UB. The unknown midpoint lies in the interval $[f/2, UB\text{-}f/2]$. Set $v = b/r$. If $v > UB\text{-}f/2$, set $v = (UB\text{-}f)/2 + f/2$. Using Algorithm MIMCKFE, solve the subproblems in which the midpoint is smaller, equal and larger than v, respectively, and add them to the equity tree. Update the Incumbent accordingly.

Step 1 (Iteration)

```
Select the equity node with the maximum upper bound from the equity tree
If the preset accuracy is not met or this solution is (equity) infeasible do{
    - divide the interval of the node selected into two equal intervals
    - add the three associated subproblems to the equity tree;
    - solve each of these three subproblems using Algorithm MIMCKFE
    - update the Incumbent if necessary;
    - go to the beginning of Step 1 again.
}end if
Otherwise, STOP the Incumbent solution is optimal. □
```

7.6 Computational Experience

In this section the computational complexity of Algorithm MIMCKFE is examined. Then, the experimental design and computational results that were obtained from testing both algorithms are presented and their performance is compared to that of a commercial package for mixed integer programming (LINGO, 2001). This analysis provides important insights into the structure of the problem and the behavior of the algorithms that can be valuable in future research.

7.6.1 Computational Complexity

Let N_k and B_k be the number of continuous and binary variables, respectively, in set k, $N = \sum_{k\in S} N_k$, $B = \sum_{k\in S} B_k$, $N_{max} = \max_{k\in S} N_k$ and $B_{max} = \max_{k\in S} B_k$. The nondominated continuous variables of a set k can be identified in time $O(N_k \log N_k)$ by a suitable algorithm for finding the convex hull of N_k points in two dimensions. The binary variables of set k can be arranged in non-increasing order of their associated slopes in time $O(B_k \log B_k)$. Then, all the variables of this set can be merged in a single list in time $O(N_k + B_k)$. Hence, the time needed to construct the variable list of set k is $O((N_k + B_k) \max (\log N_k, \log B_k)$. Thus, the time needed to construct the variable lists of all r sets is $O(\sum_{k \in S} [(N_k + B_k) \max (\log N_k, \log B_k)]) = O((N + B) \max (\log N_{max}, \log B_{max}))$. Once these lists are constructed, the time needed for merging them to obtain the master list ML is $O((N + B) \log r)$ (see Cormen et al., 2001). Therefore, the total time required is $O((N + B) \max (\log N_{max}, \log B_{max}, \log r))$.

The work needed in Step 0 of the Algorithm MIMCKFE dominates the work needed in Steps 1 and 2, which is linear in the total number of variables. Step 3 requires constant time and Steps 5 and 6 require time which is linear in the total number of variables. Together with Step 4 however, in the worst case, Steps 5 and 6 are executed exponentially many times, since the maximum number of subproblems generated is an exponential function of the total number of binary variables. Note that the time required to find the upper bound of Proposition 7.4 is $O(r)$, since it takes constant time to find it based on one fractional-valued binary variable and there are at most r such variables.

For the heuristic procedure, the dominating operation is the construction of the variable lists which requires time $O((N + B) \max (\log N_{max}, \log B_{max}, \log r))$. Once these lists are constructed, the time required to do the forward and backward pass is linear in the total number of variables.

7.6.2 Experimental Design

The above algorithms were coded in C/C++ and tested on a Pentium IV/1.8 GHz processor. Various combinations for the number of sets as well as the number of continuous and binary variables within each set were used. For all problems, the number of continuous variables was equal to the number of binary variables in each set and this number was kept constant in all sets.

Parameters p_{ki}, q_{kj}, c_{ki} and d_{kj} were uniformly distributed between 0 and N_k. While the following tables show the initial mix of binary and continuous variables, the problems that result after elimination of the dominated variables, contain significantly fewer continuous variables. This is because the expected percentage of integer dominated variables within a multiple choice set increases from 70% when $N_k = 10$, to 96% when $N_k = 150$ (Sinha and Zoltners, 1979). This reduction in the number of continuous variables worsens the performance of the algorithm in a similar manner as already explained for Problem MIMCK. Parameter l_k was set equal to 1 for all multiple choice sets. Finally, budget b was set at $\frac{1}{2}\sum_{k\in S}(\min_{i\in R_k} c_{ki} + \max_{i\in R_k} c_{ki}) + 0.25BN_k$. Thus, the average budget amount available for allocation to each multiple choice set and each binary variable was $0.5N_k$ and $0.25\ N_k$, respectively. This decision for the average budget amount allocated to each binary variable ensures a tight budget constraint, i.e., a scarce budget. Computational experience indicates that the problems generated this way show more computational interest.

The performance of the branch and bound algorithm for the MIMCKFE is affected by a number of different parameters. The location of the interval $[L,U]$ is very important. If U lies to the left of the value b/r, then the optimal solution has a positive (unused) budget residual. If on the other hand L lies to the right of the value b/r, the problem is infeasible, since $rL > b$. Therefore, to generate interesting instances for Problem MIMCKFE, the midpoint of the interval $[L,U]$ should be relatively close to b/r. For all computational results presented in the next tables, the midpoint of this interval was set exactly equal to b/r.

The width of the interval $[L,U]$ also affects the performance of the algorithm, mainly because it affects the likelihood that a solution violates the equity constraints if these are ignored. In other words, the larger the width of this interval, the more likely it is that the equity constraints will be satisfied by an optimal solution to the relaxed problem without equity constraints. As a result, the difficulty of the problem seems to increase as the width of this interval decreases. It should also be noted that, for similar reasons, the difficulty of the problem seems to decrease when only one of the two bounds, L and U, is imposed.

7.6.3 Computational Results

Tables 7.1 and 7.2 present the computational results for Algorithm MIMCKFE. For each problem size, 10 different instances were tested. The decision for the parameters r, B_k and N_k was made in such a way that the complete behavior of the problem is depicted. The value of L was set equal to 0.8 b/r and the value of U was set equal to 1.2 b/r. This way, the width of the interval $[L,U]$ was equal to 0.4 b/r and the maximum allowable difference on the costs of any two sets varied between $0.005b$ for the problems with $r = 80$ and $0.02b$ for the problems with $r = 20$.

Table 7.1 presents results regarding the quality of the solutions obtained. The first two columns (LP Avg and LP Max) present the average and the maximum difference between the objective function value of the linear relaxation of the problem and the optimal value as a relative percentage. The next two columns (Heur Avg and Heur Max) present the same results for the heuristic solution. For instances it could not solve, the algorithm was stopped as soon as the limit of 50,000 tree nodes was reached. The next column (Gap Avg) presents the average relative percentage by which the upper bound obtained before the algorithm was stopped differed from the optimal objective function value. This result was computed over the number of instances for which this performance was exhibited for each problem size. This number is shown in the last column of Table 7.1 (# failed).

Table 7.2 presents results for the CPU time needed to obtain the optimal solution using the various algorithms and the size of the corresponding branch and bound tree for the algorithm proposed. The first two columns (LIN Avg and LIN Max) present the average and maximum time needed by LINGO. The next two columns (MIM Avg and MIM Max) show these times for Algorithm MIMCKFE while the next two (Heur Avg and Heur Max) for the heuristic. The last two columns (Nodes Avg and Nodes Max) present the average and maximum size of the branch and bound tree. Note that unsolved instances were not included. The heuristic terminated with a feasible solution for all instances tested.

Table 7.1: Quality of the solutions obtained by the various algorithms for MIMCKFE

r	N_k	B_k	LP Avg (%)	LP Max (%)	Heur Avg (%)	Heur Max (%)	Gap Avg (%)	# failed
20	100	100	0.00183	0.00479	0.00090	0.00213		0
20	200	200	0.00012	0.00020	0.00042	0.00140		0
20	300	300	0.00004	0.00008	0.00005	0.00016		0
20	400	400	0.00003	0.00006	0.00009	0.00045		0
40	100	100	0.00181	0.00507	0.00099	0.00375	0.00134	3
40	200	200	0.00006	0.00036	0.00009	0.00041	0.00004	1
40	300	300	0.00001	0.00002	0.00004	0.00008		0
40	400	400	0.00001	0.00001	0.00003	0.00007		0
60	100	100	0.00152	0.00372	0.00060	0.00168	0.00040	6
60	200	200	0.00010	0.00034	0.00008	0.00048	0.00003	2
60	300	300	0.00001	0.00001	0.00003	0.00016		0
60	400	400	0.00001	0.00004	0.00001	0.00002	0.00001	1
80	100	100	0.00162	0.00265	0.00058	0.00260	0.00043	8
80	200	200	0.00005	0.00035	0.00010	0.00079	0.00013	2
80	300	300	0.00003	0.00011	0.00005	0.00028	0.00002	2
80	400	400	0.00000	0.00000	0.00001	0.00002		0

Table 7.2: CPU times (in seconds) for the various algorithms and size of the B&B tree of the proposed algorithm for MIMCKFE

r	N_k	B_k	LIN Avg	LIN Max	MIM Avg	MIM Max	Heur Avg	Heur Max	Nodes Avg	Nodes Max
20	100	100	2.4	4	2.565	15.690	0.028	0.030	4113	21929
20	200	200	5.7	12	3.238	26.576	0.061	0.070	3323	20811
20	300	300	9.5	15	6.461	20.031	0.102	0.120	2434	8637
20	400	400	22.1	87	17.963	133.660	0.143	0.220	4609	24631
40	100	100	6.5	12	1.509	4.406	0.065	0.140	3002	10149
40	200	200	11.1	14	0.922	3.014	0.122	0.140	1102	2663
40	300	300	17.5	22	0.928	2.334	0.202	0.230	1243	3507
40	400	400	30.0	39	2.201	10.233	0.289	0.321	1984	6385
60	100	100	9.5	10	1.395	1.973	0.088	0.110	3048	4885
60	200	200	19.1	27	1.311	4.235	0.193	0.230	1509	4451
60	300	300	34.6	91	21.733	160.595	0.306	0.321	5560	30091
60	400	400	51.4	82	15.256	93.672	0.435	0.461	3617	15151
80	100	100	16.0	17	5.760	9.636	0.115	0.130	9216	14231
80	200	200	24.1	35	4.585	14.771	0.258	0.290	4050	15959
80	300	300	57.0	124	29.623	213.075	0.413	0.431	5477	28545
80	400	400	63.9	96	10.528	62.676	0.576	0.620	2929	14265

A number of interesting observations can be made from Table 7.1. The linear relaxation bounds are extremely tight for all problem sizes. This is an implication of the problem sizes tested. As the number of variables in each set increases, these bounds become tighter. That is, for fixed r, the linear relaxation bounds become tighter as N_k and B_k increase. This is due to two main reasons. On the one hand, the various sets tend to resemble each other this way. On the other hand, there are more variables with similar associated slopes within each set and therefore more combinations of variables that can result in similar solutions. Thus, the gap between the objective function value of the linear relaxation and that of the exact solution decreases. The effect of increasing the number of sets is not so apparent. It is clear from Proposition 7.2 that the number of sets is directly related to the number of binary variables that have a fractional value at the optimal solution of the linear relaxation of the problem. In general, the higher this number is, the larger the gap between the objective function value of the linear relaxation solution and that of the exact solution is expected to be.

The quality of the solutions obtained using the heuristic procedure is very high. This is another implication of the problem sizes tested. The effectiveness of the heuristic depends on the way that the total number of variables is distributed among the sets. The heuristic solution is obtained mainly by rounding the critical binary variables of the problem. As r decreases, the number of binary variables that are rounded decreases too. Therefore, this is expected to improve the quality of this solution. On the other hand, it is clear from Table 7.1 that, as the number of variables in each set increases, the quality of the solutions obtained by the heuristic procedure improves too. This is mainly due to the fact that the effect of rounding these variables becomes negligible, since there are many more variables with similar associated slopes.

The distribution of the variables among the sets affects the performance of the algorithm too. As can be seen from Table 7.1, the problem sizes where the algorithm exhibits its worst performance are not random. In general, it is true that as r increases, the likelihood that this worst case behavior will be exhibited increases too. This is an implication of Proposition 7.2, since r is an upper bound to the total number of binary variables that have a fractional value in the optimal linear relaxation solution. On the other hand, the effect of increasing N_k and B_k is not as significant, since r always acts as a barrier to this number. It is clear from Table 7.1 that the worst case behavior is exhibited for large values of r. It is also very interesting to note that, for fixed r, the number of instances where the worst case behavior is exhibited increases as the values of N_k and B_k decrease.

The results of Table 7.2 show a good average performance for Algorithm MIMCKFE. This is however highly variable, in contrast to the behavior of LINGO which seems to be more predictable. Certain instances require significantly more time for the algorithm and have not been included. For some problem sizes, Algorithm MIMCKFE exhibits a better performance than LINGO, while for other, LINGO does. It should be noted however that LINGO was able to terminate in reasonable time even for those instances where Algorithm MIMCKFE exhibited its worst behavior.

The good performance of Algorithm MIMCKFE for some problem sizes is mainly due to the tightness of the linear relaxation upper bounds. The algorithm exploits the special structure of the problem and takes advantage of these bounds when they are tight. Additionally, the algorithm is also able to obtain the optimal solution much faster when the equity constraints at the root of the branch and bound tree are not binding for all sets. This is because, in such cases, the number of binary variables with a fractional value in the optimal linear relaxation solution is small.

The high efficiency of LINGO is mainly due to the use of lifting constraints to tighten the linear programming relaxation. As a result, LINGO explores a much smaller number of tree nodes. On the other hand, Algorithm MIMCKFE is much more efficient in finding the linear relaxation solution to each subproblem of the branch and bound tree due to the efficient specialized algorithm that solves these. The algorithm however, fails when it has to explore an exponentially large number of tree nodes. Valid inequalities could not be used in Algorithm MIMCKFE, since this would have disallowed the utilization of the specialized algorithm for the linear relaxation. How such inequalities can be utilized to develop a more efficient algorithm is a topic that shows great interest for future research.

Another characteristic of Algorithm MIMCKFE is that the upper bound on the optimal solution decreases very slowly as the branch and bound tree is explored. This is mainly due to the strategy used for selecting the next node to explore in the branch and bound tree. The node with the largest upper bound among all active nodes is always selected in order to limit the size of the tree. This strategy was preferable, since the reoptimization of the tree subproblems is very fast and the memory usage by the algorithm is very low.

It should also be noted that the performance of Algorithm MIMCKFE is good when the solution provided by the heuristic procedure is also good. This is because the various parameters of the problem have a similar effect on these two procedures. As already mentioned, the likelihood that the heuristic procedure will terminate with a feasible solution also depends strongly on these parameters. More specifically, this likelihood increases as the value of r decreases and the values of N_k and B_k increase. The value of r increases the number of binary variables rounded by the heuristic and therefore it needs to be small in order for the resulting solution to be good. On the other hand, as the number of variables in each set increases, the heuristic has a larger number of options when making the necessary decisions.

In the problem sizes of Tables 7.1 and 7.2, the number of sets is small compared to the number of variables in each set. It turns out that r is a crucial parameter for the difficulty of the problem. In fact, the performance of all algorithms worsens very fast as the number of sets increases. We believe that, among others, future research should focus on how this difficulty can be handled effectively.

As the number of variables in each set increases, the performance of LINGO worsens faster than that of Algorithm MIMCKFE. This is because the latter takes advantage of the fact that the optimal linear relaxation solution contains at most r fractional-valued binary variables. On the other hand, as the number of sets increases, the performance of Algorithm MIMCKFE worsens much faster than that of LINGO. This is due to the benefit of using the lifting constraints in LINGO.

Given that the difficulty of the problem increases significantly when the exact location of the interval of uncertainty is not known, the size of the problems tested for MIMCKE was reduced by half. To get a better sense of the resulting difficulty and of the effect that the equity constraints have on the optimal objective function value of the problem, Table 7.3 is presented. The results in this table are based on only one instance for each problem size because they are simply intended to give a qualitative flavor about the problem. This table shows the time needed for LINGO to obtain the optimal solution to MIMCKE and the optimal objective function value for the problem with and without equity constraints. The problem instances were generated similarly as before and the value of f was set equal to $0.01b$. The last column of Table 7.3 shows the relative percentage reduction in the objective function value resulting from the incorporation of the equity constraints. This reduction does not vary significantly for different instances of the same problem size.

Table 7.3: Effect of equity constraints on the optimal objective function value

r	N_k	B_k	Time (secs)	Obj	Obj no f	% Red
10	50	50	3	10503.66	10573.24	0.6581
10	100	100	3	42882.54	42910.67	0.0656
10	150	150	12	91255.54	91295.22	0.0435
10	200	200	10	166969.1	167068.8	0.0597
20	50	50	875	21599.74	21634.6	0.1611
20	100	100	12	84194.68	84222.54	0.0331
20	150	150	461	187389.1	187459.7	0.0377
20	200	200	1143	325357.2	325497.0	0.0429
30	50	50	18	31674.5	31708.18	0.1062
30	100	100	10	125694.4	125723.0	0.0227
30	150	150	9	280930.0	280971.4	0.0147
30	200	200	16	497318.7	497318.7	0.0000
40	50	50	14	42973.61	42992.3	0.0435
40	100	100	7	168460.8	168460.8	0.0000
40	150	150	12	375254.2	375254.4	0.0001
40	200	200	21	658472.3	658472.3	0.0000

It is clear from the results of Table 7.3 that the performance of LINGO is really poor for some instances. This becomes evident in the problems with (r,N_k,B_k) = (20,50,50), (20,150,150) and (20,200,200) where LINGO needs 875, 461 and 1143 seconds, respectively, to terminate. When the equity constraints were dropped from these problems, LINGO needed only 2, 4 and 5 seconds, respectively, to find the optimal solution. These cases, provide a clear indication of the great difficulty introduced by the incorporation of the equity constraints.

The reduction in the objective function value due to the incorporation of the equity constraints is relatively small for the problems shown in Table 7.3. As a relative percentage, this reduction decreases as the number of sets increases, mainly because the absolute value of the width of the interval of uncertainty becomes larger.

Tables 7.4 and 7.5 present the computational results for Problem MIMCKE. Due to the fact that LINGO often exhibited an exponential behavior, a relative accuracy of 10^{-3} in Table 7.4 and 10^{-4} in Table 7.5 was used. The same relative accuracy was used for Algorithm MIMCKE, to be able to directly compare the results obtained by the two algorithms. Ten instances for each problem size were tested and f was set equal to $0.01b$, as before. The heuristic procedure was embedded in Algorithm MIMCKE, since in this way the desired accuracy could be obtained much faster.

Table 7.4 presents the results for the case that the relative accuracy is set at 10^{-3}. The first column (MIM Avg) presents the average time that Algorithm MIMCKE needed to find the optimal solution and the second (LIN Avg), the average time LINGO did. The third column (# better) presents the number of instances out of 10 where the objective function value of the solution provided by Algorithm MIMCKE was better than that provided by LINGO. The fourth column (% MIM Improv) presents the average relative percentage by which the solution provided by Algorithm MIMCKE was better than that obtained by LINGO, for those instances that this occurred. Similarly, the fifth column (% LIN Improv) presents the reverse average relative percentage for the cases where LINGO was better.

Table 7.4: Computational results for MIMCKE when the relative accuracy is set at 10^{-3} (time in seconds)

r	N_k	B_k	MIM Avg	LIN Avg	# better	% MIM Improv	% LIN Improv
10	50	50	0.05	3.0	0		0.028
10	100	100	0.01	1.0	9	0.042	0.005
10	150	150	0.02	1.0	10	0.036	
10	200	200	0.03	2.4	10	0.021	
20	50	50	0.01	1.2	8	0.021	0.019
20	100	100	0.03	2.0	9	0.042	0.014
20	150	150	0.05	3.8	10	0.033	
20	200	200	0.06	4.8	8	0.010	0.001
30	50	50	0.02	1.8	5	0.036	0.005
30	100	100	0.05	3.4	6	0.019	0.005
30	150	150	0.08	6.4	10	0.013	
30	200	200	0.10	8.8	10	0.008	
40	50	50	0.02	2.0	6	0.047	0.006
40	100	100	0.06	4.4	7	0.019	0.003
40	150	150	0.10	14.8	10	0.011	
40	200	200	0.13	14.8	10	0.008	

The results of Table 7.4 clearly illustrate that Algorithm MIMCKE was able on the average to obtain better solutions than LINGO, much faster. For most of these instances, the algorithm obtained solutions with a higher objective function value than LINGO in only a fraction of the time that LINGO required. This is mainly due to the utilization of the heuristic procedure which was fast in providing good solutions within the preset accuracy. This is exactly the reason why the average time for the algorithm seems to increase polynomially with the size of the problem. It should be noted however that, for the problems with $r = 10$, $N_k = 50$ and $B_k = 50$, LINGO provided better solutions for all instances tested. For fixed r, the difference between the solutions obtained by the two algorithms seems to decrease as the values of N_k and B_k increase.

Table 7.5: Computational results for MIMCKE when the relative accuracy is set at 10^{-4} (time in seconds)

r	N_k	B_k	MIM Avg	LIN Avg	# better	% MIM Improv	% LIN Improv	# failed
10	50	50	9.03	1.8	0		0.005	0
10	100	100	23.08	13.0	0		0.023	8
10	150	150	1.65	3.2	0		0.009	6
10	200	200	0.22	5.2	5	0.002	0.003	0
20	50	50		79.3				10
20	100	100		4.8				10
20	150	150	0.05	5.2	4	0.002	0.002	0
20	200	200	0.07	7.0	10	0.003		0
30	50	50		77.8				10
30	100	100	0.05	4.2	3	0.003	0.0011	2
30	150	150	0.08	6.6	0		0.001	0
30	200	200	0.10	10.0	5	0.003	0.004	0
40	50	50	0.09	3.6	2	0.001	0.002	4
40	100	100	0.06	4.2	7	0.005	0.002	0
40	150	150	0.10	8.8	8	0.004	0.001	0
40	200	200	0.13	15.6	10	0.004		0

Table 7.5 presents the same results when the accuracy was set at 10^{-4}. A new last column was added because the Algorithm MIMCKE did not solve some instances at this new accuracy level. This column shows the number of instances for each problem size for which the algorithm failed to return a solution within the limit of 50,000 nodes for the

branch and bound tree. The average time for the algorithm was computed over the instances for which the algorithm terminated.

It is clear from Table 7.5 that the number of instances where Algorithm MIMCKE was able to come up with a better solution than LINGO decreases significantly compared with the case where the accuracy was set at 10^{-3}. Additionally, the performance of Algorithm MIMCKE is really poor for some of the instances with a small number of variables in each set. It should be noted that for several of these instances LINGO's performance was also poor, although LINGO was always able to terminate with a solution in reasonable time. The performance of Algorithm MIMCKE improves as the number of sets increases mainly because the absolute value of the width of the interval of uncertainty becomes larger and the heuristic procedure can be effectively utilized.

The high variability depicted in the results of Tables 7.4 and 7.5 is a clear indication that even for the same problem size, the efficiency of the two algorithms depends strongly on the specific instance of the problem. For this reason, there is no clear pattern indicating how the average time needed to obtain the optimal solution changes as the size of the problem changes. The difference between the solutions provided by the two algorithms is now smaller, intuitively, since a smaller value for the relative accuracy is used.

The results are analogous when an even stronger accuracy is required. The superiority of LINGO becomes clearer and the number of cases where Algorithm MIMCKE fails becomes even larger. It should be noted, however, that LINGO's performance is also really poor for many problems. This is an implication of the difficulty introduced by the equity constraints. The efficiency of Algorithm MIMCKE depends strongly on the efficiency of Algorithm MIMCKFE. The worst case performance of Algorithm MIMCKFE which is exhibited rather often, makes the applicability of Algorithm MIMCKE very limited.

For these reasons, the proposed algorithms can only be efficient when certain problem sizes are solved or when reasonably good approximate solutions are sufficient. On the other hand, when exact solutions are needed, LINGO should be preferred, although its performance can be really poor in many cases, too.

A couple of important effects arising from the incorporation of the equity constraints should be noted at this point. The first is that these constraints may result in an optimal solution with a positive (unused) budget residual even though not all the considered activities have been implemented. This happens when the utilization of this unused budget results in a violation of the equity constraints that cannot be handled successfully.

The second is that incorporation of the equity constraints may result in lower profit decisions within a variable set. This happens when, by the structure of the problem, an inferior solution within a set must be preferred in order for the problem to remain feasible. This could mean for example that a budget amount may need to be “wasted” in a set in order to bring the cost of this set within some desired limits. For these reasons, the number of activities within each set should be as large as possible, because in this case the effect from the incorporation of the equity constraints is minimized.

"It is not enough to have a good mind; the main thing is to use it well." **Rene Descartes**

"The art of being wise is the art of knowing what to overlook." **William James**

"Defeat never comes to any man until he admits it." **Josephus Daniels**

"The man who does not read good books has no advantage over the man who cannot read them." **Mark Twain**

Chapter 8:

Future Research

8.1 Introduction

This chapter contains a discussion on the possible ways in which the present research can be extended in the future. In relevance to the transportation application, a number of additional models are proposed that result after incorporating different elements in the models addressed in this dissertation. In this way, new mixed integer formulations arise that haven't been studied in the past. These models exhibit broad theoretical and practical interest as natural extensions of the models addressed in this dissertation.

Another direction in which future research can be directed is the improvement of the algorithms developed in this dissertation, or the development of more efficient algorithms for the models treated in this work. Since these models are introduced for the first time here, there is probably much that can be done to this direction.

Another way the present research can be extended is by incorporating elements from the models that were addressed in this dissertation to other models used in different applications. For example, the important issue of equity can be incorporated into a large number of different resource allocation problems in which the considered activities can be grouped into disjoint sets. The problem of keeping some balance on the resource amounts consumed from different groups of activities is a problem often encountered by decision makers. The specific formulation may be different than the ones in this dissertation but the present work can be very helpful for modeling and solving the specific problem.

Finally, the adaptation of the developed algorithms to handle different problems seems also very promising. This becomes evident when we consider that the main structure of the models addressed in this dissertation can also arise in many different applications. Many of the techniques that were used to treat these models are quite general and can be used to improve the performance of algorithms dealing with different problems.

8.2 Transportation Extensions

The transportation model of Section 1.3 can be used as a basis for the development of more complex models that incorporate additional aspects of the problem. For example, mutually exclusive alternatives may exist for each of the discrete highway points where some intervention is considered. In that case, multiple choice constraints for the binary variables should also be added to the model.

Nonlinear functions can be used to model the anticipated profit and cost of the continuous improvements when the proportionality assumption is not realistic. The effect of traffic growth resulting from the implemented projects can also be taken into account. It is also very interesting to study the relationship between the accident reduction factors of single and combined improvements.

In the present model, setup costs for continuous improvements were ignored. In future research setup costs can be considered, if not negligible. When the input data are subject to estimation errors, a stochastic variation of the model is more suitable. Finally, to accommodate a multi-year highway safety improvement program as well as incorporate changes of traffic over time, the current model can be extended to a dynamic multi-period model.

8.3 Extensions for MIMCK

Although Algorithm MIMCK is very efficient, it could be potentially improved further in the future. The question of whether the problem can be solved without ordering the variables in each multiple choice set remains open. To this direction, some theory from the work dealing with the so-called core of a knapsack problem (see Pisinger, 1999) could turn out to be very useful.

Another way in which the algorithm could be improved is by deriving tighter upper bounds for the solution of each of the B&B search tree subproblems than the one provided by the LP relaxation solution. Work that has been published on this subject for the 0-1 knapsack problem (e.g. Martello et al., 1999) can provide the fundamentals needed to proceed to that direction.

8.4 Extensions for LMCKE

The question of how the present algorithm for LMCKE can be further improved needs to be investigated. It is possible that such improvements can be made to both phases of the algorithm. The variability of Phase II on problems with dominated variables is also a subject that needs further research and potential improvement. Another question that remains open for future research is whether a lower bound on the optimal cost of each set can be derived, and if not, under which conditions the unused budget at the optimal solution of the problem is equal to 0. This is important because then the lower bound of Corollary 7.2 can be utilized in the algorithm.

Besides the algorithm developed in this dissertation, another algorithm can also be developed for LMCKE. This new algorithm works by maintaining primal feasibility and trying to reach optimality. To do this, after the construction of the multiple choice lists, the equity constraints should not be dropped. Instead, the variables should be increased in such a way as to retain feasibility. To this effect, the variables should always be increased in a way that the maximum resource difference between any two sets does not exceed the maximum allowable value, *f*. Thus, it may be necessary for some iterations of the algorithm to increase simultaneously more than one variable from different sets. The algorithm terminates when either the total available budget is used (scarce budget) or when no other variable exists that can be increased (surplus budget). The solution at that point is optimal.

As already mentioned, the two algorithms can be thought of as dual to each other. The algorithm presented in this dissertation, finds first a superoptimal solution violating the equity constraints and works toward (equity) feasibility while maintaining superoptimality. The suggested algorithm, tries to reach optimality while always remaining (equity) feasible. It would be very interesting to develop and code the second algorithm and then compare its performance against the performance of the existing algorithm. Useful insight could be provided this way and many interesting questions could be answered. One such question that could be posed, for example, is what are the characteristics of the problems for which the one or the other algorithm performs better.

8.5 Extensions for MIMCKFE

The difficulty of this problem necessitates the development of a more efficient algorithm that can handle many more instances of the problem successfully. The main problem with the current algorithm is that, in the worst case, an exponentially large number of tree nodes are generated. Thus, the time needed to reoptimize these subproblems occasionally becomes prohibitive even though the solution to each of these subproblems is obtained very fast using the specialized LP algorithm. Therefore, future research should focus on how to make the size of this B&B tree smaller. The incorporation of cutting plane techniques into the present algorithm seems very promising to this direction, especially if this could take advantage of the special structure of the problem.

Another direction in which future research should be directed is the derivation of tighter upper bounds for the optimal solution of each of the tree subproblems which would also make the size of the B&B tree smaller. Additionally, more strategies can be tried for selection of the next tree subproblem to be explored and of the next binary variable to branch on.

Another topic for future exploration is the improvement of the heuristic that was designed for the problem. Future research could be directed to two main directions. The first is the modification of the heuristic in order to reduce the number of problems for which it terminates without finding a feasible solution. Mainly, the procedure should repeat its search in a more sophisticated way by making additional passes and trying different combinations until a feasible solution is obtained. Secondly, it is worth researching whether the solutions provided by the heuristic can be further improved using a more clever algorithmic procedure. Of course these improvements for the heuristic translate to an extra cost in coding effort and computational efficiency.

"It is better to deserve honors and not have them than to have them and not deserve them." **Mark Twain**

"... seek and ye shall find." **The Book of Matthew**

"Necessity is the mother of invention." **Anon**

"Education is what survives when what has been learned has been forgotten."

B. F. Skinner

Chapter 9: Summary

9.1 Introduction

In this chapter the work presented in this dissertation is briefly summarized. The particular implementation conclusions obtained, were summarized at the end of each of the corresponding chapters.

In this dissertation a resource allocation model was introduced with application in transportation management for allocating funds to highway improvements. The model contains both discrete and continuous activities that can be partitioned into disjoint sets. This makes possible the modeling of both the improvements that can be implemented continuously over a section of a highway and of the traditional ones that refer to a specific intervention at a certain point of a highway. Continuous and binary variables were used to represent these two improvement types, respectively, and the problem was formulated as a 0-1 mixed integer knapsack model. Linear multiple choice constraints were added that handle the interactions that arise between the continuous activities of each set. Equity constraints were also included that ensure a certain balance on the resource amounts allocated to different activity sets. Various subproblems that arise based on this general model were studied and a number of important properties was developed for each of them. Based on this theory, various algorithmic procedures were developed that can be used to obtain the optimal solution of each of these problems. These algorithms were then tested against existing algorithms that can be used alternatively.

9.2 Dissertation Summary

The first problem addressed was the 0-1 mixed integer knapsack problem with linear multiple choice constraints. This is a generalization of two widely known problems, the linear multiple choice knapsack and the binary knapsack problems. Several model properties were developed which were utilized to design a B&B solution algorithm. The algorithm solves at each node of the B&B tree a LP relaxation, using an adaptation of an existing algorithm for the linear multiple choice knapsack problem. The special relationship between the solutions of parent and children subproblems is exploited by the algorithm. This results in high efficiency and low storage space requirements. Computational results demonstrated the efficiency of the algorithm. Analysis of these results provided valuable insights into the properties of the problem.

Then, a new variation of the linear multiple choice knapsack problem was introduced. The problem arises when equity constraints are incorporated into the traditional linear multiple choice knapsack problem. A mathematical formulation was presented and it was proven that this problem structure exhibits several fundamental properties. These were used to develop an optimal two-phased greedy algorithm for its solution. In the first phase, the algorithm enhances an existing method for the linear multiple choice knapsack problem to obtain an initial superoptimal solution. Phase two starts with this solution and, at each iteration, it brings closer together the multiple choice sets that significantly differ in the resource amounts that they consume. This is done in such a way that superoptimality is maintained throughout.

The computational complexity of the algorithm was examined. The computational results presented also illustrated the efficiency of the algorithm. It outperformed a commercial linear programming package and its superiority increased with problem size. The algorithm performs well because it exploits the special structure of the problem. It focuses on the multiple choice sets rather than on individual decision variables.

Finally, two versions of the general model of Section 1.3 in which all the individual elements are included were studied. In this model, an available budget must be allocated to disjoint sets of discrete and continuous activities, while also keeping some balance with respect to the budget amounts allocated to different sets. Several properties for each of the two versions of the model were developed. This methodology was then used to design a branch and bound algorithm for the first version of the model. The procedure solves a linear relaxation at each node of the branch and bound tree using a specialized highly efficient algorithm. Computational experience indicates that the performance of the algorithm depends strongly on the structure of the size of the problem. Although it can be efficient in many cases, it often exhibits exponential behavior which limits its applicability.

For this reason, an optimization-based heuristic was developed based on this algorithm that has a very good performance for many problem sizes. The algorithm was cleverly embedded within a binary search procedure in order to handle the second version of the model. Although the resulting algorithm does not have a superior performance when compared to a commercial software package for mixed integer programming, it can be used in conjunction with the heuristic procedure to obtain good approximate solutions very fast.

The behavior of the model under consideration was explored in depth with the analysis of many computational results obtained using the two algorithms and a commercial package for mixed integer programming. From the analysis of these results, the main characteristics of the model were identified and its interesting behavior was explored.

We believe that the insights gained into the structure of the problems studied in this dissertation and the advantages of the algorithms developed will prove useful in real world applications that involve a large number of decision variables. The general model arose from an application in transportation management for allocating funds to highway improvements. This is only one of a more general class of problems involving balancing the resource amounts consumed by different groups of activities which can be addressed by using the present work.

"A scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die and a new generation grows up that is familiar with it " **Maxwell Planck**

"The greater the difficulty, the more the glory in surmounting it " **Epicurus**

"An expert is someone who knows more and more about less and less, until eventually he knows everything about nothing." **Anon**

"Success generally depends upon knowing how long it takes to succeed." **Montesquieu**

"A lot of fellows nowadays have a B.A., M.D., or Ph.D. Unfortunately, they don't have a J.O.B." **Fats Domino**

"Each problem that I solved became a rule, which served afterwards to solve other problems." **Rene Descartes**

"A journey of a thousand miles begins with a single step." **Confucius**

"The roots of true achievement lie in the will to become the best that you can become."
Harold Taylor

"To accomplish great things, we must not only act, but also dream; not only plan, but also believe." **Anatole France**

"The problems that exist in the world today cannot be solved by the level of thinking that created them." **Albert Einstein**

"If a man's wit be wandering, let him study the mathematics."
Francis Bacon, Essays 1625

APPENDIX I

A comprehensive alphabetical list of all the abbreviations used in this Dissertation:

B&B: Branch and Bound

LMCK: Linear Multiple Choice Knapsack

LMCKE: Linear Multiple Choice Knapsack with Equity Constraints

LP: Linear Programming

MC: Multiple Choice

MIMCK: Mixed Integer Knapsack with Linear Multiple Choice Constraints

MIMCKE: Mixed Integer Knapsack with Linear Multiple Choice and Equity Constraints

MIMCKFE: Mixed Integer Knapsack with Linear Multiple Choice and Fixed Equity Constraints

APPENDIX II

A list of the files containing the source code for all the algorithms developed in this Dissertation:

A) For Problem MIMCK:

- **MIMCKP.c** (main program)
- **MIMCKPImplementation.c** (function definitions)
- **MIMCKPTypes.h** (data structure definitions)

B) For Problem LMCKE:

- **LmcEquity.c** (main program)
- **LmcEquityImplementation.c** (function definitions)
- **LmcEquityTypes.h** (data structure definitions)

C) For Problem MIMCKFE:

- **MIMCKEFixed.cpp** (main program)
- **MIMCKEFixedImplementation.cpp** (function definitions)
- **MIMCKEFixedTypes.h** (data structure definitions)

D) For Problem MIMCKE:

- **MIMCKE.cpp** (main program)
- **MIMCKEImplementation.cpp** (function definitions)
- **MIMCKETypes.h** (data structure definitions)

REFERENCES

Aggarwal V., Deo N. and Sarkar D. (1992), "The Knapsack Problem with Disjoint Multiple-Choice Constraints", *Naval Research Logistics*, 39, 213-227.

Aittoniemi L. (1982), "Computational Comparison of Knapsack Algorithms", Presented at the *XIth International Symposium on Mathematical Programming*, Bonn.

Armstrong R. D., Kung D. S., Sinha P. and Zoltners A. A. (1983), "A Computational Study of a Multiple-Choice Knapsack Algorithm", *ACM Transactions on Mathematical Software*, 9, 184-198.

Balas E. and Zemel E. (1980), "An Algorithm for Large Zero-One Knapsack Problems", *Operations Research*, 28, 1130-1154.

Balintfy J. L., Loss G. T., Sinha P. and Zoltner A. A. (1978), "A Mathematical Programming System for Preference and Compatibility Maximized Menu Planning and Scheduling", *Mathematical Programming*, 15, 63-76.

Barbaresso J. C., Bair B. O., Mann C. R. and Smith J. (1982), "Selection Process for Local Highway Safety Projects", *Transportation Research Record*, 847, 24-29.

Barr R. S. and Ross G. T. (1975), "A Linked List Data Structure for a Binary Knapsack Algorithm", Research Report CCS 232, Center for Cybernetic Studies, University of Texas.

Bellman R. (1957), Dynamic Programming, Princeton University Press, Princeton, New Jersey.

Brown D. B. (1976), "Allocation of Federal Highway Safety Funds using Dynamic Programming", *AIIE Transactions*, 8, 461-466.

Brown D. B. (1980), "Use of Dynamic Programming in Optimally Allocating Funds for Highway Safety Improvements", *Transportation Planning and Technology*, 6, 131-138.

Brown D. B., Bulfin R. and Deason W. (1990), "Allocating Highway Safety Funds", *Transportation Research Record*, 1270, 85-88.

Bulfin R. L., Parker R. G. and Shetty C. M. (1979), "Computational Results with a Branch and Bound Algorithm for the General Knapsack Problem", *Naval Research Logistics Quarterly*, 26, 41-46.

Chandra A. K., Hirschberg D. S. and Wong C. K. (1976), "Approximate Algorithms for some Generalized Knapsack Problems", *Theoretical Computer Science*, 3, 293-304.

Chvatal V. (1980), “Hard Knapsack Problems”, *Operations Research*, 28, 402-411.

Cormen T. H., Leiserson C. E., Rivest R. L. and Stein C. (2001), Introduction to Algorithms, MIT Press, Cambridge Ma.

Dantzig G. B. (1957), “Discrete Variable Extremum Problems”, *Operations Research*, 5, 266-277.

Dantzig G. B. (1963), Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.

Donaldson, G. A. (1988), “Safety Spending: usually Begrudged, often Misallocated”, Proceedings of the *ASCE Conference on Highway Safety at the Crossroads*, San Antonio, Texas, 140-150.

Dudzinski K. and Walukiewicz S. (1984), “A Fast Algorithm for the Linear Multiple-Choice Knapsack Problem”, *Operations Research Letters*, 3, 205-209.

Dudzinski K. and Walukiewicz S. (1987), “Exact Methods for the Knapsack Problem and its Generalizations ”, *European Journal of Operations Research*, 28, 3-21.

Dyer M. E. (1984), “An O(n) Algorithm for the Multiple-Choice Knapsack Linear Program”, *Mathematical Programming*, 29, 57-63.

Dyer M. E., Kayal N. and Walker J. (1984), "A Branch and Bound Algorithm for Solving the Multiple-Choice Knapsack Problem", *Journal of Computational and Applied Mathematics*, 11, 231-249.

Dyer M. E., Riha W. O. and Walker J. (1995), "A Hybrid Dynamic Programming/Branch and Bound Algorithm for the Multiple-Choice Knapsack Problem", *Journal of Computational and Applied Mathematics*, 58, 43-54.

FHWA (1997), Federal Highway Administration, "Synthesis of Human Factors Research on Older Drivers and Highway Safety". http://www.bts.gov/ntl/DOCS/97095/.

Farid F., Johnston D. W., Laverde M. A. and Chen C-J (1994), "Application of Incremental Benefit-Cost Analysis for Optimal Budget Allocation to Maintenance, Rehabilitation and Replacement of Bridges", *Transportation Research Record*, 1442, 88-100.

Fayard D. and Plateau G. (1975), "Resolution of the 0-1 Knapsack Problem: Comparison of Methods", *Mathematical Programming*, 8, 272-307.

Fayard D. and Plateau G. (1982), "An Algorithm for the Solution of the 0-1 Knapsack Problem", *Computing*, 28, 269-287.

Garey M. R. and Johnson D. S. (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

Garfinkel R. S. and Nemhauser G. L. (1972), Integer Programming, Wiley, New York.

Gass S. I. and Shao Jr. S. P. (1985), "On the Solution of Special Generalized Upper-Bounded Problems: The LP/GUB Knapsack Problem and the λ-form Separable Convex Objective Function Problem", *Mathematical Programming Study*, 24, 104-115.

Geoffrion A. M. and Marsten R. E. (1972), "Integer Programming Algorithms: A Framework and State of the Art Survey", *Management Science*, 18, 465-491.

Gilmore P.C. and Gomory R. E. (1961), "A Linear Programming Approach to the Cutting Stock Problem I", *Operations Research*, 9, 849-858.

Gilmore P.C. and Gomory R. E. (1963), "A Linear Programming Approach to the Cutting Stock Problem II", *Operations Research*, 11, 863-888.

Gilmore P.C. and Gomory R. E. (1965), "Multi-stage Cutting Stock Problems of two and more Dimensions", *Operations Research*, 13, 94-120.

Gilmore P.C. and Gomory R. E. (1966), "The Theory and Computation of Knapsack Functions", *Operations Research*, 14, 1045-1074.

Glover F. and Klingman D. (1979), "An O(n log n) Algorithm for LP Knapsacks with GUB Constraints", *Mathematical Programming*, 17, 345-361.

Greenberg H. and Hegerich R. L. (1970), "A Branch Search Algorithm for the Knapsack Problem", *Management Science*, 16, 327-332.

Guignard M. M. and Spielberg K. (1972), "Mixed Integer Algorithms for the 0-1 Knapsack Problem", *IBM Journal of Research and Development*, 16, 424-430.

Hillier F. S. and Lieberman G. J. (2001), Introduction to Operations Research, McGraw-Hill, New York.

Horowitz E. and Sahni S. (1974), "Computing Partitions with Applications to the Knapsack Problem", *Journal of the Association for Computing Machinery*, 21, 277-292.

Hu T. C. (1969), Integer Programming and Network Flows, Addison-Wesley, New York.

Ibaraki T. (1987a), "Enumerative Approaches to Combinatorial Optimization- Part 1", *Annals of Operations Research*, 10.

Ibaraki T. (1987b), "Enumerative Approaches to Combinatorial Optimization- Part 2", *Annals of Operations Research*, 11.

Ibarra O. H. and Kim C. E. (1975), "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems", *Journal of the Association for Computing Machinery*, 22, 463-468.

Ingargiola G. P. and Korsh J. F. (1973), "A Reduction Algorithm for Zero-One Single Knapsack Problems", *Management Science*, 20, 460-463.

Ingargiola G. P. and Korsh J. F. (1977), "A General Algorithm for One-Dimensional Knapsack Problems", *Operations Research*, 25, 752-759.

Johnson D. S. (1974), "Approximation Algorithms for Combinatorial Problems", *Journal of Computer and System Sciences*, 9, 256-278.

Johnson E. L. and Padberg M. W. (1981), "A Note on the Knapsack Problem with Special Ordered Sets", *Operations Research Letters*, 1, 18-22.

Johnson M., Zoltners A. A. and Sinha P. (1979), "An Allocation Model for Catalog Space Planning", *Management Science*, 25, 117-129.

Jones A. P. and Soland R. M. (1969), "A Branch and Bound Algorithm for Multi-Level Fixed Charge Problem", *Management Science*, 16, 67-76.

Kilbridge M. D. and Wester L. (1962), “A Review of Analytical Systems of Line Balancing”, *Operations Research*, 10, 626-638.

Klincewicz J. G. and Luss H. (1986), “A Lagrangian Relaxation Heuristic for Capacitated Facility Location with Single-Source Constraints”, *Journal of Operational Research Society*, 37, 495-500.

Kolesar P. (1966), “Assignment of Optimal Redundancy in Systems subject to Failure”, Operations Research Group Technical Report, Columbia University, New York.

Kolesar P. (1967), “A Branch and Bound Algorithm for the Knapsack Problem”, *Management Science*, 13, 723-735.

Korte B. H. and Vygen J. (2000), Combinatorial Optimization: Theory and Algorithms, Springer-Verlag, New York.

Lauriere M. (1978), “An Algorithm for the 0-1 Knapsack Problem”, *Mathematical Programming*, 14, 1-10.

Lawler E. L. (1979), “Fast Approximation Algorithms for Knapsack Problems”, *Mathematics of Operations Research*, 4, 339-356.

Lin E. Y-H. (1998), “A Bibliographical Survey on Some Well Known Non-Standard Knapsack Problems”, *INFOR*, 36, 274-317.

LINGO (2001), User's Guide. LINDO Systems, Inc., Chicago, IL. http://www.lindo.com/ .

Lorie J. and Savage L. (1955), “Three Problems in Capital Rationing”, *Journal of Business*, 38, 229-239.

Magazine M. J. and Oguz O. (1981), “A Fully Polynomial Approximation Algorithm for the 0-1 Knapsack Algorithm”, *European Journal of Operations Research*, 8, 270-273.

Martello S. and Toth P. (1977), “An Upper Bound for the Zero-One Knapsack Problem and a Branch and Bound Algorithm”, *European Journal of Operations Research*, 1, 169-175.

Martello S. and Toth P. (1979), “The 0-1 Knapsack Problem”, In Christofides N., Mingozzi A., Toth P., Sandi C. (eds), Combinatorial Optimization, Wiley, Chichester, 237-279.

Martello S. and Toth P. (1987), “Algorithms for Knapsack Problems”, In Martello S., Laporte G., Minoux M., Ribeiro C (eds), Surveys in Combinatorial Optimization, *Annals of Discrete Mathematics* 31, North Holland, Amsterdam, 213-257.

Martello S. and Toth P. (1988), "A new Algorithm for the 0-1 Knapsack Problem", *Management Science*, 34, 633-644.

Martello S. and Toth P. (1990), Knapsack Problems, Algorithms and Computer Implementations. Chichester, England: John Wiley & Sons.

Martello S. and Toth P. (1997), "Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems", *Operations Research*, 45, 768-778.

Martello S., Pisinger D. and Toth P. (1999), "Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem", *Management Science*, 45, 414-424.

Martello S., Pisinger D. and Toth P. (2000), "New Trends in Exact Algorithms for the 0-1 Knapsack Problem", *European Journal of Operational Research*, 123, 325-332.

Melching C. S. and Liebman J. S. (1988), "Allocating Railroad Maintenance Funds by Solving Binary Knapsack Problems with Precedence Constraints", *Transportation Research: Part B*, 22, 181-194.

MHD (1994), Massachusetts Highway Department. http://www.magnet.state.ma.us/.

MTA (1998), Massachusetts Turnpike Authority. http://www.massturnpike.com/.

Nauss R. M. (1976), “An Efficient Algorithm for the 0-1 Knapsack Problem”, *Management Science*, 23, 27-31.

Nauss R. M. (1978), “The 0-1 Knapsack Problem with Multiple Choice Constraints”, *European Journal of Operational Research*, 2, 125-131.

Nemhauser G. L. and Wolsey L. A. (1988), Integer and Combinatorial Optimization, Wiley, Chichester.

Opiela K. S. (1985), “Evaluating the Effectiveness of Highway Safety Improvements”, Proceedings of the *Conference on the Effectiveness of Highway Safety Improvements*, Nashville, Tennessee, pp. 180-188.

Pal R. and Sinha K. C. (1998), “Optimization Approach to Highway Safety Improvement Programming”, *Transportation Research Record*, 1640, 1-9.

Papadimitriou C. H. and Steiglitz K. (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice Hall, Englewood Cliffs, New Jersey.

Pisinger D. (1995a), “A Minimal Algorithm for the Multiple Choice Knapsack Problem”, *European Journal of Operations Research*, 83, 394-410.

Pisinger D. (1995b), “An Expanding-Core Algorithm for the Exact 0-1 Knapsack Problem”, *European Journal of Operational Research*, 87, 175-187.

Pisinger D. (1997), “A Minimal Algorithm for the 0-1 Knapsack Problem”, *Operations Research*, 45, 758-767.

Pisinger D. (1999), “Core problems in Knapsack Algorithms”, *Operations Research*, 47, 570-575.

Pisinger D. and Toth P. (1999), “Knapsack Problems”, *Handbook of Combinatorial Optimization, Volume 1*, Du D.-Z. and Pardalos P. M. (eds), Kluwer Academic Publisher, Boston.

Sahni S. (1975), “Approximate Algorithms for the 0-1 Knapsack Problem”, *Journal of the Association for Computing Machinery*, 22, 115-124.

Salkin H. M. and de Kluyver C. A. (1975), “The Knapsack Problem: A Survey”, *Naval Research Logistics Quarterly*, 22, 127-144.

Salkin H. M. (1975), Integer Programming, Addison-Wesley, New York.

Sarin S. and Karwan M. H. (1989), “The Linear Multiple Choice Knapsack Problem”, *Operations Research Letters*, 8, 95-100.

Schrijver A. (1986), Theory of Linear and Integer Programming, Wiley, Chichester.

Sinha K. C. and Hu K. (1985), "Assessment of Safety Impacts of Highway Projects", Proceedings of the *Conference on the Effectiveness of Highway Safety Improvements*, Nashville, Tennessee, pp. 31-40.

Sinha K. C., Kaji T. and Liu C. C. (1981), "Optimal Allocation of Funds for Highway Safety Improvement Projects". *Transportation Research Record*, 808, 24-30.

Sinha P. and Zoltners A. A. (1979), "The Multiple Choice Knapsack Problem", *Operations Research*, 27, 503-515.

Sinha P. and Zoltners A. A. (1982), "Integer Programming Model and Algorithmic Evolution: A Case from Sales Resource Allocation", *TIMS/Studies in the Management Science*, 18, 99-116.

Skinner R. E. Jr. (1985), "RRR Design Standards: Cost Effectiveness Issues", Proceedings of the *Conference on the Effectiveness of Highway Safety Improvements*, Nashville, Tennessee, pp. 41-50.

STPP (1998), Surface Transportation Policy Project, "Potholes and Politics", Washington, DC. http://www.transact.org/Reports/Potholes/title.htm.

STPP (2000), Surface Transportation Policy Project, "Changing Direction-Federal Transportation Spending in the 1990s", Washington, DC. http://www.transact.org/Reports/ Cd/tea21color.pdf.

Suhl U. (1978), "Algorithm and Efficient Data Structures for the Binary Knapsack Problem", *European Journal of Operational Research*, 2, 420-428.

Syslo M. M., Deo N. and Kowalik J. S. (1983), Discrete Optimization Algorithms with Pascal Programs, Prentice-Hall, Englewood Cliffs, New Jersey.

Sweeney, D. J. and Murphy R. A. (1981), "Branch and Bound Methods for Multi-Item Scheduling", *Operations Research*, 29, 853-864.

Taha H. A. (1975), Integer Programming, Academic Press, New York.

U.S. DOT (2001), United States Department of Transportation. Performance report 2000 - Performance plan 2002. http://ostpxweb.dot.gov/budget/Perfplan02.pdf.

Veliev G. P. and Mamedov K. S. (1981), "A Method for Solving the Knapsack Problem", *USSR Computational Mathematics and Mathematical Physics*, 21, 75-81.

Witzgall C. (1980), "On One-Row Linear Programs", A. V. Fiacco & K. O. Kortane (Eds) Ext. Methods and Systems Analysis, 384-414.

Zemel E. (1980), “The Linear Multiple Choice Knapsack Problem”, *Operations Research*, 28, 1412-1423.

Zemel E. (1984), “An O(n) Algorithm for the Linear Multiple Choice Knapsack Problem and Related Problems”, *Information Processing Letters*, 18, 123-128.

Zoltners A. A. (1978), “A Direct Descent Binary Knapsack Algorithm”, *Journal of the Association for Computing Machinery*, 25, 304-311.

Zoltners A. A. and Sinha P. (1980), “Integer Programing Models for Sales Resource Allocation”, *Management Science*, 26, 242-260.