Multiobjective Scalarization Methods applied to the Standard Knapsack Problem.

By

Yash Prajapati

University College Cork – Ireland
Department of Computer Science
College of Science, Engineering and Food Science
Major: MSc. Data Science and Analytics
Course: CS6500 – Final Research Project 2023

Academic Supervisor:
Dr. Steve Prestwich

Second Reader:
Harry Nguyen

4th September 2023

Abstract

The Standard Knapsack Problem is a challenging combinatorial optimization problem having applications in many fields. This thesis studies the use of multiobjective optimization approaches to solve the Knapsack Problem, notably Linear Scalarization, Weighted Metric Method, and Chebyshev Scalarization. In addition, the TABU search metaheuristic is implemented to improve solution space discovery.

The study had three main goals:

a) implement and adapt scalarization methods to the Knapsack Problem,

b) incorporate TABU search to improve solution diversity and quality, and

c) design systematic experiments to evaluate the performance of the integrated methods under varying problem parameters and constraints.

Python was used to implement the algorithms. The experiment was carried out in three stages by generating synthetic Knapsack Problem instances and running the techniques in various settings. The results revealed Chebyshev Scalarization's superiority in consistently identifying the best solutions most number of times across all the phases of experimentation. The Weighted Metric Method was also capable of adapting to shifting limitations. The performance of the Linear Scalarization approach varied depending on the problem instance.

In summary, by rigorous testing, this thesis provides a thorough framework for applying and analysing multiobjective optimization strategies for the Knapsack Problem. The findings shed light on the advantages and disadvantages of scalarization approaches under various scenarios. The study helps to guide the selection and use of appropriate multiobjective optimization algorithms based on the characteristics and limitations of the problem.

Declaration

I confirm that, except where indicated through the proper use of citations and references, this is my original work and that I have not submitted it for any other course or degree.

Signed: Yash Prajapati

Date: 4th August 2023

122100327

Yash Prajapati

04/09/2023

Acknowledgement

I would like to thank my thesis supervisor, Dr. Steve Prestwich, for his encouragement, patience, and helpful advice during my study journey. Your knowledge, insights and experience have been priceless, and I could not have asked for a more knowledgeable and encouraging mentor.

I am extremely grateful to the University College Cork (UCC) for providing me with access to critical research materials in digital format. Their resources have been crucial in the advancement of this research.

I owe an incalculable debt of gratitude to my family, who have shown me enormous love and unwavering support throughout my academic endeavours. My parents and my sister, your sacrifices and unending encouragement have been a guiding light in my life, driving my determination and drive. Your understanding, levity, and inspiring talks have provided me with sustenance over the long nights of writing and rigorous rewriting.

I would finally like to express my sincere thanks to my friends who have helped and motivated me till the very end of this thesis and this entire academic year.

Contents

List of Figures

List of Code Snippets

1 INTRODUCTION

1.1 Motivation

Day-to-day life decisions and orientations are steered by choices. While sentiments and feelings have a crucial part in making choices, numerous situations require formalizing the process of decision-making to solve complex problems. This objective is encompassed by combinatorial optimization, a division of operational analysis that furnishes techniques and methods to solve optimization problems emerging across various domains. Within this paper, a distinct class of optimization problem is examined: the Knapsack Problem.

The Knapsack problem is something that everyone has unknowingly come across, whether it is from grocery shopping to packing the schoolbag. We always evaluate the items that need to be packed in the knapsack or bag according to its value and the weight while keeping the knapsack weight capacity into consideration. The knapsack problem is a NP hard problem in combinatorial optimization with numerous applications. As industries and fields continue to evolve, we as the decision-makers are faced with the challenge of making informed choices that balance multiple criteria. The Knapsack Problem serves as a fundamental representation of such decision-making scenarios, where the need to optimize the objectives under capacity constraints is pervasive.

Even though we attempt to serve several objectives while solving the knapsack problem, we often tend to only maximize the value of the items that need to be included into the knapsack and this in turn makes us neglect the weight capacity constraint of the knapsack. This is similar to how people often choose what to eat and drink without considering the simple nutritional benefits of food options. Therefore, we should consider all the objectives when it comes to solving the knapsack problem, i.e., Maximizing the value of the items selected in the knapsack, Minimizing the weight that might exceed the knapsack weight capacity in order to keep the knapsack weight constraint in mind.

Moreover, the thesis aims to advance the understanding of how different multiobjective optimization methods perform in the context of the Knapsack Problem, thus aiding in the selection and application of appropriate algorithms for similar problems.

1.2 Problem Statement

The Knapsack Problem is a fundamental optimization problem that arises in a variety of decision-making settings, requiring the selection of a subset of objects from a given set while adhering to the limited capacity of a knapsack [1]. In contrast to the traditional single-objective Knapsack problem, where the primary objective is to maximize the total value of selected items while adhering to the capacity constraints, this problem introduces the complication of simultaneously optimizing multiple conflicting objectives [2].

In the Knapsack Problem, the overarching aim is to identify a set of items that optimizes multiple objectives, with the primary objectives being:

- ❖ Maximizing Total Value: Select a subset of items that maximizes the total value. The value of an item represents its significance or benefit.
- ❖ Minimizing Excess Weight: Ensure that the sum of the weights of the selected items does not exceed the capacity of the knapsack. The objective is to minimize the amount by which the weight exceeds the capacity.

To accommodate the constraints introduced by the minimization of excess weight, an additional aspect of the problem is considered:

- ❖ Penalization for Weight Excess: In cases where the total weight of the selected items surpasses the knapsack's capacity, a penalty is applied to the solution [3]. The penalty is computed by subtracting the total weight from the knapsack's capacity. This penalty mechanism aims to incentivize solutions that maintain a balance between maximizing value and minimizing weight excess.

The Knapsack Problem, therefore, requires the exploration of a solution space where trade-offs between maximizing value and minimizing excess weight are balanced. The challenge lies in finding a set of solutions that reside on the Pareto-optimal front, offering varying degrees of value and weight trade-offs. These solutions collectively represent the feasible compromise solutions that cater to different decision-maker preferences.

1.3 Research Approach

The primary goals of this research are to investigate and analyse the application of multiobjective optimization techniques to the Knapsack Problem, assess the performance of different scalarization methods, and enhance the solution process through the integration of a metaheuristic approach [1]. Specifically, the research aims to achieve the following objectives:

- ❖ Method Exploration and Implementation:
    - ➢ Investigate and understand the Linear Scalarization Method, Weighted Metric Method, and Chebyshev Method as multiobjective optimization techniques.
    - ➢ Implement these techniques to solve the Knapsack Problem, adapting them to the specific requirements of the problem.
- ❖ Metaheuristic Integration:
    - ➢ Incorporate the TABU search metaheuristic as a complementary approach to explore the solution space of the Knapsack Problem.
    - ➢ Develop the integration of the TABU search algorithm with the scalarization methods to enhance solution quality and diversity.
- ❖ Data Generation and Experiment Design:

    - Generate synthetic data sets to simulate a range of problem instances, varying the parameters such as item values, weights, knapsack capacity, and number of items.
    - Design a systematic experimentation plan to thoroughly evaluate the performance of the integrated methods across different problem instances.
- Algorithm Performance Evaluation:
    - Execute the integrated methods, including the three scalarization techniques with TABU search, on the generated problem instances.
    - Measure and analyse the quality of the solutions obtained, focusing on the characteristics of the Pareto-optimal fronts, convergence speed, and diversity of solutions.
- Comparative Analysis:
    - Compare and contrast the performance of the Linear Scalarization Method, Weighted Metric Method, and Chebyshev Method in terms of their effectiveness in producing Pareto-optimal solutions.
    - Identify strengths and limitations of each method under varying problem instance settings.
- Parameter Sensitivity Study:
    - Investigate the sensitivity of the integrated methods to different parameters, such as item values, weights, and knapsack capacity, as well as the number of TABU search iterations.
    - Analyse how changes in these parameters impact the quality of solutions and the behaviour of the algorithms.
- Conclusive Insights and Recommendations:
    - Summarize the findings from the experimentation and analyses.
    - Provide evidence-based insights and recommendations for selecting the most suitable method based on problem characteristics and preferences.

Through the achievement of these research objectives, this study contributes to the understanding of multiobjective optimization for the standard Knapsack Problem and offers practical guidance for researchers and practitioners seeking effective approaches in similar multiobjective optimization scenarios.

1.4 Thesis Structure

- Chapter 2 introduces the background of the thesis, important technical concepts and technologies used.
- Chapter 3 involves the Literature Review
- Chapter 4 discusses the Methodology used in this thesis and the Design of the experimentation.
- Chapter 5 expands on the implementation of the Experimentation
- Chapter 6 explores the results obtained in the thesis.
- Chapter 7 highlights the conclusion and scope of future work.

2 BACKGROUND

2.1 Dataset

For this thesis an important aspect was the Dataset from which we are going to get the values and weights for the items to attempt to solve the Knapsack problem. The values or the benefits of an item and the weights of those items were generated using python. Python is a versatile and widely-used programming language, forms the cornerstone of the dataset generation process. The programming language has a wide set of libraries and the ease to use enables for the generation of a data that mirrors the complexity of the Knapsack Problem. A pivotal tool in the Python programming arsenal is the random module and by using the random function we introduce a controlled randomness to the data. The controlled randomness ensures and facilitates the creation of diverse problem instances.

In the pursuit of solving the Knapsack Problem, creating a custom dataset in Python provides a plethora of benefits that are critical for rigorous research and aligning with the complexities of the challenge. One of the most significant advantages of generating a custom dataset is the ability to carefully customize problem instances to the research objectives. Researchers can manipulate numerous parameters such as the number of items, their values, weights, and constraints by creating datasets from scratch, ensuring that the dataset closely replicates the complexity and characteristics of the real-world Knapsack Problem scenarios under examination. Creating a custom dataset guarantees that the research process is completely transparent. Other researchers can use the same dataset generating approach to duplicate and validate the results, enhancing reproducibility and increasing the scientific community's confidence in the research findings.

2.2 Technical concepts

2.2.1 Multiobjective Optimization

Several real-life optimization problems such as in the fields of agriculture, manufacturing, construction, etc. involve multiple objectives. Often these objectives are conflicting in nature hindering with the solution of one another. All these objectives are required to be handled all at once. This in turn gives us the concept of Multiobjective Optimization [4]. In contrast to single-objective optimization, which seeks a single best solution, multiobjective optimization aims to find a set of solutions that represent different trade-offs between competing objectives [5]. This set of solutions is known as the Pareto-optimal front or Pareto frontier, and it showcases the optimal compromises achievable between the objectives.

Let us consider a particular Multiobjective Optimization problem in the given form,

$$minimize/maximize \quad \{f1(x), f2(x), f3(x), . \ . \ . \ . \ . \ , fn(x)$$

$$\text{subject to } x \in X$$

where $x \in R^n$ is a decision variable, f_i: R n → R for i ∈ I = {1, ..., n} with n ≥ 2 are objective functions, and $X \subset R^n$ is called a feasible set. The image of the feasible set is called the objective function space.

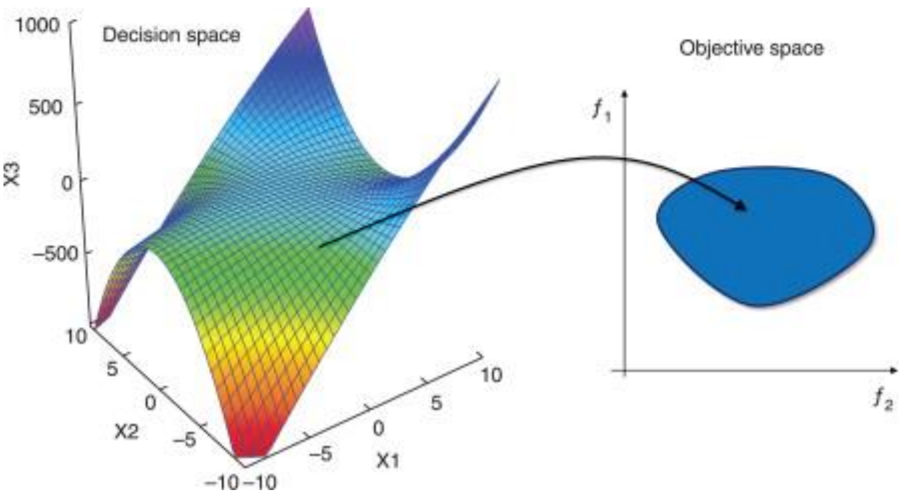


Figure 1: Decision Space and Objective Space for Multiobjective Optimization Problem

2.2.2 Knapsack Problem

The Knapsack problem is a classic optimization problem in combinatorial optimization. It entails choosing items from a set, each of the items carry a specific value or benefit and weight with the goal of maximizing the total value of the items selected. This objective must be met by adhering to the knapsack capacity constraint [6]. The problems name 'knapsack' is derived from the analogy of packing items into a bag or a knapsack with limited capacity. The applications of this problem are found across various fields such as resource allocation, project scheduling, portfolio optimization, etc. The challenge emerges from the need to be able to make optimal selections while taking into account the limitations of available resources and considering the trade-off between value and weight.

Unlike the single objective knapsack problem, we take a step beyond by introducing the Knapsack problem with multiple conflicting objectives. And just like in the single objective knapsack problem, there is an interest in maximizing the total value of the items that are selected in the knapsack. Simultaneously, there is another goal to minimize the total weight of the selected items [1]. The crucial aspect when solving the knapsack problem is selecting an item with higher value might also have higher cumulative weight, whereas focusing on lightweight items could potentially sacrifice the value. This trade-off between conflicting objectives gives rise to a collection of potential solutions that represent various compromises between maximizing value and minimizing weight.

2.2.3 Pareto Optimality

In the context of the multiobjective optimization, a pareto optimal solution is the one if there is no other solution that is better in all objectives and worse in none. It essentially symbolises a compromise in which one objective's improvement comes at the expense of another [7]. Pareto optimality denotes a balance that cannot be improved without making trade-offs.

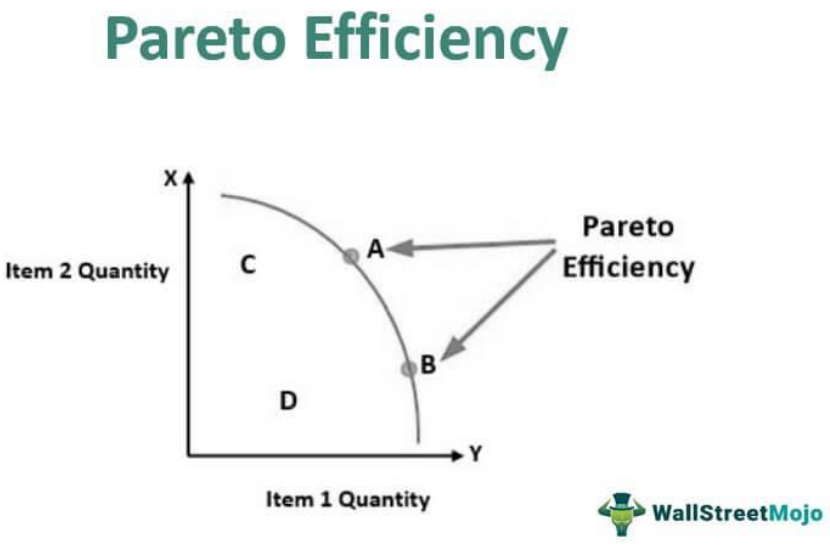


Figure 2: Pareto Efficiency or Pareto Optimality

2.2.4 Methods used

There are several ways for dealing with the complexity of multiobjective optimization problems. These methods can be generally classified on the basis of their approach to handling multiple conflicting objectives. Evolutionary algorithms, Scalarization methods, constraint handling approaches, and more [8].

Overview of Linear Scalarization, Weighted Metric Method and Chebyshev Scalarization

- ❖ Linear scalarization
  - ➢ By linearly integrating the objectives with specified weights, this method converts the multiobjective optimization problem into a single objective optimization problem [9].
- ❖ Weighted Metric Method
  - ➢ The weighted metric method like the linear scalarization method assigns weights to the objectives. However, it calculates the distances between each solution and an ideal solution and a reference point in the objective space [10].
- ❖ Chebyshev Scalarization
  - ➢ The Chebyshev scalarization method entails solving a single objective problem with the purpose of minimizing the maximum weighted distance between the solution

and the utopian solution. Utopian solution refers to the perfect solution or the most ideal outcome for a given problem [11].

2.2.5 Metaheuristic Approach

Metaheuristic algorithms are optimization techniques that aid in the search for large, complex problem spaces. Unlike exact methods, which require specific problem structures and mathematical features, metaheuristics are universal problem-solving strategies that can tackle a wide range of problems, often when finding optimal solutions is difficult or computationally infeasible.

TABU search is a well-known Metaheuristic algorithm that focuses on optimization problems by exploring the solution space iteratively. It combines the features of exploration and exploitation to find better solutions more effectively. The term 'TABU' refers to a memory mechanic that maintains a note of the recent search moves and restricts revisiting certain solutions to encourage diversification [12]. Its capacity to handle a wide range of problems and adapt to various search strategies makes it a viable and valuable for addressing optimization problems.

2.3 Technology used

Python is a popular programming language and it serves as the core platform for conducting the research activities outlined in this thesis. It is known for its versatility and ease of use, allowing researchers to seamlessly translate abstract concepts into practical implementation. Its extensive libraries and several resources online offer an array of tools essential for tackling complex optimization problems.

Within the context of this thesis, several Python libraries were used:

- NumPy
  - NumPy, a basic scientific computing library, it is used for a wide variety of mathematical operations on arrays. It aided in the synthetic data generation process, by using the random module in the NumPy library [13].
- SciPy
  - The SciPy library enhanced the research capabilities by including a variety of optimization algorithms, statistical functions, and mathematical tools. The optimization module in SciPy aided in the customization of algorithms, allowing them to be more tailored to the complexities of the of the Knapsack problem [14].
- Matplotlib
  - Matplotlib has emerged as a powerful visualization tool, allowing for the visualization of data. The graphs and insightful plots from matplotlib help in effective communication of the research findings [15].

3 LITERATURE REVIEW

The knapsack problem is a well-known combinatorial optimization problem with numerous applications in the real world. It entails filling a knapsack with items that have both a weight and a value, with the goal of maximizing the overall value of the items in the knapsack without surpassing its weight capacity.

The typical knapsack problem only examines a single objective the total value of selected items. However, many real knapsack problems necessitate simultaneously optimizing many competing objectives. For example, one might want to maximize value while also minimizing weight, or maximize value while minimizing the number of items selected [16]. This has resulted in extensive study on multiobjective formulations of the knapsack problem.

Early work on multiobjective knapsack problems focused on bi-objective models with value and weight objectives [17]. In 1987, Serafini introduced one of the first multiobjective genetic algorithms for this problem [18]. Other early attempts presented the problem as a multi-criteria decision-making model and used techniques such as goal programming [19].

Recent research has looked on knapsack problems with three or more objectives. In 1997, Gandibleux et al. established a strategy for determining the Pareto optimal set for a tri-objective knapsack problem with value, weight, and volume objectives [20]. In 2004, Doerner et al. tested various evolutionary multiobjective algorithms on bi- and tri-objective knapsack problems and discovered that the NSGA-II performed best overall [21].

The development of solution methods that may effectively approximate the Pareto frontier in problems with complicated objective spaces and constraints has been a primary emphasis in multiobjective knapsack research. Metaheuristics such as evolutionary algorithms and swarm optimization have been shown to be effective in this regard [22]. Multiobjective ant colony optimization has also been applied to knapsack problems with success [23].

Overall, multiobjective optimization has emerged as a valuable method for resolving real-world knapsack problems with competing objectives. The current focus of research is on designing algorithms that can handle high-dimensional objective spaces and big problem cases efficiently. Many concerns remain unanswered about approaches for balancing solution quality and computing effort on complex multiobjective knapsack formulations.

Scalarization is a popular method for dealing with multiobjective optimization problems. It creates a scalarized objective function to turn a multiobjective problem into a single objective problem. To the multiobjective knapsack problems, scalarization approaches such as linear scalarization, weighted sum, weighted metric, epsilon constraint, and Chebyshev have been used.

In 2004, Marler and Arora [8] compared weighted sum and weighted metric approaches on bi-objective knapsack problems. They discovered that using a weighted measure method increased the diversity and distribution of solutions on the Pareto frontier. On tri-objective knapsack examples, Lust and Teghem used the weighted sum, epsilon constraint, weighted min-max, and weighted metric approaches in 2010 [1]. Their findings revealed that epsilon constraint and weighted min-max yielded more Pareto optimum solutions.

Several studies have looked into the Chebyshev approach for multiobjective knapsacks in particular. Hu et al. introduced a chebyshev-based evolutionary algorithm in 2003 and demonstrated that it outperformed weighted techniques [24]. Tan et al. created a chebyshev-based particle swarm optimization for bi-objective problems in 2007, and discovered that it was effective on large knapsack examples [25]. Recently, Liang et al. used an adaptive chebyshev technique to solve a profit-cost-risk knapsack model [26].

Marler and Arora determined that the weighted metric method was the best overall for bi-objective knapsacks when evaluating scalarization strategies [8]. The epsilon constraint method produced the most Pareto optimum solutions across tri-objective examples, according to Lust and Teghem [1]. The Chebyshev method appears to be particularly promising for equally dispersing solutions over the Pareto frontier [25,26].

Tabu search is a metaheuristic that guides the search through the solution space using adaptive memory structures. In recent years, it has been used to solve multiobjective knapsack problems.

Gandibleux et al. devised a tabu search strategy for a tri-objective knapsack problem with objectives of value, weight, and volume in 1997, which was one of the first implementations [27]. They employed tactics such as tabu lists, diversification, and intensification. According to the results, tabu search produced good estimates of the Pareto set.

In 2003, Loria et al. introduced a tabu search method for bi-objective knapsack problems [28]. Their adaptive memory strategy enabled them to strike a balance between exploration and exploitation. It outperformed a genetic algorithm in comparison tests.

In 2010, Yagmahan and Yenisey [29] employed a fuzzy variant of tabu search for a profit-cost knapsack model. In comparison to weighted genetic algorithms, their method produced solutions with greater diversity across the Pareto frontier.

Tabu search and scalarization methods are combined in some hybrid algorithms. On multiobjective unconstrained problems, Hansen, for example, used tabu search in conjunction with chebyshev scalarization [30]. The tabu search was guided by adaptive weight vectors.

In conclusion, tabu search shows potential as a metaheuristic technique for obtaining good Pareto set approximations on complex multiobjective knapsack problems. Future research could benefit from hybridization with scalarization approaches.

4 METHODOLOGY

In Chapter 4, we get deep into the core of our research methodology. This chapter provides the foundation of our research, in which we methodically explain the methodologies and strategies used to solve the many problems posed by the Knapsack Problem.

4.1 Linear Scalarization

While tackling the Knapsack problem using the Multiobjective Optimization Methods, Linear Scalarization emerges as prominent technique. The Linear scalarization method works on the sole premise of transforming a Multiobjective optimization problem into a Single Objective Optimization Problem [9]. Each of the objective is given a specific weight that represents its relative importance, allowing decision-makers to reflect the preferences via the weight allocation. This transformation allows the techniques of a traditional single objective optimization to be used to obtain a solution. As well as streamlining the solution process within a familiar framework.

In the case of solving the Knapsack Problem using the Multiobjective Optimization techniques, the Linear Scalarization Method is applied by formulating the single objective to maximize the total value of the selected items while adhering to the knapsack's capacity constraint. Also, a penalty mechanism is incorporated to make adjustments to the aggregated objective function [3,9]. When the total weight of the selected items exceeds the knapsacks capacity, a penalty is imposed on the solution. This penalty aims to discourage solutions that violate the constraint by penalizing the aggregated objective value.
The penalty value is computed as follows:

$$Penalty = \max\left(0, \sum_{i=1}^{n} w_i . x_i - W\right) \qquad (4.1)$$

Where,

w_i denotes the weight of item i.

x_i denotes the binary decision variable where x_i=1 item is selected, x_i=0 item is not selected in the knapsack

W denotes the Knapsacks capacity.

The modified single objective single objective function encompasses both the weighted sum of the values and the introduced penalty function. The objective is to maximize the total value of the items selected while considering the knapsack capacity constraint and to minimize the penalty.

The objective function is given as:

$$Maximize: \sum_{i=1}^{n} v_i \cdot x_i - Penalty$$

$$Subject\ to: \sum_{i=1}^{n} w_i . x_i \ \leq W$$

Where,

v_i denotes the value of item i.

w_i denotes the weight of item i.

x_i denotes the binary decision variable where x_i=1 item is selected, x_i=0 item is not selected in the knapsack

W denotes the Knapsacks capacity [9].

The addition of the penalty mechanism adds an extra layer of sophistication to the Linear Scalarization method. This technique fosters the finding of solutions that strike a more optimal balance between value maximization and weight constraint by penalizing solutions that violate the knapsack capacity constraint. The penalty mechanism is a useful tool for aligning solutions with real-world feasibility while still accommodating the inherent trade-offs between objectives.

4.2 Weighted Metric Method

The weighted metric method is a well-known technique for making informed decisions by taking into account multiple conflicting objectives at the same time. The Weighted Metric Method is a multiobjective optimization approach that uses a weighted sum to convert multiple objectives into a single aggregated objective [10]. By assigning weights to each objective decision makers set their preferences for different objectives. These weights are relative to the importance assigned to each objective in the decision-making process.

The foundation of the Weighted Metric Method lies in the definition of a reference point that embodies the Utopian solution. The Utopian solution is an ideal state in objective space in which all objectives are satisfied perfectly. The Utopian solution or the Utopian point signifies the pinnacle of achievement in terms of objectives, but it is often unattainable in practice [31].

The distance metric is a weighted deviation-based distance metric between a solution's objective vector and a reference point. It takes the absolute difference between the objective value and the utopian solution multiplied by the weights assigned to the objectives.

To address the capacity constraints violation, a penalty mechanism is implemented. When the overall weight exceeds the knapsack capacity the solution is penalized. The penalty value is determined by the difference of the total weight of the selected items and the weight capacity of the knapsack and it is intended to discourage the solutions that violate constraints [3]. The same penalty is used as (4.1).

The objective function is given as follows:

$$Maximize: weighted\ value\ deviation + weighted\ weight\ deviation + Penalty$$

$$Subject\ to: \sum_{i=1}^{n} w_i . x_i \ \leq W$$

Where,

$$weighted\ value\ deviation =$$

$$|total\ value\ of\ selected\ items - utopian\ point\ for\ value| * weight\ factor$$

$$weighted\ weight\ deviation =$$

$$|total\ weight\ of\ selected\ items - utopian\ point\ for\ weight| * weight\ factor$$

v_i denotes the value of item i.

w_i denotes the weight of item i.

x_i denotes the binary decision variable where x_i=1 item is selected, x_i=0 item is not selected in the knapsack

W denotes the Knapsacks capacity [10].

The incorporation of the penalty mechanism into the Weighted Metric Method increases its applicability and significance in the real world. The strategy compensates for the trade-offs between objectives while recognising the practicality of the solutions in decision making process by assessing alternatives that comply to the capacity limits.

4.3 Chebyshev Scalarization

The Chebyshev Scalarization takes the centre stage in the spectrum of multiobjective optimization techniques applied to the Knapsack Problem. By aiming for the solutions that minimizes the maximum weighted deviation from a reference point along each objective dimension, the Chebyshev Scalarization Method takes a unique approach to multiobjective optimization [11]. This strategy enables decision makers a clear trade-off mechanism and make it easier to generate a diverse selection of Pareto-optimal alternatives.

The Chebyshev Scalarization begins with the definition of a reference point or the Utopian point that represents the desired balance of objectives. Objective weights are important in shaping the direction of optimization because they allow decision makers to express their preferences [31].

The calculation of the Chebyshev scalarization method is the computation of the maximum weighted deviation of each solution from the Utopian point along each objective dimension. This encapsulates the core of trade-offs and illustrates how much each target deviates from its ideal value.

A penalty mechanism is integrated, recognizing the relevance of feasible solutions. Solutions that exceed the knapsack's capacity constraint are penalized in order to keep the optimization process pragmatic. The penalty is calculated as the difference between the total weight of the selected items and the weight capacity of the knapsack [3]. The same penalty is used as in (4.1).

The objective function is given as follows:

$$Minimize: \max(weighted\ value\ deviation, weighted\ weight\ Deviation) + Penalty$$

$$Subject\ to: \sum_{i=1}^{n} w_i . x_i \leq W$$

Where,

$$weighted\ value\ deviation =$$

$$|total\ value\ of\ selected\ items - utopian\ point\ for\ value| * weight\ factor$$

$$weighted\ weight\ deviation =$$

$$|total\ weight\ of\ selected\ items - utopian\ point\ for\ weight| * weight\ factor$$

v_i denotes the value of item i.

w_i denotes the weight of item i.

x_i denotes the binary decision variable where x_i=1 item is selected, x_i=0 item is not selected in the knapsack

W denotes the Knapsacks capacity [11].

With the penalty mechanism included, the Chebyshev Scalarization method emerges as a formidable tool for multiobjective optimization. This method combines the elegance of theoretical foundations with practical considerations, giving decision makers a nuance platform for tackling complicated optimization scenarios.

4.4 TABU search

TABU search is a versatile metaheuristic algorithm that is noted for its ability to explore the solution space and uncover high-quality solutions to optimization problems. It adopts a systematic search method that balances exploration with exploitation, making it idea for handling combinatorial optimization problems like the Knapsack Problem [12, 28].

The Scalarization methods mentioned above are also integrated into the TABU search. For problems like the Knapsack problem scalarization methods aid in the discovery of ideal combination of items that strike a balance between value maximization and weight minimization.

Developing a scalarized objective that converts the multiobjective optimization into a single objective optimization problem.

The scalarized objective is then integrated into the TABU search algorithm to lead the search for optimal solutions. The scalarized objective serves as a fitness function during the TABU search process, directing the selection of candidate solutions to consider. The scalarized objective guides the algorithm's exploration, allowing it to effectively search the solution space while maintaining solution diversity.

The TABU search algorithm iteratively explores the solution space. For each iteration, it generates neighbours by toggling the inclusion/exclusion of individual items in the current solution. It calculates the value and weight of the neighbour and checks if it is feasible. If the neighbour is feasible, has better value, and is not in the TABU list, it becomes the best candidate solution [32].

The TABU list is the memory mechanism in the TABU search. It is critical in maintaining diversity during the search. By limiting revisits to certain solutions, the TABU list encourages the algorithm to explore new regions of the solution space, fostering diversification and reducing early convergence to suboptimal solutions [12].

The current solution Is then updated to the best candidate solution. The current solution is added to the TABU list, ensuring that all solutions visited are recorded. If the TABU list exceeds a particular size, the oldest solutions are deleted.

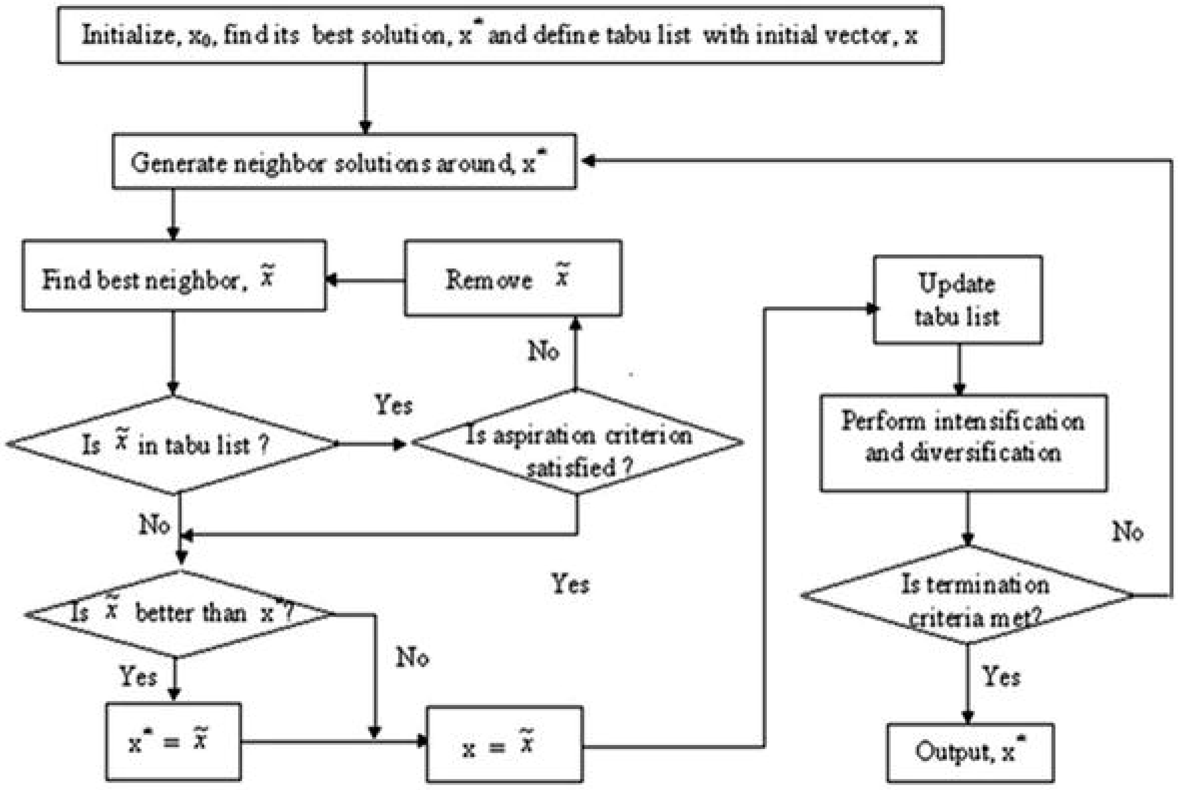

Initialize, x_0, find its best solution, x^* and define tabu list with initial vector, x
Generate neighbor solutions around, x^*
Find best neighbor, $\tilde{x}$
Remove $\tilde{x}$
Is $\tilde{x}$ in tabu list ?
Yes
No
Is aspiration criterion satisfied ?
No
Yes
Is $\tilde{x}$ better than x^*?
Yes
No
$x^* = \tilde{x}$
$x = \tilde{x}$
Update tabu list
Perform intensification and diversification
Is termination criteria met?
No
Yes
Output, x^*

Figure 3: TABU search algorithm Flowchart

The combination of TABU search and multiobjective optimization techniques heralds a comprehensive approach to solving complex optimization challenges withing Knapsack Problem [27]. This approach enables decision-makers to discover a plethora of Pareto Optimal solutions that address real-world feasibility by leveraging TABU search's memory-based exploration, solution variety and aspiration criteria.

5 IMPLEMENTATION AND EXPERIMENTATION

5.1 Implementation

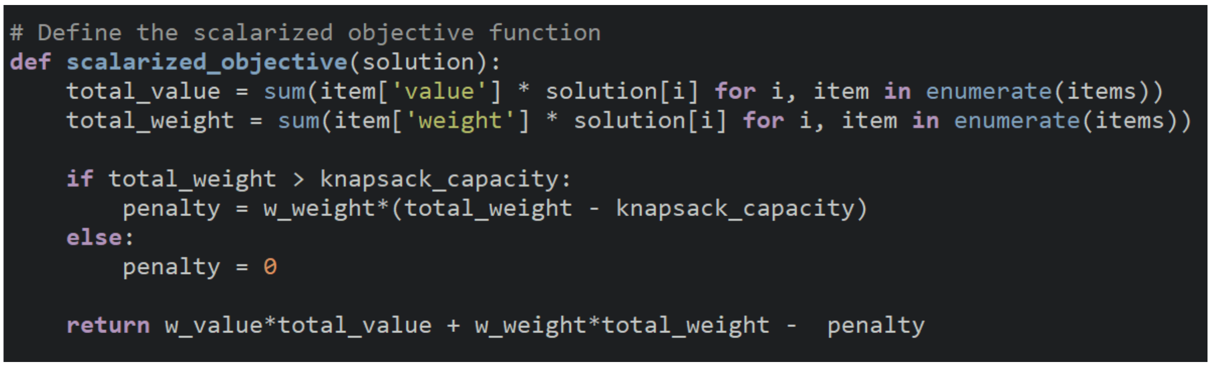
The theoretical concepts and methodological insights discussed in the previous chapter are translated into practical reality through the implementation of multiobjective optimization methods as code.

The full and detailed source code can be accessed through the GitHub repository link provided below:

https://github.com/YashPrajapati25/CS-6500---Final-thesis---Multiobjective-Scalarization-Methods-applied-on-Knapsack-Problem

5.1.1 Linear Scalarization Method Implementation

The Linear Scalarization method is built around specifying the objective weights and constraints. By measuring the relative relevance of each target and ensuring feasibility, these components lead the optimization process. Each of the objective is given a particular weight that helps the decision makers to set their preferences for the respective objective.



```
# Define the scalarized objective function
def scalarized_objective(solution):
    total_value = sum(item['value'] * solution[i] for i, item in enumerate(items))
    total_weight = sum(item['weight'] * solution[i] for i, item in enumerate(items))

    if total_weight > knapsack_capacity:
        penalty = w_weight*(total_weight - knapsack_capacity)
    else:
        penalty = 0

    return w_value*total_value + w_weight*total_weight -  penalty
```

Code Snippet 1: Linear Scalarization Objective Function

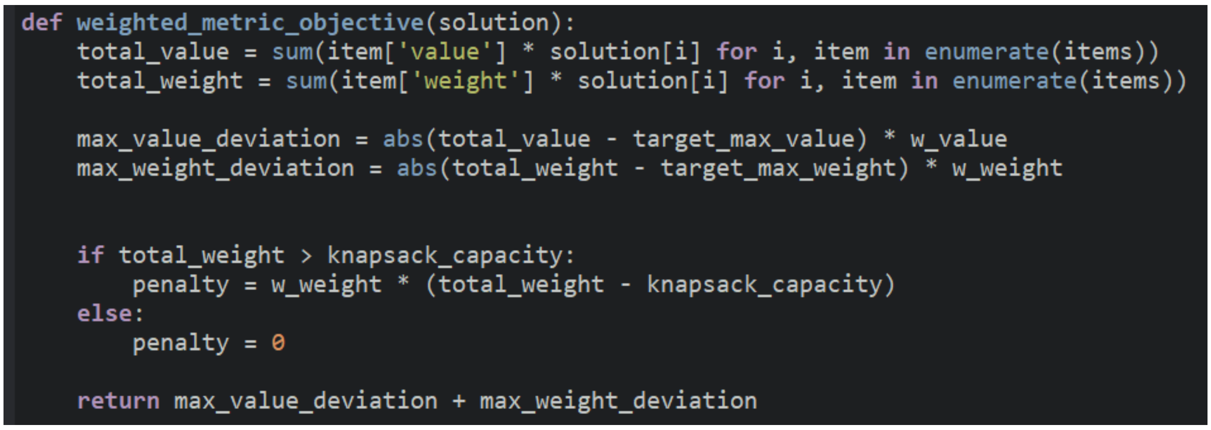
The basis of the linear scalarization method embodies the concept of merging objectives into a single objective. The function computes the total value and total weight of the solutions selected items. If the weight exceeds the capacity of the knapsack, a penalty is calculated and removed from the overall goal. The result is a scalarized objective that takes into account both value maximization and feasibility via the penalty mechanism.

The penalty mechanism within the linear scalarization method is calculated and incorporated when the capacity constraint is violated. This enhancement ensures that the process accounts for real-world feasibility while striving to optimize the objectives.

5.1.2 Weighted Metric Method Implementation

The Weighted metric method begins with the establishment of reference points and objective weights. These elements embody the preferences of decision makers and guide the optimization process by influencing the direction of exploration in the objective space.

The calculation of the distance metrics that measure the deviation of the objective vector from the reference point is a crucial principle of the Weighted metric method. These deviations are weighted according to the assigned weights.



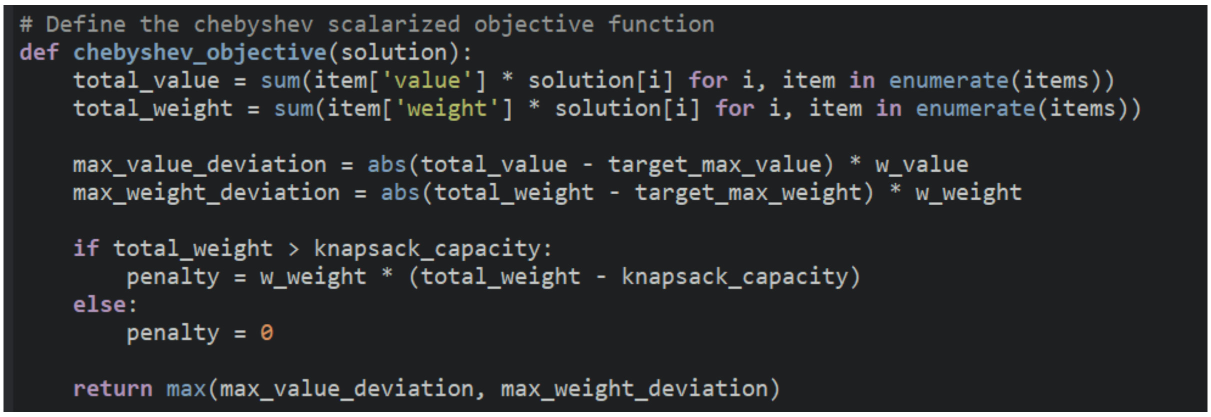
```python
def weighted_metric_objective(solution):
    total_value = sum(item['value'] * solution[i] for i, item in enumerate(items))
    total_weight = sum(item['weight'] * solution[i] for i, item in enumerate(items))

    max_value_deviation = abs(total_value - target_max_value) * w_value
    max_weight_deviation = abs(total_weight - target_max_weight) * w_weight


    if total_weight > knapsack_capacity:
        penalty = w_weight * (total_weight - knapsack_capacity)
    else:
        penalty = 0

    return max_value_deviation + max_weight_deviation
```

Code Snippet 2: Weighted Metric Method Objective Function

The function computes the total value and total weight of the items that are selected. The absolute difference between both the objectives and their respective utopian values is calculated and is multiplied by the weight factor assigned the objectives. The assigned weights enable the decision makers to assign preferences to the one of the objectives over the other. The penalty mechanism is also included for the solution that violate the capacity weight constraint. The feasibility of the solution is ensured by the penalty mechanism and the satisfaction of the objective needs.

5.1.3 Chebyshev Scalarization Method Implementation

The Chebyshev scalarization method commences with defining the reference points and the objective weights. These elements direct the optimization process, shaping trade-offs between conflicting objectives. The determination of the maximum weighted deviations is the distinctive feature of the Chebyshev Scalarization Method.



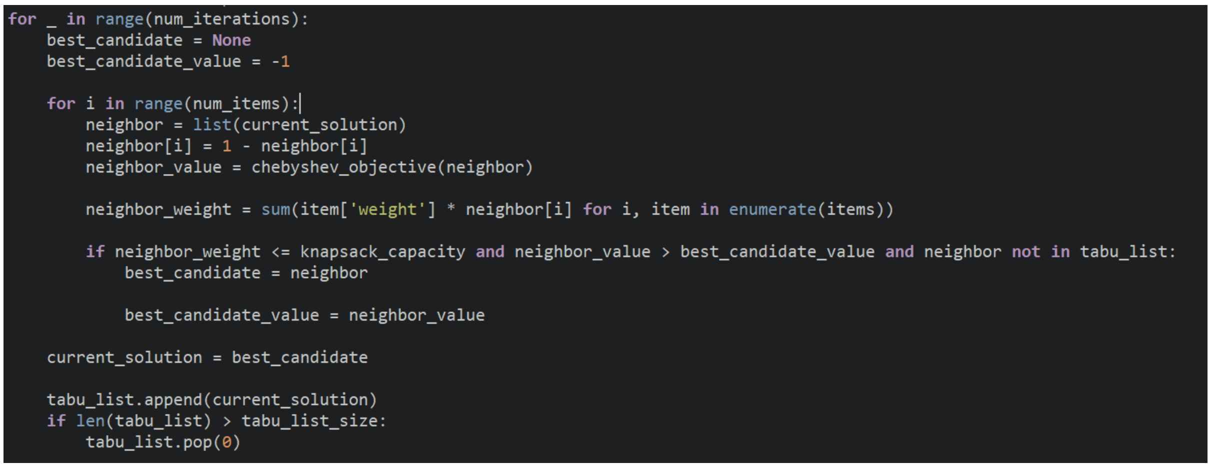
```python
# Define the chebyshev scalarized objective function
def chebyshev_objective(solution):
    total_value = sum(item['value'] * solution[i] for i, item in enumerate(items))
    total_weight = sum(item['weight'] * solution[i] for i, item in enumerate(items))

    max_value_deviation = abs(total_value - target_max_value) * w_value
    max_weight_deviation = abs(total_weight - target_max_weight) * w_weight

    if total_weight > knapsack_capacity:
        penalty = w_weight * (total_weight - knapsack_capacity)
    else:
        penalty = 0

    return max(max_value_deviation, max_weight_deviation)
```

Code Snippet 3: Chebyshev Scalarization Objective Function

The function calculates the total value and total weight of the selected items. The weighted deviations for both the objectives are calculated. A penalty mechanism is integrated to penalize the solutions that exceed the knapsack capacity. Finally, the scalarized objective returns a value that is the maximum of the two weighted deviations.

5.1.4 TABU Search Algorithm Implementation

The essence of TABU search is to generate neighbouring solutions in order to explore the solution space. The TABU list serves as the memory mechanism for the TABU search. Solutions that have recently been investigated are added to this list, while solutions that are already on the list are excluded from consideration. This strategy discourages returning to the same solutions and promotes exploration diversity.



```python
for _ in range(num_iterations):
    best_candidate = None
    best_candidate_value = -1

    for i in range(num_items):
        neighbor = list(current_solution)
        neighbor[i] = 1 - neighbor[i]
        neighbor_value = chebyshev_objective(neighbor)

        neighbor_weight = sum(item['weight'] * neighbor[i] for i, item in enumerate(items))

        if neighbor_weight <= knapsack_capacity and neighbor_value > best_candidate_value and neighbor not in tabu_list:
            best_candidate = neighbor

            best_candidate_value = neighbor_value

    current_solution = best_candidate

    tabu_list.append(current_solution)
    if len(tabu_list) > tabu_list_size:
        tabu_list.pop(0)
```

Code Snippet 4: TABU search Algorithm

The algorithm iterates for the specified number of iterations, wherein each iteration seeks to find an improved solution. Within each iteration, a loop generates neighbouring solutions by toggling the inclusion/exclusion of items in the current solutions. This generates potential solutions that are adjacent in the solution space. The objective function is assessed for each generated neighbour to determine its value, and the total weight of the neighbour is calculated based on the weights of the selected items. The procedure determines whether the generated neighbour is viable (its weight does not exceed the capacity of the knapsack), has a higher value than the current best candidate solution, and is not on the TABU list (memory mechanism). If these conditions are met, the neighbour is elevated to the position of best candidate solution and its value is adjusted correspondingly. The current solution is updated to the best candidate solution obtained in this iteration after assessing all potential neighbours.

5.2 Experimentation

In this key subsection, we enter into the research's practical application and experimentation phase. The major goal is to assess the efficacy of the proposed multiobjective optimization approaches - Linear Scalarization, Weighted Metric, and Chebyshev Scalarization - in solving the difficult Knapsack Problem. We present the approach, algorithmic implementations, and systematic process used to carry out the tests, establishing the groundwork for the thorough analysis that will be presented in later chapters.

5.2.1 Experimental Methodology

A well-defined and standard technique was used across all methodologies to assure the uniformity and credibility of the experimentation process. Because of its powerful libraries and ease of use, Python, a flexible and widely used programming language, was chosen as the implementation platform. This uniform setting eliminated any potential biases that might have arisen from the use of multiple programming languages or tools, confirming the validity of our comparison research.

5.2.2 Phase 1 of Experimentation

The parameters defined while executing the code are as follows:

- ❖ Code parameters:
    - ➢ Number of runs are 100.
        - ▪ This implies that the entire experimentation will run 100 times. Each run represents a unique random generation of items and subsequent execution of the multiobjective optimization method.
- ❖ Item generation parameters:
    - ➢ Range of Value an item can take 1 – 30.
        - ▪ The values of the items generated can take any integer values between 1 to 30. This parameter simulates the variety of the value of items that can be included in the knapsack.
    - ➢ Range of weight of an item is 1 – 20.
        - ▪ The weight of the items generated can take any integer values between 1 to 20. This parameter simulates the variety of the weights of items that can be included in the knapsack.

- Knapsack problem parameters:
  - Knapsack capacity is set at 100 units.
    - This parameter represents that the knapsack can hold up to 100 units, i.e., knapsacks maximum weight capacity.
  - Number of items generated are 30 items.
    - For each run there will be 30 unique items generated which are available for selection.
- TABU Search parameters
  - Number of TABU iterations are 100.
    - The TABU search algorithm is executed for 100 iterations in each run. This parameter defines how many iterations the algorithm will perform in its search for Pareto optimal solutions.
  - TABU list size is 5.
    - This parameter determines how many candidate solutions are stored in TABU list to prevent revisiting the same solution.
- Weight factors for scalarized objective function
  - These weight factors for both value and weight objectives are set to 0.5 for each run. Representing an equal distribution between maximizing value and minimizing the weight.

Overall, these parameters set a stage for a systematic and reproducible experimentation process, allowing for thorough evaluation of the multiobjective optimization methods while considering the variability in items value and weight that could occur in the real-world scenario.

The entire phase 1 of experimentation is run across all three methods, Linear Scalarization method, Weighted Metric Method, and Chebyshev Scalarization method. The same parameters were used for all three objective functions, including the number of runs, item generation parameters, knapsack problem parameters, and TABU search parameters. This consistent setting assures that the observed performance discrepancies between these methods are mostly attributable to changes in the objective functions themselves, allowing for a fair comparison.

5.2.3 Phase 2 of Experimentation

The parameters defined while executing the code are as follows:

- Code parameters:
  - Number of runs are 100.
    - This implies that the entire experimentation will run 100 times. Each run represents a unique random generation of items and subsequent execution of the multiobjective optimization method.
- Item generation parameters:

- Range of Value an item can take 1 – 30.
  - The values of the items generated can take any integer values between 1 to 30. This parameter simulates the variety of the value of items that can be included in the knapsack.
- Range of weight of an item is 1 – 10.
  - The weight of the items generated can take any integer values between 1 to 10. This parameter simulates the variety of the weights of items that can be included in the knapsack.

In phase 2, the range of item weight has been reduced from 1 – 20 to 1 – 10. Reducing the weight range can lead to a narrower distribution of item weights in the knapsack, potentially making it easier to reach the knapsack's weight capacity.

- Knapsack problem parameters:
  - Knapsack capacity is set at 50 units.
    - This parameter represents that the knapsack can hold up to 50 units, i.e., knapsacks maximum weight capacity.
  - Number of items generated are 20 items.
    - For each run there will be 20 unique items generated which are available for selection.

In phase 2, both the knapsack capacity and the number of items generated have been reduced. This change introduces a more constrained knapsack problem, which could potentially help evaluate the robustness of the algorithm.

- TABU Search parameters
  - Number of TABU iterations are 1000.
    - The TABU search algorithm is executed for 1000 iterations in each run. This parameter defines how many iterations the algorithm will perform in its search for Pareto optimal solutions.
  - TABU list size is 3.
    - This parameter determines how many candidate solutions are stored in TABU list to prevent revisiting the same solution.

In phase 2, there is a significant change made to the number of TABU iterations from 100 to 1000, and the TABU list size was reduced from 5 to 3. This change could lead to a longer and more extensive search process, potentially resulting in improved exploration of the solution space.

- Weight factors for scalarized objective function
  - These weight factors for both value and weight objectives are set to 0.5 for each run. Representing an equal distribution between maximizing value and minimizing the weight.

These modifications are intended to study how multiobjective optimization methods perform under various problem constraints and search parameters. The outcomes and insights collected from Phase 2 of the experimentation will be used to determine if they are better or not.

5.2.4 Phase 3 of Experimentation

The parameters defined while executing the code are as follows:

- Code parameters:
  - Number of runs are 100.
    - This implies that the entire experimentation will run 100 times. Each run represents a unique random generation of items and subsequent execution of the multiobjective optimization method.
- Item generation parameters:
  - Range of Value an item can take 1 – 30.
    - The values of the items generated can take any integer values between 1 to 30. This parameter simulates the variety of the value of items that can be included in the knapsack.
  - Range of weight of an item is 1 – 10.
    - The weight of the items generated can take any integer values between 1 to 10. This parameter simulates the variety of the weights of items that can be included in the knapsack.
- Knapsack problem parameters:
  - Knapsack capacity is set at 100 units.
    - This parameter represents that the knapsack can hold up to 100 units, i.e., knapsacks maximum weight capacity.
  - Number of items generated are 30 items.
    - For each run there will be 30 unique items generated which are available for selection.
- TABU Search parameters
  - Number of TABU iterations are 10000.
    - The TABU search algorithm is executed for 10000 iterations in each run. This parameter defines how many iterations the algorithm will perform in its search for Pareto optimal solutions.
  - TABU list size is 3.
    - This parameter determines how many candidate solutions are stored in TABU list to prevent revisiting the same solution.
- Weight factors for scalarized objective function
  - These weight factors for both value and weight objectives are set to 0.5 for each run. Representing an equal distribution between maximizing value and minimizing the weight.

In Phase 3 we only make a few tweaks, more specifically the tweaks that might potentially positively affect the experimentation process and help us converge to a better result. The major change made is increasing the TABU iterations from 1000 to 10000. This makes the algorithm slower it has a significantly longer search time. But this is beneficial for exploring the solution space more extensively, potentially leading to the discovery of better solutions.

Since in phase 2 we ensured the testing of the robustness of the algorithm, we revert the knapsack parameters back to the 100 units and 30 items generated. Finally, the items parameters and tabu list size is also kept same as these were found to be potentially beneficially for the overall algorithm.

In all 3 phases of experimentation the same parameters are kept across all three methods namely, Linear Scalarization method, Weighted Metric Method, and Chebyshev Scalarization method. The changes are made only between the phases of experimentation, in particular phase the parameters are kept same for consistency and comparability between the 3 methods. The changes between the three phases of experimentation and the effect of those changes on the results will be discussed in the following chapter.

6 RESULTS

In chapter 6, we unveil the outcomes of our extensive experimentation which are aimed to assess the effectiveness of the three multiobjective optimization methods in handling the intricate Knapsack Problem: Linear Scalarization, Weighted Metric Method and Chebyshev scalarization. Our thorough investigation, which encompassed two independent periods, provided us with priceless insights into the performance of these approaches under varying constraints and parameters.

6.1 Phase 1 Assessment

In Phase 1 of the experimentation, we kept the same parameters across all three optimization methods. This method created the groundwork for a fair comparison, guaranteeing that any discrepancies in performance could be attributed to changes in the objective function used.

- ❖ Results from Phase 1:
  - ➢ Chebyshev Scalarization emerged as the standout performer, consistently finding the best solutions more that 55% of the time. This result underscores its robustness and ability to identify non-dominated solutions effectively.
  - ➢ Linear Scalarization displayed a variable performance, discovering best solutions between 45% to 60% of the time. This variability suggests that while it is effective, the performance may be sensitive to specific problem instances.
  - ➢ Weighed Metric Method, while not as consistent as the Chebyshev Scalarization method, demonstrated promise by identifying best solutions approximately 35% to 45% of the time.

6.2 Phase 2 Assessment

In Phase 2 of the experimentation, we made a few tweaks to the parameters to assess how these multiobjective optimization methods adapt to different constraints and search parameters. This phase aimed to provide a deeper understanding of their performance under a broader range of scenarios.

- ❖ Results from Phase 2:
  - ➢ Chebyshev Scalarization continued to excel, showing its remarkable adaptability. It consistently found best solutions more than 80% of the time, indicating its suitability for more constrained problem spaces.
  - ➢ Weighted Metric Method saw substantial improvement in Phase 2, identifying best solutions in over 50% of the cases. This implies its capacity to adapt effectively different constraints.

- Linear Scalarization exhibits similar variable performance, with best solutions identified between 40% to 55% of the time. This suggests that its effectiveness still depends on the specifics of the problem instance.

6.3 Phase 3 Assessment

In Phase 3 of the experimentation, we have finally made changes to those parameters that will potentially make the results better. The phase 3 also assess how these multiobjective optimization methods adapt to the increased size of the TABU search parameter. This phase is aimed at gaining a comprehensive understanding of the methods performance.

- Results from Phase 3:
  - Chebyshev Scalarization continues to outperform the other two methods by finding the best solutions more than 83% of the time. This indicates that, under 10000 TABU iterations, a TABU list of size 3, a knapsack capacity of 100 units and 30 generated items, Chebyshev scalarization is highly effective at discovering Pareto optimal solutions. This is due to its ability to explore and balance both value and weight objectives effectively.
  - Weighted Metric Method also performed well in the phase 3 of the experimentation, finding best solutions more than 55% of the time. While it didn’t achieve the same level of success as the Chebyshev scalarization, it demonstrates its effectiveness under the specific conditions.
  - Linear Scalarization found best solutions 45% to 55% of the time. While it did not perform as well as the other two methods in this phase, it is essential to consider that different methods may excel under different problem settings.

The results from our three phased experimentation provide significant insights into the strengths and weaknesses of these multiobjective optimization methods. And beyond the laboratory settings our findings have substantial implications in the real-world decision-making scenarios. Understanding which method performs best under given conditions can help decision-makers in a variety of fields make educated decisions.

During Phase 3 of the experiment, as we began to investigate the performance of multiobjective optimization methods with a larger TABU search size of 10,000 iterations, a noteworthy finding arose. The computational demands of the code increased dramatically, resulting in longer execution durations. It is critical to recognize that the computational intensity was a compromise made in order to investigate the effectiveness of these methods over larger search spaces. While the code's run-time increased, we gained a thorough understanding of how these approaches adapt to increasingly complicated settings. This trade-off between computing time and vast solution space exploration highlights the real-world obstacles that decision-makers may face when applying these optimization strategies to practical issues. Despite the lengthy run-time, the insights gathered from this phase are useful in that they give information on the approaches' adaptability to complex issue contexts.

It is also worth noting that all the three multiobjective optimization methods have found the best solutions. While these solutions might be potentially Pareto optimal, the absence of algorithm like NSGA-II means there is uncertainty regarding their exact optimality. NSGA-II and similar algorithms are specifically designed to explore the entire Pareto front, offering a more comprehensive understanding of the trade-offs between different objectives.

7 CONCLUSION AND FUTURE WORK

In this chapter, we derive results from considerable experiments aimed at evaluating the effectiveness of three multiobjective optimization methods—Linear Scalarization, Weighted Metric Method, and Chebyshev Scalarization—in handling the complex Knapsack Problem. Our investigation was divided into three unique phases, each of which was aimed to investigate the performance of various approaches under varying limitations and characteristics.

As we continue the final part of our thesis, we pause to consider the vital lessons obtained during our experimentation. These insights are not only useful for decision-makers looking for effective approaches to multiobjective problems, but they also set the framework for future optimization research.

7.1 Conclusion

7.1.1 Chebyshev Scalarization Dominance

Chebyshev Scalarization repeatedly established its superiority across all three phases. The ability of this method to find the most number of best solutions under varied constraints and parameters demonstrates its adaptability and dependability. Chebyshev Scalarization may be a viable option for decision-makers faced with difficult multiobjective situations.

Chebyshev scalarization has the property of being able to reach Pareto optimal solutions. By minimizing the maximum deviation from the utopian solution, Chebyshev scalarization can reach more diverse Pareto Optimal solutions [1]. The weighted metric method and the linear scalarization method may only find a subset of the Pareto optimal solutions. Also, in the weighted metric method and the linear scalarization require setting multiple preference weights, Chebyshev scalarization is much easier to use as it only requires setting a reference point. Although Chebyshev scalarization requires to optimize the max function this makes the Chebyshev scalarization more computationally expensive.

7.1.2 Weighted Metric Method’s Adaptability

The Weighted Metric Method demonstrated its adaptability by enhancing performance in Phase 2 and maintaining effectiveness in Phase 3. This adaptability implies that it could be a useful tool when dealing with multiobjective situations with changing constraints.

7.1.3 Linear Scalarization’s Variability

Linear Scalarization displayed variable performance throughout all phases. While it worked well in Phases 1 and 2, its outcomes in Phase 3 were less consistent. Because of this variation, Linear Scalarization may be more suited for specific problem cases rather than broadly applicable scenarios.

Finally, our comprehensive testing has shed light on the multifaceted landscape of multiobjective optimization methods, with each approach presenting a distinct set of advantages and disadvantages. Chebyshev Scalarization has continuously proven to be a reliable and resilient method capable of dealing with a wide range of restrictions. The Weighted Metric Method demonstrated adaptability by enhancing performance when constraints changed. Linear Scalarization, on the other hand, while beneficial in certain cases, produced varying results, indicating its applicability for specific problem instances.

As we end this investigation, it is critical to remember that no single strategy is universally preferable. Instead, the method chosen should align with the specific demands of the problem at hand. Our study has provided useful insights into the actual implementation of these methodologies, thereby empowering decision-makers in a variety of sectors.

7.2 Future Work

7.2.1 Diversifying Metaheuristic Approaches

While our research focuses primarily on the TABU search algorithm, the wide terrain of metaheuristic optimization algorithms provides opportunity for comparison and investigation. Several alternative techniques, including Genetic techniques (GAs), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), have proven to be efficient in diverse optimization contexts.

Combining metaheuristic techniques with genetic algorithms, such as TABU search (GA-TABU), can result in hybrid algorithms with complementary strengths. Exploring these hybrid approaches may give novel answers to the Multiobjective Knapsack Problem.

7.2.2 Real-World Applications

It is critical to broaden the area of study to include real-world applications in order to validate the applicability of multiobjective optimization methods. Several domains provide application opportunities:

- ❖ Project Management: Multi-Objective Knapsack Problems can arise in project management scenarios in which project managers must efficiently deploy resources to satisfy several project objectives, such as minimizing cost and time while maximizing quality.
- ❖ Portfolio Optimization in Finance: Financial portfolio optimization is selecting the appropriate asset mix to achieve several goals, such as maximizing returns while limiting risk. Using multiobjective optimization algorithms to manage portfolios helps improve investing strategies.

- ❖ Supply Chain Optimization: Optimizing resource allocation throughout multiple phases of the supply chain is crucial in supply chain management. Costs, lead times, and other supply chain parameters can all be balanced through multiobjective optimization.

7.2.3 Additional Constraints

Adding constraints to the Multiobjective Knapsack Problem can help to broaden the research landscape. Budget constraints, resource availability, and even environmental factors can all add to the problem's complexity. Investigating how multiobjective optimization methods perform when subjected to certain limitations can yield useful insights.

7.2.4 Multiobjective Evolutionary Algorithms (MOEAs)

MOEAs such as NSGA-II (Non-dominated Sorting Genetic Algorithm II) and SPEA2 (Strength Pareto Evolutionary Algorithm 2) provide alternate ways to multiobjective optimization. When these MOEAs are compared to standard metaheuristic approaches, researchers can better appreciate their relative advantages and limits.

7.2.5 Visualization Techniques

Creating visualization approaches for the Multiobjective Knapsack Problem can help decision-makers grasp the trade-offs between competing goals. Visualization tools can assist consumers make informed selections based on their preferences by providing insights into the Pareto front.

7.2.6 Multidimensional Multiobjective Knapsack Problem

The Multidimensional Multiobjective Knapsack Problem, in which each item has many attributes (dimensions), increases the complexity and applicability to real-world events. Investigating this expanded problem space can lead to new research directions.

In summary, the subject of multiobjective optimization remains vibrant and ripe for further investigation and innovation. The directions mentioned in this chapter serve as a road map for future research efforts aimed at improving understanding and use of multiobjective optimization methods in addressing difficult real-world problems. Multiobjective optimization will play an increasingly important role in defining efficient, sustainable, and informed decision-making processes as academics continue to push the boundaries of knowledge.

References

1 Lust, T., & Teghem, J. (2010). The multiobjective multidimensional knapsack problem: a survey and a new approach. International Transactions in Operational Research.

2 Abdel-Basset, M., El-Shahat, D., Faris, H., & Mirjalili, S. (2019). A binary multi-verse optimizer for 0-1 multidimensional knapsack problems with application in interactive multimedia systems. *Computers & Industrial Engineering*, *132*, 187-206. https://doi.org/10.1016/j.cie.2019.04.025

3 Della Croce, F., Pferschy, U., & Scatamacchia, R. (2019). New exact approaches and approximation results for the Penalized Knapsack Problem. *Discrete Applied Mathematics*, *253*, 122-135. https://doi.org/10.1016/j.dam.2017.11.023

4 Stewart, T. *et al.* (2008). Real-World Applications of Multiobjective Optimization. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds) Multiobjective Optimization. Lecture Notes in Computer Science, vol 5252. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88908-3_11

5 Zakaria, M. Z., Jamaluddin, H., Ahmad, R., & Loghmanian, S. M. (2012). Comparison between multi-objective and single-objective optimization for the modeling of dynamic systems. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*. https://doi.org/10.1177/0959651812439969

6 Della Croce, F., Salassa, F., & Scatamacchia, R. (2017). An exact approach for the 0–1 knapsack problem with setups. *Computers & Operations Research*, *80*, 61-67. https://doi.org/10.1016/j.cor.2016.11.015

7 Pareto efficiency. In *Wikipedia*. https://en.wikipedia.org/wiki/Pareto_efficiency

8 Marler, R.T., & Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization.

9 Scalarizing multiple objective with Linear Scalarization. https://www.supplychaindataanalytics.com/scalarizing-multi-objective-optimization-problems-method-comparison

10 Weighted Metric Method. https://optimization.cbe.cornell.edu/index.php?title=Weighted_metric_method

11 Schmidt, M., Schöbel, A., & Thom, L. (2019). Min-ordering and max-ordering scalarization methods for multi-objective robust optimization. *European Journal of Operational Research*, *275*(2), 446-459. https://doi.org/10.1016/j.ejor.2018.11.048

12 Optimization Techniques – TABU Search. https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25

13 Introduction to Numpy in Python. https://towardsai.net/p/l/introduction-to-numpy-in-python

14 Scientific Python: Using SciPy for Optimization. https://realpython.com/python-scipy-cluster-optimize/

15 Matplotlib: Visualization with Python. https://matplotlib.org/
16 Ehrgott, M., & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum.
17 Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE transactions on Evolutionary Computation.
18 Serafini, P. (1987). Simulated annealing for multi objective optimization problems. TIMS/ORSA Joint National Meeting, St. Louis, Missouri, USA.
19 Schniederjans, M. J. (1995). Goal programming: methodology and applications. Springer Science & Business Media.
20 Gandibleux, X., Mezdaoui, N., & Fréville, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. Advances in multiple objective and goal programming.
21 Doerner, K. F., Gutjahr, W. J., Hartl, R. F., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. Annals of Operations research.
22 Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). Evolutionary algorithms for solving multi-objective problems (2nd ed.). New York: Springer.
23 Angus, D., & Woodward, C. (2009). Multiple objective ant colony optimisation. Swarm Intelligence.
24 Hu, X., Eberhart, R. C., & Shi, Y. (2003). Engineering optimization with particle swarm. In Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE.
25 Tan, K. C., Khor, E. F., Lee, T. H., & F. Colombano, S. (2007). Adaptive specification of chromosomes in genetic algorithms for multiobjective optimization. In International Conference on Adaptive and Natural Computing Algorithms. Springer, Berlin, Heidelberg.
26 Liang, Y., Zhou, J., Kang, L., & Cui, Y. (2016). Adaptive epsilon chebyshev scalarization methods for multiobjective programming. Information Sciences.
27 Gandibleux, X., Mezdaoui, N., & Fréville, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. Advances in multiple objective and goal programming.
28 Loria, J. A. S., Coello Coello, C. A., & Alanis, A. Y. (2003, July). A multiobjective tabu search algorithm for solving the multiobjective 0/1 knapsack problem. In Mexican International Conference on Artificial Intelligence Springer, Berlin, Heidelberg.
29 Yagmahan, B., & Yenisey, M. M. (2010). A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Systems with Applications.
30 Hansen, M. P. (1997). Tabu search for multiobjective optimization: MOTS. In Proceedings of the 13th international conference on multiple criteria decision making Springer, Berlin, Heidelberg.
31 Szparaga, A., Stachnik, M., Czerwińska, E., Kocira, S., Dymkowska-Malesa, M., & Jakubowski, M. (2019). Multi-objective optimization based on the utopian point

method applied to a case study of osmotic dehydration of plums and its storage. *Journal of Food Engineering*, *245*, 104-111. https://doi.org/10.1016/j.jfoodeng.2018.10.014

32 Laguna, M. (2018). Tabu Search. In: Martí, R., Pardalos, P., Resende, M. (eds) Handbook of Heuristics. Springer, Cham. https://doi.org/10.1007/978-3-319-07124-4_24