ROBUST OPTIMIZATION OF LINEAR OPTIMIZATION PROBLEMS AND AN APPROXIMATION APPROACH TO SOLVE ROBUST KNAPSACK PROBLEM

By

Blake Smith

Lakmali Weerasena
Assistant Professor of Mathematics
(Major Advisor)

Aniekan Ebiefung
Professor of Mathematics
(Committee Member)

Ossama Saleh
Professor of Mathematics
(Committee Member)

Sumith Gunasekera
Associate Professor of Statistics
(Committee Member)

ROBUST OPTIMIZATION OF LINEAR OPTIMIZATION PROBLEMS AND AN APPROXIMATION APPROACH TO SOLVE ROBUST KNAPSACK PROBLEM

By

Blake Smith

A Thesis Submitted to the University of
Tennessee at Chattanooga in Partial
Fulfillment of the Requirements of the Degree
of Master of Science: Mathematics

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

May 2019

.

ABSTRACT

The goal of classical KP, is to find a subset of items whose total weight does not exceed the knapsack capacity, and whose profit is a maximum. In the robust KP the goal is to find a subset of items whose total weight does not exceed the knapsack capacity, and remains near maximum for the worst scenario. Solving the robust KP exactly is difficult due to this data uncertainty and combinatorial structure of the problem. In this research, a polynomial-time algorithm is proposed to approximately obtain a near optimal solution for the robust KP with a provable quality. The quality is described by an error term and it is derived for the proposed algorithm. It is shown that the error depends on the characteristics of the problem. We verify the accuracy of the algorithm theoretically and computationally. rithm. It is shown that the error depends on the characteristics of the problem. We verify the accuracy of the algorithm theoretically and computationally.

DEDICATION

This Thesis is dedicated to my parents, Jim and Julie Smith, who have been an incredible source of love and encouragment.

ACKNOWLEDGEMENTS

I would first like to thank the Department of Mathematics at the University of Tennessee at Chattanooga for giving me the opportunity to pursue a graduate degree in mathematics. The support of the students and faculty in the past two years has been incredibly helpful in my journey. I especially want to thank my thesis advisor, Dr. Lakmali Weerasena for her guidance through graduate school, in particular, throughout the research process. Without your support and insight in the past two years this thesis would not have been written. I would also like to thank my committee members, Prof. Aniekan Ebiefung, Prof. Ossama Saleh, and Dr. Sumith Gunasekera for your critiques and advice. A special thanks also goes to the SIMCenter-Center of Excellence in Applied Computational Science and Engineering at University of Tennessee Chattanooga for providing funds to give me more time to work on this research. I would also like to thank my professors Dr. Debra Gladden and Dr. Caroline Boulis, from Lee University, without whom I would never have had the confidence to pursue a graduate degree in the first place. To my friends, too many to thank individually, thank you for being there for me to constantly encourage me to continue to work hard in my studies. Finally and dearly, I would like to thank my family for your unending support through the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

CO, Cominatorial Optimization

KP, Knapsack Problem

LPP, Linear Programing Problem

RLLP, Robust Linear Programing Problem

pmf, probability mass function

MLE, maximum likelihood estimate

EF, Efficiency Matrix

LIST OF SYMBOLS

$z(x)$, Objective function value

c, Coefficient vector for objective function

A, Coefficient matrix for constraints

b, Coefficient vector for right hand side of constraints

δ, Amount of perturbation for coefficients, c, A, and b

$\bar{a}_{ij}$, Uncertainty for value a_{ij} in matrix A

$\bar{b}_i$, Uncertainty for value b_i in vector b

$\bar{c}_j$, Uncertainty for value c_j in vector C

U_A, Uncertainty set for matrix A

U_b, Uncertainty set for vector b

U_c, Uncertainty set for vector c

$\mathcal{U}$, Set of uncertainty set's U_A, U_b, U_c

$\bar{x}$, Optimal solution of uncertainty model

x^*, Optimal solution for a problem

x_s^*, optimal solution for a scenario s

w_i, Weight of item i

p_i, Profit added to the objective function for item i

W, Total weight capacity for the knapsack problem

CHAPTER 1

INTRODUCTION

Deterministic optimization is a one of the main branches in optimization, the process in which a best value of a given function satisfying some conditions is obtained. Linear, nonlinear, convex, and discrete optimizations are common areas of the classical deterministic optimization. Although, the deterministic optimization problems provide the best solution for a given scenario the most important assumption, when solving deterministic optimization problem, is that all input data of the problem is known in advance. Unfortunately, this assumption cannot be fulfilled when real-world optimization problems are considered, especially due to the emergence of the data uncertainty. Data uncertainty can come from a multitude of reasons. In some cases data is not yet known, thus, when solving the optimization problem, the forecast for the data is used. In turn, because these forecasts are not certain, they are subject to prediction errors. The second possibility for data uncertainty is that data cannot be measured with complete accuracy. In many of these cases, the data has an average, which is used as the nominal value, and then from there we solve for measurement errors. These two are the common reasons for data uncertainty in optimization problems, but many other factors can cause it as well.

Thus, developing methods to solve optimization problems with uncertain data plays an important role in optimization, and optimization of problems with uncertain data turns to be an important sub-field of optimization. Stochastic optimization and robust optimization have been

studied in the literature to solve optimization problems with uncertainty. Like in deterministic optimization methods, these methods do not provide one or multiple solutions that yield the same best value for a given function. Stochastic optimization is used when a finite number of scenarios are possible for the optimization problem, and some information about the probabilistic distribution of effective data is available. On the other hand, robust optimization is used to solve optimization problems with data uncertainty with the assumption that the objective and constraint functions are only assumed to belong to certain sets in function space or with the assumption that only finite number of scenarios are possible. These sets are called uncertainty sets. In general, the objective of robust optimization is to make a decision that is feasible no matter what the constraints and objective functions turn out to be. The robust optimal solution must be feasible for the worst-case objective function. Different definitions of robustness are given in the literature, and robust optimization methods to solve linear optimization problems with uncertain data was proposed in the early 1970s. Soster [18] proposed a method to solve linear optimization problems with set inclusive constraints, and later this concept has been well studied and extended. Later 90s, Tal and Nemirovski [4] studied convex optimization with an ellipsoidal uncertainty set for all possible input data, and they [5] also proposed a method to find robust solutions of linear optimization problems contaminated with uncertain data. They showed that optimal solutions of linear optimization problems may become severely infeasible if the nominal values of input data is slightly perturbed. Many of these methods and their variants have been addressed in the literature from a computational viewpoint (see, [7], [3], [6]).

In this research, we focus on robust optimization of one of the classical combinatorial optimization (CO) problems. CO plays an important role in optimization. For CO problems, we

are interested in finding an optimal object from a finite number of feasible solutions. These feasible solutions can be obtained using some combinations of the decision variables' components. Some well-known CO problems include the traveling salesman problem (TSP), the set covering problem (SCP), the minimum spanning tree problem (MSTP), and the knapsack problem (KP). Obtaining the optimal solution of these problems is computationally expensive, and sometimes even not possible using a polynomially solvable algorithm. The problems that are not polynomially solvable are called NP-hard problems. Karp [12] showed that TSP, SCP and KP are NP-hard CO problems. Because of these reasons a variety of methods have been proposed to find near optimal solutions, and these methods are called approximation algorithms.

Approximation methods can be categorized into sub groups as heuristic methods, metaheuristic methods, and ϵ-approximation methods. Heuristic methods and metaheuristic methods are empirical search methods and provide feasible solutions for the optimization problem but not necessary optimal. Furthermore, these methods do not guarantee a consistent error bound between the approximated objective function value and the optimal function value. On the other hand ϵ-approximation method also provide a feasible solution for the problem but guarantee a consistent error bound between the approximated objective function value and the optimal function value. The quality of the approximation is described by ϵ. Specifically, during the past couple of decades, the interest in solving CO problems has been growing ([1], [10]), and surveys summarizing those efforts are given by Nemhauser et. al. [16], Kasperski et. al. [13], and many others.

We study classical KP with uncertainty of the objective functions. When we include the concept of uncertainty for NP-hard CO problems, solving them becomes more challenging. In the

KP, we are given a set of items, a collection called the knapsack, with a fixed capacity. Each item has a certain weight value and a profit value. We determine the number of each item to include in the knapsack so that the total weight of the items is less than or equal to the weight of the knapsack and that the total profit value is as large as possible. We consider an uncertain variant of the KP, where the profit of each item is not exactly known in advance but can be described by finite number of scenarios. The main purpose of this study is to develop a performance guaranteed approximation algorithm to obtain a near optimal solution for the KP with a finite number of scenarios. In other words, no matter what scenario appears, our solution remains feasible, and we, furthermore, provide an upper bound for the maximum error. We analyze the worsening of the optimal solution value with respect to the classical problem, and exactly determine its worst-case performance depending on uncertainty for all parameter configurations. In addition, we derive the quality ϵ for the worst-case, with respect to the optimal value of the objective function. We also propose an error function to describe the quality of the algorithm.

This thesis is organized as follows. In Chapter 2, we review the robust optimization of the classical linear optimization problems. In Chapter 3, we review one of the approximation method available for classical KP and provide some numerical studies to demonstrate the algorithm. In Chapter 4, we present our formulation of the KP with a finite number of scenarios and discuss the proposed approximation algorithm and its proprietaries. In Chapter 5, we present numerical results obtained using the proposed algorithm. In Chapter 6, we summarize our work and discusses future research directions.

CHAPTER 2

A COMPARISON BETWEEN CLASSICAL AND ROBUST OPTIMIZATION

In this section we discuss some relevant mathematical concepts and notations related to linear optimization that are used extensively throughout the thesis. We will also discuss the concept of linear optimization with data uncertainty called robust linear optimization and provide geometric and numerical examples to demonstrate this concept.

2.1 Preliminaries and basic definitions

Let $\mathcal{R}^p$ be a finite dimensional Euclidean vector space. We first introduce some basic notations. For $y^1, y^2 \in \mathcal{R}^p$, to define an ordering relation on $\mathcal{R}^p$, the following notation will be used for $p > 1$.

1. $y^1 \leqq y^2$ if $y^1_k \leq y^2_k$ for all $k = 1, 2, \ldots, p$;

2. $y^1 \leq y^2$ if $y^1_k \leq y^2_k$ for all $k = 1, 2, \ldots, p$ and $y^1 \neq y^2$;

3. $y^1 < y^2$ if $y^1_k < y^2_k$ for all $k = 1, 2, \ldots, p$.

In particular, using componentwise orders, the nonnegative orthant of $\mathcal{R}^p$ is defined as $\mathcal{R}^p_{\leqq} = \{y \in \mathcal{R}^p : \ y \geqq 0\}$, the nonzero orthant of $\mathcal{R}^p$ is defined as $\mathcal{R}^p_{\geq} = \{y \in \mathcal{R}^p : \ y \geq 0\}$ and positive orthant of $\mathcal{R}^p$ is defined as $\mathcal{R}^p_{>} = \{y \in \mathcal{R}^p : \ y > 0\}$.

We begin our discussion by introducing a Linear Programing Problem (LPP). Consider the following LPP.

$$\begin{aligned} \text{max} \quad & z(x) = c_1x_1 + c_2x_2 + c_nx_n \\ \text{subject to:} \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{1}$$

Here $c_1x_1 + c_2x_2 + c_nx_n$ is the objective function to be maximized and will be denoted by $z(x)$. The coefficients in the vector c where $c = (c_1, c_2, \ldots, c_n) \in \mathcal{R}^{n\times 1}$ are known cost constants and $x_1, x_2, \ldots, x_n$ are the decision variables. The inequality $\sum_{j=1}^{n} a_{ij} \leq b_i$ denotes the i^{th} constraints. The coefficient a_{ij} for $i = 1, \ldots, m$, $j = 1, \ldots, n$ are the constraint coefficients. These coefficients make a matrix.

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \in \mathcal{R}^{m\times n}.$$

The column vector $b = (b_1, b_2, \ldots, b_m) \in \mathcal{R}^{m\times 1}$ whose i^{th} component (b_i) is called right side of i^{th} constraint, represents the maximum requirement to satisfy. The constraints $x_1, x_2, \ldots, x_n \geq 0$ are the non negativity constraints. A set of values of $x = (x_1, x_2, \ldots, x_n) \in \mathcal{R}^{n\times 1}$ satisfying all constraint is called a feasible solution. The matrix form of a linear optimization problem can be

given as follows:

$$\begin{aligned} \max \quad & z(x) = c_1x_1 + c_2x_2 + c_nx_n \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2}$$

We provide Theorem 1 given in [2].

Theorem 1 *The fundamental theorem of linear programming states that the maximum and minimum of a linear function, if it exist, over a convex polygonal region occurs at a corner points of the feasible region.*

To represent an optimization problem as a LPP, several assumptions such as proportionality, additivity, divisibility and deterministic are needed. Thus, one of the main assumptions in classical optimization problems is certainty of input data. On the other hand, robust optimization, as described in the introduction, deals with uncertainty sets that describe possible values coefficients may take. Thus, when solving robust optimization problems there are multiple possible cases that could occur for the set of constraints. Therefore, we must consider all cases and optimize the objective function under any given set of constraints. In robust optimization we assume each input data belongs to a certain set called an uncertainty set.

2.2 Robust Optimization of Uncertain Linear Optimization Problems

A robust approach to solving LPPs with uncertain input data has been extensively studied widely over the last two decades. We would like to obtain a suboptimal solution for the nominal values of the input data in order to ensure that the solution remains feasible and near optimal for

all possible data changes. We refer the study done by BenTal and Nemirovski in 2000 [5] on linear optimization problems to understand the importance of robustness in practical applications.

There exist two different ways to describe optimization problems with uncertain data. The first type is the interval scenario case. We observed that each coefficient a_{ij}, b_i, and c_j for $i = 1, \ldots, m, j = 1, \ldots, n$ could take any value on an interval surrounding a nominal value. Consider a nominal value a_{ij} for some row i and column j of coefficient matrix A. Let δ_{ij} denote the perturbation of a_{ij} and $\bar{a}_{ij}$ denotes the coefficient in uncertain LPP. Then, as described by BenTal and Nemirovski in 2000 [5], $\bar{a}_{ij} \in [a_{ij} - \delta_{ij}, a_{ij} + \delta_{ij}]$. Similarly we obtain that $\bar{b}_i \in [b_i - \delta_i, b_i + \delta_i]$ where δ_i denotes the perturbation of b_i and $\bar{b}_i$ denotes the coefficient in uncertain LPP and $\bar{c}_j \in [c_j - \delta_j, c_i + \delta_j]$ where δ_j denotes the perturbation of c_j and $\bar{c}_j$ denotes the coefficient in uncertain LPP. In discrete scenario cases the problem is described explicitly. The second type is the discrete scenario case. The, discrete case has a finite number of scenarios included in set S for objective functions. Each scenario $s \in S$ is described as a vector $c^s = (c_1^s \ldots c_n^s)$ with $c_i^s \in \mathcal{R}$ for $i = 1, \ldots n$. The value of the objective function for any $x \in X$ under scenario $s \in S$ can be described as $z(x, s) = \sum_{i=1}^{n} c_i^s x_i$. The optimal solution for $x \in X$ under scenario $s \in S$ is given by x_s^* and its optimal value is denoted by $z(x, x_s^*) = \sum_{i=1}^{n} c_i^s x_s^*$. When we solve a LPP with uncertain data we would like to find the optimal solution for the worst scenario. In other words, with robust optimization problems, we seek to maximize the minimum $z(x, s) = \sum_{i=1}^{n} c_i^s x_i$, can be stated by $\max_{x \in X} \min_{s \in S} z(x, s)$.

Now we provide the formulation of the Robust Linear Optimization problem. Let $a_{ij}, b_i.c_j$ be the nominal values of the LP given in (2). Let $\delta_{ij}, \delta_i, \delta_j$ be the perturbation corresponding to the nominal values $a_{ij}, b_i.c_j$. $U_A = A + \delta_{\bar{A}}$ where $\delta_{\bar{A}}$ is an $m \times n$ matrix such that

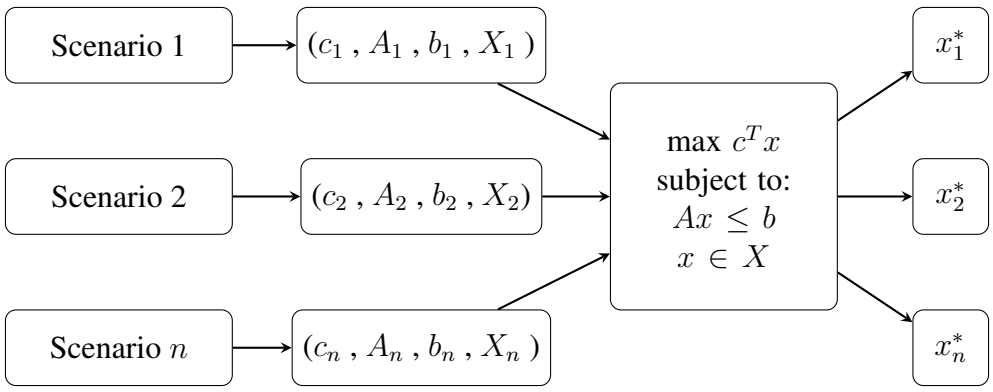

Figure 1 Robust Optimization Scenarios

$\bar{a}_{ij} \in [a_{ij} - \delta_{ij}, a_{ij} + \delta_{ij}]$, $U_b = b + \delta_{\bar{b}}$ where $\delta_{\bar{b}}$ is an $m \times 1$ vector such that $\bar{b}_i \in [b_i - \delta_i, b_i + \delta_i]$ and $U_c = C + \delta_{\bar{c}}$ where $\delta_{\bar{c}}$ is an $n \times 1$ vector such that $\bar{c}_j \in [c_j - \delta_j, c_j + \delta_j]$. Thus for a given robust LP, the uncertainty set $\mathcal{U}$ can be expressed as $\mathcal{U} = \{U_A, U_b, U_c\}$. A Robust Linear Optimization problem is defined in (3)

$$\begin{aligned} &\text{maximize } z(x) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ &\text{subject to} \\ &Ax \leq b \\ &\forall\, (A, b, C) \in \mathcal{U} = \{U_A, U_b, U_c\} \end{aligned} \tag{3}$$

An instance of LP generated by (3) for a given perturbation is called a scenario and by solving the model in (3) we would like to find the best feasible solution for any scenario resulting from model (3). We observe that any solution that satisfies the model in (3) is a feasible solution for the model in (2). Figure (1) shows that for different scenarios there will be different solutions to satisfy the RLLP. For example, let $A_1 \in U_A, b_1 \in U_b, c_1 \in U_c$ in this scenario we will get the optimal solution is x_1^*.

Observation:

Let $\bar{x}$ be the optimal solution of uncertainty model in (3). Then $\bar{x}$ is feasible for the model in (2).

Verification for observation:

Since $\bar{x}$ is optimal for the worst case scenario of RLLP given in (3), we have

$$\sum_{j=1}^{n}(a_{ij}+\delta_{ij})\bar{x}_j \leq b_i - \delta_i \text{ for } i = 1, 2, \ldots, n \tag{4}$$

Now let $\bar{a}_{ij} \in U_A,\ \bar{b}_i \in U_b$ be the input values that generate the feasible constraints for model in (4). Then for any feasible solution x, we have

$$\sum_{j=1}^{n}\bar{a}_{ij}x_j \leq \bar{b}_i \text{ for } i = 1, 2, \ldots, n.$$

By definition of uncertainty sets U_A and U_b we have

$$\begin{aligned} a_{ij} - \delta_{ij} \leq \bar{a}_{ij} \leq a_{ij} + \delta_{ij} \\ b_i - \delta_i \leq \bar{b}_i \leq b_i + \delta_i \end{aligned} \tag{5}$$

Combining inequalities (4) and (5) we have that $\bar{x}$ is feasible for LPP given in (2).

In the next section we describe a geometric procedure for solving a liner programing problem with data uncertainty. Even though this method is only suitable for problems with two variables, it provides great insight into solving linear optimization problems with data uncertainty.

2.2.1 Illustrative Example: Geometric procedure

Let's consider an optimization problem with two variables $x = [x_1, x_2] \in \mathcal{R}^{2\times 1}$ and nominal input data given in A, b and c where

$$A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \\ 4 & 3 \end{bmatrix} \in \mathcal{R}^{3\times 2}, b = \begin{bmatrix} 15 & 12 & 25 \end{bmatrix} \in \mathcal{R}^{3}, c = \begin{bmatrix} 3 & 2 \end{bmatrix} \in \mathcal{R}^{2}. \tag{6}$$

Since there are two decision variables we solve the problem geometrically by graphing the feasible region. Once we have the feasible region, we find corner points and evaluate the objective function at those corner points. Figure 2 illustrates the feasible region and the corner points for the problem data given in (6).

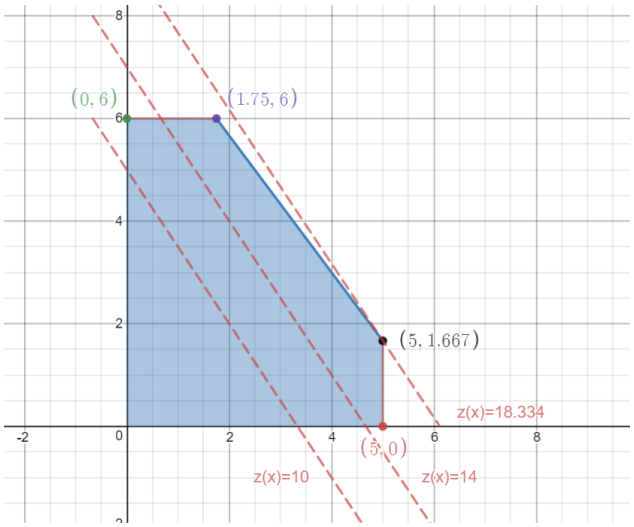

Figure 2 Feasible Region for Example 1

We observe 5 corner points for the feasible region at

$x^{(1)} = (0, 0),\ x^{(2)} = (0, 6),\ x^{(3)} = (1.75, 6),\ x^{(4)} = (5, 1.667),\ x^{(5)} = (5, 0)$. The dotted line in

Table 1 Corner Points for Example 1

	Corner Points	$z(x)$
$x^{(1)}$	$(0, 0)$	0
$x^{(2)}$	$(0, 6)$	12
$x^{(3)}$	$(1.75, 6)$	17.25
$x^{(4)}$	$(5, 1.667)$	**18.334**
$x^{(5)}$	$(5, 0)$	13

Figure 2 shows the objective function at different levels. Since z is to be maximized , the doted line must be moved parallel to itself in the direction that maximizes the objective function most. This is c and, hence the line is moved in the direction of $c = [3, 2]$ as much as possible while maintain the contact with the feasible region. From this geometric figure we see the optimal solution occurs at $x^{(4)} = (x_1, x_2)$. Table 1 shows the objective function values evaluated at these corner points. Further, we observe from this table the point $x^{(4)} = (x_1, x_2) = (5, 1.667)$ will produce a maximum z value of 18.33.

Now, instead of having nominal coefficients for the constraints, lets estimate that there will be up to 10% uncertainty for each input data in matrix A and b. Then the uncertainty set of A is $A + 0.1\%A$ and $\bar{a}_{ij} \in [a_{ij} - 0.1a_{ij}, a_{ij} + 0.1a_{ij}]$ and that of b is $b + 0.1\%b$ and $\bar{b}_i \in [b_i - 0.1b_i, b_i + 0.1b_i]$. These correspond to uncertainty intervals. Under this uncertainty the RLLP related to data given in 6 will be as follows:

$$U_A = A + \delta_A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \\ 4 & 3 \end{bmatrix} + \begin{bmatrix} \pm 0.3 & \pm 0 \\ \pm 0 & \pm 0.2 \\ \pm 0.4 & \pm 0.3 \end{bmatrix} \in \mathcal{R}^{3\times 2}$$

$U_b = b + \delta_b = (15, 12, 25) + (\pm 1.5, \pm 1.2, \pm 2.5) \in \mathcal{R}^{3\times 1}$.

The general form of RLLP can be given as follows.

$$
\begin{aligned}
\max \text{z}= \quad & 3x_1 && +2x_2 && \\
\text{subject to :} & && && \\
& [3 \pm 0.3]x_1 && && \leq [15 \pm 1.5] \\
& && [2 \pm 0.2]x_2 && \leq [12 \pm 1.2] \\
& [4 \pm 0.4]x_1 && +[3 \pm 0.3]x_2 && \leq [25 \pm 2.5] \\
& && x_1, x_2 && \geq 0
\end{aligned}
\tag{7}
$$

Under these conditions there are infinitely many possible scenarios for the given problem, we will examine two of them, namely Scenario 1 and Scenario 2.

Scenario 1: Let coefficient matrix U_A and the corresponding right side U_b be

$$
U_A = \begin{bmatrix} 2.9 & 0 \\ 0 & 2.3 \\ 3.6 & 3.3 \end{bmatrix} \in \mathcal{R}^{3\times 2}, U_b = (15.95, 11.5, 27.3).
$$

Scenario 2: Let coefficient matrix U_A and the corresponding right side U_b be

$$
U_A = \begin{bmatrix} 3.2 & 0 \\ 0 & 1.85 \\ 4.2 & 2.7 \end{bmatrix} \in \mathcal{R}^{3\times 2}, U_b = (13.6, 12.95, 23.25).
$$

Figures 3 and 4 provides the feasible regions corresponds to these scenarios.

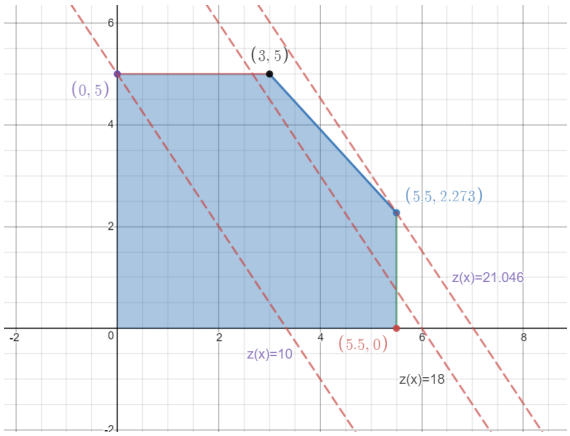

Figure 3 Feasible Region for Scenario I

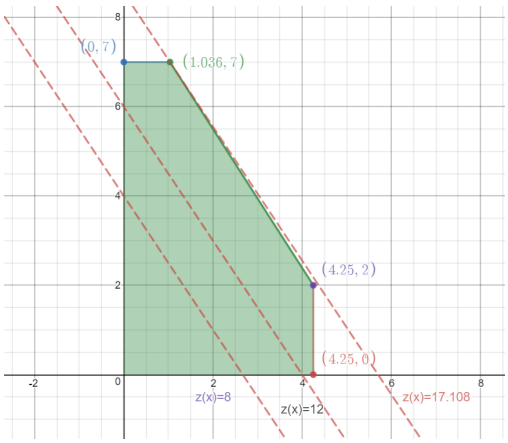

Figure 4 Feasible Region Scenario II

Table 2 Corner Points for Scenario I

	Corner Points	$z(x)$
$x^{(1)}$	$(0, 0)$	0
$x^{(2)}$	$(0, 5)$	10
$x^{(3)}$	$(3, 5)$	19
$x^{(4)}$	$(5.5, 2.273)$	**21.05**
$x^{(5)}$	$(5.5, 0)$	16.5

Table 3 Corner Points Scenario II

	Corner Points	$z(x)$
$x^{(1)}$	$(0, 0)$	0
$x^{(2)}$	$(0, 7)$	14
$x^{(3)}$	$(1.036, 7)$	**17.11**
$x^{(4)}$	$(4.25, 2)$	16.75
$x^{(5)}$	$(4.25, 0)$	12.75

Observing the feasible regions for both cases, we see that with a small amount of perturbation the possible feasible regions for the problem can vary greatly. In Scenario 1 there are corner points at $x^{(1)} = (0, 0), x^{(2)} = (0, 5), x^{(3)} = (3, 5), x^{(4)} = (5.5, 2.273)$ and $x^{(5)} = (5.5, 0)$ and the maximum z value occurs at $x^{(4)} = (x_1, x_2) = (5.5, 2.273)$ z=21.05. Thus we have $z(x^*_{s_1}, s_1) = 21.05$ where $x^*_{s_1}$ denotes the optimal value for Scenario 1. In Scenario 2 there are corner points at $x^{(1)} = (0, 0), x^{(2)} = (0, 7), x^{(3)} = (1.036, 7), x^{(4)} = (4.25, 2)$ and $x^{(5)} = (4.25, 0)$ and the maximum z value occurs at $x^{(3)} = (x_1, x_2) = (1.036, 7)$ $z = 19.18$. Thus we have $z(x^*_{s_2}, s_2) = 21.05$ where $x^*_{s_1}$ denotes the optimal value for Scenario 2. These are two cases that could possibly happen under the given constraints, however, because it is uncertain which case will occur this problem must be solved by analyzing the worst case scenario and optimizing that

case. In order to guarantee a solution will remain feasible no matter what values are taken from the parameters, we will apply the most conservative values from each uncertainty set. This means taking the maximum values for $a_{ij} \in U_A$ and the minimum values for $b_i \in U_b$. Applying these numbers our robust case will result in solving the following LP:

Worst Scenario: Let worst coefficient matrix U_A and the corresponding right side U_b be

$$U_A = A + \max_{\forall a_{ij}} \delta_A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \\ 4 & 3 \end{bmatrix} + \begin{bmatrix} 0.3 & 0 \\ 0 & 0.4 \\ 0.4 & 0.3 \end{bmatrix}$$

and

$$U_b = b + \min_{\forall b_i} \delta_b = (13, 10, 22) + (0.5, 0.8, 0.5) \in \mathcal{R}^{3\times 1}.$$

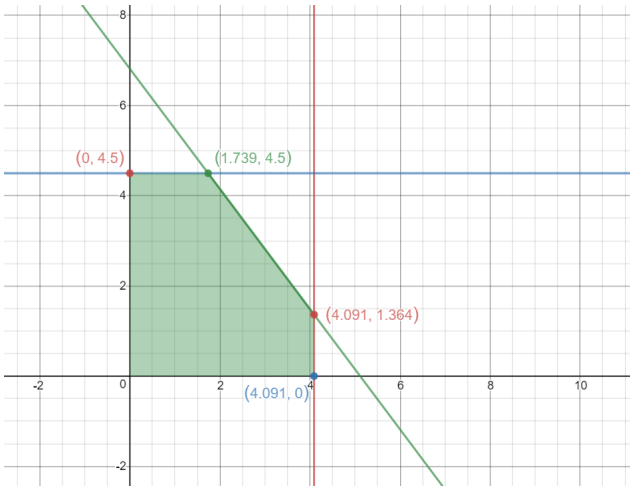

Figure 5 Feasible Region For Worst Scenario

In this scenario we see there are corner points at $x^{(1)} = (0, 0), x^{(2)} = (0, 4.5), x^{(3)} = (1.739, 4.5), x^{(4)} = (4.091, 1.364)$ and $x^{(5)} = (4.091, 0)$. The maximum z value will occur at $x^{(5)} = (x_1, x_2) = (4.091, 1.364)$ and z=15.00. Thus we have

Table 4 Corner Points for Robust Scenario

	Corner Points	z
$x^{(1)}$	$(0, 0)$	0
$x^{(2)}$	$(0, 4.5)$	9
$x^{(3)}$	$(1.739, 4.5)$	14.22
$x^{(4)}$	$(4.091, 1.364)$	**15.00**
$x^{(5)}$	$(4.091, 0)$	12.27

$z(x_s^*, s_2) = 15.00$ where x_s^* denotes the optimal value for worst scenario.

In the Figure 6 we see the feasible region of the worst case scenario compared to the other three feasible regions observed. It is apparent that these corner points will be feasible in the three scenarios, however, the corner points for the previous three cases would not be feasible for the worst case.

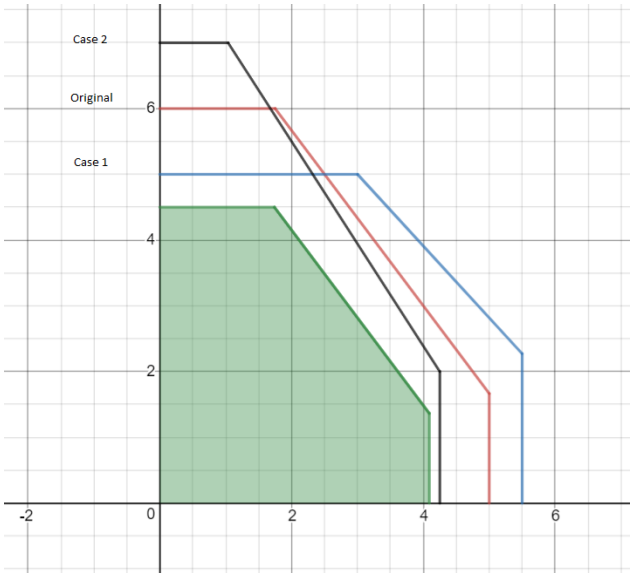

Figure 6 Feasible Region For All Observed Cases

So far we have discussed optimization of linear programming problems when the uncertainty of the input data is described using an interval. Now instead of allowing for an interval of numbers, lets assume each coefficient has three possible values. Lets say

$a_{11} \in \{2.8, 3, 3.2\}, a_{22} \in \{1.5, 2, 3.5\}, a_{31} \in \{2.5, 4, 5\}, a_{32} \in \{2.9, 3, 4\}$ and

$b_1 \in \{12.8, 15, 16\}, b_2 \in \{10.5, 12, 13.5\}, b_3 \in \{22, 25, 26\}$. For this example we will also assume uncertainty in our objective function as well. Let's assume $c_1 \in \{3, 3.5\}$ and $c_2 \in \{2, 2.5\}$ With uncertainty in our objective function we will now observe the following four possible scenarios for the objective function.

1. Scenario 1: $z_1(x) = z(x, s_1) = 3x_1 + 2x_2$
2. Scenario 2: $z_2(x) = z(x, s_2) = 3.5x_1 + 2x_2$
3. Scenario 3: $z_3(x) = z(x, s_3) = 3x_1 + 2.5x_2$
4. Scenario 4: $z_4(x) = z(x, s_4) = 3.5x_1 + 2.5x_2$

Our problem now has a finite number of possible scenarios, to solve this problem we will once again assume the most conservative values by choosing the highest values for each a_{mn}, the lowest values for b_m and the lowest values for c_n. Thus the problem we will seek to optimize will be set up as follows:

Discrete Case: Let worst coefficient matrix U_A and the corresponding right side U_b be

$$U_A = \begin{bmatrix} 3.2 & 0 \\ 0 & 3.5 \\ 5 & 4 \end{bmatrix} \in \mathcal{R}^{3\times 2}, U_b = (12.8, 13.5, 22) \in \mathcal{R}^3, U_c = (3, 2) \in \mathcal{R}^2.$$

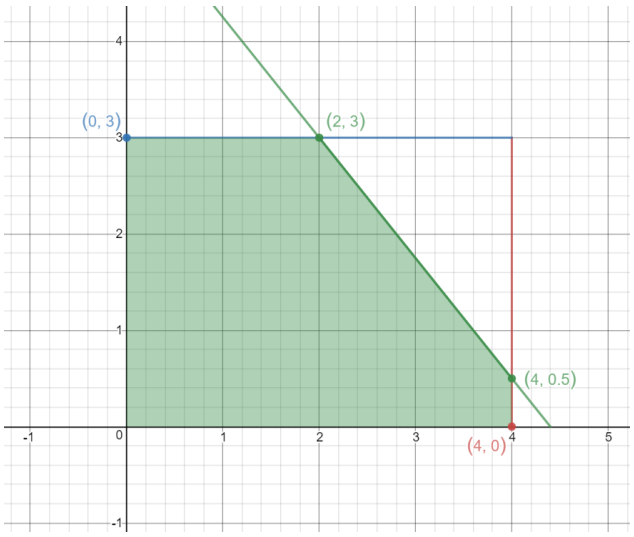

Figure 7 Feasible Region For Discrete Problem

Table 5 Corner Points for Discrete Scenarios

	Critical Points	$z_1(x)$	$z_2(x)$	$z_3(x)$	$z_4(x)$
$x^{(1)}$	$(0,0)$	0	0	0	0
$x^{(2)}$	$(0,3)$	6	6	9	9
$x^{(3)}$	$(2,3)$	12	13	13.5	14.5
$x^{(4)}$	$(4,0.5)$	13	15	13.25	15.25
$x^{(5)}$	$(4,0)$	12	14	12	14

From figure 7 we see that this example of a discrete case has corner points at $x^{(1)} = (0,0), x^{(2)} = (0,3), x^{(3)} = (2,3)x^{(4)} = (4,0.5)$ and $x^{(5)} = (4,0)$. Because there are four different scenarios we want to look at the greatest value in all four scenarios and optimize the worst case scenario. In this example the four scenarios have optimal solutions where in $z_1(x) = 13, z_2(x) = 15, z_3(x) = 13.25,$ and$z_4(x) = 15.25$. Clearly the smallest objective value occurs in the first scenario where $z_1(x)$ has a maximum value at the points $x^{(4)} = (x_1, x_2) = (4, 0.5)$ and $z = 13$.

In the next section we perform an in depth analysis of a real world robust optimization problem. We analyze how the solution and objective function can be effected by small amounts of

perturbation of input data.

2.2.2 Practical Application

In 2009, Ben-Tal et al. [3] introduce an optimization problem in which a certain company produces two types of drug, DrugI and DrugII, containing active agent A. Active agent A is extracted from raw materials, RawI and RawII, which can be used as source of the active agent. The goal of this problem is to maximize the profit while satisfying given constraints. Tables 6, 7, and 8 give details about drug production, contents of the raw materials, and resources.

Table 6 Drug Production

Parameter (Per 1,000 Packs)	DrugI	Drug II
Selling Price	$6,200$	$6,900$
Content of Agent A	0.5	0.6
Manpower Required	90	100
Equipment Required	40	50
Operational Costs	700	800

Table 7 Contents of Raw Materials

Raw Materials	Purchasing Price per kg	Content of Agent A per kg
RawI	100	0.01
RawII	199.9	0.02

Table 8 Resources

Budget, $	Manpower	Equipment	Capacity of Raw Material Storage
$100,000$	$2,000$	800	$1,000$

Using the information given to us in Tables 6, 7, and 8 we formulate the optimization problem as follows:

$$
\begin{array}{llllll}
\max\ z(x) = & 100x_1+ & 199.90x_2- & 5500x_3- & 6100x_4 & \\
\text{Subject to :} & & & & & \\
& -0.01x_1- & 0.02x_2+ & 0.5x_3+ & 0.6x_4 & \leq 0 \quad \text{[balance of active agent]} \\
& x_1+ & x_2 & & & \leq 1000 \quad \text{[storage constraint]} \\
& & & 90x_3+ & 100x_4 & \leq 2000 \quad \text{[manpower constraint]} \\
& & & 40x_3+ & 50x_4 & \leq 800 \quad \text{[equipment constraint]} \\
& 100x_1+ & 199.9x_2+ & 700x_3+ & 800x_4 & \leq 100000 \quad \text{[budget constraint]} \\
& & & x_1, x_2, & x_3, x_4 & \geq 0
\end{array}
\tag{8}
$$

The matrix form of this problem is as follows: $c = \begin{bmatrix} 100 & 199.90 & -5500 & -6100 \end{bmatrix} \in \mathcal{R}^4$, $b = \begin{bmatrix} 0 & 1000 & 2000 & 800 & 100000 \end{bmatrix} \in \mathcal{R}^4$

$$
A = \begin{bmatrix}
-0.01 & -0.02 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 90 & 100 \\
0 & 0 & 40 & 50 \\
100 & 199.9 & 700 & 800
\end{bmatrix} \in \mathcal{R}^{5\times 4}
$$

Since we have more than two decision variables we cannot solve this geometrically. Thus, we solved this problem using Matlab "linprog" function. We obtained the optimal solution $(x_1, x_2, x_3, x_4) = (0, 438.789, 17.557, 0)$ where the maximum value for the objective function is 8,819.66.

In the classical approach to optimization the coefficients for the objective function and constraints are fixed numbers. But what happens if in the first constraint of our problem the coefficient for x_1 can vary by .5% and the coefficient for x_2 can vary by 2%? Then, the uncertainty set $\bar{a}_{11} \in [.00995, .01005]$ and $\bar{a}_{12} \in [.0196, .024]$. If we use the same solution of $x_1 = 0, x_2 = 438.789$, then, in order to stay within the constraints, x_3 would decrease to 17.201. This, in turn, would lower the value of the optimal value to 6891.58, a 21% loss from the original optimal value. In order to see how different amounts of perturbation would effect the original optimal solution we introduce different perturbations and obtain the optimal solutions based on that. Tables 9 and 10 display optimal solutions for different amounts of perturbation. The ϵ represents what percentage of possible perturbation observed. For example in Table 9 we analyze the optimal solution if x_1 can perturbation by up to .5%, 1% and 2.5%, respectively, and when $\epsilon = -.1$ we look at when x_1 is .05%, .01%, and .025% lower than the nominal value. Table 10 display the perturbation sets for x_2 if values can vary by 1%, 2%, and 3%, respectively. Table 11 represents optimal solution if we jointly perturb x_1 by $2.5\%$and x_2 by 1%.

Table 9 Perturbations of x_1

	0.5%					1%					2.5%				
ϵ	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$
-1.000	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.900	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.800	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.700	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.600	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.500	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.400	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.300	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.200	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
-0.100	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
0.000	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660	0.000	438.789	17.552	0.000	8,819.660
0.100	0.000	438.789	17.552	0.000	8,819.660	877.085	0.000	17.559	0.000	8,867.330	876.924	0.000	17.582	0.000	9,010.390
0.200	877.085	0.000	17.559	0.000	8,867.330	876.978	0.000	17.575	0.000	8,962.710	876.655	0.000	17.621	0.000	9,248.710
0.300	877.031	0.000	17.567	0.000	8,915.020	876.870	0.000	17.590	0.000	9,058.070	876.386	0.000	17.659	0.000	9,486.880
0.400	876.978	0.000	17.575	0.000	8,962.710	876.762	0.000	17.605	0.000	9,153.400	876.117	0.000	17.698	0.000	9,724.900
0.500	876.924	0.000	17.582	0.000	9,010.390	876.655	0.000	17.621	0.000	9,248.710	875.848	0.000	17.736	0.000	9,962.780
0.600	876.870	0.000	17.590	0.000	9,058.070	876.547	0.000	17.636	0.000	9,343.990	875.580	0.000	17.774	0.000	10,200.500
0.700	876.816	0.000	17.598	0.000	9,105.740	876.440	0.000	17.652	0.000	9,439.250	875.312	0.000	17.813	0.000	10,438.100
0.800	876.762	0.000	17.605	0.000	9,153.400	876.332	0.000	17.667	0.000	9,534.490	875.044	0.000	17.851	0.000	10,675.500
0.900	876.708	0.000	17.613	0.000	9,201.060	876.225	0.000	17.682	0.000	9,629.710	874.776	0.000	17.889	0.000	10,912.800
1.000	876.655	0.000	17.621	0.000	9,248.710	876.117	0.000	17.698	0.000	9,724.900	874.508	0.000	17.927	0.000	11,150.000

Table 10 Perturbations of x_2

	1%					2%					3%				
ϵ	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$
-1.0	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.9	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.8	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.7	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.6	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.5	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.4	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.3	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.2	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
-0.1	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930	877.193	0.000	17.544	0.000	8771.930
0.0	0.000	438.789	17.552	0.000	8819.660	0.000	438.789	17.552	0.000	8819.660	0.000	438.789	17.552	0.000	8819.660
0.1	0.000	438.735	17.567	0.000	8915.100	0.000	438.681	17.582	0.000	9010.510	0.000	438.627	17.598	0.000	9105.900
0.2	0.000	438.681	17.582	0.000	9010.510	0.000	438.573	17.613	0.000	9201.270	0.000	438.466	17.644	0.000	9391.940
0.3	0.000	438.627	17.598	0.000	9105.900	0.000	438.466	17.644	0.000	9391.940	0.000	438.304	17.690	0.000	9677.760
0.4	0.000	438.573	17.613	0.000	9201.270	0.000	438.358	17.675	0.000	9582.510	0.000	438.143	17.736	0.000	9963.370
0.5	0.000	438.520	17.629	0.000	9296.610	0.000	438.251	17.705	0.000	9772.990	0.000	437.982	17.782	0.000	10248.800
0.6	0.000	438.466	17.644	0.000	9391.940	0.000	438.143	17.736	0.000	9963.370	0.000	437.821	17.828	0.000	10534.000
0.7	0.000	438.412	17.659	0.000	9487.230	0.000	438.035	17.767	0.000	10153.700	0.000	437.660	17.874	0.000	10818.900
0.8	0.000	438.358	17.675	0.000	9582.510	0.000	437.928	17.797	0.000	10343.900	0.000	437.499	17.920	0.000	11103.700
0.9	0.000	438.304	17.690	0.000	9677.760	0.000	437.821	17.828	0.000	10534.000	0.000	437.338	17.966	0.000	11388.300
1.0	0.000	438.251	17.705	0.000	9772.990	0.000	437.713	17.859	0.000	10724.000	0.000	437.178	18.012	0.000	11672.600

However, not all optimal solutions in the perturbation tables are feasible for the nominal problem so we must check the feasibility of each optimal solution with respect to each perturbations. In Table 12 we display the feasibility of each perpetrated optimal solution point for the original problem, where 1 means that point is feasible in all scenarios and 0 means there is a scenario in which it will not be feasible. We see here that there are many points that will not be

Table 11 Joint perturbations of x_1 and x_2

	$x_1 = 0.5\%$ $x_2 = 2\%$					$x_1 = 1\%$ $x_2 = 3\%$					$x_1 = 2.5\%$ $x_2 = 1\%$				
ϵ	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$	x_1	x_2	x_3	x_4	$z(x)$
-1.000	877.732	0.000	17.467	0.000	8294.570	878.272	0.000	17.390	0.000	7816.620	0.000	439.329	17.397	0.000	7863.980
-0.900	877.678	0.000	17.475	0.000	8342.330	878.164	0.000	17.405	0.000	7912.250	0.000	439.275	17.413	0.000	7959.660
-0.800	877.624	0.000	17.482	0.000	8390.090	878.056	0.000	17.421	0.000	8007.870	0.000	439.221	17.428	0.000	8055.310
-0.700	877.570	0.000	17.490	0.000	8437.840	877.948	0.000	17.436	0.000	8103.460	0.000	439.167	17.444	0.000	8150.930
-0.600	877.516	0.000	17.498	0.000	8485.580	877.840	0.000	17.452	0.000	8199.020	0.000	439.113	17.459	0.000	8246.540
-0.500	877.462	0.000	17.505	0.000	8533.320	877.732	0.000	17.467	0.000	8294.570	0.000	439.059	17.475	0.000	8342.110
-0.400	877.408	0.000	17.513	0.000	8581.050	877.624	0.000	17.482	0.000	8390.090	0.000	439.005	17.490	0.000	8437.670
-0.300	877.355	0.000	17.521	0.000	8628.780	877.516	0.000	17.498	0.000	8485.580	0.000	438.951	17.505	0.000	8533.200
-0.200	877.301	0.000	17.529	0.000	8676.500	877.408	0.000	17.513	0.000	8581.050	0.000	438.897	17.521	0.000	8628.710
-0.100	877.247	0.000	17.536	0.000	8724.220	877.301	0.000	17.529	0.000	8676.500	0.000	438.843	17.536	0.000	8724.200
0.000	0.000	438.789	17.552	0.000	8819.660	0.000	438.789	17.552	0.000	8819.660	0.000	438.789	17.552	0.000	8819.660
0.100	0.000	438.681	17.582	0.000	9010.510	0.000	438.627	17.598	0.000	9105.900	876.924	0.000	17.582	0.000	9010.390
0.200	0.000	438.573	17.613	0.000	9201.270	0.000	438.466	17.644	0.000	9391.940	876.655	0.000	17.621	0.000	9248.710
0.300	0.000	438.466	17.644	0.000	9391.940	0.000	438.304	17.690	0.000	9677.760	876.386	0.000	17.659	0.000	9486.880
0.400	0.000	438.358	17.675	0.000	9582.510	0.000	438.143	17.736	0.000	9963.370	876.117	0.000	17.698	0.000	9724.900
0.500	0.000	438.251	17.705	0.000	9772.990	0.000	437.982	17.782	0.000	10248.800	875.848	0.000	17.736	0.000	9962.780
0.600	0.000	438.143	17.736	0.000	9963.370	0.000	437.821	17.828	0.000	10534.000	875.580	0.000	17.774	0.000	10200.500
0.700	0.000	438.035	17.767	0.000	10153.700	0.000	437.660	17.874	0.000	10818.900	875.312	0.000	17.813	0.000	10438.100
0.800	0.000	437.928	17.797	0.000	10343.900	0.000	437.499	17.920	0.000	11103.700	875.044	0.000	17.851	0.000	10675.500
0.900	0.000	437.821	17.828	0.000	10534.000	0.000	437.338	17.966	0.000	11388.300	874.776	0.000	17.889	0.000	10912.800
1.000	0.000	437.713	17.859	0.000	10724.000	0.000	437.178	18.012	0.000	11672.600	874.508	0.000	17.927	0.000	11150.000

feasible under the original problem. However, the only points that will remain feasible under all scenarios are points when $\epsilon = -1$. Thus for x_1 perturbation of $.05\%$ and x_2 perturbation of 2% $x_1 = 877.732, x_2 = 0, x_3 = 17.467$and $x_4 = 0$ these values will result in an objective value $z(x) = 8294.57$ only a 6% difference from the original objective value. Similarly, if we were to say x_1 could perturbate up to 1% and x_2 could perturbate by up to 3% we our solution would be $x_1 = 878.272, x_2 = 0, x_3 = 17.39,$ and $x_4 = 0$ resulting in an objective value $z(x) = 7816.62$, an 11.37% difference from the original objective value. Finally, when x_1 can perturbate by up to 2.5% and x_2 can perturbate by 1% our solution is $x_1 = 0, x_2 = 439.329, x_3 = 17.397,$ and $x_4 = 0$ for an objective value $z(x) = 7863.98$, a 10.8% difference from the original objective value.

Table 12 Feasibility with respect to the nominal data given in Problem 8

	Perturbation of x_1			Perturbation of x_2			Perturbations of x_1 and x_2		
ϵ	.5%	1%	2.5%	1%	2%	3%	.5%, 2%	1% 3%	2.5% 1%
-1.0	1	1	1	1	1	1	1	1	1
-0.9	1	1	0	1	1	0	1	1	1
-0.8	1	1	1	1	0	0	1	1	1
-0.7	1	1	0	0	0	1	1	1	1
-0.6	0	0	1	1	1	1	1	1	1
-0.5	1	1	0	1	1	1	1	1	1
-0.4	1	1	1	0	1	1	1	1	1
-0.3	0	1	0	1	1	1	1	1	1
-0.2	1	1	1	1	0	1	1	1	1
-0.1	1	1	1	0	1	1	1	1	1
0.0	1	1	1	1	1	1	1	1	1
0.1	1	1	1	0	0	0	0	0	0
0.2	0	1	1	0	0	0	0	0	0
0.3	0	0	0	0	0	0	0	0	0
0.4	0	0	1	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0	0
0.6	0	0	1	0	0	0	0	0	0
0.7	0	0	1	0	0	0	0	0	0
0.8	0	0	1	0	0	0	0	0	0
0.9	0	0	1	0	0	0	0	0	0
1.0	0	0	1	0	0	0	0	0	0

CHAPTER 3

KNAPSACK PROBLEM

As mentioned in the introduction KP is one of the well-known combinatorial optimization problems with many real word applications. In classical KP we assume that all coefficients that appear in the objective function, constraints and right side of the mathematical formulation of the KP are known in advanced. The goal of classical KP, is to find a subset of items whose total weight does not exceed the knapsack capacity, and whose profit is a maximum.

Studying KP is merited especially since the KP has many real-life applications in fields such as scheduling and packing. In particular, the capital budgeting problem is a common problem for its applications as will be discussed in the following section in relation to the the classical KP.

In this problem we are given a certain amount of capital and have identified various investment opportunities. Each investment has a different cost and different expected profits. This can be formulated as a knapsack problem. The mathematical formulation of the knapsack problem is given in the following section.

3.1 Problem Formulation

In the KP there are n number of items contained in the set E with the index set $I = \{i : i = 1, 2, \ldots, n\}$.

Let $x \in \mathcal{R}^n$ be the decision variable defined as follows;

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is selected for Knapsack} \\ 0 & \text{otherwise,} \end{cases} \quad \text{for } i \in I.$$

Let $w_i > 0$ and $p_i > 0$ for $i \in I$ be the capacity and the profit of the item i, respectively. Let W be the capacity of the Knapsack. As mentioned above, the goal of the KP is to select a subset of M items from the set E such that $\sum_{i \in M} w_i \leq W$ maximizing the total profit $\sum_{i \in M} p_i$. It is assumed that each item has size at most W. Thus the feasible region X is defined as

$$X = \{x \in \mathcal{R}^n : \sum_{i \in I} w_i x_i \leq W \text{ for } i \in I\} \text{ and } x_i \in \{0, 1\} \text{ for } i \in I\}.$$

The formulation of KP can be presented as follows:

$$\begin{aligned} \text{maximize } z(x) = & \sum_{i=1}^{n} p_i x_i \\ \text{subject to } & \sum_{i \in I} w_i x_i \leq W \\ & x_i \in \{0, 1\} \text{ for } i \in I \end{aligned} \tag{9}$$

We use Example 1 to discuss standard solution approach for the KP.

Example 1: Suppose that there are four investment opportunities to choose from with costs in thousands of dollars $w_1 = 6, w_2 = 3, w_3 = 4$ and $w_4 = 1$ and they provide profits in thousands of dollars of $p_1 = 2, p_2 = 3, p_3 = 4$ and $p_4 = 5$, respectively. How can one maximize profit if the total cost cannot exceed \$9000? This problem can be formulated as follows:

$$\begin{aligned} &\text{maximize } z(x) = 2x_1 + 3x_2 + 4x_3 + 5x_4 \\ &\text{subject to} \\ &6x_1 + 3x_2 + 4x_3 + x_4 \leq 9 \\ &x_i \in \{0, 1\}, i \in I = \{1, \ldots, 4\} \end{aligned} \tag{10}$$

In order to determine which combination of packages will bring the company the maximum profit we may consider the following. In this problem there are only ${}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4 = 16$ different ways to to pick up to 4 items, however as seen in Table **??** they are not all feasible. For example, Row 8 of Table 13 shows the solution $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0)$ is not a feasible solution since it violates the capacity constraint while Row 7 of Table 13 shows solution $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$ is a feasible solution since it satisfies the capacity constraint. Because there are so few possible combinations as possible

Table 13 Possible Combinations to load 4 items

		x_1	x_2	x_3	x_4	value of z	Feasible
1	Load 0 Items	0	0	0	0	0	Yes
2		1	0	0	0	2	Yes
3	Load 1 Items	0	1	0	0	3	Yes
4		0	0	1	0	4	Yes
5		0	0	0	1	5	Yes
6		1	1	0	0	5	Yes
7		1	0	1	0	6	No
8	Load 2 Items	1	0	0	1	7	Yes
9		0	1	1	0	7	Yes
10		0	1	0	1	8	Yes
11		0	0	1	1	9	Yes
12		1	1	1	0	9	No
13	Load 3 Items	1	1	0	1	10	No
14		1	0	1	1	11	No
15		0	1	1	1	12	Yes
16	Load 4 Items	1	1	1	1	14	No

outcomes we evaluate the objective function value at each feasible combination and select feasible combination with the highest profit. From Table 13 we observe the best feasible combination is to select items 2, 3 and 4 which corresponds to the feasible solution $(x_1, x_2, x_3, x_4) = (0, 1, 1, 1)$.

However, as we increase the number of items to choose from the possible number of combinations increases exponentially. When there are n number of items to choose for the

Knapsack the total number of possible combinations is given by

$$\sum_{i=0}^{n} {}^{n}C_i \text{ where} {}^{n}C_i = \frac{n!}{(n-r)!r!}.$$

From Table 14 shown below we see that as we increase n the possible items to choose from increases by 2^n, thus solving the KP by examining each possible solution becomes unreasonable for larger n values. Some efficient algorithms involving branch and bound techniques

Table 14 Number of combinations for given n

Number of items	Number of possible combinations
$n = 4$	$\sum_{i=0}^{4} {}^{4}C_i$=16
$n = 5$	$\sum_{i=0}^{5} {}^{5}C_i$=32
$n = 10$	$\sum_{i=0}^{10} {}^{10}C_i$=1024
$n = 20$	$\sum_{i=0}^{20} {}^{20}C_i$=1048576
$n = 50$	$\sum_{i=0}^{50} {}^{50}C_i$=1.126E15
$n = 100$	$\sum_{i=0}^{100} {}^{100}C_i$=1.268E30

([14],[15],[9]) have been proposed to solve these types of problems exactly. Even that when the n is reasonably large exactly solving this problem are computationally expensive and variety of studies ([11], [17] and many others) are available in the literature to obtain close solutions so called approximations. As mentioned in the introduction, approximating the solution of combinatorial optimization problem with a performance guarantee is a motivating challenge.

In the following section we provide the definition of ϵ-approximation algorithm and review

ϵ-approximation algorithm proposed in (ref) for the classical KP.

3.2 Approximation for classical KP

The concept of ϵ-approximate algorithm for a COP is defined by many authors. We are interested in finding a near optimal solution, that is, a solution that yields the objective value that is worse than the optimal objective value by a factor of $\epsilon \geq 0$. We make use of a constant ϵ to quantify the representation error.

Definition 1 *Let $\bar{x}$ be a near optimal solution yield by an algorithm for an optimization problem given in model (9). Let x^* be the optimal solution of this problem. Let $\epsilon \geq 0$. The algorithm is called ϵ-approximation algorithm if*

$$z(x^*) \leq (1+\epsilon) z(\bar{x})$$

The classical KP is a well-studied problem and different methods have been proposed in the literature to address it. In the following section we outline the ϵ-approximation available for KP.

3.3 An ϵ-approximation algorithm to classical Knapsack Problem

The key idea of this algorithm is based on the following observation: when selecting an item to be in a maximum profit collection of items, it is important to consider not only the profit of the item but also the weight of the item. This approach to the KP seeks to determine which items offer the most profit for their given weight. This can be done by finding the ratio of profit divided

by weight, then ordering each item from highest ratio to lowest ratio. With the notations defined above $\frac{p_i}{w_i}$ provides the ratio of item i for some $i \in I$. We call this the effectiveness of the item. After finding all these ratios, we sort the items by the ratio so that $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \frac{p_n}{w_n}$. Once the items are organized from highest to lowest ratio, we begin selecting items until we can no longer add items without exceeding the maximum possible weight W. We select items according to this sorted order until the weight constraint is no longer satisfied. It is shown that this algorithm performs poorly in some cases. For example consider two items, that is $i = 1, 2$. Let $W \geq 2$ and $p_1 = 2, p_2 = W, w_1 = 1$ and $w_2 = W$. Then the algorithm selects first item even though its profit is smaller than that of the second item. The modified version of this algorithm has been proposed in [8]. It says best solution obtained by this sorting procedure or the most profitable item. Further if we assume $w_i \leq \epsilon W$ for all $i \in I$, and for some ϵ, then the it is proven that proposed algorithm is a ϵ algorithm. Thus the quality of this algorithm is described ϵ which is given in Theorem 1. The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1 Approximation Approach to Classical KP

Input: n, P, A, W
$\bar{x}$ is an zero row vector in $\mathcal{R}^n$
Initialize: $\bar{I} = \emptyset, z(\bar{x}) = 0, \bar{W} = 0$
for $i = 1 \rightarrow n$ **do**
    $Ratio(i) = \dfrac{p_i}{w_i}$
end for
while $\bar{W} \leq W$ **do**
    Set $t = 1$
    Let $i^* = \max\limits_{i \in I} \left\{ \dfrac{p_i}{w_i} \right\}^T$
    Update $\bar{I} = \bar{I} \cup \{i^*\}$
    Set $\bar{x}(i^*) = 1$
    Update $z(\bar{x}) \rightarrow z(\bar{x}) + p_{i^*}$
    Update $\bar{W} \rightarrow \bar{W} + w_{i^*}$
    $t \rightarrow t + 1$
end while
if $\bar{W} > W$ **then**
    Let i_l be the index of the last item added
    Update $z(\bar{x}) \rightarrow z(\bar{x}) - p_{i_l}$
    Update $\bar{W} \rightarrow \bar{W} - w_{i_l}$
    $\bar{x}(i_l) = 0$
end if
Return: $\bar{x}, \bar{W}$ and $z(\bar{x})$

Theorem 2 *Let w_i be the weight of the item $i \in I$. If $w_i \leq \epsilon W$ for all $i \in I$, the modified greedy algorithm provides an $(1 + \epsilon)$-approximation for the KP.*

The performance of this algorithm is derived mathematically in this this article. We provide the outline of this proof in Appendix A since this study motivates our algorithm proposed in Section 4. We also use *Example 2* to describe the algorithm. *Example 2:* Refer to *Example 1* again. We have $w_1 = 6, w_2 = 3, w_3 = 4$ and $w_4 = 1$ for sizes of items and $p_1 = 2, p_2 = 3, p_3 = 4$ and $p_4 = 5$ for weights. The ratios are $p_1/w_1 = 1/3$, $p_2/w_2 = 1$, $p_3/w_3 = 1$ and $p_4/w_5 = 5$. In order to determine which the best items are one can order these items from greatest ratio to lowest ratio,

thus, the order will be $w_4/p_4 \geq w_2/p_2 \geq w_3/p_3 \geq w_1/p_1$. Since p_4/w_4 provides the highest ratio we first add item 4 and $z = 5$ and the remaining size of the KP is $9 - 1 = 8 > 0$. Since we have enough space for the next item, we add the next best item which is item 2. With this section $z = 5 + 3 = 8$ and the remaining size of the KP is $8 - 3 = 5 > 0$. Continuing the process we next add item 3. With this section $z = 8 + 4 = 12$ and the remaining size of the KP is $5 - 4 = 1 > 0$. It is clear that in the next step if we add the reaming item which is item 1, the total weight will exceed 9, thus we cannot add any more items. Thus the solution return by this procedure is $\bar{x} = (x_1, x_2, x_3, x_4) = (0, 1, 1, 1)$ with the total profit is 12. Observe that using this approach we have come up with the same optimal solution as described in *Example 2*. Thus we have $z(\bar{x}) \leq 2z(x^*)$.

In order to check the performance of this approach we simulated some test problems and verify the accuracy of the algorithm using them. In the following table we analyze how this approximation method compares to the exact method. We use the random integer generator in Matlab to generate n items with random profits and weights. We found the total weight two different ways, the first, we multiplied the smallest item weight by 10. The second weight we generated by taking the largest item weight and multiplying it by 2. We then solved the knapsack problem exactly using the "intprog" function in Matlab, and using the proposed algorithm. We then calculated the error term.

From Table 15 we observe that when we set W to the minimum value of $\min_{i \in I}\{w_i\}$ multiplied by 10 the $z(x^*)$ and $z(\bar{x})$ are the same. For example when $n = 25$, $z(x^*) = z(\bar{x}) = 339$ and the error term is 1. This will not be the situation every time but when $W = \min_{i \in I}\{10w_i\}$ is a smaller value we observe an increase in accuracy. Similarly when the weight is double of the $\max_{i \in I}\{w_i\}$,

Table 15 ϵ- approximation algorithm for KP

	$W = \min_{i \in I}\{10w_i\}$			$W = \max_{i \in I}\{2w_i\}$		
n	$z(x^*)$	$z(\bar{x})$	ϵ	$z(x^*)$	$z(\bar{x})$	ϵ
10	248	248	1	293	248	1.21
15	145	145	1	537	537	1
25	339	339	1	516	516	1
50	207	207	1	630	577	1.09
100	327	327	1	1050	991	1.06

W becomes much larger and the accuracy declines. We observe that for all cases $\epsilon \leq 2$ and this verifies the algorithm.

In the following section we discuss our approach to approximate the optimal solution of Robust KP with p number of scenarios.

CHAPTER 4

APPROXIMATING THE KP WITH FINITE NUMBER OF SCENARIOS

We explain the importance of studying KP with data uncertainty by continuing the capital budgeting problem discussed in Section 3.1. Now let's assume that because of market uncertainty the profits are not exactly known. Depending on the climate of the market some investments will do better while others will do worse. Assume that the profits of investments are influenced by the occurrence of well-defined future events (e.g., different levels of market reactions, public decisions of constructing or not a facility which would impact on the investment projects). Then the problem becomes how to select investments in order to maximize the expected profit in the worst case scenario. Now each possible occurrence can be thought of as one objective function to be maximized and this leads to KP with a finite number of scenarios. The generic formulation of these types of problem is called robust KP and is defined below.

4.1 Formulation of KP with finite number of scenarios

Assume that KP has $q > 1$ number of conflicting scenarios with the index set $K = \{k : k = 1, 2, \ldots, q\}$. Let $w_i^k > 0$ be the profit of an item $i \in I$ with respect to the scenario $k \in K$. The total profit of selected items with respect to the scenario $k \in K$ is given by $\sum_{i \in I^*} w_i^k$ where $I^* \subset I$. In the Robust KP with multiple scenarios the goal is to find a sub collection of items from I such that the profit with respect to all scenarios are maximized.

The Robust KP can be presented as follows:

$$\begin{aligned} \text{max } z(x) = \ & [z_1(x), \ \ z_2(x) \ \ \ldots, \ \ z_q(x)] \\ & \text{subject to } \ x \in X. \end{aligned} \tag{11}$$

where $z_k = \sum_{i=1}^{n} p_i^k x_i$ and X is defined in (9). In matrix form Robust KP can be represented as

$$\begin{aligned} \text{max } z(x) &= Px \\ \text{subject to } \ Ax &\leq W. \end{aligned} \tag{12}$$

where

$$P = \begin{bmatrix} p_1^1 & p_1^2 & \ldots & p_1^q \\ p_2^1 & p_2^2 & \ldots & p_2^q \\ \vdots & \vdots & & \vdots \\ p_n^1 & p_n^2 & \ldots & p_n^q \end{bmatrix}$$

is an $n \times q$ matrix that contains profits of each item with respect to each scenario and

$A = [w_1, w_2, \ldots, w_n]$ is a row vector in $\mathcal{R}^n$.

Since we would like to identify a collection of items that gives an optimal solution for the

worst scenario, Problem given in (11) can be reformulated as

$$
\begin{aligned}
&\text{maximize } \theta \\
&\text{subject to} \\
&\theta \leq z_1(x) = p_1^1 x_1 + p_2^1 x_2 + \cdots + p_n^1 x_n \\
&\theta \leq z_2(x) = p_1^2 x_1 + p_2^2 x_2 + \cdots + p_n^2 x_n \\
&\qquad\qquad \vdots \\
&\theta \leq z_q(x) = p_1^q x_1 + p_2^q x_2 + \cdots + p_n^q x_n \\
&Ax \leq W.
\end{aligned}
\tag{13}
$$

Problem (13) can be interpreted as identify the optimal solution that is feasible and optimal for all possible scenarios. In other words, it is equivalent to saying that we maximize the worst scenario. Since the number of scenarios add an extra difficulty to solve the robust KP, we propose an algorithm to solve this problem. We now provide the concept of ϵ- approximation for robust KP and we describe the quality of the algorithm by a constant ϵ. The quality term is valid term for all possible scenarios.

We now develop the algorithm to approximate an optimal solution of the Robust KP with finite number of scenarios.

4.2 The algorithm for Robust KP with finite scenarios

We first present the concept of the vector effectiveness of an item. We observe that when constructing the algorithm it is beneficial to select the items with large profits and small weights

and thus, this is equivalent to selecting items corresponding to large profit to weight ratio as proposed in classical KP. We construct a $q \times n$ matrix called the Ratio Matrix, denoted by EF, that provides the best effectiveness for all items with respect to all scenarios. The scalar effectiveness of an item with respect to scenario k is denoted by EF_i^k and given in Definition 2.

Definition 2 *Let i be an item for some $i \in I$. Let p_i^k and w_i be the profit and the weight of the item with respect to scenario k. The effectiveness of the item i with respect to scenario k is $EF_i^k = \frac{p_i^k}{w_i^k}$*

When we use multiple scenarios $z_1, z_2, \ldots, z_q$, the effectiveness of an item is a vector of effectiveness ratios, where each ratio is the effectiveness with respect to one scenario. Let EF_i be the vector effectiveness of an item i for $i \in I$ and it is defined as follows:

$$EF_i = \left[EF_i^1, EF_i^2, \ldots, EF_i^k, \ldots, EF_i^q \right] \tag{14}$$

Therefore, for a given Robust KP we will define an effectiveness matrix, denoted by EF, as given in (15) where k^{th} column of EF gives the effectiveness vector of scenario k for some $k \in K$.

$$EF = \begin{bmatrix} EF_1^1 & EF_1^2 & \ldots & EF_1^k & \ldots & EF_1^q \\ EF_2^1 & EF_2^2 & \ldots & EF_2^k & \ldots & EF_2^q \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ EF_i^1 & EF_i^2 & \ldots & EF_i^k & \ldots & EF_i^q \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ EF_n^1 & EF_n^2 & \ldots & EF_n^k & \ldots & EF_n^q \end{bmatrix} \in \mathcal{R}^{n \times q}. \tag{15}$$

The k^{th} column of the matrix $[EF_1^k, EF_2^k, \ldots, EF_n^k]$ of EF provides the profit to weight ratio for

each scenario. We find the effectiveness vector for item i for $i \in I$ for a scenario $k \in K$ as in equation (14) and arrange them in ascending order. Let $EF^{(k)}$ denote the ordered effectiveness for a scenario $k \in K$ which is given in equation (16).

$$EF^{(k)} = \left[EF^k_{(i^k_1)}, EF^k_{(i^k_2)}, \ldots, EF^k_{(i^k_i)}, \ldots, EF^k_{(i^k_n)} \right]^T \tag{16}$$

where $\left((i^k_1), (i^k_2), \ldots, (i^k_q) \right)$ is a sequence of $(1, 2, \ldots, k, \ldots, q)$. The matrix given in (17) contains all the ordered effectiveness for each scenario $k \in K$. This matrix is denoted by $EF^{(o)}$

$$EF^{(o)} = \begin{bmatrix} EF^1_{(i^1_1)} & EF^2_{(i^2_1)} & \ldots & EF^k_{(i^k_1)} & \ldots & EF^q_{(i^q_1)} \\ EF^1_{(i^1_2)} & EF^2_{(i^2_2)} & \ldots & EF^k_{(i^k_2)} & \ldots & EF^q_{(i^q_2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ EF^1_{(i^1_i)} & EF^2_{(i^2_i)} & \ldots & EF^k_{(i^k_i)} & \ldots & EF^q_{(i^q_i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ EF^1_{(i^1_n)} & EF^2_{(i^2_n)} & \ldots & EF^k_{(i^k_n)} & \ldots & EF^q_{(i^q_n)} \end{bmatrix} \in \mathcal{R}^{n \times q}. \tag{17}$$

In the matrix given in (17), the entry (i^k_i) denotes the i^{th} order candidate for scenario k for some $k \in K$. We use Definition (3) to select items to obtain an approximated solution for the Robust KP.

Definition 3 *Let* $EF(t) = \left[EF^1_{(i^1_t)}, EF^2_{(i^2_t)}, \ldots, EF^k_{(i^k_t)}, \ldots, EF^q_{(i^q_t)} \right]$ *and* $i^{k_1}_t$ *and* $i^{k_2}_t$ *be two candidates of items of* $(t)^{th}$ *order.*

1. *An item $i_t^{k_1}$ is preferred over item $i_t^{k_2}$ in the set of candidates at t^{th} order if*

$$EF^{(t)}_{i_t^{k_1}} \leq EF^{(t)}_{i_t^{k_2}}.$$

It is denoted by

$$EF^{(t)}(i_t^{k_1}) \underset{\min}{\succeq} EF^{(t)}(i_t^{k_2})$$

2. *Let*

$$i = \min_{i \in I} \left[EF^1_{(i_t^1)}, EF^2_{(i_t^2)}, \ldots, EF^k_{(i_t^k)}, \ldots, EF^q_{(i_t^q)} \right]$$

provide the t^{th} order effectiveness. An item i is preferred over other items in the set of candidates at t^{th} order if

$$EF^{(t)}_i \leq EF^{(t)}_{\bar{i}}$$

for all $\bar{i} \in I \setminus \{i\}$

Example 3: Assume that the two items i_1 and i_2 have three profits with respect to three scenarios. Let $p^1_{i_1} = 2, p^2_{i_1} = 5, p^3_{i_1} = 3$ and $p^1_{i_2} = 3, p^2_{i_2} = 2, p^3_{i_2} = 5$ and $w_{i_1} = 5, w_{i_2} = 4$. The effectiveness of i_1 is $EF(i_1) = [2/5, 5/5, 3/5]$ and that of i_2 is $EF(i_1) = [3/4, 2/4, 5/4]$. This implies that the selecting i_1 is preferred than selecting i_2 for order 1 since $EF^{(1)}(i_1) < EF^{(1)}(i_2)$, selecting i_2 is preferred than selecting i_1 for order 2 since $EF^{(2)}(i_2) < EF^{(2)}(i_1)$, and selecting i_1 is preferred than selecting i_2 for order 3 since $EF^{(3)}(i_2) < EF^{(3)}(i_1)$ according to Definition 3. That is we have $EF^{(1)}(i_1) \underset{\min}{\succeq} EF^{(1)}(i_2)$, $EF^{(2)}(i_2) \underset{\min}{\succeq} EF^{(2)}(i_1)$ and $EF^{(3)}(i_1) \underset{\min}{\succeq} EF^{(3)}(i_2)$, respectively.

Now we present our algorithm that approximately solves the Robust KP. The concept of effectiveness of an item defined in Definition 3 leads to the development of our approximation

algorithm.

The i^{th} row of EF in (15) gives the i^{th} best ratio for each scenario $k \in K$ and $\frac{p_i^{k_k^i}}{w_i}$ give the best k^{th} ratio of item i. The preferred item i^* for some $i \in I$ is identified according to Definition 3.

The psudocode of the algorithm is given in Algorithm 2. The symbols $\bar{I}$, $\bar{W}$ and P denote the set of currently selected items to be included in the approximated solution $\bar{x}$, $\bar{W}$ denote the initial weight of the KP. According to the relation given in Definition 3, we select items according to this sorted order until the weight constraint $Ax \leq W$ is satisfied. At the termination of the algorithm, the solution $\bar{x}$ with corresponding weight vale and the profit will be returned.

Algorithm 2 Approximation Approach to Robust KP

Input: n, q, P, A, W
Initialize: EF as a zero matrix in $\mathcal{R}^{n\times q}$, $\bar{x}$ is an zero row vector in $\mathcal{R}^n$
Initialize: $\bar{I} = \emptyset, z(\bar{x}) = 0, \bar{W} = 0$
for $k = 1 \rightarrow q$ **do**
  for $i = 1 \rightarrow n$ **do**
    $EF(i,k) = \dfrac{p_i^k}{w_i}$
  end for
end for
for $k = 1 \rightarrow q$ **do**
  Set k^{th} column of RM to sorted k^{th} column of EF
end for
while $\bar{W} \leq W$ **do**
  Set $t = 1$
  Let $i^* = \min\limits_{i\in I} \left\{ \dfrac{p^{k_i^t}}{w_i} \right\}^T \leftarrow$ Choose smallest ratio
  Update $\bar{I} = \bar{I} \cup \{i^*\} \leftarrow$ Add item to Index set
  Set $\bar{x}(i^*) = 1 \leftarrow$ Add item to solution
  Update $z(\bar{x}) \rightarrow z(\bar{x}) + p_{i^*}^k$
  Update $\bar{W} \rightarrow \bar{W} + w_{i^*}$
  $t \rightarrow t + 1$
end while
if $\bar{W} > W$ **then**
  Let i_l be the index of the last item added $\leftarrow$ If the weight exceeds the allowed select the last item added
  Update $z(\bar{x}) \rightarrow z(\bar{x}) - p_{i_l}^k \leftarrow$ Remove item's profit
  Update $\bar{W} \rightarrow \bar{W} - w_{i_l} \leftarrow$ Remove item's weight
  $\bar{x}(i_l) = 0 \leftarrow$ Remove item from solution
end if
Return: $\bar{x}, \bar{W}$ and $z(\bar{x})$

It is possible that Algorithm 2 terminates with some unused weight. As a result there may be unselected items that could be added to the solution and improve the value. Let $\delta = W - \bar{W}$ be the amount of unused weight after Algorithm 2 terminates. Let $\bar{I}$ be the index set of items with weights less than or equal to δ and $\bar{P}, \bar{A}$ be profits and the weights of the items in $\bar{I}$.

The theoretical observation about Algorithm 1 discussed in the next section.

Algorithm 3 Improvement procedure to Robust KP

Input: $\bar{n}, q, \bar{P}, \bar{A}, \delta$
Initialize: EF as a zero matrix in $\mathcal{R}^{n\times q}$, $\bar{x}$ is an zero row vector in $\mathcal{R}^n$
Initialize: $\bar{I} = \emptyset, z(\bar{x}) = 0, \bar{W} = 0$
while $\delta \geq 0$ **do**
    Call Algorithm 1 with new $\bar{I}$
    Let $\bar{i}$ be the selected items by Algorithm 1
    Add these item to $\bar{I}$
    Update $z(\bar{x}) \rightarrow z(\bar{x}) + p^k_{i^*}$
    Update $\bar{W} \rightarrow \bar{W} + w_{i^*}$
end while
Return: $\bar{x}, \bar{W}$ and $z(\bar{x})$

4.3 Theoretical Performance of the Algorithm 1

One of the main observation about Algorithm 2 is given in Theorem 1. In this theorem we estimate a lower bound for the objective value of the k^{th} scenario for some $k \in K$.

Theorem 3 *Let* $\Big((1),(2),\ldots,(t-1)\Big)$ *be the order of the selected items and* (t) *is the index of the first item that is not selected by Algorithm 2. Let* $\bar{x}$ *be the solution provided by Algorithm 2 for this selection. Then*

$$W\delta_t - p^k_{(i^k_t)} \leq z^k(\bar{x})$$

where $\delta_t = \min\{EF^1_{(i^1_t)}, EF^2_{(i^2_t)}, \ldots, EF^q_{(i^q_t)}\}$.

proof 1 *Let* k *be any given scenario for some* $k \in K$. *For any order* t *we have*

$$\delta_t = \min\{EF^1_{(i^1_t)}, EF^2_{(i^2_t)}, \ldots, EF^k_{(i^k_t)}, \ldots, EF^q_{(i^q_t)}\} \leq EF^k_{(i^k_t)}.$$

By Definition 3, we have
$\left[EF^k_{(i^k_1)} = \dfrac{p^k_{(i^k_1)}}{w_{(i^k_1)}} \geq \delta_t,\ EF^k_{(i^k_2)} = \dfrac{p^k_{(i^k_2)}}{w_{(i^k_2)}} \geq \delta_t, \ldots, EF^k_{(i^k_{t-1})} = \dfrac{p^k_{(i^k_{t-1})}}{w_{(i^k_{t-1})}} \geq \delta_t\right]$. *Therefore,*

$$EF^k_{(i^k_t)} = \frac{p^k_{(i^k_t)}}{w_{(i^k_t)}} \geq \delta_t \textit{ for all } t = 1, 2, \ldots, t \tag{18}$$

By adding inequalities given in (18)

$$p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)} \geq \left(w_{(i^k_1)} + w_{(i^k_2)} + w_{(i^k_{t-1})} + w_{(i^k_t)} \right) \delta_t \tag{19}$$

Rearranging terms given in (19)

$$\delta_t \leq \frac{1}{\left(w_{(i^k_1)} + w_{(i^k_2)} + w_{(i^k_{t-1})} + w_{(i^k_t)} \right)} \left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)} \right) \tag{20}$$

Since t is the index of the first item that is not selected by Algorithm 2, we have

$w_{(i^k_1)} + w_{(i^k_2)} + w_{(i^k_{t-1})} + w_{(i^k_t)} > W$. *Therefore, we obtain*

$\frac{1}{w_{(i^k_1)} + w_{(i^k_2)} + w_{(i^k_{t-1})} + w_{(i^k_t)}} < \frac{1}{W}$. *Then inequality (19) implies the following:*

$$\delta_t \leq \frac{1}{W} \left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)} \right). \tag{21}$$

We know that t is the index of the last item that did not accept by the algorithm. Therefore the objective value of the k^{th} scenario is $z^k(\bar{x}) = p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})}$. Therefore (21) implies that

$$W\delta_t - p^k_{(i^k_t)} \leq z^k(\bar{x}). \tag{22}$$

This completes the proof.

Based on the finding in Theorem 1 we present Corollary 1. We prove that Algorithm 2 yields a

solution $\bar{x}$ such that objective vector $z = [z_1, z_2, \ldots, z_n]$ can be bounded below by a vector which depends on the total weight W, $\delta_t = \min\{EF^1_{(i^1_t)}, EF^2_{(i^2_t)}, \ldots, EF^k_{(i^k_t)}, \ldots, EF^q_{(i^q_t)}\} \leq EF^k_{(i^k_t)}$ and profit of the first item that is not selected by Algorithm 2.

Corollary 1 *The objective vector $z = [z_1, z_2, \ldots, z_n]$ can be bounded.*

proof 2 *From Theorem 1, we obtain $W\delta_t - p^k_{(i^k_t)} \leq z^k(\bar{x})$ for $k = 1, 2, \ldots, p$. This implies that*

$$W\delta_t \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} - \begin{pmatrix} p^1_{(i^1_t)} \\ p^2_{(i^2_t)} \\ \vdots \\ p^q_{(i^q_t)} \end{pmatrix} \leq \begin{pmatrix} z^1(\bar{x}) \\ z^2(\bar{x}) \\ \vdots \\ z^q(\bar{x}) \end{pmatrix}$$

Now we assume that for each $i \in I$, the weight of the item i for some $i \in I$ is less than or equal to ϵW. That is $w_i \leq \epsilon W$ for some $\epsilon \in (0.5, 1)$. We prove that Algorithm 2 yields a solution $\bar{x}$ such that $[z^1(\bar{x}), z^2(\bar{x}), \ldots, z^q(\bar{x})]^T \leq \bar{\epsilon}[z^k(x^{1^*}), z^k(x^{2^*}), \ldots, z^k(x^{q^*})]^T$ where $x^{1^*}, x^{2^*}, \ldots, x^{k^*}$ are the optimal solutions of each individual scenario $k \in K$with $\bar{\epsilon} = \dfrac{1-\epsilon}{\epsilon}$. The error term $\bar{\epsilon}$ provides the maximum tolerance of the algorithm. Thus Algorithm 2 can be interpreted as an ϵ- approximation algorithm According to Definition 4.

Theorem 4 *Let $w_i \leq \epsilon W$ for $i \in I$. Let $\bar{x}$ be the solution provided by Algorithm 1. Let $x^{1^*}, x^{2^*}, \ldots, x^{q^*}$ be the optimal solutions of KP given in model 12. Let $\bar{x}$ be the solution provided by Algorithm 2 and x^* is the optimal solution of the KP. Then Algorithm 2 provides a solution satisfying the*

$$\bar{\epsilon} p^k_{\min} \leq z^k(\bar{x}) \leq z^k(x^*)$$

where $\bar{\epsilon} = \frac{1-\epsilon}{\epsilon}$ and $p^k_{\min}$ is the minimum profit of the k^{th} scenario.

proof 3 *Consider inequality (23) given under the proof of Theorem 3.*

$$\delta_t \leq \frac{1}{W}\left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)}\right). \tag{23}$$

We have $\delta_t = \dfrac{p^k_{(i^k_t)}}{w_{(i^k_t)}}$. Thus from (23) we obtain that

$$p^k_{(i^k_t)} \leq \frac{w_{(i^k_t)}}{W}\left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)}\right). \tag{24}$$

Since we assume that for each $i \in I$, $w_i \leq \epsilon W$, from (24) we obtain that

$$p^k_{(i^k_t)} \leq \epsilon\left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})} + p^k_{(i^k_t)}\right). \tag{25}$$

Therefore we can conclude that

$$\begin{aligned} p^k_{(i^k_t)} - \epsilon p^k_{(i^k_t)} &\leq \epsilon\left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})}\right) \\ \Rightarrow p^k_{(i^k_t)} &\leq \frac{\epsilon}{1-\epsilon}\left(p^k_{(i^k_1)} + p^k_{(i^k_2)} + p^k_{(i^k_{t-1})}\right) \end{aligned} \tag{26}$$

Since $p^k_{(i^k_1)} + p^k_{(i^k_2)} + \cdots + p^k_{(i^k_{t-1})} = z^k(\bar{x})$ combining with inequality (27) we obtain that

$$\Rightarrow p^k_{(i^k_t)} \leq \frac{\epsilon}{1-\epsilon} z^k(\bar{x}) \tag{27}$$

For any feasible solution $\bar{x} \in X$ we have $z^k(\bar{x}) \leq z^k(x^{k^*})$ since x^{k^*} is the optimal value of

that scenario.

$$\frac{1-\epsilon}{\epsilon} p^k_{\min} \leq z^k(\bar{x}) \leq z^k(x^*) \tag{28}$$

The statement we arrive at (28) is true for any $k \in K$. Let $\bar{\epsilon} = \dfrac{1-\epsilon}{\epsilon}$ Thus we obtain

$$\bar{\epsilon} p^k_{\min} \leq z^k(\bar{x}) \leq z^k(x^*) \tag{29}$$

Corollary 2 *Let x^* be the optimal solutions of Min-Max KP given in model 13. Let $\bar{x}$ be the solution provided by proposed algorithm. Let $w_i \leq W$ for $i \in I$. Then*

$$\bar{\epsilon}[p^1_{\min}, p^2_{\min}, \ldots, p^q_{\min}]^T \leq [z^1(\bar{x}), z^2(\bar{x}), \ldots, z^q(\bar{x})]^T \leq [z^1(x^*), z^2(x^*), \ldots, z^q(x^*)]^T$$

proof 4 *According to the construction of the algorithm we have for any given*

$$\frac{1-\epsilon}{\epsilon} p^k_{\min} \leq z^k(\bar{x}) \leq z^k(x^*).$$

Thus we obtain

$$\frac{1-\epsilon}{\epsilon} \begin{pmatrix} p^1_{\min} \\ p^2_{\min} \\ \vdots \\ p^q_{\min} \end{pmatrix} \leq \begin{pmatrix} z^1(\bar{x}) \\ z^2(\bar{x}) \\ \vdots \\ z^q(\bar{x}) \end{pmatrix} \leq \begin{pmatrix} z^1(x^*) \\ z^2(x^*) \\ \vdots \\ z^q(x^*) \end{pmatrix} \tag{30}$$

Inequality (30) confirms that

$$\bar{\epsilon}[p^1_{\min}, p^2_{\min}, \ldots, p^q_{\min}]^T \leq [z^1(\bar{x}), z^2(\bar{x}), \ldots, z^q(\bar{x})]^T \leq [z^1(x^*), z^2(x^*), \ldots, z^q(x^*)]^T.$$

We describe the accuracy of algorithm using a function and we define it as an error function.

Definition 4 *Let x be a vector $\mathcal{R}^q$. A function $t : \mathcal{R} \to \mathcal{R}$ such that*

(1) for all $y \in Y, y \leqq t(y)$, and

(2) for all $y^1, y^2 \in Y$, if $y^1 \leqq y^2$ then $t(y^1) \leqq t(y^2)$,

is called a error function.

Example 4 Let $Y \subset \mathcal{R}^2$ be set of feasible solutions for some Robust KP with two scenarios. Then $Y = \{y^1 = (6,6)^T, y^2 = (5,8)^T, y^3 = (7,9)^T, y^4 = (8,9)^T, y^5 = (9,7)^T, y^6 = (10,10)^T\}$. A function $t : \mathcal{R}^2 \to \mathcal{R}^2$, defined as $t(y) = 2y$, satisfies Definition 3 and therefore is an error function.

Connecting with Theorem 4 and Definition 4 we have that $t(y) = \dfrac{1-\epsilon}{\epsilon} y$. Thus the maximum possible error when using Algorithm 1 is $\dfrac{1-\epsilon}{\epsilon}$. In the next section we will perform computation work to observe how our algorithm performs.

CHAPTER 5

COMPUTATIONAL WORK

The computational results obtained using the proposed algorithm are presented in this section. We also provide an instance of Robust KP with 4 scenarios and 10 items to illustrate the proposed algorithm. The algorithm was implemented using MATLAB interface while all experiments were carried out on a personal computer with a I-3 processor and 2 GB RAM. Matlab 'Linprog' and 'Intprog' functions were used to solve KP problems exactly when required.

5.1 Demonstration of the algorithm using educational example

Table 16 Input data: profits for 4 scenarios and weights

	1	2	3	4	5	6	7	8	9	10
Scenario 1	5	10	2	3	2	2	9	6	6	2
Scenario 2	9	7	4	6	5	1	3	2	2	3
Scenario 3	5	1	10	10	5	5	4	10	4	2
Scenario 4	8	4	3	5	1	2	10	10	6	1
Weight	3	4	9	1	1	2	7	8	7	5

Let z_1, z_2, z_3 and z_4 denote the given 4 scenarios and let $x_1, \ldots, x_{10}$ be decision variables associated with the test problem. Further, assume that the Knapsack capacity is 39. Mathematical

formulation for this test problem with 4 scenarios is presented in (31).

$$\begin{aligned} &\max z(x) = [z_1(x),\ z_2(x),\ z_3(x),\ z_4(x)] \\ &\text{subject to} \\ &3x_1 + 4x_2 + 9x_3 + x_4 + x_5 + 2x_6 + 7x_7 + 8x_8 + 7x_9 + 5x_{10} \leq 39 \end{aligned} \tag{31}$$

where

$$\begin{aligned} z_1 =& 5x_1 + 10x_2 + 2x_3 + 3x_4 + 2x_5 + 2x_6 + 9x_7 + 6x_8 + 6x_9 + 2x_{10} \\ z_2 =& 9x_1 + 7x_2 + 4x_3 + 6x_4 + 5x_5 + x_6 + 3x_7 + 2x_8 + 2x_9 + 3x_{10} \\ z_3 =& 5x_1 + x_2 + 10x_3 + 10x_4 + 5x_5 + 5x_6 + 4x_7 + 10x_8 + 4x_9 + 2x_{10} \\ z_4 =& 8x_1 + 4x_2 + 3x_3 + 5x_4 + x_5 + 2x_6 + 10x_7 + 10x_8 + 6x_9 + x_{10} \end{aligned} \tag{32}$$

In Table 17, the column labels $\frac{p_i^k}{w_i}$ show the effectiveness for $k \in K$ and $i \in I$. For example, as seen in first row of Table 17 item one has weight of 3, and profit of 5, respectively under scenario one. Thus, the corresponding ratio is $\frac{5}{3}$=1.667. Under scenario two item one has a profit of 9, and weight is 3, thus ratio is $\frac{9}{3} = 3$.

Table 17 Ratios for the test problem

i	W	p_i^1	$\frac{p_i^1}{w_i}$	p_i^2	$\frac{p_i^2}{w_i}$	p_i^3	$\frac{p_i^3}{w_i}$	p_i^4	$\frac{p_i^4}{w_i}$
1	3	5.000	1.667	9.000	3.000	5.000	1.667	8.000	2.667
2	4	10.000	2.500	7.000	1.750	1.000	0.250	4.000	1.000
3	9	2.000	0.222	4.000	0.444	10.000	1.111	3.000	0.333
4	1	3.000	3.000	6.000	6.000	10.000	10.000	5.000	5.000
5	1	2.000	2.000	5.000	5.000	5.000	5.000	1.000	1.000
6	2	2.000	1.000	1.000	0.500	5.000	2.500	2.000	1.000
7	7	9.000	1.286	3.000	0.429	4.000	0.571	10.000	1.429
8	8	6.000	0.750	2.000	0.250	10.000	1.250	10.000	1.250
9	7	6.000	0.857	2.000	0.286	4.000	0.571	6.000	0.857
10	5	2.000	0.400	3.000	0.600	2.000	0.400	1.000	0.200

Once we have all the ratio's we need to sort these ratios according to descending order to obtain the the matrix RM.

Table 18 Ratios Sorted From Greatest to Least

$i \in I$	Ratio for $q = 1$	$i \in I$	Ratio for $q = 2$	$i \in I$	Ratio for $q = 3$		Ratio for $q = 4$
4	3.000	4	6.000	4	10.000	4	5.000
2	2.500	5	5.000	5	5.000	1	2.667
5	2.000	1	3.000	6	2.500	7	1.429
1	1.667	2	1.750	1	1.667	8	1.250
7	1.286	10	0.600	8	1.250	2	1.000
6	1.000	6	0.500	3	1.111	5	1.000
9	0.857	3	0.444	7	0.571	6	1.000
8	0.750	7	0.429	9	0.571	9	0.857
10	0.400	9	0.286	10	0.400	3	0.333
3	0.222	8	0.250	2	0.250	10	0.200

Columns of Table 18 with labels $EF(k,:)$ for $k \in K$ gives the best sequence of items with respect to scenario k. For example the sequence $(4, 2, 5, 1, 7, 6, 9, 8, 10, 3)$ is the best ordering of items to select in order to increase the profit of the Scenario 1. As we see in Table 18 the item with highest effectiveness for all scenarios is item 4, therefore, this will be the first item selected to include in solution $\bar{x}$. Thus $\bar{x}_4 = 1, \bar{W} = 1, \bar{I} = \{4\}$. However, with respect to Scenario 1 the second best item is item 2, with respect to Scenarios 2 and 3 the second second best item is item 5, and with respect to Scenario 4 the second best item is item 1. Since $\min_{i \in I}\{2, 5, 5, 2.667\}$

corresponds to item 2, the second best item will be item 2. Continuing this manner, according to Algorithm 1 the best sequence of items to select for Robust KP given in 31 is $(4, 2, 7, 8, 10, 6, 3)$ which is given in Table 19.

Table 19 Order that items are selected

Item	p_i^1	p_i^2	p_i^3	p_i^4	w_i	$\bar{W}$
4	3	6	10	5	1	1
2	10	10	7	1	4	5
7	9	3	4	10	7	12
8	6	2	10	10	8	20
10	2	3	2	1	5	25
6	2	1	5	2	2	27
3	2	4	10	3	9	36

In the final step we will look at the solution from solving the problem exactly and solving the problem using the greedy algorithm. The exact solution for this problem is obtained by solving 13. The optimal solution set for the exact problem is given by $x^* = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (1, 1, 1, 1, 1, 0, 1, 1, 0, 1)$ and for the greedy algorithm $\bar{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (0, 1, 1, 1, 0, 1, 1, 1, 0, 1)$. The values for these solutions in each scenario and the error is given by the table below. We can see here that

Table 20 Optimal profit vs. approximated profit

Scenario	$z(x^*)$	$z(\bar{x})$	(ϵ)
1	39	34	1.147
2	39	26	1.500
3	47	42	1.119
4	42	35	1.200

the greatest error is in scenario 2 and is 1.5. The other three scenarios have relatively low margins of error. Now we run the improvement procedure for this test problem.

5.1.1 Improvement Procedure

Using Algorithm 2, we see that the total weight we used is 36 while the total possible weight is 39. However, we can observe item 1 has a weight of 3 and item 5 has a weight of 1 and these two items have not been selected by Algorithm 2. Thus, we could add either item and still remain under or equal to the total weight allowed by the problem. Thus we use the improvement procedure discussed in Algorithm 3 to this problem with unselected items $\bar{I}$=$\{1, 5\}$.

Table 21 Improvement Procedure

$i \in I$	Ratio for $q = 1$	$i \in I$	Ratio for $q = 2$	$i \in I$	Ratio for $q = 3$		Ratio for $q = 4$
5	2.000	5	5.000	5	5.000	1	2.667
1	1.667	1	3.000	1	1.667	5	1.000

In table 21 we see that the first item we will pick is 5 because the lowest ratio in the first row. When item 5 is added to the solution the new total weight of items will be 37. Thus when the algorithm tries to pick item 1 it will be removed from the solution because that will put the total weight over 39.

5.2 Simulation Study

We analyzed the performance of the algorithm for large robust KP with different number of items and scenarios by randomly generating test problems. The characteristics of these problems are given in Table 22. The first columns indicates the number of the items for $q = 4, q = 10, q = 15, q = 20, q = 25$ and $q = 50$ scenarios. Further, we set the capacity of the Knapsack equal to half of the sum of all item weights of the items. For each block in this table, $z(x^*)$ and $z(\bar{x})$ provide the optimal objective function value for the Min-max problem given in

model 13, the objective function value of the problem yield by the algorithm, respectively. The ϵ denotes the error ratio.

Table 22 Simulation Results for Robust KP

n	$q=4$			$q=10$			$q=15$		
	$z(x^*)$	$z(\bar{x})$	ϵ	$z(x^*)$	$z(\bar{x})$	ϵ	$z(x^*)$	$z(\bar{x})$	ϵ
10	35	27	1.296	31	22	1.409	25	10	2.500
25	87	78	1.115	83	77	1.078	84	59	1.423
50	173	150	1.167	175	131	1.336	186	148	1.257
100	390	324	1.204	378	286	1.322	355	269	1.320
n	$q=20$			$q=25$			$q=50$		
	$z(x^*)$	$z(\bar{x})$	ϵ	$z(x^*)$	$z(\bar{x})$	ϵ	$z(x^*)$	$z(\bar{x})$	ϵ
10	25	23	1.087	23	11	2.091	23	17	1.352
25	73	47	1.553	78	52	1.500	71	48	1.479
50	165	99	1.667	169	122	1.385	169	106	1.594
100	380	313	1.214	352	272	1.294	341	267	1.277

We observe from this table that the error is high when there are many scenarios with few variables. But always the error ratio is smaller than 2 except for the two cases with $(n, q) = (10, 15)$ and $(n, q) = (10, 25)$. We also observe that the computational time is reasonable for the proposed algorithm on the above mention computer and it is always less than 2 seconds. Overall, this analysis indicates that the proposed algorithm performs well.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Overview

In classical linear programming problems, the most important assumption is that all input data is known. However, in real world problems the data is not always certain. Some possible causes for data uncertainty include not yet knowing data, and data forecasts are never certain. Measuring errors when collecting data can also cause data uncertainty. These are just two of many possible causes surrounding data uncertainty. Robust optimization is a branch of optimization that deals with uncertainty in optimization problems. Under robust optimization the objective and constraint functions are assumed to belong to uncertainty sets. In general, the objective of robust optimization is to find a solution that is near optimal and feasible under any scenario. This means under the worst case scenario, a solution must be feasible and near the optimal solution. The knapsack problem is a well-known combinatorial optimization problem. In the knapsack problem, we are given a set of items, a collection called the knapsack, with a fixed capacity. Each item has a certain weight value and a profit value. The goal of the knapsack problem is to optimize the profit of the knapsack while staying under a given weight. Because the knapsack problem is computationally expensive, and sometimes not solvable using a polynomially solvable algorithm, approximation methods have been proposed. This research focuses on developing an approximation algorithm for the knapsack problem with finite number of scenarios. Our method

is developed to optimize the objective function value of the worst case scenario. We validate the algorithm theoretically, in which we mathematically derive the maximum possible error for the proposed algorithm. We also validate the algorithm computationally.

6.2 Summary

In Chapter 1, we provide a motivation to our research. In Chapter 2, we discuss the difference between classical and robust optimization methods. We first provide an elementary example, which can be solved geometrically, to show that classical optimization methods will not work when there is data uncertainty. We also analyze the behavior of the optimal solution for a well-cited problem in the literature under many possible perturbations. We observe that the optimal solution with the nominal data becomes infeasible when we slightly perturb the input data. In Chapter 3, we introduce the knapsack problem, which is a well-known combinatorial optimization problem. The KP has many real world applications, and we provide the capital budgeting problem as an illustrative example. The combinatorial structure of the KP makes solving the exact solution for this problem hard. We continue discussion by providing the approximation algorithms proposed in the literature to solve the KP. In Chapter 4, we further discuss the uncertainty variant of the KP and formulate the problem for this variant. After introducing the new problem, we provide our approximation algorithm. With the algorithm, we also derive theoretical performance of the proposed algorithm. In Chapter 5, we perform computational work to analyze the performance of our algorithm. We first provide an educational example with four scenarios and ten items to demonstrate the algorithm. We then conduct an advanced computational work to analyze and see how it performs under circumstances with

various scenarios and items. We compare the approximated value of the objective function with the exact optimal value.

6.3 Future Works

One of the major drawbacks of the proposed method is its inability to analyze the profit of the item individually. At the moment the algorithm yields a feasible solution to the robust KP by analyzing the profit to weight ratio. For example, recall the case with two items, that is $i = 1, 2$. Let $W \geq 2$ and $p_1 = 2, p_2 = W, w_1 = 1$ and $w_2 = W$. Then the algorithm selects first item even though its profit is smaller than that of the second item. Thus, we need to include a preference relation to the proposed algorithm to check the profit of the item in addition to the ratio. We also need to compare the performance of our algorithm with the existing algorithm in the literature. Collaborating with my advisor, Dr. Weerasena, at least one of high quality advanced papers stemming from this research will be submitted to top peer-reviewed Operations Research/Mathematical journals. In addition, a paper will be submitted to the INFORMS (Institute for Operations Research and the Management Sciences) conference on Business Analytic and Operations Research, which is held annually with participants and presenters representing academic and non-academic area, to be held at the Washington State Convention Center, Washington from October 20 to October 23, 2019. Furthermore, rather than just analyzing the finite number of scenarios, we can propose uncertainty sets for weights of the items and extend the proposed algorithm to find robust solutions. Finally, not only data generated form the some simulated test scenarios, but some real world problem data can also be used to validate the proposed algorithm.

REFERENCES

[1] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. Complexity and approximation: Combinatorial optimization problems and their approximability properties. Springer Science & Business Media, 2012.

[2] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. Linear programming and network flows. John Wiley & Sons, 2011.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. Robust optimization, volume 28. Princeton University Press, 2009.

[4] Aharon Ben-Tal and Arkadi Nemirovski. Robust convex optimization. Mathematics of operations research, 23(4):769–805, 1998.

[5] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. Mathematical programming, 88(3):411–424, 2000.

[6] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. SIAM review, 53(3):464–501, 2011.

[7] Dimitris Bertsimas and Melvyn Sim. The price of robustness. Operations research, 52(1):35–53, 2004.

[8] Ashok K Chandra, Daniel S. Hirschberg, and Chak-Kuen Wong. Approximate algorithms for some generalized knapsack problems. Theoretical Computer Science, 3(3):293–304, 1976.

[9] Harold Greenberg and Robert L Hegerich. A branch search algorithm for the knapsack problem. Management Science, 16(5):327–332, 1970.

[10] Juraj Hromkovic. Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics. Springer Science & Business Media, 2013.

[11] Oscar H Ibarra and Chul E Kim. Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM (JACM), 22(4):463–468, 1975.

[12] Richard M Karp. Reducibility among combinatorial problems. In Complexity of computer computations, pages 85–103. Springer, 1972.

[13] Adam Kasperski and Pawel Zielinski. An approximation algorithm for interval data minmax regret combinatorial optimization problems. Inf. Process. Lett., 97(5):177–180, 2006.

[14] Peter J Kolesar. A branch and bound algorithm for the knapsack problem. Management science, 13(9):723–735, 1967.

[15] Silvano Martello and Paolo Toth. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. European Journal of Operational Research, 1(3):169–175, 1977.

[16] George L Nemhauser and Laurence A Wolsey. Integer programming and combinatorial optimization. Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin, 20:8–12, 1988.

[17] R Gary Parker and Ronald L Rardin. Discrete optimization. Elsevier, 2014.

[18] Allen L Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations research, 21(5):1154–1157, 1973.

VITA

Blake Smith was born in Fullerton, CA, to the parents of Jim and Julie Smith. He attended Acacia Elementary School, Ladera Vista Junior High School, and Troy High School in Fullerton, CA. After graduation he attended Lee University in Cleveland, TN where he studied mathematics with an emphasis in actuarial sciences. He completed the Bachelors of Science program in May 2017. Following graduation Blake accepted a teaching assistatship at The University of Tennessee at Chattanooga in the in the Mathematics department. Blake will graduate with a Masters of Science degree in Mathematics in May 2019.