Faculty of Natural and Mathematical Sciences
Department of Informatics

King's College London
Strand Campus, London,
United Kingdom

7CCSMPRJ

Individual Project Submission 2020/21

Name: Diego Toledano

Student Number: 20113058

Degree Programme: MSc Advance Computing

Project Title: Bi-level and Decoupling Approaches for the Robust Knapsack Problem

Supervisor: Dimitrios Letsios

Word Count: 14565

[✓] I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: [illegible]

Date: August 29, 2021

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ Individual Project

Bi-level and Decoupling Approaches for the Robust Knapsack Problem

Name: **Diego Toledano**
Student Number: 20113058
Course: MSc Advance Computing

Supervisor: Dimitrios Letsios

This dissertation is submitted for the degree of MSc in MSc Advance Computing.

Abstract

Mathematical problems are amazing as they allow us to model on paper some real life interrogations. Furthermore, Mathematical optimization is better as it permit us to use programming to solve our mathematical models. The biggest issue face in optimization is uncertainty, as it is hard to model the unknown or imprecise data. Dealing with uncertainty is a must and involves robust approaches. A robust approach to solving mathematical optimization problems have existed for a long time already and have been applied to a good deal of optimization problem. One of them is the famous Knapsack Problem.

In this paper we perform an empirical comparison of multiple factors, models and approach of the Knapsack Problem. Mainly by opposing the bi-level approach and the decoupling approach. We present two models for this two approach that stem uncertainty using robust counterparts. Therefore, this paper present a model of robustness based on Dimitris Bertsimas and Melvyn Sim's idea. As well as a study of some factors that compose that said robustness. Following our results , the two approaches have their opportunities and obstacles. Moreover, our results allow us to figured out why the 0-1 Knapsack Problem is the exclusive base when we talk about variants of the problem. Studying the Knapsack Problem and its robust counterparts gives us a better understand of how it works under uncertainty and ease people to increase the number of applications for it. However, at the end, one question remains: Considering all the works and publications on the subject, does it still makes sense to study the Knapsack Problem, or have we mastered everything about it?

Acknowledgements

The completion of this thesis would not have been possible without the help of many persons. Thus, I would like to express my sincerest gratitude and warm appreciation to the followings persons :

Dr. Dimitrios Letsios, theisis supervisor, for his expertise and help regarding the subject.

Dr. Khuong Nguyen, for his expertise and structural advices regarding thesis and reports.

Mr. Enric Pardo for his thought, knowledge and introduction regarding mathematical optimizations.

Mr. Alain Toledano, for lending his computer when needed, his proofreading and thought about the topic.

Mrs. Sylvia Toledano, for giving out knowledge and validation regarding law issues.

My Friends for the encouragement and enthusiasm about the project.

My Classmates for all the discussions and debates regarding structures, time management and resources used for this project.

Mr. Ruben Toledano, for lending his computer when needed and being an amazing brother, giving out fun and distractions.

Nomenclature

0-1KP	0-1 Knapsack Problem
Γ	Percentage of value that are subject to robustness
ξ	Percentage of uncertainty the model is confronted with
Ψ	Adjustable parameters controlling uncertainty sets
Ω	Adjustable parameters controlling uncertainty sets
B-LRKP	Bi-Level Robust Knapsack Problem
DB&BRKP	Decoupling Branch & Bound Knapsack Problem
F	Set of feasible solutions
$f()$	Objective function
G0-1KP	Greedy 0-1 Knapsack Problem
IKP	Integer Knapsack Problem
J	Variables subject to uncertainty
K	0-1 Programming Set
K_I	Integer Programming Set
K_M	Mixed 0-1 Programming Set
K_{MI}	Mixed Integer Programming Set
k	3-CNF Targeted sum
M0-1KP	Mixed 0-1 Knapsack Problem
MIKP	Mixed Integer Knapsack Problem
m	Number of item
n	Number of item
RIKP	Robust Integer Knapsack Problem
RKP	Robust Knapsack Problem
RM0-1KP	Robust Mixed 0-1 Knapsack Problem
RMIKP	Robust Mixed Integer Knapsack Problem
$\mathbb{R}$	Real numbers
S	A multiset
s_i	Clause's value
s'_i	Clause's value
T	Targeted sum
U	Uncertainty set
U_2	Ellipsoidal Uncertainty Set
U_∞	Box Uncertainty Set
$U_{2\infty}$	Box+Ellipsoidal" Uncertainty Set
v_i	Value of the i^{th} item
v_j	Value of the j^{th} item
W	Overall weight we do not have to exceed
w	Set of weights
w_i	Weight of the i^{th} item
w_j	Weight of the j^{th} item
$\bar{w}_i$	Robust Weight of the i^{th} item

X	Discrete set of candidates
x	Set of items
x_i	i^{th} item
x_j	j^{th} item
y_i	Positive occurrence
$\bar{y}_i$	Negative occurrence
$\mathbb{Z}$	Integer Number

Contents

List of Figures

List of Tables

1 Introduction

To begin, the Introduction sets up here. It focuses on why this paper has been written. Consequently, we have the motivations, the applications and the impact of the problem.

1.1 Motivations

Dealing with uncertainty forces us to reinvent our way of thinking and our way of setting problems. Adding uncertainty into the mix increases the fitting aspect of our models to real-life examples. As modelling, it involves randomness, this means that solving approaches have to deal with randomness and become more complex and interesting.

Moreover, the Knapsack Problem is an easy subject to teach to students and to discuss. Despite being a simple problem, everyone can relate to it as it has a wide range of more or less complex applications. As we are dealing with uncertainty and thus randomness, we are performing our experiment in a Monte Carlo Simulation style by running our methods many times on many instances. As we want better and faster results. Two options arise, either have faster hardware or decrease the time complexity of our computations. As this problem applies to many situations, the solving approach will be used by non-expert people on commodity hardware. Consequently, this project will raise interest for the Knapsack Problem and improve either hardware's utilization efficiency and/or time complexity optimizations.

On a more personal tone, solving problems and puzzles are one of the biggest passions I have. In addition to that, since I was young, I have always been in love with computers and mathematical related subjects. Hence, taking part in this journey where I can help to improve our knowledge about the Knapsack Problem is something that came out very naturally in my mind when I have had to choose a project. This is a good opportunity for me to help the community and learn how to deal with a big project like this, which will improve my skills at problem and optimization solving. I think that this will help me in my career.

1.2 Applications

One of the main motivations for this project is the wide range of applications the Knapsack Problem fits into, going from usual tasks to more complex applications. As there are many applications, here are some of them :

The main application for this problem is the application we use for describing the problem and where its name came from. It consists in figuring out what is the best way of optimizing the space of a knapsack *(e.g. a backpack, a data center, a warehouse, a* *plane and such more)* with respect to its size and the importance of the items we fill it with. One way of using the Knapsack Problem is to apply it to cryptography. This has been done by Ralph Merkle and Martin Hellman to what is called *the Merkle–Hellman knapsack cryptosystem* [1] where the objective function of the Knapsack Problem is used to create their ciphertext and solving a version of the problem is required for their description.

Another subject the Knapsack Problem can be applied to is Finance. In fact, the problem can be used with the diversification aspect of the Modern Portfolio Theory of Harry Markowitz, where uncertainty is used to model risk and fluctuation of stocks. Fereshteh Vaezi, Seyed Jafar Sadjadi and Ahmad Makui combine that idea with the Knapsack Problem in their article *A portfolio selection model based on the knapsack problem under uncertainty* [2] where they try to maximize the return on investment of stocks without having their overall prices overcome our budget.

In addition, our Knapsack Problem is used by industries as it can be applied to the raw cutting stock problem - trying to figure out the best way of cutting raw materials without loss -. This can be seen in Pitjaya Tangtatswas thesis called, *Algorithm for the cutting stock problem with multiple raw and limited number of cutting knives* [3]. Here, the Knapsack Problem is modelled as maximizing the profit of each cut, while the sum of each width of all the cuts should not be bigger than the width of the raw material. It can also be used in electromagnetic science by doing spectrum allocation. This use of the Knapsack Problem works using multidimensionality, I would not go in details about how the problem is used because multidimensionality is not something this project is focusing on. However, it can be discovered in the paper *There is No EPTAS for Two-dimensional Knapsack* [4] written by Ariel Kulik and Hadas Shachnai and its application can be explored thanks to Yang Song, Chi Zhang and Yuguang Fang in *Multiple multidimensional knapsack problem and its applications in cognitive radio networks* [5].

As you can see the Knapsack Problem has many applications, in fact, a study made by the Stony Brook University *Algorithm Repository in 1999* [6] showed that the Knapsack Problem came out nineteenth in favouritism and third in need of use. Finally, we can sum them up in categories : allocation/mapping, decision-making/management and general problem-solving.

1.3 Impact of the problem

The impact of the Knapsack Problem depends on its applications and use of it later. If we are using the Knapsack Problem regarding an allocation point of view, it will increase space efficiency by having a better organization, this problem can help reduce storage overflow. In doing so, it can allow us to have less storage space and save money on that. Using the knapsack problem with decision-making style issues allows us to save time on taking those complex decisions as the solver *(e.i. The software, using an algorithm to solve issues modelled after the Knapsack Problem)* does everything for you at the speed of computation. Overall, I would say that the main impact and/or reward for solving the knapsack problem is generally to save time and money.

However, it is very dependent on applications. Who knows, maybe NASA will use this problem to know how much fuel to put on their rocket ship, hence, the impact could be related to space exploration. Who knows, this problem will be used by Google or Amazon to manage their data centres, use less storage, hence, the impact could be more ecological. Who knows, IBM will use this problem for bit manipulation in their next quantum computer, with a computer that can help to resolve string theory.

> The Robust Knapsack Problem can have a massive impact on mankind, we just need to find how.

2 Project Specification

Afterwards, lie here is the Project Specification. It is about definitions, of the Knapsack Problem, of uncertainty, of the research questions and research objectives of this project and the structure of this paper.

2.1 The Knapsack Problem

Imagine, that you just became a thief and that for your first mission, you have decided to burgle a museum. As this is the first time you are burgling, you have made moronic decisions, breaking into the Louvre Museum in Paris which has a massive alarm system and only taking a backpack with you. Beside the security aspect of this mission, your biggest concern is how you are going to tuck all your belongings inside your backpack, as your goal is to get away with the most valuable artworks without either overloading your bag nor breaking it. How do you choose amongst all those beautiful pieces of art and maximize your loot? This is where the Knapsack Problem comes into play. In its original form, the Knapsack Problem is an optimization problem that consists in maximizing the overall profit of objects we fill up a receptacle with, subject to the capacity of the latter. This problem can be easily translated to a mathematical problem :

$$\begin{aligned} &maximize \quad \sum_{i=0}^{n} v_i x_i \\ &subject\,to \quad \sum_{i=0}^{n} w_i x_i \leq W \end{aligned} \tag{1}$$

This model embodies maximizing values v_i of our n items x_i without having their weight w_i overcoming the overall weight W of our receptacle. Sets, in our problem, govern laws about item quantity. The most known one is the 0-1 Knapsack set. This set will be used for all the different Knapsack Problem variants in this paper. The latter comes as follows :

$$K = \{x \in \mathbb{Z} : 0 \leq x \leq 1\} = \{0, 1\} \tag{2}$$

Noted K , this set can be resumed in three words : *"All or Nothing"* items cannot be divided. As sets bespeak quantity of items, K means either we load the item one time, or we do not load it. Consequently, the acknowledged 0-1 Knapsack Problem - often said to be the classical one - is in such a way as we include K into the mix :

$$\begin{aligned} & \textit{maximize} && \sum_{i=0}^{n} v_i x_i \\ & \textit{subject to} && \sum_{i=0}^{n} w_i x_i \leq W \\ & && x_i \in \left\{ x' \in \mathbb{Z} : 0 \leq x' \leq 1 \right\} \end{aligned} \tag{3}$$

2.2 Uncertainty

Some questions have life changing answers while others do not, here is one of them : A heavy surgical intervention would not be done until the blood pressure of a patient - let us call him Harry - increases. A first medical analysis has been done and output a value of 2.86. The following day, another laboratory performed the same medical analysis, but output a value of 2.9 instead. The tricky question is then :

Should we perform the medical operation on Harry?

On the one hand, some say we must perform the operation since the value did indeed change, as 2.9 is bigger than 2.86. While on the other hand, some say we should not do it as this is due to uncertainty. The second laboratory could have reported a value of 2.86 and rounded up their result. The truth is that scientists must be as honest as possible when reporting their results on paper. Hence, they should report as precise as possible the uncertainty of their data. Knowing that, reporting a value of 2.9 should not be questioned. Harry will have his surgery. This doubt is caused by uncertainty. In a general manner, uncertainty of a data is caused by the precision of measuring tools. According to David M. Wheeler in his paper *Uncertainty - What is it ?* [7] he comes up with the conclusion that uncertainty is :

> A common theme in the definitions is that uncertainty is linked to lack of information, or unknowns. [...] ignorance and uncertainty are neither negative nor positive; rather, they are neutral terms describing a state of being.

His overall outcome is that uncertainty is something we should live with. Consequently, we try to overcome uncertainty by taking it into account while modelling our problems. According to our Knapsack Problem, dealing with uncertainty and adding it to our problem allows us to have a better match with real world examples, one way of doing it we roughly take the worst possible variables into account. This will be discussed later in this paper.

2.3 Research questions and research objectives

While doing research about our topics, some major questions arose :

- How to solve the Knapsack Problem ?
- What is the fastest way of solving the Knapsack Problem ?
- What is the simplest way of solving the Knapsack Problem ?
- What is uncertainty ?
- How to deal with uncertainty ?
- Is uncertainty a problem ?
- Is the 0-1 Knapsack Problem the best way to study the problem ?
- Does it still make sense to study the problem?

Consequently, by the end of this project, I would like to :

- Have an answer for all the interrogation above.
- Learn more about mathematical optimization.
- Lean about robustness.
- Have an efficiency way of solving the Knapsack Problem.
- Have a good model of the uncertainty under the Knapsack Problem.
- Learn more skills on how to deal with big projects.
- Improve my skills and knowledge about solving problems.

By the end of this project, I would like to know more about the implication and the consequences of dealing with and solving the Knapsack Problem. Since the Knapsack Problem have a lot of applications, some of them can be benefic, while others can be threatening. Hence, I would like to learn about how to ensure applications do not lead to issues.

2.4 Structure of this report

This project is divided into eleven parts or chapters that are :

- Firstly, there is the Introduction part where I explain the motivations for these projects as well as the applications and the different impact and usage of the Knapsack Problem.

- Then, the Project Specification where we will find descriptions about the two major component of this project, the Knapsack Problem in itself and Uncertainty. We will also read about the questions and objectives that I faced while doing this project as well as, we will find there this structure of this report.
- The third chapter looks at the project schedule, where we will understand how things were planned with the use of a timeline. In this chapter, we will also find the key issues I have faced during my project and research.
- The fourth chapter is about the big background of the Knapsack Problem. It looks at the Computational complexity of the Knapsack Problem, explaining how it is called an NP-complete problem. Moreover, it is about the Discrete and Robust optimization, which are two fields of mathematics the Knapsack Problem fell into.
- The fifth chapter is the literature review of this project. Here we will learn about what has been done regarding the Knapsack Problem, Robustness and other topics related to this project.
- The sixth chapter goes with a mathematical description of all the models used within this project. We will have, the Knapsack Problem, the Fundamentals Knapsack Problems that use different sets restrictions, the Robust Knapsack Problem with how uncertainty was modelled for this project, the Bi-level Knapsack Problem and the Decoupling Knapsack Problem.
- The seventh chapter explains the technical information of this project. Going from how the data were collected, how the data were used and how I am communicating the results.
- The eighth chapter is all about the analysis and the evaluation of the computed results. Therefore, we will find multiple comparison about results computed before based on graphs.
- The ninth chapter is about Software engineering and especially how I have used a Revision Control System and how I have done tests. In addition to that, over there, we will find a UML diagram that explains how I have been working.
- The tenth chapter looks at the professional issues we could face using the Knapsack Problem, we will find four types of issues, legal issues, social issues, ethical issues and robust issues.
- Finally, the last chapter is the conclusion of this project, where we will find a summary of my work and my futures intentions if I had more time.

3 Project Schedule

Following is the Project Schedule. It is about time management regarding this paper. Here we have the timeline and the every issues faced.

3.1 Timeline

The actual timeline for this project is as follows

I have spent the first few months of this project doing mainly research, as when I was selecting a project I did not know anything about the Knapsack Problem nor robustness. The next thing I have been doing was playing with the python based modelling language Pyomo, since this was the first time using it for me, I had to compute some proofs of concepts. The first one was a basic optimization problem with known variable. The goal here was to make sure Pyomo and the solver CPLEX where installed. Then, the second proof of concepts was a basic optimization problem with unknown variable. The goal was here to have a better understanding of how things were supposed to work. The third proof of concept that I needed to perform was a bi-level style optimization problem. AS you will read in the lines below, this was a bit tricky at first. The fourth and final proof of concept was to perform the greedy algorithm with the Knapsack Problem. Now that I have done my research and that I know how to compute the experiments, I have stated to model and compute the 0-1 Knapsack Problem as well as the Robust Counterpart. Following that, the Decoupling Branch & Bound Robust Knapsacks came into play, followed by the Bi-level Robust Knapsack Problem. To finish with the modelling part, I have made up the Fundamentals Knapsack Problem with the robust counterpart in Pyomo. Once everything has been modelled, I have been collecting results with the instances. After that, some mean values have been computed. To finish, graphs has been made with those results. This can be seen on the Gantt chart bellow.

3.2 Gantt Chart

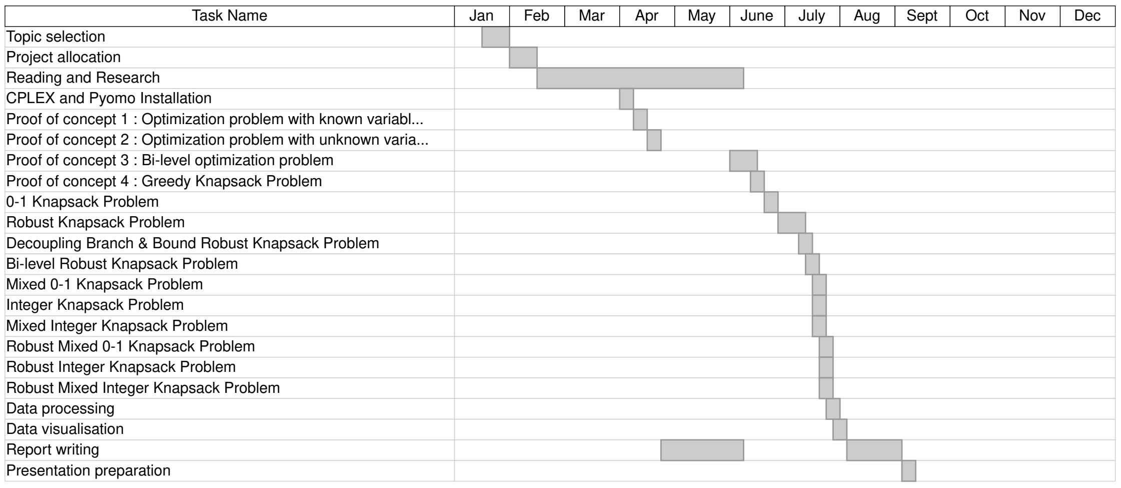

Figure 1: Gantt Chart.

3.3 Issues and Risks

While I was working on this project, I have only faced some issues and risks.

The first issue I have face was learning on how to use the python based modelling language Pyomo. It was not really an issue, but since it was the first time I was using it, everything was discovery. Consequently, I had to spend some time learning on how to use it. The second issue I have faced occurred while I was modelling bi-level base optimization formulas into Pyomo. Here, the issue was that Pyomo do not know how to properly deal with bi-level model. Hence, there was some few tricks and transformation that needed to be applied to transforms the models. The last issue I have faced is one of the biggest of the project was randomness. Let me explain. In fact, since I am using a Gamma factor that decide the percentage of value that are subject to robustness, this process is random. The other time randomness has come into play was when doing fundamentals models, as values can either be from one set or from another one. As an example, values can either be from the Positive Integer set or from the Non-Negative Real set for the Mixed Integer Knapsack Problem. The process of defining which variable belong to which specific set is random. Consequently, a Monte-Carlo simulation style approach has been done in order to stem randomness.

4 Background

The Background explains contexts and insights. Therefore, arrive discrete and robust optimization and time complexity of the Knapsack Problem.

4.1 Discrete Optimization

Discrete Optimization is a subfield of Mathematical Optimization and can be divided into two sub-parts : Combinatorial Optimization and Integer Programming.

4.1.1 Combinatorial Optimization

Combinatorial Optimization is a subfield of Mathematical Optimization and a subset of Discrete Optimization. This filed is about problems that consists of finding the optimal item from a finite set of items. It can be modelled as follows :

$$maximize/minimize_{x \subseteq X} \{f(x) : x \in F\} \tag{4}$$

This model stand for finding the set that maximize or minimize the objective function $f()$ over the subset x of the discrete set of candidates, X such as x is a feasible solution by belonging to F. The number of options that solve this problem is too large to be brute forced, hence solving techniques occur, the two most universally applicable ones are the Branch & Bound algorithm and the Branch & Cut algorithms which are two Decoupling approaches. Moreover, this problem has many applications such as, the Travelling Salesman, the Minimum Spanning Tree and the Knapsack Problem. The Knapsack Problem relates to a Combinatorial Optimization Problem as $f()$ is the sum of all values of items and the constraint of the total weight being less than our capacity is denoted by F.

4.1.2 Integer Programming

Integer programming is a subfield of Mathematical Optimization and a subset of Discrete Optimization. It consists of having all variables in an optimization problem to be integer variables. In terms of model, since integer programming only apply as a constraint, it is modelled a set :

$$K_I = \{x \in \mathbb{Z}\} \tag{5}$$

We also have variants, the two most used one are, Mixed-Integer Programming which is as follows :

$$K_{MI} = \left\{x \in \mathbb{Z}^+, y \in \mathbb{R}^+\right\} \tag{6}$$

And 0-1 Programming, which is also as follows :

$$K = \{x \in \mathbb{Z} : 0 \leq x \leq 1\} = \{0, 1\} \tag{7}$$

More variants will be explored later in this paper. Integer programming problems have many applications such as, Capital Budgeting Extension, Blending Problem, Scheduling Problem and the Knapsack Problem. The Knapsack Problem relate to Integer Programming since the most common Knapsack Problem is the 0-1 Knapsack Problem which does use the 0-1 set K constraint stated above. However, we will see later that the Knapsack Problem can be applied to many more different variants of K_I.

4.2 Robust Optimization

Robust Optimization is a subfield of Mathematical Optimization. However, compared to other types of optimization, Robust Optimization is depicted by having to deal with uncertainty. As said above, uncertainty is when data is either imprecise, missing or incorrect. In fact, it can be divided into two parts, microscopic uncertainty and macroscopic uncertainty. Microscopic uncertainty stands for numerical and measurement errors, while macroscopic uncertainty is about forecast errors and environment changes and disturbances.

Whereas in mathematics, where uncertainty is represented by the variability of the values used in problems. That variability is figured with sets, where the values lie within a given uncertainty set. This given uncertainty set have some variant, the most common used one are :

Box Uncertainty Set :

$$U_\infty = \{\xi : \|\xi\|_\infty \leq \Psi\} = \{\xi : |\xi_j| \leq \Psi, \forall j \in J_i\} \tag{8}$$

Ellipsoidal Uncertainty Set :

$$U_2 = \{\xi : \|\xi\|_2 \leq \Omega\} = \left\{\xi : \sum_{j \in J_i}^{n} \xi_j^2 \leq \Omega\right\} \tag{9}$$

"Box+Ellipsoidal" Uncertainty Set :

$$U_{2\infty} = \left\{ \xi : \sum_{j \in J_i}^{n} \xi_j^2 \leq \Omega, |\xi_j| \leq \Psi, \forall j \in J \right\} \tag{10}$$

Where we have Ψ and Ω being adjustable parameters controlling the uncertainty sets, J being the vector of our variables subjects to uncertainty and ξ being our perturbation factor, which we will talk more about later in this paper. According to our Knapsack Problem, the uncertainty model we will be using is inspired from the model of Dimitris Bertsimas and Melvyn Sim in their paper, *The Price of Robustness* [8] where the value that lies within our uncertainty set is only the weight of our items while its price remains fixed. The uncertainty set is modelled as then :

$$U = [w_i - \bar{w}_i, w_i + \bar{w}_i] \tag{11}$$

With w_i being the weight of the i^{th} items and $\bar{w}_i$ is the robust weight of that said item that can be computed as follows :

$$\bar{w}_i = w_i + \xi w_i \tag{12}$$

Here, the value represented by ξ will be called the perturbation factor, which denotes the percentage of uncertainty our model is confronted with. Finally, we have another value Γ called Gamma which is the percentage of values that can attain a value in U, the others remain at w_i.

Another way to model uncertainty is through objective functions. The objective function can be modified to unsure that the complexity level of uncertainty is handle. Doing so, we will end up with the same goal but different way of achieving it. This given uncertainty objective function have some variant, the most common used one are :

Absolute Robustness :

$$maximize\, minimize \sum_{i=0}^{n} v_i x_i \tag{13}$$

Maximum Regret :

$$minimize\, maximize\{(\sum_{i=0}^{n} v_i x_i)^* - (\sum_{i=0}^{n} v_i x_i)\} \tag{14}$$

Maximum Relative Regret :

$$minimize\, maximize\{\frac{(\sum_{i=0}^{n} v_i x_i)^* - (\sum_{i=0}^{n} v_i x_i)}{(\sum_{i=0}^{n} v_i x_i)^*}\} \tag{15}$$

Here, $(\sum_{i=0}^{n} v_i x_i)^*$ denote the optimal objective function amongst every given objective function. In other words, it is the objective function that output the maximum value amongst every scenario. As we will see later, we will always assume that objective functions are fixed *(i.e. not subject to uncertainty)* while the constrained function is not fixed. That is why, for our Knapsack Problem, only the weight value is subject to uncertainty sets.

4.3 Computational Complexity

As its name implies, computational complexity is a measure that quantifies resource usage of a problem. A problem can be NP-hard or NP-complete, which stand for non-deterministic polynomial-time complete. This means that the problem can be solved in polynomial time using a nondeterministic Turing machine - a machine that can do multiple actions at once -

Theorem : Knapsack is NP-complete.

Proof : In order to demonstrate that the Knapsack Problem is NP-complete, we have to use a simpler reduction of the idea by using other problems.

First, let us prove that the Subset Sum Problem can be modelled as a Knapsack Problem. The Subset Sum Problem is given a multiset S and a targeted sum T, find a subset of S that sum to T. By simply saying that values $v_i$and weights w_i of items x_i *(e.i. $v_i = w_i$)* we can remodel the problem as follows :

$$\begin{aligned} &maximize \quad && \sum_{i=0}^{n} w_i x_i \\ &subject\, to \quad && \sum_{i=0}^{n} w_i x_i \leq W \end{aligned} \tag{16}$$

Since we want to maximize the weights of items without having them to be larger than the weight W of our receptacle, the problem then seek for the set of items that are equal to W. In fact, the subset of S become the set of all w_i while the targeted sum T become our capacity W.

Second of all, let us prove that the Subset Sum Problem can be modelled as a 3-SAT

Problem. A SAT formulation is a decision formula that takes multiple booleans variables $x_1, ..., x_n$. Variables are grouped as clauses *(e.i.* $x_1 \vee x_2 \vee x_3$ *)* while operators betweens variables and clauses are logical OR $\vee$ and logical AND $\wedge$. The formula aims at checking if there is a set of x_i such that the whole formula is satisfied. The 3-CNF-SAT is then a SAT formulation where each clause consists of three variables.

Given a 3-CNF formulation with variables $x_1, ..., x_n$ and clauses $c_1, ..., c_n$. Let us give each variable a positive occurrence y_i and a negative occurrence $\bar{y_i}$ and give each clause has two values, s_i and s'_i. We then define all of them using a base 10 representation. On top of that, we add a value k which is our desired total. This special formulation works better with an example :

Given the following 3-CNF formulation :

$$(x_1 \vee \bar{x_2} \vee x_2) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \tag{17}$$

We then create a base 10 tables with columns for the variables, columns for the clauses and add the numbers described above as rows for the table which is :

Table 1: 3-SAT to Subset Sum Problem example.

	x_1	x_2	x_3	c_1	c_2
y_1	1	0	0	1	0
$\bar{y_1}$	1	0	0	0	1
y_2	0	1	0	0	1
$\bar{y_2}$	0	1	0	1	0
y_3	0	0	1	1	0
$\bar{y_3}$	0	0	1	0	1
s_1	0	0	0	1	0
s'_1	0	0	0	1	0
s_2	0	0	0	0	1
s'_2	0	0	0	0	1
k	1	1	1	3	3

We can then say that the 3-CNF formulation is feasible if, and only if, the value k is the sum of the numbers presented above. Thus, the subset of S of Sum Set Problem become the set of all rows of the table above, while the targeted sum T become the row k.

Third of all, let us see that a 3-SAT problem is also a SAT problem. Given a SAT problem with m clauses, we can transform each clause by using the following algorithm:

Algorithm 1: Reduction from SAT to 3-SAT

```
$m \leftarrow number\ of\ clause$;
for $c$ in $m$ do
    while $length(c) \leq 3$ do
        $c \leftarrow two\ clauses\ with\ one\ new\ variable$;
    end
end
```

This recursive method works better with an example :

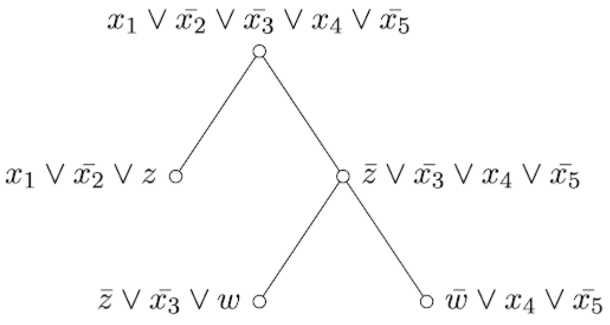
$x_1 \vee \bar{x_2} \vee \bar{x_3} \vee x_4 \vee \bar{x_5}$
$x_1 \vee \bar{x_2} \vee z$
$\bar{z} \vee \bar{x_3} \vee x_4 \vee \bar{x_5}$
$\bar{z} \vee \bar{x_3} \vee w$
$\bar{w} \vee x_4 \vee \bar{x_5}$

Figure 2: 3-SAT to SAT example.

As per with the example above, given the following CNF formulation :

$$x_1 \vee \bar{x_2} \vee \bar{x_3} \vee x_4 \vee \bar{x_5} \tag{18}$$

We end up with the following 3-CNF formulation :

$$(x_1 \vee \bar{x_2} \vee z) \wedge (\bar{z} \vee \bar{x_3} \vee w) \wedge (\bar{w} \vee x_4 \vee \bar{x_5}) \tag{19}$$

Finally, let us compute the complexity of this reduction.

As seen above, with a clause of k variables, we reduce it until we have three variables. At the i^{th} level of the tree, we have one less variable than the previous one. Hence, the tree has $k-3$ layers in total. This demonstrates that we construct $k-2$ new clauses. With a SAT problem of m clauses $k_1, ..., k_m$, the reduced 3-SAT problem will have $\sum_{i=1}^{m}(k_i - 2)$ clauses. Hence, this procedure takes $O(2\sum_{i=1}^{m}(k_i - 2))$ steps, which is a pseudo polynomial complexity. The 3-SAT problem is then NP-complete.

Consequently, if the Knapsack Problem is a case of the Subset Sum Problem, the

Subset Sum Problem can be modelled as a 3-SAT Problem and that the 3-SAT problem is then NP-complete, then, the Knapsack Problem is NP-Complete.

$$NP\text{-}COMPLETE \Rightarrow 3\text{-}SAT \Rightarrow SUBSET\ SUM \Rightarrow KNAPSACK\ PROBLEM$$

5 Literature Review

Next, is the Literature Review, it is a summary of what has been done on this topic prior to this paper. In order to solve the knapsack problem, there are a few things to look at.

5.1 Knapsack Problem

The first one is to learn about the problem itself. This can be done by reading about the origins of this problem. Doing so is complicated, since the problem does not really have a “creator”. In other words, this means that we can not come up with a proper date and a name that came up with a mathematical model to set up the Knapsack Problem. The first person that came up with a similar model to the one that we have today is George Ballard Mathews with his article *On the Partition of Numbers* [9] where the partitions of number refer to our way to do a partition of the space in our knapsack. However, the first time the name of Knapsack Problem came into words as well as a description for it is with Tobias Dantzig in his book *Numbers: The Language of Science* [10] as he describes the problem to be :

> The Knapsack Problem is the commonplace problem of packing the most valuable or useful items without overloading the luggage.

5.2 Knapsack Problem Variant

The second thing to explore is the different types of Knapsack problems, this can be done by firstly looking at is the article written by Alper Atamtürk called *Cover and Pack Inequalities for (Mixed) Integer Programming* [11] where the author describes the difference sets restriction the Knapsack Problem fell into. Moreover, regarding variants of Knapsack Problem lies the archive called *The Multidimensional Knapsack Problem: Structure and Algorithms* [12] written by Jakob Puchinger, Günther R. Raidl and Ulrich Pferschy where they describe one of the most known variant of the Knapsack Problem, the Multidimensional Knapsack Problem which is a Knapsack Problem with multiple contains.

5.3 Robustness

The third thing to look at is robustness. As this project is about the Robust Knapsack Problem, robustness is a key factor in this project. About robustness, the paper written by Jean-Claude Fernandez, Laurent Mounier, and Cyril Pachon called *A model-based approach for robustness testing* [13] is recommended. They describe the use of robustness when testing a system :

> Robustness can be defined as the ability of a software to keep an "acceptable" behaviour, expressed in terms of robustness requirements, in spite of exceptional or unforeseen execution conditions (such as the unavailability of system resources, communication failures, invalid or stressful inputs, etc.).

As robustness is mainly used in testing to ensure that the model fits better real life examples, it can be applied to many problems. One of the Robust Knapsack Problem model and the one which this paper is inspired by can be found in an article written by Dimitris Bertsimas and Melvyn Sim : *The Price of Robustness* [8] In this paper they set up a robust mathematical model for the knapsack problem that takes into account the uncertainty of data. This model will be explored in later sections of this paper. Another type of robustness called Max-Min that involved choosing the minimum optimal value amongst all maximized options. It can be found in the article writen by Gang Yu named : *On the Max-Min 0-1 Knapsack Problem with Robust Optimization Applications* [14]. Uncertainty of data is used to ensure multiple use of the problem and to have a better match with the real life applications.

As we will see later in this paper, robustness can be express in many ways, two main ones are uncertainty sets as expressed by Marc Goerigk and Anita Schoebel in their survey called *Algorithm Engineering in Robust Optimization* [15] and with objectives functions as depicted in the manuscript written by Fabrice Talla Nobibon and Roel Leus, *Complexity Results and Exact Algorithms for Robust Knapsack Problems* [16].

5.4 Different Approaches

Now that the structure of the problem is known, we should start looking at some methods to solve it. The first and the easiest one, is the greedy approach that, in the context of our problem, sorts items in decreasing value per size and starts adding items in the knapsack in order until the knapsack is full or when we can not add any more items. This has been written on paper by George Dantzig, son of Tobias, in his paper *Discrete-Variable Extremum Problems* [17].

In terms of more complex approaches to the Knapsack Problem, two of them come to mind, the Bi-level approach and the Decoupling approach. On the one hand, to learn more about the Bi-level approach in regard to our problem, there is this book written by Luce Brotcorne, Sa¨ıd Hanafi, and Ra¨ıd Mansi called *One-level reformulation of the bilevel Knapsack problem using dynamic programming* [18] in which they describe an application that we are going to see later in this paper for the robust knapsack problem that solves the problem in two steps, with a leader and a follower. On the other hand, to learn more about the Decoupling approach in regard to our problem, I recommend having a look at two things. The first one is about the branch and cut algorithm. This can be done by reading an article written by Manfred Padberg and Giovanni Rinaldi : *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems* [19] where they have taken a decoupling algorithm and applied it to the Symmetric Travelling Salesman Problems. Even though this algorithm will not be used in this paper, It allows us to have critic's look at the following process. The second one is about the branch and bound algorithm. This can be done by reading an article written by Dimitrios Letsios, Miten Mistry, and Ruth Misener : *Exact Lexicographic Scheduling and Approximate Rescheduling* [20] in which they explore the branch and bound algorithm which will be use later in this paper to solve the robust knapsack problem.

6 Mathematical Models

Afterwards, the Mathematical Models used in this project set up. We have the Knapsack Problem with its variants, how modelled robustness is and the bi-level and decoupling approaches.

6.1 The Knapsack Problem

Let us remind us some terminology :

- n is the total number of items
- x_i is the i^{th} item
- v_i is the value of the i^{th} item
- w_i is the weight of the i^{th} item
- W is the overall weight we do not have to exceed

As seen above, the classic Knapsack Problem can be modelled as follows :

$$\begin{aligned} &maximize && \sum_{i=0}^{n} v_i x_i \\ &subject\ to && \sum_{i=0}^{n} w_i x_i \leq W \end{aligned} \tag{20}$$

The Knapsack Problem has some variants, those we will be studying are based on the set constraint the decision variables belong to.

0-1 Knapsack Problem : As explained above, 0-1 Knapsack Problem is as simple as the following model :

$$\begin{aligned} &maximize && \sum_{i=0}^{n} v_i x_i \\ &subject\ to && \sum_{i=0}^{n} w_i x_i \leq W \\ & && x_i \in \left\{x' \in \mathbb{Z} : 0 \leq x' \leq 1\right\} \end{aligned} \tag{21}$$

Mixed 0-1 Knapsack Problem : The decision variable can either be from the 0-1 Knapsack K or from the Non-Negative Real set, the mixed set is then :

$$K_M = \left\{x \in \mathbb{Z} : 0 \leq x \leq 1, y \in \mathbb{R}^+\right\} = \left\{x \in \{0,1\}, y \in \mathbb{R}^+\right\} \tag{22}$$

Where we divide our decision variable set into two sets x and y. The terminology for set x then become :

- n is the total number of items
- x_i is the i^{th} item
- v_i is the value of the i^{th} item
- w_i is the weight of the i^{th} item
- W is the overall weight we do not have to exceed

When the terminology for set y is :

- m is the total number of items
- y_j is the j^{th} item
- v_j is the value of the j^{th} item
- w_j is the weight of the j^{th} item
- W is the overall weight we do not have to exceed

Hence, the Mixed 0-1 Knapsack Problem become :

$$\begin{aligned} &maximize && \sum_{i=0}^{n} v_{1i}x_i + \sum_{j=0}^{m} v_{2j}y_j \\ &subject\ to && \sum_{i=0}^{n} w_{1i}x_i + \sum_{j=0}^{m} w_{2j}y_j \leq W \\ & && x_i \in \left\{x' \in \mathbb{Z} : 0 \leq x' \leq 1\right\}, y_j \in \mathbb{R}^+ \end{aligned} \tag{23}$$

With v and w being respectively segregated into two groups each,v_1 & v_2 and w_1 & w_2.

Integer Knapsack Problem : The Integer Knapsack Problem is built in the same way as the 0-1 Knapsack Problem, except that instead of taking only 0 or 1, our decision variable set takes the whole range of integers :

$$K_I = \left\{x \in \mathbb{Z}^+\right\} \tag{24}$$

Consequently, the Integer Knapsack Problem stand as :

$$\begin{aligned} &\textit{maximize} && \sum_{i=0}^{n} v_i x_i \\ &\textit{subject to} && \sum_{i=0}^{n} w_i x_i \leq W \\ & && x_i \in \mathbb{Z}^+ \end{aligned} \tag{25}$$

Mixed Integer Knapsack Problem : The decision variable can either be from the Integer Knapsack K_I or from the Non-Negative Real set, the mixed set is then :

$$K_{MI} = \left\{x \in \mathbb{Z}^+, y \in \mathbb{R}^+\right\} \tag{26}$$

In the same way as for the Mixed 0-1 Knapsack Problem, we divide our decision variable set into two sets, x and y. Moreover, the terminology is the same. Hence, Mixed Integer Knapsack Problem is modelled as :

$$\begin{aligned} &\textit{maximize} && \sum_{i=0}^{n} v_{1i} x_i + \sum_{j=0}^{m} v_{2j} y_j \\ &\textit{subject to} && \sum_{i=0}^{n} w_{1i} x_i + \sum_{j=0}^{m} w_{2j} y_j \leq W \\ & && x_i \in \mathbb{Z}^+, y_j \in \mathbb{R}^+ \end{aligned} \tag{27}$$

6.2 The Robust Knapsack Problem

The robust Knapsack Problem is as simple as taking the Knapsack Problem, adding the following uncertainty. As described above, uncertainty regarding our Knapsack Problem is depicted through our weight value using sets theory. As a reminder, the set we will be using is :

$$U = [w_i - \bar{w_i}, w_i + \bar{w_i}] \tag{28}$$

Where w_i is the weight of the i^{th} items and $\bar{w_i}$ is the robust weight of that said item that can be computed as follows :

$$\bar{w_i} = w_i + \xi w_i \tag{29}$$

Moreover, there is our perturbation factor, ξ being the percentage of uncertainty our model is confronted with. Without forgiving Γ , the percentage of values that can attain a value in U, the remaining is at w_i. Therefore, our version of the Robust Knapsack Problem is as follows :

$$\begin{aligned}
&\textit{maximize} \quad && \sum_{i=0}^{n} v_i x_i \\
&\textit{subject to} \quad && \sum_{i=0}^{n} \bar{w}_i x_i \leq W \\
& && \bar{w}_i \in \left\{ w \cap \left\{ \Gamma \bar{w}' \right\}, \bar{w}' = w + \xi w \right\} \\
& && x_i \in \left\{ x' \in \mathbb{Z} : 0 \leq x' \leq 1 \right\}
\end{aligned} \tag{30}$$

6.3 The Bi-level model

A Bi-level model consists of two decision-makers. A leader and a follower. Those two decision-makers are hierarchically nested one into the other. In fact, the follower is the one being nested. By doing so we have the independent follower that does not take into account the leader, while we have the leader that has to take into account the follower's decision in order to make the final decision. In terms of variables, both of them have different decision variable sets. In addition, they both have different objective functions and constraints. For the Knapsack Problem, decision variable sets are separated in the same way as we are doing it for the mixed model above. As for the robustness factor, the follower is the only one dealing with uncertainty.

Using the same terminology as above, the Bi-level Knapsack Problem is modelled as follows :

$$\begin{aligned}
&\textit{maximize} \quad && \sum_{i=0}^{n} v_{1i} x_i + \sum_{j=0}^{m} v_{2j} y_j \\
&\textit{subject to} \quad && \sum_{i=0}^{n} w_{1i} x_i + \sum_{j=0}^{m} w_{2j} y_j \leq W \\
& && x_i \in \left\{ x' \in \mathbb{Z} : 0 \leq x' \leq 1 \right\} \\
& && \textit{maximize} \quad \sum_{j=0}^{m} v_{2j} y_j \\
& && \textit{subject to} \quad \sum_{j=0}^{m} w_{2j} y_j \leq W \\
& && \qquad\qquad y_j \in \left\{ x' \in \mathbb{Z} : 0 \leq x' \leq 1 \right\}
\end{aligned} \tag{31}$$

While the Bi-level Robust Knapsack Problem using the uncertainty described above is as depicted :

$$
\begin{aligned}
maximize \quad & \sum_{i=0}^{n} v_{1i}x_i + \sum_{j=0}^{m} v_{2j}y_j \\
subject\ to \quad & \sum_{i=0}^{n} w_{1i}x_i + \sum_{j=0}^{m} \bar{w}_j y_j \leq W \\
& x_i \in \left\{x' \in \mathbb{Z} : 0 \leq x' \leq 1\right\} \\
& maximize \quad \sum_{j=0}^{m} v_{2j}y_j \\
& subject\ to \quad \sum_{j=0}^{m} \bar{w}_j y_j \leq W \\
& \qquad\qquad \bar{w}_i \in \left\{w \cap \left\{\Gamma \bar{w}'\right\}, \bar{w}' = w_2 + \xi w_2\right\} \\
& \qquad\qquad y_j \in \left\{x' \in \mathbb{Z} : 0 \leq x' \leq 1\right\}
\end{aligned}
\tag{32}
$$

6.4 The Decoupling method

The Decoupling approach works in the same manner as a merge sort algorithm, as it divides and conquers. This means that the Decoupling approach is breaking the original problem into small sub problems and dividing them at the end to find them at the end to find a solution. When we are talking about Decoupling, there are two algorithms that come to mind, the first one is the Branch & Bound Algorithm while the second one is the Branch & Cut Algorithm. Since the latter is automatically used by our solver, we have to model the Branch & Bound Algorithm. It comes as follows :

Algorithm 2: Decoupling Branch & Bound Algorithm for the Robust Knapsack Problem

```
Data: $n$ the total number of items
Data: $x$ the list of items
Data: $v$ the list of values
Data: $w$ the list of weight
Data: $W$ the overall weight we do not have to exceed
Result: $sol$ the list of items
$Sort\ all\ items\ in\ decreasing\ order\ of\ \frac{v_i}{w_i+\xi w_i}$
$maxProfit \leftarrow 0$;
$sol \leftarrow new\ list$;
$Q \leftarrow new\ queue$;
$tmp \leftarrow new\ Node$;
$Q.append(tmp)$;
$tmp.profit \leftarrow 0$;
$tmp.weight \leftarrow 0$;
while $Q.empty() = False$ do
    $u \leftarrow Q.pop()$;
    if $u.level = -1$ then
        $tmp.level \leftarrow 0$;
    end
    if $u.level = n$ then
        break
    else
        $tmp.level \leftarrow u.level$;
    end
    $tmp.profit \leftarrow u.profit + tmp.profit$;
    $tmp.weight \leftarrow u.weight + tmp.weight$;
    if $tmp.weight \leq W\ \boldsymbol{and}\ tmp.profit > maxProfit$ then
        $maxProfit \leftarrow tmp.profit$;
        $sol.append(x[tmp.level])$;
    end
    $Compute\ the\ bound\ of\ the\ tmp\ Node$
    if $tmp.bound > maxProfit$ then
        $Q.append(tmp)$;
    end
    $tmp.profit \leftarrow u.profit$;
    $tmp.weight \leftarrow u.weight$;
    $Compute\ the\ bound\ of\ the\ tmp\ Node$
    if $tmp.bound > maxProfit$ then
        $Q.append(tmp)$;
    end
end
```

7 Technical Specification

Subsequently, here are the Technical Specification. They are about the technical tools and resources used. Accordingly, there is how and where data have been collected, how computed and possessed the results are and how presented they are.

7.1 Data Collection

The used data for this project have been collected through the Operations Research Group of the Department of Electrical, Electronic and Information Engineering Guglielmo Marconi of the University of Bologna. Those data can be obtained from the following link :

`http://or.dei.unibo.it/library/robust-knapsack-problem-rkp`

Over here, you will find two directories and a text file. The latter is a README.txt type file, while the other two directories are respectively many instances of the problem and the solutions of those given instances. The one I have been focusing on is the Instances directory. This directory consists of 300 instances, where each instance has the following characteristics :

- A Knapsack capacity
- A number of total items
- The type of Items used in this instance
- An ID
- A list of items with weight and profit

7.2 Processing

For each of the succeeding models, we have applied the 300 instances found above. Those models are :

- The Fundamentals Knapsack Problems
  - 0-1 Knapsack Problem
  - Mixed 0-1 Knapsack Problem
  - Integer Knapsack Problem

    - Mixed Integer Knapsack Problem

- The Fundamentals Knapsack Problems Robust Counterparts
    - Robust 0-1 Knapsack Problem *(a.k.a. Robust Knapsack Problem)*
    - Robust Mixed 0-1 Knapsack Problem
    - Robust Integer Knapsack Problem
    - Robust Mixed Integer Knapsack Problem
- The Bi-level Robust Knapsack Problem
- The Decoupling Branch & Bound Robust Knapsack Problem
- The Greedy 0-1 Knapsack Problem

Subsequently, for models that involved some manually imputed variables, we have been applying the instance to different models with variables tweaked.

As an example, the 0-1 Knapsack Model does not require any manually imputed variables. we then have only 300 solutions - one for each instance -. Whereas the Robust Knapsack Model, which do require some manually imputed variables. Since those two variables are the two factors Gamma and Perturbation, we have been choosing a value within $\{10\%, 20\%, 50\%, 75\%, 100\%\}$. We end up with a total of $5^2 = 25$ models - one for each combination of the two variables -. Hence, we have $25 * 300 = 7500$ solutions.

Once solutions have been computed for every model, means get computed, so we can use those results for the graph below.

7.3 Data Visualization

Once the processing is done and that, we are left with different means waiting to be analysed. All we have to do is to plot them into graphs. This has been done by throwing all the data into the visual analytic platform named Tableau Software, in which I have been able to come up with the following comparisons :

- The impact of robustness (0-1 Knapsack Problem vs Robust Knapsack Problem)
- The best approach for the Robust Knapsack Problem (Bi-level Robust Knapsack Problem vs Decoupling Branch & Bound Robust Knapsack Problem)
- The best fundamental model (0-1 Knapsack Problem vs Mixed 0-1 Knapsack Problem vs Integer Knapsack Problem vs Mixed Integer Knapsack Problem)
- Which factor has the best impact on the problem (Gamma vs Perturbation)
- What about a simple Greedy algorithm (Dantzig's Greedy Algorithm vs 0-1 Knapsack Problem)

7.4 Tools and equipments

In terms of tools, I have been computing everything in Python and have been modelling every problem using the open-source optimization language named Pyomo. In addition to that, I have been solving them with IBM's optimizer solver named CPLEX. In terms of equipment, I have been ensuring that every computation has been performed in the same environment. That means that the computer being used was only doing the said task. Hence, every task has : 2 Gb of allocated DDR4 RAM , 4 cores with 3.4Ghz processing speed

8 Empirical Analysis

Then, the Empirical Analysis arise, it is about the results and their comparison. Here lie graphs answering corresponding questions.

Following our comparisons' axis, we now have graphs that support our analysis. Let us explore those comparisons in detail :

8.1 The Impact of Robustness

In this comparison we are not analysing how good is robustness to allow us to fit to more concrete examples but rather, how the use of that robustness have an impact on results. This has been done by opposing the 0-1 Knapsack Problem and the Robust Knapsack Problem's optimal values and computational speed. Graphs are as follows :

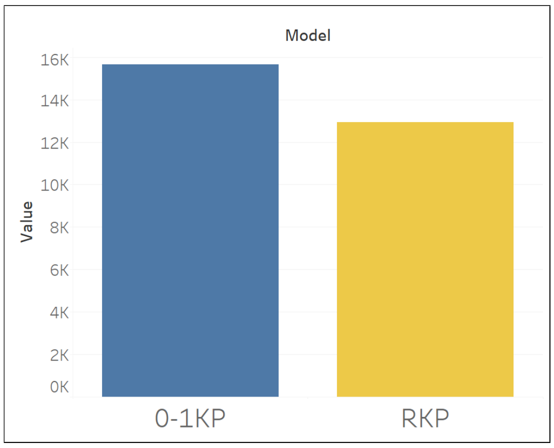

Figure 3: The 0-1 Knapsack Problem mean optimal value on the left compaired to the Robust Knapsack Problem mean optimal value on the right.

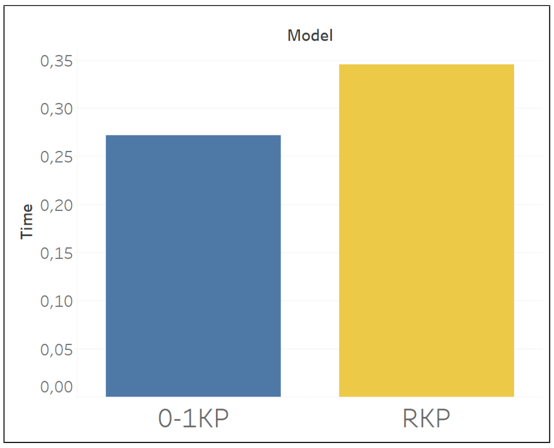

Figure 4: The 0-1 Knapsack Problem mean time on the left compaired to the Robust Knapsack Problem mean time on the right.

As we can see with the figures above, the mean optimal value of the 0-1 Knapsack Problem is higher than the Robust Knapsack Problem one. This is due to the fact that our model of robustness is based on the weights of our items and thus gives bigger restrictions on the constraints. In terms of computational time, using robustness implies having more calculations to perform before even starting to solve the problem. This is seen in the figure above, as the average computational time of the Robust Knapsack Problem is higher than the Robust Knapsack Problem one.

Using robustness have then an impact on the result and on the time, but have then advantages to handle uncertainty. Consequently, before using it we need to ask ourselves, how threatening the uncertainty will be and if the value and time loss is worth the case. At first glance, as robustness is already dealing with the uncertainty issue, we wouldnot think that some robustness issues exists. The main issue related to robustness is ,that robustness increase the complexity of some systems. If we look at the KnapsackProblem, adding robustness increase the complexity of the model and thus of the code.Consequently, the computational time gets longer. This can be seen with the graphs above :

8.2 The best approach for the Robust Knapsack Problem

This is where we figured what is the best approach to model the Robust Knapsack Problem between. This has been done by opposing the Bi-level Robust Knapsack Problem and Decoupling Branch & Bound Robust Knapsack Problem's optimal values and computational speed. Graphs are as follows :

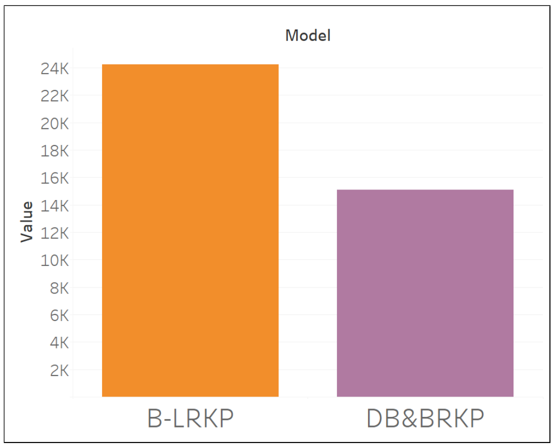

Figure 5: Bi-level Robust Knapsack Problem mean optimal value on the left compaired to the Decoupling Branch & Bound Robust Knapsack Problem mean optimal value on the right.

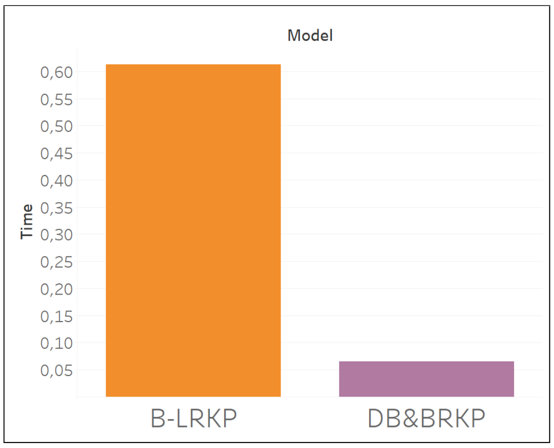

Figure 6: Bi-level Robust Knapsack Problem mean time on the left compaired to the Decoupling Branch & Bound Robust Knapsack Problem mean time on the right.

The figures above allow us to see that the Bi-level Robust Knapsack Model gives a larger mean optimal value than the Decoupling Branch & Bound Robust Knapsack Model. This effect might be due to the way the Decoupling approach is computed as in some cases, it uses some greedy aspects to solve the problem. On the computational time aspect, this is a totally different game, as the computational time of the Decoupling Branch & Bound Robust Knapsack Model gives a way smaller time than the Bi-level Robust Knapsack one. Again, this could also be due to the way the Decoupling approach is computed.

The best approach for the Robust Knapsack Problem would then depend on your needs. one is faster while the other one gives better results. If we want the best of both worlds, we would then need to be creative and mix those two approaches.

8.3 The best fundamental model

This comparison is about finding how constraints on variables can have an impact on the results. We are then bringing together the 0-1 Knapsack Problem, the Mixed 0-1 Knapsack Problem, the Integer Knapsack Problem and the Mixed Integer Knapsack Problem. Graphs are as follows :

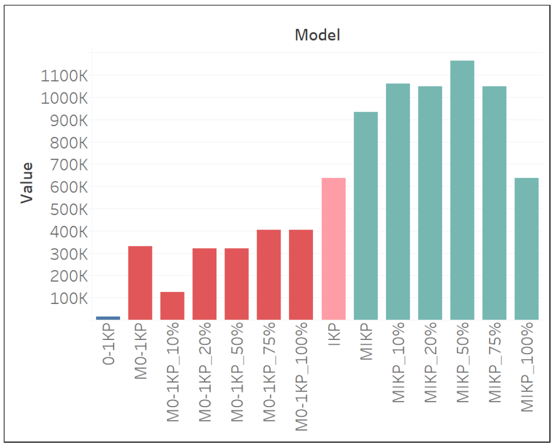

Figure 7: Fundamental Knapsack Problem Knapsack Problems mean optimal value.

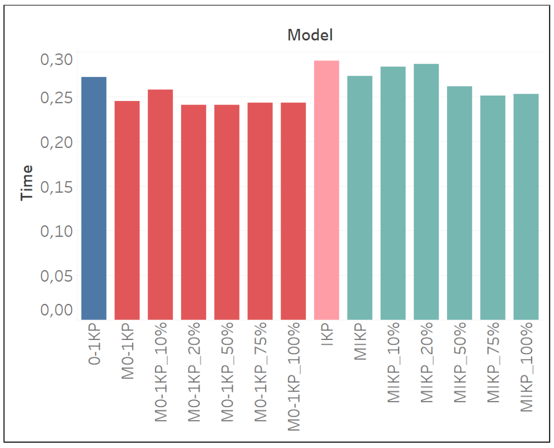

Figure 8: Fundamental Knapsack Problem Knapsack Problems mean time.

For the two figures above going left to right, we have : the 0-1 Knapsack Problem, the Integer Knapsack Problem, the Mixed 0-1 Knapsack Problem with its degree of transition and the Mixed Integer Knapsack Problem with its degree of transition.

Looking at the figures above, we are seeing that out of all the models, the 0-1 Knapsack Problem gives significantly low mean optimal value as a result. This is due because the 0-1 Knapsack Problem is the model with the stronger constraints, only 0 or 1, whereas the Mixed Integer Knapsack Problem is giving the highest mean optimal value and has the lighter constraints, only positive Integer and Non-negative Reals. In fact, we can see that the lighter the constraints on the decision variable, the bigger the mean optimal value is. Following the figure above, in increasing order of result value, we have : the 0-1 Knapsack Problem, the Mixed 0-1 Knapsack Problem, the Integer Knapsack Problem and the Mixed Integer Knapsack Problem. This does correlate with the idea explained above. In terms of computational time. I would say that looking at the figure above, despite having some special cases like the Mixed 0-1 Knapsack Problem with 10% Binary values and 90% Non-negative Reals values who rise above the other

Mixed 0-1 Knapsack Problem. Things are pretty average. That kind of rise could be due to some randomness in the decision of - as in the example above - which value should be Binary and which value should be Non-negative Reals.

It was obvious that adding reals numbers increase results as it increases the number of possibilities we have. Since the time complexity does not really vary from a model to another, the 0-1 Knapsack Model stays the best to study examples, as it is not realistic to add into our knapsack 3.1415 of an item.

8.4 Which factor has the best impact on the problem

As a reminder, the only two things I am tweaking are the two factors, Gamma Γ and Perturbation ξ . Consequently, a comparison between those two factors is needed. Here, we would not be comparing them with regard to the mean optimal value and of the mean computational time; however, we would compare them in terms of which one, once tweaked, gives a bigger change in the results. Graphs are as follows :

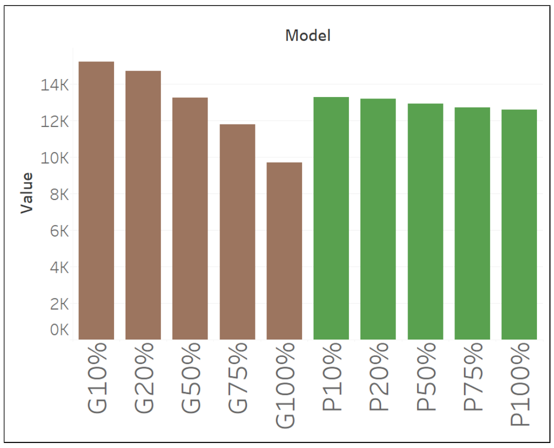

Figure 9: Gamma factor mean optimal value on the left compaired to the Perturbation factor mean optimal value on the right grouped by percentages.

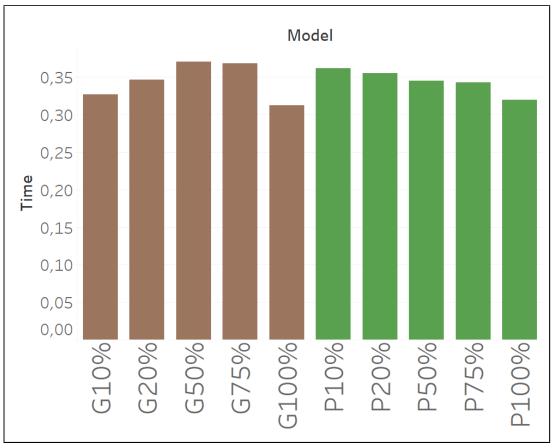

Figure 10: Gamma factor mean time on the left compaired to the Perturbation factor mean time on the right grouped by percentages.

As we can see on the optimal value figure, increasing the percentage of the Gamma factor, implies that the optimal value drops significantly compared to when we increase the percentage of the Perturbation factor which only describes a smooth curve. This phenomenon could be due to the fact that, as seen before, the optimal value for the Robust Knapsack problem is way lower than for the 0-1 Knapsack Problem. Hence, the more we tend to look like a fully *(i.e.* Γ *= 100%)* Robust Knapsack Problem, the bigger the drop is. This same phenomenon can be transposed to the mean computational time figure, as the Gamma factor's curve fluctuates more than the Perturbation one. On top of that, drop in time could also be due to having more values that become robust - in our case more robust weight - and thus, more values that are subject to stronger constraints.

8.5 What about a simple Greedy algorithm ?

When we talk about the Knapsack Problem, the first solution that comes to mind for people and also the first solution that we teach to students is the Greedy Algorithm of George Dantzig, son of Tobias Dantzig. This solution follows a greedy approach to the problem and is easy to understand. The Greedy algorithm of George Dantzig comes as follows :

Algorithm 3: Greedy algorithm for the Knapsack Problem

```
$x \leftarrow Sort\ all\ items\ in\ decreasing\ order\ of\ \frac{v_i}{w_i}$
$totalWeight \leftarrow 0$;
while $x.empty() = False$ and $totalWeight < W$ do
    if $totalWeight + w_{indexOf(x_0)} < W$ then
        $add\ x_0\ to\ the\ knapsack$
        $x.pop(0)$;
    else
        break
    end
end
```

Following that, one idea comes into mind, how good is Dantzig's algorithm compared to another model? Hence, Graphs comes as follows :

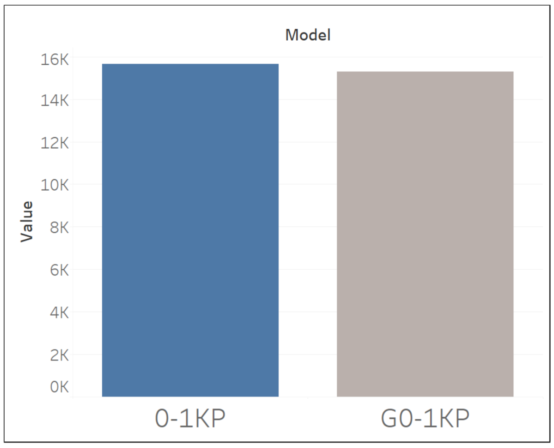

Figure 11: 0-1 Knapsack Problem mean optimal value on the left compaired to the Greedy 0-1 Knapsack Problem mean optimal value on the right.

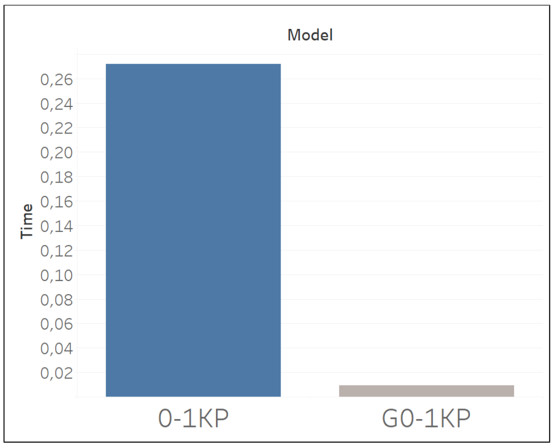

Figure 12: 0-1 Knapsack Problem mean time on the left compaired to the Greedy 0-1 Knapsack Problem mean time on the right.

Looking at the two figures above, the greedy algorithm stated by George Dantzig perform just as intended. As we can see, compared to the 0-1 Knapsack Problem, the Greedy Algorithm output roughly the same mean optimal value - slightly less but negligible -. This small increase of the 0-1 Knapsack problem is due to better results though tout all the instances, see figure 13 below.

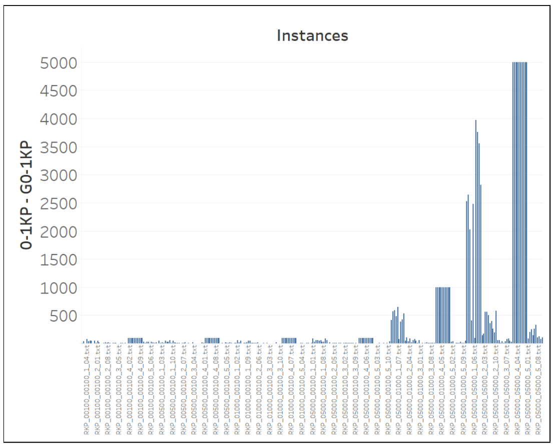

Figure 13: Greedy 0-1 Knapsack Problem mean optimal value subtracted to 0-1 Knapsack Problem mean optimal value.

However, what really stand out is the computational time, which is really low compared to the 0-1 Knapsack Problem. In fact, goings into the numbers and comparing to the other graph, the greedy approach is the fastest. Overall, I would say that Dantzig's algorithm is one of the best solution for the Knapsack Problem - have not tested the robust counterpart but expect similar results - as the ration result over speed is one of the best. In addition, if we take into account the difficulty to understand and teach that approach, the later stand out for being very simple.

9 Software Engineering

Later is Software Engineering. It is about how the Revision Control System called GitHub have been used, how tests have been performed and how the code is working.

9.1 Revision Control System

The use of a version control system rather than storing everything internally is first motivated by being in the habit of using a version control system when starting a project and motivated by the convenience of it as a version control system allows us to have better organizations on tasks with branches and on storages with the master branch and the ease of recovering versions.

I have then used the version control system of GitHub by having a special folder for each model, and types of data as well as respective branches where I commit my code for the working tasks before merging everything into the master branch at when data were collected.

9.2 Testing

As far as the testing goes for this project, it has been performed on the python code used to collect data. Hence, there are three types of things that needed to be tested, the first one the adaptation of algorithms into python and the use of them on the instance data, the second one, is the translation of the different mathematical models explained above and the collection of data with those models and the third one is the mean method that read and write into files.

Everything has been tested using black box testing as it was easier to check results for given instances. Consequently, functional testing has been performed on every unit coded. In terms of non Pyomo based lines of code, like for loops that read and write respectively instances and results, white box testing has been used. Consequently, unit tests and path and branch coverage has been performed on those lines of codes

9.3 UML

This UML diagram, see Figure 14 below, represents how every class of my project are related to each other. Looking at it, two classes stands out, the 0-1KnapsackModel one and the Mean one.

On the one hand, the 0-1KnapsackModel is the Pyomo translation of the 0-1 Knapsack Problem and is used as the baseline of the other models, this includes all the fundamental's Knapsack Problems stated above as well as the Greedy Knapsack Problem which extend to the Decoupling Branch & Bound Robust Knapsack Problem and the Bi-level Robust Knapsack Problem. Moreover, every fundamental's Knapsack Problems inherent their robust counterpart. Doing so simplified a lot of thigs as robust counterpart is simply the problem where we add the robust aspect.

On the other hand, the Mean is simply used here to collect the mean variables for the graphs used above. That is why the Mean class realize actions on every model. Finally, the Mean class is supported by two classes respectively GammaMean and PerturbationMean which as their names imply allow the Mean class to group results by factors - Γ or ξ -.

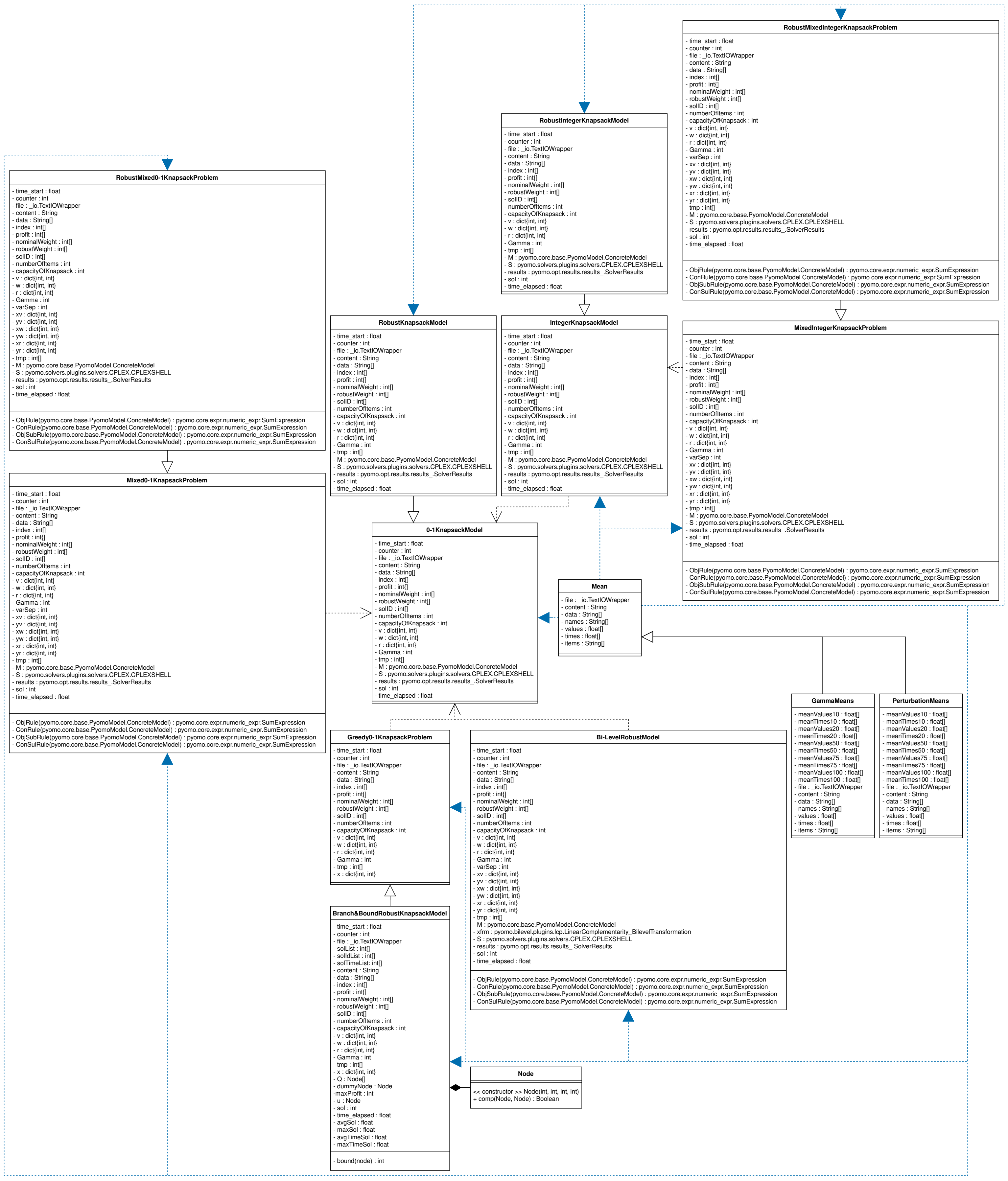

Figure 14: UML Class Diagram.

10 Professional Issues

Finally, Professional Issues stand. They are about the threatening aspects of the problem. Encountered here are limitations, including, legal, social, ethical and robust issues.

Professional issues for this project lies in its wide range of applications of the Knapsack Problem. As some of them are time saving and others are money saving. There are some applications that does not totally lie in those categories. In fact, I mean that some applications of the Knapsack Problem have some drawbacks by not being a good thing for everyone. Therefore, we can sum up all those types of drawbacks into three categories :

10.1 Legal Issues

As the name imply, legal issues are issues about the law. In addition to that, legal issues for the Knapsack Problem are about how the law would not be respected by the final utilization of our problem. Here are some examples of some legal issues that could arise with the Knapsack Problem :

The first legal issue is about gambling, in fact, Sondre Glimsdal in his thesis called *Efficient Gaussian Process Based Optimistic Knapsack Sampling with Applications to Stochastic Resource Allocation* [21] where he gives a Gaussian Process based Knapsack Problem suitable for gambling machine's randomness. Doing so, the Knapsack Problem could be able to deal with probabilities and perhaps trick the casino's gambling machine. This would not be a legal issue until the idea that people could start selling the technique and allow people to "break the bank". This is a legal issue because this does not suit the law as you are not using your technique for personal purpose.

On the same tone, another legal issue with the Knapsack Problem is also related to casino games. This can be read in the book *Encyclopedia of Computer Science and Technology* [22] written by Allen Kent and James G. Williams. This idea is not properly related to casino games. The only thing that is in the name. In fact, in this book, Allen Kent and James G. Williams are describing a method base on the Knapsack Problem to model public key cryptosystems one of the application of that cryptosystem is in mental poker - hence the relation with casinos - This is a legal issue because we can use this cryptosystem to decrypt and intercept messages that use the mental poker protocol.

10.2 Social Issues

The expression of social issues conveys the idea that the problem is affecting people within the society. Therefore, social issues related to the Knapsack Problem are about how the final application of the Knapsack Problem create some inequality between people or group of people. Here are some examples of some social issues that could arise with the Knapsack Problem :

The main types of social issues we will discuss in this part is about work inequalities and especially human replacement by machines.

The first machine replacement that could arise due to the Knapsack Problem is the about the career of librarian or documentalist and especially the part where the worker have to search for a book and sort them. This is where the Knapsack Problem comes into play and can be easily used by a machine, not to sort but to arrange books spaces on shelves for example.

The second machine replacement due to the Knapsack Problem is with trains or subway drivers. The Knapsack Problem can be used to decide which path the train should take. As an example, the Knapsack Problem can be used to find the Optimal path in the railway subway graph. This can be done applying Nadav Voloch's technique that can be found in his article *Optimal paths of knapsack-set vertices on a weight-independent graph* [23]

10.3 Ethical Issues

The terms ethical issues refer to issues that does not obey to moral philosophy. Those kinds of issues are about following the wrong behaviours. Applied to the Knapsack Problem, this concept is about the end usage of the Knapsack Problem not being the right thing that should have been done in terms of moral senses. Here are some examples of ethical issues that could arise with the Knapsack Problem however this time, the examples are not so correlated :

The first ethical issue is about genetics. In fact, this issue arise with the publication of Abbas Y. Al Bayati in his article called *An Implementation of an Initial Scale in Solving Binary Knapsack Problem Using a Genetic Algorithm* [24]. Here, Abbas Y. Al Bayati describe a way of predicting chromosomes and mutations of genes using the Knapsack Problem. Being able to predict genes output a massive ethical issues as, pushing this idea as far a possible, it means that we could choose how our children will look like, and choose its gender.

The other ethical issue that could come with the Knapsack Problem is related to transportation. In fact, one use of the Knapsack Problem is with containers storage management on cargo boats. As we are suing our problem, it allows us to load more containers on boats. The more container we add, the heavier the boat become, moreover, the heavier the boat is, the more fuel we need to use. Finally, the more fuel the boat consume, the more pollution the boat ejects.

11 Conclusion

My project was about the Robust Knapsack Problem and especially about two approaches of modelling and solving that said problem. The first one was a bi-level approach, hence, the Bi-level Robust Knapsack Problem while the second one was about a decoupling approach, hence, the Decoupling Branch & Bound Robust Knapsack Problem. Consequently, we have had some research interrogations that needed to be answered. Despite all of them being solved, some of them remains without answers.

Here are some overall results from the analysis made in this paper that try to answer those remaining unsolved questions.

The first thing that we should note is that amongst all the used techniques to model and solve the Knapsack Problem, the simplest and somehow pretty fast was of solving the problem is the greedy approach. That is why Dantzig's algorithms is the first thing we teach to people when they discover the problem.

The second thing that I have noted from the empirical analysis is that when we compared the Bi-level Robust Knapsack Problem with the Decoupling Branch & Bound Robust Knapsack Problem, the first one gives better result than the second one, however the latter is way faster than the first one. Therefore, choosing the correct approach depends on your needs, as they both have their strengths and weaknesses. The b-level approach is better for results, and the decoupling approach is better for speed.

The last thing that should be noted is that the correct way of analysing and modelling the Knapsack Problem is in fact with the 0-1 set as a decision variable set. Despite have optimal values low enough to be analysed, its computational time is roughly the same as the other fundamentals Knapsack Problems. Hence, the decision comes thanks to its simplicity to understand and its ease to be transposed in real-life.

One of the biggest issues faced in this project was uncertainty, and the whole point of this study was to find the best way of stemming it.

Uncertainty is an issue solved by robustness optimization which has been allowing us to have better match of our models with reality thus, a proper way of modelling it involve randomness, and it is still debatable. As we are adding randomness, models tends to not be precise and require further calculation. After all, I have found that the main issue is not randomness, it is hardware limitation. The faster computers get, the faster computations get. The faster computation gets, the faster randomness is bypassed.

Overall, after doing all the experiment, I have learnt that doing a project like this is not simple and that you will always face some unexpected issues like randomness or simply coding issues. I have learned to manage my time, to set up goals, and to deal with big projects. On a personal experienced point of view, this experiment helps me to get a foothold in the world of optimization. I have then improved my technics and my way of thinking regarding mathematics.

The next step will be to explore more different variant of the Knapsack Problem, such as the Quadratic Knapsack Problem or the Subset-Sum Knapsack Problem. To find out the optimal way of modelling the problem. In terms of uncertainty, the next step will be to compared different uncertainty sets and find out which one is the less prone to randomness. On top of that, we could be creative and model those sets and variants ourselves. As this problem have many applications and thus, big impacts, we could try to apply models to complex situations.

Now that we know more about the Knapsack Problem, let us imagine again and come back to our mission. As a reminder, you have decided to burgle a museum and have made the moronic decisions of only bringing your backpack. As your goal is to get away with the most valuable artworks without either overloading your bag nor breaking it. How do you choose amongst all those beautiful pieces of art and maximize your loot? Knowing that you have limited skills in alarm system and security disarmament, burgling the Louvre Museum in Paris have to be done as fast as you can. Hopefully you have read this paper and know that amongst all the knapsack problem variants and models, the fastest one and easy enough to be done without a computer - unexperimented as you are, you have not brought your computer with you - is the greedy algorithm. If only you had more time *(i.e. the alarm system is off)* and had a friend expert in art with you, you could have used a bi-level approach which is the one that gives better result. I know that you are not an art expert, but at least you know your basics. Consequently, you immediately rush for the most valuable artwork in the museum, the Mona Lisa, steal it and leave before the security arrive. Congratulation, you are now an expert thief, however, good luck on selling the Mona Lisa!

References

[1] A. Agarwal, “Encrypting messages using the merkle-hellman knapsack cryptosystem,” *IJCSNS*, vol. 11, no. 5, p. 12, 2011.

[2] F. Vaezi, S. J. Sadjadi, and A. Makui, “A portfolio selection model based on the knapsack problem under uncertainty,” *PLOS ONE*, vol. 14, pp. 1–19, Apr 2019.

[3] P. Tangtatswas, “Algorithm for the cutting stock problem with multiple raws and limited number of cutting knives,” 2017.

[4] A. Kulik and H. Shachnai, “There is no eptas for two-dimensional knapsack,” *Information Processing Letters*, vol. 110, no. 16, pp. 707–710, 2010.

[5] Y. Song, C. Zhang, and Y. Fang, “Multiple multidimensional knapsack problem and its applications in cognitive radio networks,” pp. 1 – 7, Dec 2008.

[6] S. S. Skiena, “Who is interested in algorithms and why? lessons from the stony brook algorithms repository,” *SIGACT News*, vol. 30, p. 65–74, Sept. 1999.

[7] D. Wheeler, E. Meenken, M. Espig, M. Sharifi, M. Shah, and S. Finlay-Smits, “Uncertainty -what is it?,” 02 2020.

[8] D. Bertsimas and M. Sim, “The price of robustness,” *Operations Research*, vol. 52, pp. 35–53, 02 2004.

[9] G. B. Mathews, “On the Partition of Numbers,” *Proceedings of the London Mathematical Society*, vol. s1-28, pp. 486–490, 11 1896.

[10] T. Dantzig and J. Mazur, *Number: the language of science.* Plume Book, 2007.

[11] A. Atamtürk, “Cover and pack inequalities for (mixed) integer programming,” *Annals of Operations Research*, vol. 139, pp. 21–38, Oct 2005.

[12] J. Puchinger, G. R. Raidl, and U. Pferschy, “The Multidimensional Knapsack Problem: Structure and Algorithms,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250 – 265, 2010.

[13] J.-C. Fernandez, L. Mounier, and C. Pachon, “A model-based approach for robustness testing,” pp. 333–348, 05 2005.

[14] G. Yu, “On the max-min 0-1 knapsack problem with robust optimization applications,” *Operations Research*, vol. 44, no. 2, pp. 407–415, 1996.

[15] M. Goerigk and A. Schöbel, “Algorithm engineering in robust optimization,” 2016.

[16] F. Talla Nobibon and R. Leus, “Complexity results and exact algorithms for robust knapsack problems,” *Journal of Optimization Theory and Applications*, vol. 161, pp. 533–552, May 2014.

[17] G. B. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. 266–288, 1957.

[18] L. Brotcorne, S. Hanafi, and R. Mansi, "One-level reformulation of the bilevel knapsack problem using dynamic programming," *Discrete Optimization*, vol. 10, no. 1, pp. 1–10, 2013.

[19] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991.

[20] D. Letsios, M. Mistry, and R. Misener, "Exact lexicographic scheduling and approximate rescheduling," 2018.

[21] S. Glimsdal, "Efficient gaussian process based optimistic knapsack sampling with applications to stochastic resource allocation," Master's thesis, Universitetet i Agder/University of Agder, 2013.

[22] A. Kent, J. G. Williams, and R. Kent, *Encyclopedia of computer science and technology.* Marcel Dekker, Inc., 1989.

[23] N. Voloch, "Optimal paths of knapsack-set vertices on a weight-independent graph," *WSEAS Transactions on Computers*, vol. 16, pp. 163–171, 09 2017.

[24] A. Bayati and N. Qubat, "An implementation of an initial scale in solving binary knapsack problem using a genetic algorithm," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 4, 12 2007.

[25] C. A. Pachon, J.-C. Fernandez, and D. M. Bozga, *Une approche basee sur les modeles pour le test de robustesse.* PhD thesis.

[26] M. Monaci, U. Pferschy, and P. Serafini, "Exact solution of the robust knapsack problem," *Computers & Operations Research*, vol. 40, p. 2625–2631, 11 2013.

[27] G. Cornuéjols, M. A. Trick, and M. J. Saltzman, "A tutorial on integer programming," 1995.

[28] E. W. Weisstein, "Np-problem," *MathWorld A Wolfram Web Resource*, 2021.

[29] M. Richmond, "Examples of uncertainty calculations," *Rochester Institute of Technology*, 2019.

[30] M. Monaci and U. Pferschy, "On the robust knapsack problem," vol. 23, pp. 207–210, 01 2011.

[31] T. Mahmoodi, "Teams meeting link for 7ccsmnth network theory sem2 000001/dl/01," *King's College London*, 2021.

[32] A. Bari, "7.2 0/1 knapsack using branch and bound," Feb 2018.

[33] O. Y. Özaltın, O. A. Prokopyev, and A. J. Schaefer, "The bilevel knapsack problem with stochastic right-hand sides," *Operations Research Letters*, vol. 38, no. 4, pp. 328–333, 2010.

[34] S. Jain, "0-1 knapsack problem, dp-10," *GeeksforGeeks*, 2021.

[35] S. Jain, "Implementation of 0/1 knapsack using branch and bound," *GeeksforGeeks*, 2019.

[36] J. M. Garrido, *Introduction to Computational Models with Python.* CRC Press, 2015.

[37] Pyomo, "Pyomo documentation," Aug 2021.

[38] F. Frising, "Imprécision d'une mesure - erreur absolue et relative," *Université de Namur*, 2021.

[39] M. Monaci, "Knapsack problem and variants," *DEI, University of Bologna, Italy*, Apr 2019.

[40] R. L. Harrison, "Introduction to monte carlo simulation," *AIP Conference Proceedings*, vol. 1204, no. 1, pp. 17–21, 2010.

[41] S. Wernicke, B. Sean, and N. Sarah, "Boolean satisfiability problem - intro to theoretical computer science," Feb 2018.

[42] D. P. Williamson, "Orie 6300 mathematical programming i, lecture 25," *Cornell University*, Nov 2014.

[43] R. Peng, "Cs 3510 design & analysis of algorithms, np-completeness of knapsack," *Georgia Tech, College of Computing*, Nov 2016.

[44] "Design of algorithms, np-complete problems," *Ben-Gurion University of the Negev*, Jan 2007.

[45] D. M. Barrington and A. Maciel, "Lecture 7: Np-complete problems," *IAS/PCMI Summer Session 2000 Clay Mathematics Undergraduate Program Basic Course on Computational Complexity*, Jul 2000.

[46] A. Dudley, "Boolean satisfiability problem - intro to theoretical computer science," Dec 2016.

[47] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency.* Springer, 2003.

[48] J. E. Beasley, "Or-notes," *IAS/PCMI Summer Session 2000 Clay Mathematics Undergraduate Program Basic Course on Computational Complexity*, Sep 2009.

[49] K. S. Mamedov, K. K. Mamedov, and S. K. Elchueva, "Solving the mixed-integer knapsack problem by decrease of dimension and use of dynamic programming," *Automatic Control and Computer Sciences*, vol. 49, pp. 231–238, Jul 2015.

[50] G. J. Koehler, *Ultimate X Bonus Streak Analysis*. PhD thesis, The Wizard of Odds, 2017.

[51] M. Poss, “Robust combinatorial optimization with knapsack uncertainty,” *Discrete Optimization*, vol. 27, pp. 88 – 102, Feb. 2018.

[52] B. L. Gorissen, İ. Yanıkoğlu, and D. den Hertog, “A practical guide to robust optimization,” *Omega*, vol. 53, pp. 124–137, 2015.

A Appendix

A.1 Extra Figures

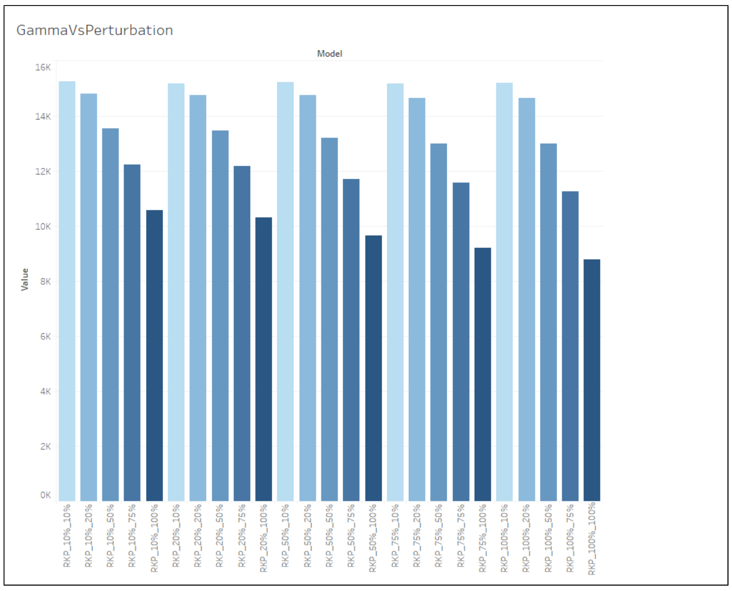

Figure 15: Gamma and Perturbation Factors Mean Value Detailed.

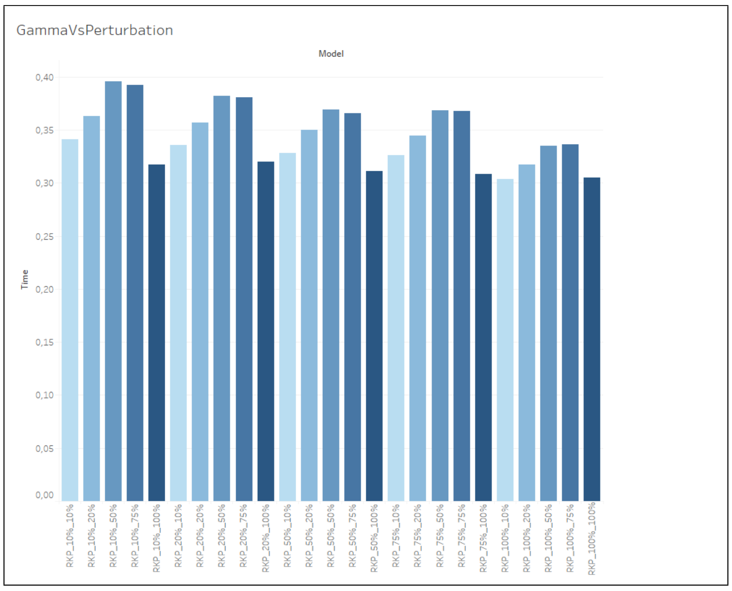

Figure 16: Gamma and Perturbation Factors Mean Time Detailed.

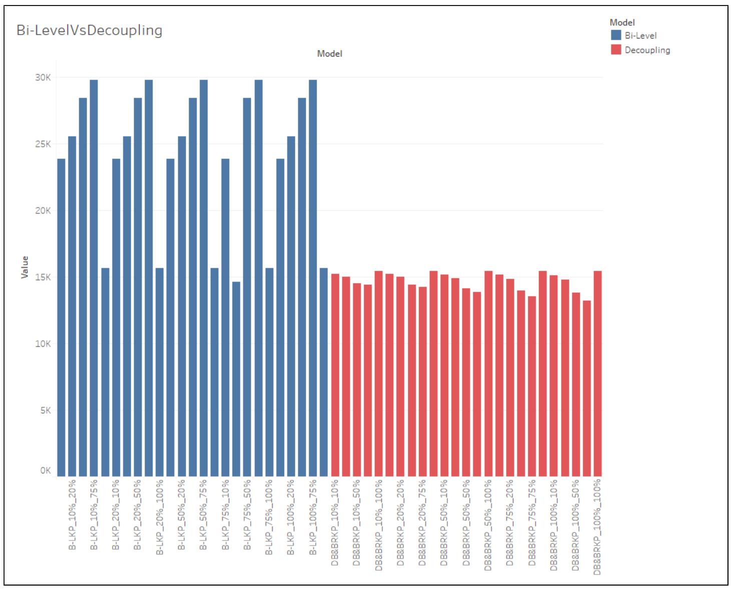

Figure 17: B-LRKP and DB&BRKP Mean Value Detailed.

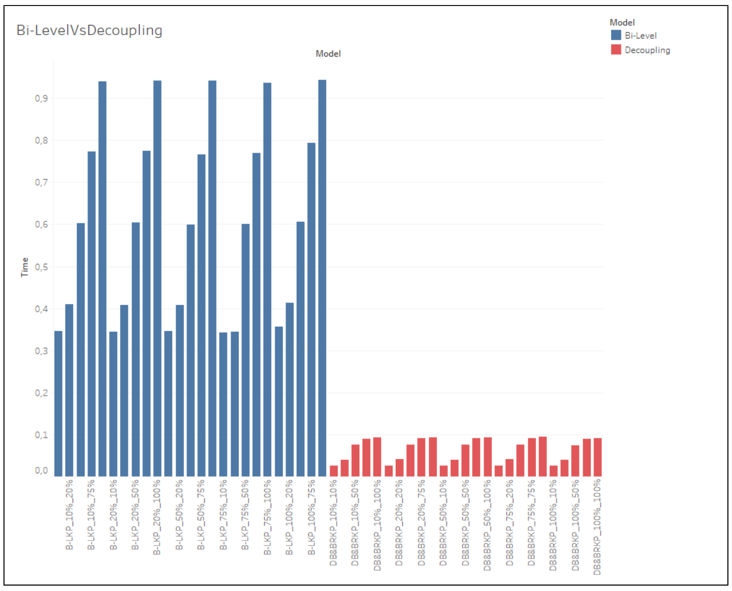

Figure 18: B-LRKP and DB&BRKP Mean Time Detailed.