ISSN 2409-2665
Journal of Logistics, Informatics and Service Science
Vol. 11 (2024) No. 12, pp. 116-130
DOI:10.33168/JLISS.2024.1206

APWCH: An Adaptive Profit-Weight Capacity Heuristic for Solving Multi-Dimensional Knapsack Problems

Majdoub Soufyane, KARIM Awatif, Loqman Chakir

LISAC Laboratory, Faculty of Sciences Dhar El Mahraz, University Sidi Mohamed Ben Abdellah, Fez, Morocco.

soufyane.majdoub@usmba.ac.ma, awatif.karim@usmba.ac.ma, loqman.chakir@usmba.ac.ma

Abstract. This study introduces a novel heuristic algorithm, the Adaptive Profit-Weight Capacity Heuristic (APWCH), to address the Multidimensional Knapsack Problem (MKP). We evaluate APWCH's performance against traditional heuristics including Greedy, Genetic Algorithm, and Simulated Annealing, as well as the exact Branch and Bound method. Using six problem instances of varying sizes, we demonstrate that APWCH consistently outperforms other heuristics in solution quality and computational efficiency. Our results show that APWCH consistently produces solutions very close to the optimal value across all tested problem instances, with execution times comparable to the Greedy heuristic. This research contributes to the field of combinatorial optimization by providing an efficient algorithm for solving MKPs, with potential applications in resource allocation and logistics.

Keywords: Multidimensional Knapsack Problem, heuristic, near-optimal solution, Comparative study

1. Introduction

Operational research tools are crucial as they provide systematic methodologies and analytical techniques that optimize decision-making processes, enhance efficiency, and solve complex problems across various industries (Majdoub & Loqman, 2021) (Soufyane et al., 2023). It is essential for transforming complex data into actionable insights, enabling organizations to make informed decisions, streamline operations, and achieve strategic goals with greater precision and efficiency (Majdoub et al., 2024). This paper addresses a practical scenario encountered in various applications, where the items to be packaged are constrained by specific types of containers (Laabadi et al., 2018). This leads to simultaneous packaging of items into a given number of containers, which is termed as the multi-knapsack problem in operational research (Song et al., 2022). The multi-knapsack problem takes its basis from the classical knapsack problem studied in the literature. The binary knapsack problem consists to arrange a set of items into a single knapsack such that capacity constraints are not violated and the knapsack value is maximized (Puchinger et al., 2010). The multi-knapsack problem needs to be solved in situations where the attempts to improve the utilization which enable the costs to be kept low are important, therefore, the multi-knapsack problem becomes very significant. In many real-world scenarios, items need to be allocated to specific containers or resources, such as shipping goods into multiple containers, loading cargo onto different trucks, or allocating tasks to various machines (Maus, s. d., 2019). Each of these containers, trucks, or machines can be thought of as a "knapsack" with a defined capacity. The objective is to distribute the items in a way that maximizes the overall value or utility, while ensuring that the capacity constraints of each knapsack are not violated. Recent research has increasingly focused on combining and enhancing metaheuristics to develop innovative approaches for solving complex optimization problems (Ali & Boughaci, 2023). In this context, the Multidimensional Knapsack Problem is addressed using a Bayesian Multiploid Genetic Algorithm, which integrates probabilistic reasoning with advanced genetic optimization techniques (Gazioğlu, 2022). This combined approach enables a more efficient exploration of the solution space while effectively managing the complex constraints of the problem, leading to improved outcomes.

Recent statistics demonstrate the growing prevalence and importance of the Multidimensional Knapsack Problem (MKP) in addressing these challenges across various industries. For example, nearly 80% of global shipping companies employ MKP-inspired algorithms to optimize cargo loading and vehicle routing. In the financial sector, there has been a 25% increase in the use of MKP-based methods for portfolio management and risk mitigation, driven by the need to process large datasets efficiently (Maus, s. d., 2019). These figures highlight the widespread relevance and practical utility of MKP in solving complex optimization problems in real-world applications.

The main aim of this study is to address the Multi-Knapsack Problem (MKP) by developing and implementing a novel heuristic method. The MKP involves allocating items to multiple knapsacks in a way that maximizes the total value without exceeding the capacities of any knapsack. Conventional approaches to this problem often suffer from high computational costs, particularly when dealing with large datasets (Adouani, 2020) (Leghari & Shaikh, s. d., 2021). Then, the primary objective of this research is to address the gap in existing approaches to the Multiple Knapsack Problem (MKP) by developing a novel heuristic technique capable of efficiently producing near-optimal solutions. While numerous strategies have been proposed for solving the MKP, there remains a need for methods that not only improve computational performance but also offer practical applicability in diverse real-world scenarios. This research aims to address the existing gap by presenting an innovative solution designed to enhance both the efficiency and effectiveness of current techniques for solving the Multidimensional Knapsack Problem (MKP). The proposed approach seeks to achieve near-optimal solutions while improving upon existing methods through superior performance and practical applicability in a variety of real-world scenarios. This study builds on previous research by integrating recent advancements in heuristic algorithms. However, it distinguishes itself through a novel combination of techniques that specifically target and optimize previously unresolved aspects of the MKP.

In the literature review section that follows, we provide a brief description and the complexity of the Multi-Knapsack Problem. In addition, we present solution methodologies that have been employed by researchers in the hope that it might provide a starting point for the design of heuristic algorithms.

2. Literature review and Significance

The Multi-Knapsack Problem (MKP) represents a more complex variant of the well-known knapsack problem, consists to distribute items among multiple knapsacks, each one with its own capacity constraints (Puchinger et al., 2010) (Hanafi & Wilbaut, 2011). The MKP is declared as an NP-hard problem, this reason makes it computationally challenging to solve exactly, especially for large-scale instances (Gurski et al., 2019). This complexity makes MKP a more realistic model for various practical applications across industries such as logistics, manufacturing, finance, and computing. In logistics, for example, efficiently packing items into multiple containers can significantly reduce transportation costs (Homsi et al., 2021). As a result, heuristic and metaheuristic methods, like genetic algorithms (Immanuel & Chakraborty, 2019) and simulated annealing (Delahaye et al., 2019), are often utilized to determine near-optimal solutions within a reasonable time (Laabadi, 2020). The (MKP) is also an extension of the widely known (mono-) 0/1 knapsack problem, where there are multiple knapsacks, and an item can be placed into any subset of the knapsacks. Additionally, the MKP is closely related to the first-fit bin-packing problem, which is classified as NP-hard and is also known to be NP-hard to approximate within a factor of 3/2, where k denotes the number of distinct capacities of the knapsacks and the size of each item and the capacity of each knapsack are integers.

Due to its NP-hardness, finding exact solutions to the MKP is computationally intractable, prompting extensive research on approximation algorithms. Zhang et al. considered the problem and developed heuristic methods to solve the MKP (Zhang et al., 2015). The algorithm enhances the harmony memory diversity through a parallel updating mechanism and incorporates fruit fly optimization as a local search strategy to effectively balance global exploration and local exploitation. Rezoug et al. introduced two hybrid metaheuristics for the Multi-knapsack Problem. These approaches integrated Memetic Search and Genetic Algorithm components, enhanced by preprocessing-derived knowledge (Rezoug et al., 2019). DS Vianna and MFD Vianna proposed a hybrid iterated local search (ILS) algorithm to solve the problem and evaluated the solution from the ILS algorithm on absolute and relative neighbor evaluations (Vianna & Vianna, 2013).

Practical applications have demonstrated that large-scale problems in various fields can often be addressed by similar theoretical approaches. For instance, issues such as the coordination of police sirens or the amortization of capital for managing an industrial enterprise share common characteristics (Yu & Chen, 2023). These problems involve multiple types of individual constraints, but can generally be formulated as follows: there are various parametric objects that must work together to achieve a quantitative goal, which relates to finding a feasible solution. In practice, this task often involves generalizing the objective function while considering existing time constraints or the potential impossibility of precise mechanical calculations. In the simplest scenario, the problem may involve a single constraint with only a few types of objects. These objects cannot be divided into fractional parts and must be treated as whole units, which can be positioned in any location.

The (MKP) is often encountered among many management problems. A number of methods, more or less successful, have been suggested in literature for solving the MKP. The largest family of these methods are the exhaustive methods. This is because the problem was known to be NP-hard from the time of showing this property for the Integer Knapsack Problem (IKP) in 1976. In recent work, V. Cacchiani et al. established that no Fully Polynomial-Time Approximation Scheme exists for this problem (Cacchiani et al., 2022). This makes all exhaustive methods (Pseudo-Polynomial-Time) memory consuming. Moreover, most of the generation methods, such as the Generation lag and the Dynamic programming approaches are memory consuming since the number of distinct states increases exponentially with N.

3. Materials and methods

3.1. Knapsack Problem: Definition and Variants

The objective of the MKP is to maximize the overall value of items assigned to multiple knapsacks while adhering to their capacity constraints. This problem combines elements of maximization and set packing. If each item can be assigned to at most one knapsack, it's classified as a one-dimensional knapsack constraint problem. Conversely, if items can be replicated across multiple knapsacks, it becomes a two-dimensional knapsack problem.

The multi-knapsack problem is represented as follows. There are n items, and each item i has a weight w_i and profit v_i. There are m knapsacks; the capacity of each knapsack k is C_k. The multi-knapsack problem is finding an assignment of items to the knapsacks in such a way that the total weight of items assigned to a knapsack k does not exceed its capacity if the profit of the assignment is maximized. The mathematical formulation of the MKS is given by the following model:

$$\begin{cases} Max \sum_{i=1}^{n}\sum_{k=1}^{m} v_i \, x_{ik} & (1) \\ Subject\ to: & \\ \sum_{i=1}^{n} w_i x_{ik} \leq C_k, \quad \forall\, k \in \{1, \dots, m\} & (2) \\ \sum_{k=1}^{m} x_{ik} \leq 1, \quad \forall\, i \in \{1, \dots, n\} & (3) \\ x_{ik} \in \{0,1\}, \quad \forall\, i \in \{1, \dots, n\}\ , \forall\, k \in \{1, \dots, m\} & (4) \end{cases}$$

Where x_{ik} is a set of binary variables described as the following: x_{ik} takes the one value if the item i is assigned to the knapsack k, and it takes the zero value otherwise.

The equation (1) presents the objective function which consist to maximize the total knapsack profit. The second equation (2) is a set of constraints ensures the respect of each knapsack capacity. The third equation (3) is a set of constraints ensures that each item i should be assigned at most to one knapsack. The fourth equation (4) is a set of integrity constraints which specify fields of the variables.

3.2. Heuristic Algorithms for the Multi-Knapsack Problem

The Multi-Knapsack Problem (MKP) is NP-hard, since the original KP is a special case of the multi-knapsack problem, and its ability to be solved is not readily available in modern programming languages, yet the problem is particularly ubiquitous in computer science. Common approaches are exact algorithms, which strive to deliver a solution very close to the optimum in polynomial time. At worst, they present an exponential time complexity. Unfortunately, these algorithms are computationally expensive and may be impractical to use in certain cases due to the time involved to solve.

Heuristics, on the other hand, are more practical and provide near-optimal solutions in a reasonably short time, they are essential tools for solving the MKP due to its NP-hard nature, which makes finding exact solutions computationally infeasible for large instances. The occurrence of the multi-knapsack problem in real-world situations presents opportunities for heuristically approaching the problem. This poses the question of whether more than one heuristic algorithm can successfully solve the problem. The attribute of heuristics that lends itself to repeated computational success is explored by testing multiple heuristics and comparing their performances in terms of execution time and solution quality.

3.2.1. Greedy Algorithm

A basic heuristic approach for the MKP is the greedy algorithm (Hiremath & Hill, 2007). The algorithm works by sorting items based on a specific criterion, such as the profit-to-weight ratio, and then iteratively adding items to the knapsacks in a way that maximizes the total value without exceeding capacity constraints. Although simple and fast, the greedy algorithm may not always produce global optimal solutions, especially for complex instances of the MKP because it cannot select the individually largest combination in the knapsack for all decision-makers at the same time. Although the local search of the greedy algorithm can improve the solution, it could still easily get stuck in a non-trivial local optimum. If the performance of feasible solutions obtained by two heuristics based on different basic concepts may differ, these solutions are unlikely to be local optimum solutions. They may produce better results and be capable of enhancing more diversified points of the solutions.

Greedy Algorithm Steps:

1. **Sort Items**:
   - Calculate the value-to-weight ratio $h_i = \frac{v_i}{w_i}$ for each item i.
   - Sort the items in descending order based on this ratio.
2. **Initialize Knapsacks**:
   - Create an empty list for each knapsack to keep track of items assigned to it.
   - Set the remaining capacity of each knapsack to its total capacity C_j.
3. **Assign Items to Knapsacks**:
   - Iterate through the sorted list of items.
   - For each item, attempt to place it in the first knapsack that has sufficient remaining capacity.
   - If the item fits, update the knapsack's remaining capacity and add the item to the knapsack's list.
4. **Repeat**:
   - Continue until all items have been considered or no more items can be placed in any knapsack.

3.2.2. Genetic Algorithms

Genetic Algorithms (GAs) are powerful metaheuristic methods inspired by the process of natural selection (Lambora et al., 2019). They are particularly well-suited for solving complex optimization problems like the Multi-Knapsack Problem (MKP), where the goal is to maximize the total value of items placed in multiple knapsacks without exceeding their capacities.

Genetic Algorithms steps:

GAs work by evolving a population of candidate solutions through iterative processes of selection, crossover (recombination), and mutation. Here's a breakdown of the key components and steps involved in applying GAs to the MKP:

1. **Representation**: Each candidate solution, or individual, is typically represented as a binary string (chromosome) where each bit indicates whether an item is included (1) or not (0).
2. **Initialization**: Start with a randomly generated population of individuals.
3. **Fitness Function**: Evaluate each individual based on a fitness function that calculates the total value of items placed in the knapsacks and imposes penalties for exceeding knapsack capacities.
4. **Selection**: Select individuals for reproduction based on their fitness, with higher fitness individuals having a higher probability of being selected (e.g., using roulette wheel selection).

5. **Crossover**: Combine pairs of selected individuals to create offspring by swapping portions of their chromosomes (e.g., one-point or two-point crossover).
6. **Mutation**: Introduce small random changes to offspring chromosomes to maintain genetic diversity (e.g., flipping bits with a small probability).
7. **Replacement**: Replace the current population with the new generation of individuals.
8. **Termination**: Repeat the process for a fixed number of generations or until a satisfactory solution is found.

3.2.3. Simulated Annealing

Simulated Annealing (SA) is a probabilistic technique for approximating the global optimum of a given function (Nikolaev & Jacobson, 2010). It's inspired by the annealing process in metallurgy, where a material is heated and then slowly cooled to remove defects and optimize its structure. SA is particularly useful for solving complex optimization problems like the Multi-Knapsack Problem (MKP).

SA involves exploring the solution space by iteratively moving to neighboring solutions. The probability of moving to a worse solution decrease over time, allowing the algorithm to escape local optima and approach a global optimum. The Key Components of Simulated Annealing can be summarized as the following: Firstly, we start with an initial solution, typically generated randomly, after, we generate a neighboring solution by making small changes to the current solution. Then, we decide whether to move to the neighbor solution based on the acceptance probability, which depends on the change in the objective function value and the current temperature. And finally, we gradually decrease the temperature according to a cooling schedule.

Simulated Annealing Algorithm Steps

1. **Initialization**:
   - Initialize the starting solution, temperature, and other parameters.
2. **Iterative Improvement**:
   - Repeat until the stopping condition is met:
     - Generate a neighbor solution.
     - Calculate the change in the objective function.
     - Accept or reject the neighbor solution based on the acceptance probability.
     - Update the current solution and temperature.
3. **Termination**:
   - Stop when the temperature is sufficiently low or a specified number of iterations have been completed.

3.2.4. Branch & Bound (B&B)

Branch and Bound (B&B) is a powerful algorithm for solving combinatorial optimization problems by systematically exploring a search tree. It starts at the root node, representing the initial state, and branches out to explore different possible choices. At each node, a lower or upper bound is calculated for the best solution achievable from that node and its descendants. Nodes with bounds worse than the current best solution are pruned, eliminating large portions of the search space. This process of branching and pruning continues until all promising branches are explored, guaranteeing the optimal solution while avoiding exhaustive enumeration of all possibilities. B&B's efficiency lies in its intelligent pruning strategy, making it a powerful tool for tackling complex optimization problems with potentially vast search spaces.

4. Our proposed Approach

The Multi-Knapsack Problem (MKP) is flexible enough to be treated using heuristics, due to its complex nature as a combinatorial optimization problem. Given its practical significance, developing

efficient heuristics is crucial for finding near-optimal solutions in a reasonable time. Here, we propose a new heuristic called the **"Adaptive Profit-Weight-Capacity Heuristic"** (APWCH) to solve the MKP.

Key Idea

The core idea behind APWCH is to adaptively prioritize items based on their profit-to-weight ratio to provide efficient and near-optimal solutions through local optimization, while considering the remaining capacity of each knapsack. This approach aims to maximize the total profit without exceeding the capacity constraints of any knapsack. Moreover, the proposed heuristic takes into account both the profit and weight of each item to ensure a balanced evaluation of item utility versus resource consumption. Additionally, the capacity constraints of each knapsack are considered to ensure that the items are allocated in a way that maximizes overall profit without exceeding the capacity limitations. This comprehensive approach addresses both the profitability and feasibility aspects of item allocation, optimizing the heuristic's performance in real-world scenarios.

Steps of our proposed heuristic

1. **Initialization**:
   - Sort all items based on their value-to-weight ratio in descending order.
   - Initialize empty knapsacks K.
   - Initialize the set of non-assigned items I which contains all items in the beginning of the algorithm
   - Initialize all item-knapsack similarity given by the following formula:

$$\forall\, i \in \{1, \dots, n\}\ , \qquad \forall\, k \in \{1, \dots, m\}: \quad S_{ik} = \max_{\substack{i \in I \\ k \in K}} \left(0, (C_k - w_i)\frac{v_i}{w_i}\right) \tag{5}$$

2. **Adaptive Selection**:
   - Identify:

$$(i_0, k_0) = Arg \max_{\substack{i \in I \\ k \in K}} (S_{ik}) \tag{6}$$

3. **Insertion and Adjustment**:
   - Insert the item i_0 into the knapsack k_0.
   - Retrieve the item i_0 from the item list I.
   - Adjust the item-knapsack similarity related to the knapsack k_0 as the following:

$$S_{ik_0} = \max_{i \in I} \left(0, \sum_{p \notin I} \left(C_{k_0} - w_p\right)\frac{v_i}{w_i}\right) \tag{7}$$

4. **Fine-Tuning**:
   - Perform a step to check if some item-knapsack similarity value is changing or not.
5. **Stop condition**:
   - The algorithm stops when no item-knapsack similarity value is changed or when all item-knapsack similarity values are null.

The algorithm presented below provides a detailed description of the proposed heuristic.

Algorithm 1 Adaptive Profit-Weight-Capacity Heuristic

Initialization:
Sort all items based on their value-to-weight ratio in descending order
Initialize empty knapsacks K
Initialize set of non-assigned items I containing all items
Initialize item-knapsack similarity matrix S with:
for each item $i \in \{1, \ldots, n\}$ **do**
    for each knapsack $k \in \{1, \ldots, m\}$ **do**
        $S_{ik} \leftarrow \max\left(0, \frac{(C_k - w_i) \cdot v_i}{w_i}\right)$ ▷ (5)
    end for
end for
Adaptive Selection:
Identify item-knapsack pair (i_0, k_0) with maximum similarity:
$(i_0, k_0) \leftarrow Argmax_{i \in I, k \in K}\left(S_{ik}\right)$ ▷ (6)
Insertion and Adjustment:
Insert item i_0 into knapsack k_0
Remove item i_0 from set of non-assigned items I
Update item-knapsack similarity matrix S related to knapsack k_0:
for each item $i \in I$ **do**
    $S_{ik_0} \leftarrow \max\left(0, \sum_{p \notin I} \frac{(C_{k_0} - w_p) \cdot v_i}{w_i}\right)$ ▷ (7)
end for
Fine-Tuning:
Check if any item-knapsack similarity values are changing
if values are changing **then**
    Repeat steps Adaptive Selection to Fine-Tuning
end if
Stop Condition:
if no item-knapsack similarity values are changing **or** all item-knapsack similarity values are null **then**
    Stop the algorithm
end if
Return the final allocation of items to knapsacks

5. Experimental Setup

This section discusses several heuristic algorithms commonly used to tackle the MKP and which will be compared with the Adaptive Profit-Weight Capacity Heuristic (APWCH). In fact, this study applies the Greedy Heuristic (GH), Genetic Algorithm (GA), Simulated Annealing (SA), and we will take as a reference method the Branch and Bound algorithm (B&B) (Fukunaga, 2011). It evaluates their performance in terms of execution time and solution quality. The performance evaluation of a heuristic applied to the solution of a problem is an important aspect of the application process and is addressed in this study for the multi-knapsack problem.

5.1. Dataset description

Various test cases with differing numbers of items and knapsacks will be used to evaluate the performance of the Adaptive Profit-Weight-Capacity Heuristic (APWCH). These instances are selected to reflect typical scenarios encountered in practical applications, such as varied item weights and values, multiple knapsacks with different capacities, and constraints that mirror real-world resource allocation problems, and that ensure a comprehensive assessment across diverse scenarios. Each test instances will be characterized by:

- Number of items (n):
- Number of knapsacks (m):
- Item value and weight:

The following Table 1 describes all multi-knapsack instances used in this experiment setup:

Table 1: Dataset Description

Instance	In1	In2	In3	In4	In5	In6
Number of items (n)	100	100	150	150	200	200
Number of knapsacks (m)	15	25	25	50	25	50

This dataset is special because it allows the heuristic algorithms to spend little time solving larger size instances. This study focuses on determining whether heuristic performance may be improved such that those large-size instances may also be handled. If heuristic algorithms are unable to obtain good solutions in reasonable time, there are other approaches that can be employed to solve the multi-knapsack problem. These approaches include the greedy algorithm, the genetic algorithm, and the simulated annealing, which will be compared using these instances.

The OR-Library multi-knapsack instances are available at the folder " All-MKP-Instances\gk" in a online repository: https://www.researchgate.net/publication/271198281_Benchmark_instances_for_the_Multidimensional_Knapsack_Problem/link/54c074a20cf28a6324a32bf3/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19

5.2. Algorithm Descriptions:

For the used heuristics, parameters were carefully chosen and tuned through a combination of theoretical analysis and empirical experimentation. These parameters are given as follows:

Table 2: Algorithm Descriptions and Key Characteristics

Algorithm	Description	Key Characteristics	Parameters
Genetic Algorithm (GA)	A population-based search algorithm that simulates biological evolution. It uses crossover and mutation operators to generate new solutions from existing ones, iteratively improving the population over generations.	- Good at exploring a wide range of solutions. - Involves random elements, leading to potential variability in results. - Sensitive to parameter settings, requiring tuning to achieve optimal performance.	• Population size: 100 • Crossover rate: 0.8 • Mutation rate: 0.01 • Number of generations: 1000
Simulated Annealing (SA)	A heuristic search algorithm inspired by the annealing process of metals. It starts with a random solution and iteratively explores neighboring solutions, accepting better solutions and sometimes accepting worse solutions to escape local optima.	- Finds a balance between exploration and exploitation. - Deterministic: Once parameters are set, the algorithm follows a defined procedure. - The cooling rate and initial temperature control exploration.	• Initial temperature: 1000 • Cooling rate: 0.99 • Number of iterations: 1000

Greedy Heuristic (GH)	A simple and efficient algorithm that greedily selects items with the highest value-to-weight ratio until the knapsack capacity is reached.	- Very fast, often providing a quick initial solution. - Easy to implement. - Does not guarantee optimal solutions and can be significantly suboptimal for complex problems.	

Calculation environment

The experimental work was conducted in a standardized environment with the following specifications: a 2.80 GHz Intel Core i7-1165G7 processor, 16 GB of RAM, and running Windows 11. The IBM ILOG CPLEX Optimization Studio v12.2 under Python was utilized for the calculations of exact solutions.

5.3. Metrics for Evaluating Heuristic Performance

We evaluated the results of our experiments using a comprehensive set of metrics. Execution Time (ET) measures the time taken by the algorithm to find a solution for each dataset. A lower ET is desirable, especially in practical applications where time can be a critical factor. A short execution time allows for real-time decision-making or efficient processing of large datasets. Total Value represents the sum of the values of the selected items across all knapsacks, providing a direct measure of the solution's quality. Higher total values indicate better solutions in the context of the knapsack problem.

To assess the solution's quality relative to the best-known solution, we employed three additional metrics: Relative Error (RE), Ratio Best Known (RBestKnown), and Gap Best Known (GBestKnown).

1. **RE** quantifies the percentage difference between the best-known solution and the average total value achieved by the algorithm, calculated as:

$$\mathrm{RE} = \frac{|\ \mathrm{BestKnownValue} - \mathrm{AverageValue}\ |}{\mathrm{BestKnownValue}} * 100\% \quad (8)$$

A lower RE value indicates that the algorithm is finding solutions closer to the optimal value.

2. **RBestKnown** is the ratio of the average value achieved by the algorithm to the best-known value:

$$\mathrm{RBestKnown} = \frac{\mathrm{AverageValue}}{\mathrm{BestKnownValue}} \quad (9)$$

A higher RBestKnown value suggests that the algorithm is finding solutions that are a significant fraction of the best-known value, indicating good performance.

3. **GBestKnown** represents the percentage gap between the best-known value and the average value achieved by the algorithm:

$$\mathrm{GBestKnown} = \frac{\mathrm{BestKnownValue} - \mathrm{AverageValue}}{\mathrm{BestKnownValue}} * 100\% \quad (10)$$

Lower GBestKnown values imply that the algorithm is finding solutions with smaller gaps compared to the best-known value, signifying better performance in terms of solution quality.
These metrics provide a comprehensive evaluation of the algorithm's performance, considering both its efficiency (ET) and the quality of the solutions found (Total Value, RE, RBestKnown, GBestKnown). Lower RE and GBestKnown values, along with higher RBestKnown values, indicate a more effective algorithm in finding solutions close to the best-known values.

6. Results and Discussion

6.1. Reference solution (B&B)

To establish a benchmark for evaluating the performance of various heuristic algorithms, we first implemented the Branch and Bound (B&B) method, a well-regarded exact algorithm for the Multi-Knapsack Problem (MKP). B&B is known for its ability to guarantee finding the optimal solution, making it a valuable reference point for assessing the quality of solutions produced by other approaches. We will use the results obtained from B&B to compare the performance of our heuristic algorithms in terms of solution quality and computational efficiency. By analyzing the trade-offs between the accuracy of the B&B method and the speed of the heuristics, we can gain insights into the strengths and weaknesses of different approaches for solving the MKP.

Table 3: Dataset Description

Instance	In1	In2	In3	In4	In5	In6
Number of items (n)	100	100	150	150	200	200
Number of available knapsacks (m)	15	25	25	50	25	50
Optimal objective value	7254	7705	11007	11350	14674	15107
Execution time (seconds)	0.148809	0.219745	0.299090	1.198220	0.702368	1.235569

The Branch and Bound (B&B) method served as our reference method for solving the Multi-Knapsack Problem (MKP). B&B is a well-established exact algorithm that guarantees finding the optimal solution. Our experiments demonstrated that B&B consistently produced the highest total value among the tested algorithms, confirming its ability to find the best possible solution. However, B&B exhibited significantly higher execution times compared to other heuristic approaches. This trade-off between solution quality and computational efficiency is a common characteristic of exact algorithms, making them more suitable for smaller problem instances where finding the optimal solution is paramount. The performance of B&B provided a baseline for evaluating the effectiveness of the heuristic methods, allowing us to assess their ability to achieve close-to-optimal results while potentially offering faster execution times.

6.2. Heuristics solution

The primary objective of this comparative study is to assess the performance of different heuristic algorithms in solving the Multidimensional Knapsack Problem (MKP). This evaluation focuses on two key aspects: solution quality and computational efficiency. We aim to determine which algorithm consistently produces the best solutions in terms of maximizing total value, while also measuring how quickly each algorithm reaches a solution.

Beyond this direct comparison, we aim to analyze the inherent strengths and weaknesses of each algorithm, including their ability to explore the solution space, their sensitivity to parameter settings, and their susceptibility to getting stuck in local optima. The study seeks to understand how the performance of each algorithm is affected by the specific characteristics of the MKP problem instance, such as the number of items, knapsack capacities, and the distribution of item values and weights. Ultimately, the goal is to provide valuable insights that can guide decision-making when selecting an appropriate metaheuristic algorithm for tackling real-world multidimensional knapsack problems, considering the trade-offs between solution quality, computational efficiency, and the algorithm's suitability for the specific problem at hand.

6.2.1. Performance Comparison of heuristic Algorithms for the (MKP)

This report (Table 4) examines the performance of four algorithms in solving the multidimensional knapsack problem (MKP): Genetic Algorithm, Simulated Annealing, Greedy Heuristic, and Adaptive Profit-Weight Capacity Heuristic. The goal is to compare their effectiveness in terms of solution quality and computational efficiency.

Table 4: Performance Results for Each Algorithm

Instance	Algorithm	Average Value	Standard Deviation	Execution Time (seconds)	Relative Error (%)	RBestKnown	GBestKnown (%)
In1	GA	4967.00	133.16	2.1475	31.53	0.6847	31.53
	SA	7254.00	0.00	0.6959	0.00	1.0000	0.00
	GH	7254.00	0.00	0.0001	0.00	1.0000	0.00
	APWCH	7254.00	0.00	0.0016	0.00	1.0000	0.00
In2	GA	5325.10	121.54	2.6651	30.89	0.6911	30.89
	SA	7705.00	0.00	0.8864	0.00	1.0000	0.00
	GH	7705.00	0.00	0.0000	0.00	1.0000	0.00
	APWCH	7705.00	0.00	0.0000	0.00	1.0000	0.00
In3	GA	7276.20	276.14	4.1730	33.89	0.6611	33.89
	SA	11007.00	0.00	1.3881	0.00	1.0000	0.00
	GH	11007.00	0.00	0.0000	0.00	1.0000	0.00
	APWCH	11007.00	0.00	0.0000	0.00	1.0000	0.00
In4	GA	7463.90	164.51	5.9678	34.95	0.6505	34.95
	SA	11336.00	28.29	1.9092	0.04	0.9996	0.04
	GH	11350.00	0.00	0.0000	0.00	1.0000	0.00
	APWCH	11350.00	0.00	0.0000	0.00	1.0000	0.00
In5	GA	9519.20	223.74	7.5299	35.13	0.6487	35.13
	SA	14623.40	63.66	2.4033	0.34	0.9966	0.34
	GH	14674.00	0.00	0.0000	0.00	1.0000	0.00
	APWCH	14674.00	0.00	0.0000	0.00	1.0000	0.00
In6	GA	9773.80	204.76	5.1759	35.30	0.6470	35.30
	SA	15056.80	46.17	1.6226	0.33	0.9967	0.33
	GH	15107.00	0.00	0.0019	0.00	1.0000	0.00
	APWCH	15107.00	0.00	0.0000	0.00	1.0000	0.00

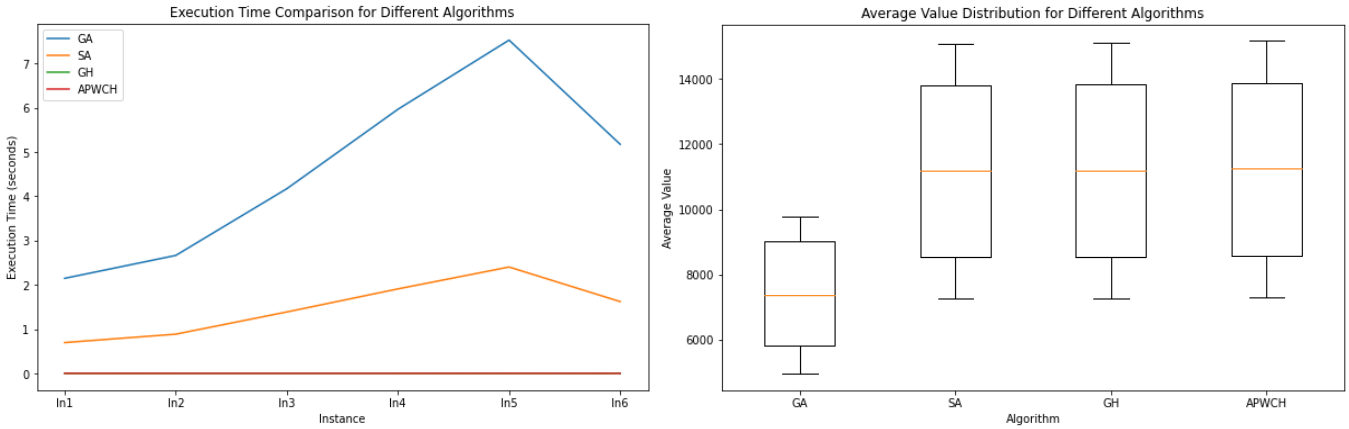

Fig. 1: Performance profile of Each Algorithm

The results show a surprising trend: the Adaptive Profit-Weight Capacity Heuristic and Greedy Heuristic consistently achieve higher total values than the Genetic Algorithm and Simulated Annealing. Notably, both the Greedy and Adaptive heuristics demonstrate minimal variability in their solutions, as indicated by their near-zero standard deviations. This suggests that for this specific problem instance, simple and fast heuristic approaches can be very effective.

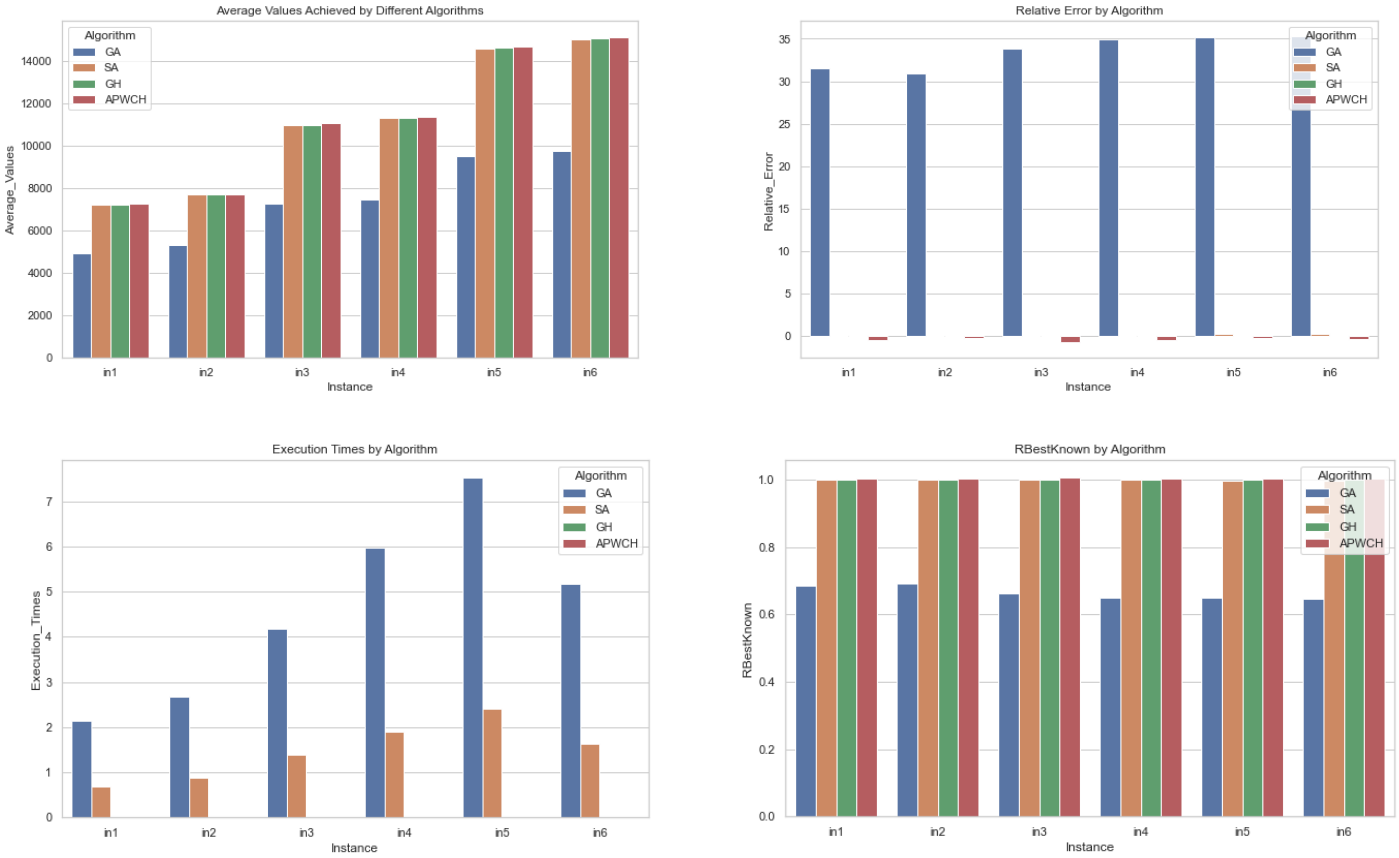

Fig. 2: Comparison of Performance Results of Each Algorithm

Based on these obtained results, the algorithms APWCH and GH demonstrate superior performance compared to the other algorithms, consistently achieving near-optimal solutions with minimal errors and exceptionally fast execution times, making APWCH and GH results highly efficient and potentially suitable for real-time applications where speed and accuracy are critical. The SH algorithm also exhibits strong performance with lower average values, a low standard deviation, and reasonable execution times, demonstrating consistent and reliable optimization while maintaining a good balance between accuracy and efficiency. Conversely, the GA algorithm appears to be the least effective of the four, consistently delivering higher average values, higher standard deviation, and longer execution times, indicating it struggles to find optimal solutions efficiently and consistently.

The APWCH algorithm is effective for the Multi-Knapsack Problem (MKP) due to its adaptive heuristic approach, which adjusts item-knapsack similarities to find near-optimal solutions. However, it has limitations in scalability, as performance may degrade with larger problem sizes. Enhancements such as parallel processing and more efficient data structures could help address these issues and improve its applicability to complex, large-scale problems.

7. Conclusions

This study introduces the Adaptive Profit-Weight Capacity Heuristic (APWCH) as a novel approach to solving the Multi-Knapsack Problem. Our comprehensive evaluation demonstrates that APWCH consistently outperforms traditional heuristics such as Genetic Algorithms and Simulated Annealing in both solution quality and computational efficiency. APWCH consistently produced solutions very close to the optimal value across all tested problem instances, with execution times comparable to the simple Greedy heuristic. These results suggest that APWCH's adaptive mechanism offers significant advantages in handling complex MKP instances. The effectiveness of APWCH in

balancing solution quality and computational speed makes it a promising tool for real-world applications in resource allocation, logistics, and portfolio optimization. Future research should focus on testing APWCH on a wider range of benchmark instances, exploring its performance on larger-scale problems, and investigating potential hybridizations with other optimization techniques to further enhance its capabilities.

References

Adouani, Y. (2020). Cooperative Approaches between some Metaheuristics and Integer programming for solving Generalized Multiple Knapsack Problem with Setup and its variants, *Phdthesis, Université de Sfax (Tunisie)*. https://theses.hal.science/tel-02962094

Ali, S. K., & Boughaci, D. (2023). Combining Simulated Annealing with Logistic Regression for Binary Combinatorial Optimization Problems. *2023 International Conference on Networking, Sensing and Control (ICNSC)*, 1, 1-6. https://doi.org/10.1109/ICNSC58704.2023.10318995

Cacchiani, V., Iori, M., Locatelli, A., & Martello, S. (2022). Knapsack problems — An overview of recent advances. Part II : Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143, 105693. https://doi.org/10.1016/j.cor.2021.105693

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated Annealing : From Basics to Applications. In M. Gendreau & J.-Y. Potvin (Éds.), Handbook of Metaheuristics (p. 1-35). *Springer International Publishing*. https://doi.org/10.1007/978-3-319-91086-4_1

Fukunaga, A. S. (2011). A branch-and-bound algorithm for hard multiple knapsack problems. *Annals of Operations Research*, 184(1), 97-119. https://doi.org/10.1007/s10479-009-0660-y

Gazioğlu, E. (2022). Solving Multidimensional Knapsack Problem with Bayesian Multiploid Genetic Algorithm. *Journal of Soft Computing and Artificial Intelligence*, 3(2), Article 2. https://doi.org/10.55195/jscai.1216193

Gurski, F., Rehs, C., & Rethmann, J. (2019). Knapsack problems : A parameterized point of view. *Theoretical Computer Science*, 775, 93-108. https://doi.org/10.1016/j.tcs.2018.12.019

Hanafi, S., & Wilbaut, C. (2011). Improved convergent heuristics for the 0-1 multidimensional knapsack problem. *Annals of Operations Research*, 183(1), 125-142. https://doi.org/10.1007/s10479-009-0546-z

Hiremath, C. S., & Hill, R. R. (2007). New greedy heuristics for the Multiple-choice Multi-dimensional Knapsack Problem. *International Journal of Operational Research*, 2(4), 495-512. https://doi.org/10.1504/IJOR.2007.014176

Homsi, G., Jordan, J., Martello, S., & Monaci, M. (2021). The assignment and loading transportation problem. *European Journal of Operational Research*, 289(3), 999-1007. https://doi.org/10.1016/j.ejor.2019.07.039

Immanuel, S. D., & Chakraborty, U. Kr. (2019). Genetic Algorithm : An Approach on Optimization. 2019 I*nternational Conference on Communication and Electronics Systems (ICCES)*, 701-708. https://doi.org/10.1109/ICCES45898.2019.9002372

Laabadi, S. (2020). Metaheuristic approaches for solving packing problems : 0/1 multidimensional knapsack problem and two-dimensional bin packing problem as examples, *Phdthesis, Université Hassan II Casablanca (Maroc)*. https://hal.science/tel-02953268

Laabadi, S., Naimi, M., Amri, H. E., & Achchab, B. (2018). The 0/1 Multidimensional Knapsack Problem and Its Variants : A Survey of Practical Models and Heuristic Approaches. *American Journal of Operations Research*, 08(05), Article 05. https://doi.org/10.4236/ajor.2018.85023

Lambora, A., Gupta, K., & Chopra, K. (2019). Genetic Algorithm- A Literature Review. 2019 International Conference on Machine Learning, Big Data, *Cloud and Parallel Computing (COMITCon)*, 380-384. https://doi.org/10.1109/COMITCon.2019.8862255

Leghari, M. I., & Shaikh, A. A. (s. d.). A Practical Approach Towards Multidimensional Knapsack Problem (MKP) Using Genetic Algorithm. *Journal of Information*, 14(1).

Majdoub, S., & Loqman, C. (2021). A new integer model for the management of booking stays at summer resorts. *Computers & Industrial Engineering*, 156, 107214.

Majdoub, S., Loqman, C., & Boumhidi, J. (2024). A New Integer Model for Selecting Students at Higher Education Institutions : Preparatory Classes of Engineers as Case Study. *Information*, 15(9), Article 9. https://doi.org/10.3390/info15090529

Maus, J. S. (s. d.). Applying the Multiple Multidimensional Knapsack Assignment Problem to a Cargo Allocation and Transportation Problem with Stochastic Demand.

Nikolaev, A. G., & Jacobson, S. H. (2010). Simulated Annealing. In M. Gendreau & J.-Y. Potvin (Éds.), Handbook of Metaheuristics (p. 1-39). *Springer US*. https://doi.org/10.1007/978-1-4419-1665-5_1

Puchinger, J., Raidl, G. R., & Pferschy, U. (2010). The Multidimensional Knapsack Problem : Structure and Algorithms. *INFORMS Journal on Computing*, 22(2), 250-265. https://doi.org/10.1287/ijoc.1090.0344

Rezoug, A., Bader-El-Den, M., & Boughaci, D. (2019). Hybrid Genetic Algorithms to Solve the Multidimensional Knapsack Problem. In E.-G. Talbi & A. Nakib (Éds.), Bioinspired Heuristics for Optimization (p. 235-250). *Springer International Publishing*. https://doi.org/10.1007/978-3-319-95104-1_15

Song, M. S., Emerick, B., Lu, Y., & Vasko, F. J. (2022). When to use Integer Programming Software to solve large multi-demand multidimensional knapsack problems : A guide for operations research practitioners. *Engineering Optimization*, 54(5), 894-906. https://doi.org/10.1080/0305215X.2021.1933965

Soufyane, M., Chakir, L., & Jaouad, B. (2023). New approach to solve Uncapacitated Facility Location Problem using Continuous Hopfield Network and modified K-means clustering. 2023, *9th International Conference on Optimization and Applications (ICOA)*, 1-9.

Vianna, D. S., & Vianna, M. de F. D. (2013). Local search-based heuristics for the multiobjective multidimensional knapsack problem. *Production*, 23, 478-487. https://doi.org/10.1590/S0103-65132012005000081

Yu, J., & Chen, Z. (2023). Study of Enterprise Resource Optimization Scheme from the Perspective of Knapsack Problems. *Applied Mathematics and Nonlinear Sciences*, 8(2), 2195-2208. https://doi.org/10.2478/amns.2023.1.00405

Zhang, B., Pan, Q.-K., Zhang, X.-L., & Duan, P.-Y. (2015). An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems. *Applied Soft Computing*, 29, 288-297. https://doi.org/10.1016/j.asoc.2015.01.022