**Algorithmic Approaches to Solving Knapsack Problem**

**Abstract:**

The knapsack problem is a classic optimization challenge where the objective is to maximize the total value of items packed into a knapsack without exceeding its weight capacity It comes in several variants, including the 0–1 Knapsack Problem (0-1KP), the Multidimensional Knapsack Problem (MDKP), and the Quadratic Knapsack Problem (QKP). This Review paper conducts a detailed exploration and analysis of algorithmic strategies developed for solving the knapsack problem (KP). The paper delves into various algorithmic approaches, including advanced dynamic programming, heuristic and metaheuristic algorithms like genetic algorithms and simulated annealing. The goal is to provide a comprehensive comparison and evaluation of these diverse algorithmic approaches, examining their performance, efficiency, and applicability in various real-world scenarios. By highlighting the strengths, weaknesses, and recent developments in knapsack problem-solving algorithms, this review aims to guide future research and help practitioners make informed choices.

1. Introduction:

The knapsack problem (KP) is a foundational and well-researched subject in combinatorial optimization due to its numerous real-world applications, its theoretical relevance, and many exact and approximate algorithms have been developed to solve it [1]. The 0-1 knapsack problem (0-1KP) is the traditional KP. Furthermore, numerous other knapsack problems are extensions of the 0-1 KP. These include the discount 0-1 knapsack problem (D{0-1}KP), the multidimensional knapsack problem (MDKP), the quadratic knapsack problem (QKP), the set union knapsack problem (SUKP), the bounded knapsack problem (BKP), and many others[2]. Various real-world decision-making procedures utilize the KPs. The problems of capital budgeting allocation, real estate property maintenance, freight loading, resource allocation, project selection, combinatorial auctions, available-to-promise, and cutting stock are a few instances of KPs' practical applications[3]. Knapsack problems hold significance due to their capacity to model various real-world situations and scenarios. The knapsack problem finds practical applications in scenarios such as collective decision-making among individuals choosing items to carry, optimizing the deployment of sensors in a network using a multiple knapsack model, and addressing challenges related to streaming layered video content over peer-to-peer networks[1].

The knapsack problem, a classic optimization challenge, has prompted the development of various algorithmic approaches, each offering unique strategies to efficiently allocate resources. These approaches showcase a nuanced array of methodologies, considering factors such as computational complexity, optimality guarantees, and problem scalability.

This paper explores diverse algorithmic strategies developed to address the Knapsack Problem. We begin with an examination of advanced dynamic programming techniques, which have evolved significantly since their inception, offering more efficient solutions to the classic 0/1 Knapsack Problem [4]. These techniques have been refined to address complexities and variations in modern optimization scenarios.

In the realm of heuristic and metaheuristic algorithms, genetic algorithms and simulated annealing stand out for their effectiveness in solving complex Knapsack Problems, as highlighted in[5]. Genetic algorithms excel in exploring vast solution spaces through evolutionary-inspired processes, making them ideal for finding efficient combinations in the Knapsack Problem. Simulated annealing, with its methodical exploration and gradual focus, effectively avoids local optima, which is crucial for complex, high-dimensional problems. Both approaches, while not always yield optimal results, provide high-quality solutions efficiently, demonstrating their significance in modern optimization challenges.

This review paper has a central objective: to conduct a thorough comparison and evaluation of diverse algorithmic approaches employed for solving the knapsack problem. By analyzing the performance and efficiency of these algorithms, we aim to provide insights into their respective strengths and weaknesses, helping researchers and practitioners make informed choices based on factors such as time complexity, solution quality, and scalability. Additionally, the review will highlight recent developments in knapsack problem-solving algorithms, contributing to the ongoing discourse on optimization strategies and guiding future research endeavors in this domain.

This review paper is structured as follows: Section 2 provides an overview of the knapsack problem, including its variations and algorithms for solving it. Section 3 presents related work in the field. Finally, the paper concludes in Section 4 with a summary of the review.

2. Background:

2.1 Knapsack Problem:

The knapsack problem is a classic combinatorial optimization challenge characterized as NP-hard. In this problem, there are 'n' items, each denoted by 'i' (1 ≤ i ≤ n), with associated profits (pi), weights (wi), and a fixed knapsack capacity (C). The primary objective is to determine a subset of items represented by a binary vector 'x' ∈ {0, 1}n, where an item 'i' is included (xi = 1) or excluded (xi = 0). The goal is to maximize the total profit (P(x)) of the selected items while ensuring that their combined weight (W(x)) does not exceed the given knapsack capacity[6].

The two main types of the knapsack problem are:

* **0/1 Knapsack Problem:**

The 0/1 Knapsack Problem exemplifies a combinatorial optimization challenge where the task is to identify the optimal solution from myriad possibilities. The primary objective is to maximize the cumulative value of items accommodated within a knapsack, subject to the constraint that the sum of their weights does not exceed the knapsack's capacity. This problem type adheres to a binary decision framework, allowing each item to be either included in the knapsack or excluded. Notably, each item is unique, with no duplications allowed, and there are only two choices for each item: inclusion or exclusion. Moreover, the constraint prohibits the partial inclusion of items in the knapsack, ensuring a discrete and non-divisible approach to the optimization problem[3].

* **Fractional Knapsack Problem:**

The Fractional Knapsack Problem (FKP) is a type of combinatorial optimization where the primary objective is to maximize the total value of items placed in a knapsack, given a constraint on the sum of their weights not exceeding the knapsack's capacity. Unlike the 0/1 Knapsack Problem, FKP allows items to be divided into fractions. Each item is unique, and the decision for each item is binary: either include it or exclude it. However, the decision variable is a real number, enabling fractional inclusion in the knapsack and providing a more flexible optimization approach compared to the discrete nature of the 0/1 Knapsack Problem[7].

2.2 Algorithmic Approaches:

1. **Dynamic Programming (DP):** stands as a classical and powerful technique employed to tackle intricate problems. The fundamental principle of DP involves breaking down a complex problem into more manageable and smaller subproblems. Each of these subproblems is then solved individually, with their solutions meticulously stored in a table or an array. This approach of solving subproblems just once and storing their solutions is a hallmark of DP, enabling the avoidance of redundant computations and significantly enhancing the efficiency of solving larger and more intricate instances of the original problem. In essence, DP leverages this systematic breakdown and solution storage mechanism to efficiently navigate through the intricacies of complex problem spaces[4].
2. **Greedy Algorithms:** Whether consciously or unconsciously applied, the Greedy algorithm emerges as a frequently utilized strategy in everyday life. The overarching goal is consistently directed toward maximizing utility. This algorithm is particularly apt for scenarios where the objective function is presented with the implicit assumption that the variable with the highest coefficient correlates with the highest profit. Consequently, the Greedy algorithm prioritizes assigning the highest value to this specific variable. In instances of single-constrained problems, the algorithm calculates utility per unit and strategically aims to fill its capacity. It achieves this by favoring the selection of the variable that promises the highest benefit. In essence, the Greedy algorithm operates with a focus on immediate gains, selecting options that offer the most significant value under the prevailing constraints, aligning with the overarching objective of maximizing utility[5].
3. **Branch and Bound:** The Branch and Bound algorithm is a comprehensive approach to addressing challenging problems directly. This method takes a holistic view of the problem space, systematically exploring various branches to find optimal solutions. In cases where certain branches do not yield values that contribute to a solution, the algorithm intelligently disregards them, focusing its efforts on branches that hold meaningful values. Even if only a single branch provides valuable evaluations, the algorithm prioritizes solving those branches, thereby streamlining the solution-finding process by strategically ignoring less promising avenues[8].
4. **Genetic Algorithms:** The Genetic Algorithm (GA) serves as a heuristic search and optimization algorithm, drawing inspiration from natural evolution [45]. The GA initiates its process with a set of potential solutions, referred to as chromosomes, constituting the population. The algorithm then generates a new population derived from the solutions of the existing population, aiming for an improved set of solutions. Selection of solutions for the creation of new offspring is contingent upon their fitness; solutions deemed more suitable have a higher probability of reproduction. This cyclic process continues iteratively until a specified condition is met. Key components integral to the GA include the encoding of chromosomes, the fitness function that evaluates solution effectiveness, the selection process, recombination techniques, and the overarching evolution scheme that governs the algorithm's progression[3].
5. **Integer Linear Programming (ILP):** In certain problem-solving scenarios utilizing Linear Programming (LP), there is a preference or necessity to derive integer results. For instance, decisions like whether to include a course in a curriculum or not can be effectively represented by decision variables taking binary values, such as 1 for selection and 0 for non-selection. Similarly, in manufacturing, the quantity of products to be produced often makes more sense when expressed as whole numbers rather than fractions. Recognizing the limitations of fractional values, an Integer Programming (IP) method has been specifically designed for such problem instances. This approach ensures that decision variables, which could signify critical choices or quantities, are constrained to integer values. This is particularly crucial in situations where fractional values, such as 3.6 computers, lack practical significance or feasibility. By introducing the concept of integer programming, this method enhances the applicability of linear programming to scenarios where decisions and quantities naturally align with whole numbers[5].

**3. Literature Review:**

Recently, some research has been conducted into various new approaches to solve different kinds and applications of the knapsack problem. Most of the efforts have gone into enhancing the dynamic programming algorithms and intelligence-based methods for better efficiency with lower computational complexity.

In this context, Della Croce et al. [9] presented an exact method, based on dynamic programming, to solve the Penalized Knapsack Problem (PKP). Such an approach was based on a core algorithm using upper bound tests to manage variables as effectively as possible and outperformed commercial solvers, such as CPLEX 12.5, solving instances with up to 10,000 items. Similarly, Schulze et al. [10] introduced a polynomial-time approximation scheme for the Rectangular Knapsack Problem (RKP), achieving a fixed approximation ratio of 4.5 and yielding improved results in practical applications.

Some studies focus on hybrid and heuristic approaches. Paulauskas and Jakubauskas [11] propose a hybrid approach in which the dynamic programming is combined with metaheuristic methods and achieves better performance on multi-objective knapsack problems. Most hybrid approaches have balanced solution quality and computational efficiency for large-scaled and complicated problem instances. Zhu and Yan [12] introduced a type of Artificial Bee Colony (ABC) optimization, especially to address multidimensional knapsack problems with an improved performance compared to classic evolutionary algorithms.

Furthermore, some researchers have applied their theories to practical uses and specific case studies in order to further validate the robustness of these algorithms. For instance, Xu et al. [13] have tried a modified branch-and-bound approach for solving real-time logistics optimization problems and dealt with large-scale constraints successfully. Chen et al. [14]designed a knapsack-based resource allocation framework in cloud computing with improved greedy algorithms for optimizing cost and resource utilization.

Wei & Hao [15] proposed an advanced threshold search-based memetic algorithm, TSBMA, tailored for the DCKP problem. The memetic algorithm combined evolutionary global search with local improvement to achieve an effective refinement of the solution. The algorithm was a binary vector-based item selection algorithm, while the constraints were respected by the problem. Similarly, Khan et al.[16] addressed a variant of the knapsack problem by introducing a (2 + ε)-approximation algorithm for a multidimensional knapsack problem with geometric and vector constraints. This approach also involved packing rectangular items, characterized by profit and weights, into a square knapsack without violating the constraints. Alongside, Gálvez et al. [17] developed a (4/3 + ε)-approximation algorithm for the 2-Dimensional Knapsack problem (2DK), which worked in polynomial time, considering both weighted and unweighted variants of 2DK with or without 90-degree rotations. Both of these studies targeted tackling complicated knapsack problems with the help of efficient polynomial-time approximation algorithms; hence, their capability regarding multi-dimensional and multi-variant constraints could be demonstrated.

Yuan & Li [18] also proposed the greedy approach, G4, toward solving MOKP by transforming an infeasible solution into a feasible solution and, within the knapsack capacity constraints, maximizing the quality of a feasible solution. This strategy is considered as showing much better enhancements in the solution of MOKP compared to other previously conventional approaches, similar to how Coniglio et al. [19], in a combinatorial branch-and-bound algorithm, addressed a Knapsack Problem with Conflicts, KPC. Both studies focused on refining existing methods to solve complex knapsack problems efficiently.

In a similar vein, Sánchez-Díaz et al.[20] presented a hyper-heuristic model for the Knapsack Problem (KP), leveraging an evolutionary-powered hyper-heuristic framework that combined multiple packing heuristics. The paper's approach agreed with that of He & Wang's[21] in introducing GTOA, which integrated algebraic group operations into the evolutionary process for solving several variants of the knapsack problem. Both studies showed that the algorithm each had proposed was adaptive and powerful to solve knapsack problems efficiently.

Along the same vein, Moradi et al.[22] developed a population-based simulated annealing algorithm for the 0-1 knapsack problem (KP01), which enhanced the traditional simulated annealing methodology with population-based features and iterative refinement to navigate the solution space effectively. This approach, like Chebakov & Serebryanaya's [23] multicriteria model for solving the knapsack problem, sought to optimize the selection process through innovative strategies and models.

Also, Kwon et al. [24] proposed a method for solving the BLKP using a Graph Neural Network, which utilized the PNA method to predict the solution of the leader and reformulated the bilevel problem as a single-level optimization problem. It demonstrated large gains in efficiency, similar to the methods reviewed above, illustrating the possibilities of modern optimization techniques in solving real-world knapsack problems.

Laalaoui & Mhalla [25]introduced the Sharknap algorithm to address the Knapsack Sharing Problem (KSP). Specifically, they utilized a Dichotomous-based exact method that decomposes KSP into smaller 0/1 Knapsack Problems (KP). Moreover, they implemented a "critical class" concept to optimize solver efficiency and scalability.

Similarly, Sur et al. [26] proposed a Deep Reinforcement Learning (DRL)-based approach for the Multiple Knapsack Problem (MKP), which uses a Markov Decision Process and a capacity-aware selection strategy. In particular, their algorithm, implemented with the Asynchronous Advantage Actor-Critic (A3C) method, highlighted the role of machine learning in optimization.

In addition, Bramantya et al.[27] developed a genetic algorithm for the 2D knapsack problem, employing Crossover, Mutation, and Selection to evolve populations. Consequently, their approach emphasized efficiency and adaptability in addressing real-world optimization challenges.

Furthermore, Lamanna et al. [28] introduced Two-phase Iterative Kernel Search (TIKS) for the Multidimensional Multiple-choice Knapsack Problem (MMKP). Specifically, the framework divided the solution process into feasibility- and quality-oriented phases, thereby demonstrating a structured approach to complex problem-solving.

On the other hand, Aïder et al.[29] addressed the quadratic multiple knapsack problem (QMKP) with a branch-and-solve strategy. To achieve this, they combined fix-and-solve techniques with local branching methods, effectively balancing solution exploration and refinement.

Likewise, Feng & Wang [30] presented a Binary Moth Search Algorithm with Self-Learning (SLMS) for the 0–1 Multidimensional Knapsack Problem (MKP). Notably, the algorithm incorporated a self-learning mechanism to enhance search capabilities and optimize population diversity.

Finally, GAZİOĞLU [31] introduced the Bayesian Multiploid Genetic Algorithm (BMGA) for the Multidimensional Knapsack Problem (MKP). Importantly, the method integrated Bayesian Optimization with genetic algorithms, offering a sophisticated approach to managing multidimensional constraints.

Ghadi et al.[32] introduced the Group Counseling Optimizer (GCO), a novel algorithm inspired by human group counseling dynamics, to solve the 0/1 Knapsack Problem. By leveraging evolutionary strategies, GCO offers a practical and efficient alternative to conventional dynamic programming methods.

Similarly, Jovanovic & Voß [33] proposed a Greedy Randomized Adaptive Search Procedure (GRASP) for the Quadratic Knapsack Problem (QKP), enhanced with an innovative local search algorithm. This approach expanded solution exploration while maintaining computational efficiency, demonstrating its utility in tackling complex optimization tasks.

Hertrich & Skutella [34] employed Recurrent Neural Networks (RNNs) with rectified linear units (ReLUs) to address the NP-hard Knapsack Problem. Their method, inspired by dynamic programming formulations, highlighted the potential of neural networks in solving combinatorial optimization challenges.

Khandekar & Nargundkar [35]applied a dynamic programming (DP) algorithm to optimize food order selection, conceptualized as a multi-objective unbounded knapsack problem. This approach effectively addressed real-world resource allocation scenarios, emphasizing the versatility of DP in handling multiple objectives.

Abdel-Basset et al. [36] developed the Binary Kepler Optimization Algorithm (BKOA) and its enhanced variant (HBKOA) to tackle the 0–1 Knapsack Problem (KP01). Their work emphasized improved exploration and exploitation capabilities, showcasing the algorithm’s adaptability to high-dimensional optimization challenges.

Finally, Wei et al.[37] introduced the Responsive Strategic Oscillation Algorithm (RSOA) to address the Disjunctively Constrained Knapsack Problem (DCKP). By integrating feasible and infeasible searches, RSOA demonstrated its applicability in practical domains such as scheduling and resource allocation.

**Table 1. Summary of Literature Review Papers**

| Author | Year | Knapsack Problem Type | Algorithm | Results / Run Time |
| --- | --- | --- | --- | --- |
| Della Croce et al. | 2019 | 0–1 Penalized Knapsack Problem | Dynamic Programming (DP2) | Superior to CPLEX 12.5; up to 10,000 items |
| Schulze et al. | 2020 | Rectangular Knapsack Problem | Polynomial time algorithm | 4.5 approximation ratio; O(n log n) complexity |
| Ali et al. | 2020 | 0/1 Knapsack Problem | Greedy and Dynamic Programming | Greedy: O(nlogn), DP: O(nC) |
| Shahbandegan & Naderi | 2020 | Multidimensional Knapsack Problem | Binary Butterfly Optimization Algorithm | Highly competitive; specific run times not mentioned |
| Abdel-Basset et al. | 2021 | 0–1 Knapsack Problem | Binary Equilibrium Optimization | Notable performance, especially in small- and medium-scale instances |
| Benford | 2021 | Knapsack Problem | Reinforcement Learning Algorithms | DDQN as the standout; specific run times not mentioned |
| Wei & Hao | 2021 | Disjunctively Constrained Knapsack Problem | Threshold Search-Based Memetic Algorithm (TSBMA) | Superior performance; initialization O(n³) |
| Khan et al | 2021 | Multidimensional Knapsack Problem | (2 + ε)-approximation algorithm | Polynomial time; efficiency in complex scenarios |
| Gálvez et al | 2021 | 2-Dimensional Knapsack Problem | (4/3 + ε)-approximation algorithm | Polynomial time; improved partitioning technique |
| Yuan & Li | 2021 | Multi-objective Knapsack Problem | Greedy Strategy G4 | Improved quality of solutions; slightly higher runtime for S-MOKPs |
| Coniglio et al. | 2021 | Knapsack Problem with Conflicts | Combinatorial branch-and-bound algorithm | Faster by up to two orders of magnitude than best method |
| Sánchez-Díaz et al. | 2021 | Knapsack Problem | Hyper-heuristic model | Effective in both balanced and unbalanced instances |
| He & Wang | 2021 | Knapsack Problems | Group Theory-based Optimization Algorithm (GTOA) | Outperformed traditional evolutionary algorithms; varying complexities |
| Moradi et al. | 2022 | 0-1 Knapsack Problem | Population-based Simulated Annealing (PSA) | Outperformed existing SA-based and other meta-heuristic solvers |
| Chebakov & Serebryanaya | 2022 | Knapsack Problem | Multicriteria Model | Optimized total weight selection; specific run times not mentioned |
| Kwon et al. | 2022 | Bilevel Knapsack Problem | Graph Neural Network-based method | 500 times faster than traditional; 1.7% optimality gap |
| Laalaoui & Mhalla | 2022 | Knapsack Sharing Problem | Sharknap | Solved instances up to 100,000 items and 1,000 classes in <1 second |
| Sur et al. | 2022 | Multiple Knapsack Problem | Deep Reinforcement Learning | Superior to random and greedy methods; specific run times not mentioned |
| Bramantya et al. | 2022 | 2-dimensional Knapsack Problem | Genetic Algorithm | Near-optimal results; <0.2 seconds for tasks |
| Lamanna et al. | 2022 | Multidimensional Multiple-choice Knapsack Problem | Two-phase Iterative Kernel Search (TIKS) | Significant improvement over existing heuristics |
| Aïder et al. | 2022 | Quadratic Multiple Knapsack Problem | Branch-and-solve strategy | Superior to exact Cplex solver on challenging instances; varied time constraints |
| Feng & Wang | 2022 | 0–1 Multidimensional Knapsack Problem | Binary Moth Search Algorithm based on Self-Learning (SLMS) | Effective; O(n^2) complexity for fixed constraint numbers |
| GAZİOĞLU | 2023 | Multidimensional Knapsack Problem | Bayesian Multiploid Genetic Algorithm (BMGA) | Superior "Best" and "Mean" solution metrics; computational complexity not specified |
| Ghadi et al. | 2023 | 0/1 Knapsack Problem | Group Counseling Optimizer (GCO) | Viable and efficient; O(g × n × m) complexity |
| Jovanovic & Voß | 2023 | Quadratic Knapsack Problem | Greedy Randomized Adaptive Search Procedure (GRASP) | Competitive with established solutions; specific run times not mentioned |
| Hertrich & Skutella | 2023 | Knapsack Problem | Recurrent Neural Networks (RNNs) | Exact solutions in pseudo-polynomial time O(np\*) |
| Khandekar & Nargundkar | 2023 | Multi-objective Unbounded Knapsack Problem | Dynamic Programming | Generated optimal solutions across all test cases |
| Abdel-Basset et al. | 2023 | 0–1 Knapsack Problem | Binary Kepler Optimization Algorithm (BKOA) | Superior exploration and exploitation; improved performance in high-dimensional instances |
| Wei et al. | 2023 | Disjunctively Constrained Knapsack Problem | Responsive Strategic Oscillation Algorithm (RSOA) | Discovered 39 new lower bounds; matched all previously known best results |

4. Discussion:

Recent studies on the knapsack problem have presented diverse methodologies, each offering unique advantages and limitations tailored to specific scenarios. Dynamic programming-based approaches, such as those proposed by Della Croce et al.[9] for the Penalized Knapsack Problem (PKP) and Schulze et al.[10] for the Rectangular Knapsack Problem (RKP), emphasize precision and efficiency in solving structured optimization problems. These approaches outperform commercial solvers and provide scalable solutions, showcasing their robustness for large datasets. In comparison, hybrid and heuristic strategies like those introduced by Paulauskas and Jakubauskas [11] and Zhu and Yan [12] demonstrate the ability to balance solution quality with computational efficiency. These methods are particularly effective for multi-objective and multidimensional problems, where purely deterministic algorithms may falter due to complexity or scale.

A key distinction among the reviewed works lies in their focus on either theoretical advancements or practical applications. For example, Xu et al.[13] employed a modified branch-and-bound algorithm to address real-time logistics optimization, demonstrating the applicability of theoretical techniques in dynamic, real-world environments. Similarly, Wei and Hao [15] and Kwon et al. [24] adopted advanced metaheuristics and machine learning models like memetic algorithms and graph neural networks, respectively, to tackle bilevel and constrained knapsack problems, emphasizing adaptability and scalability. In contrast, innovations like the Sharknap algorithm by Laalaoui and Mhalla [25] and the Binary Kepler Optimization Algorithm by Abdel-Basset et al.[36] highlight advancements in scalability and exploration capabilities for high-dimensional problems. Collectively, these studies illustrate a dynamic interplay between enhancing algorithmic sophistication and addressing domain-specific challenges, marking significant progress in solving the knapsack problem across theoretical and applied contexts.

**4. Conclusion:**

The knapsack problem remains a key challenge in combinatorial optimization with wide-ranging applications. This review examined diverse algorithmic strategies, including dynamic programming, heuristic, and metaheuristic methods like genetic algorithms and simulated annealing, as well as recent advancements using machine learning and hybrid approaches. While exact methods offer precision, heuristic and metaheuristic techniques balance efficiency and scalability, making them ideal for complex problems. Emerging innovations, such as reinforcement learning and graph neural networks, further enhance solution quality for multidimensional scenarios. Tailoring algorithmic choices to specific problem requirements is crucial, with future research focusing on hybrid models and real-time applications to advance optimization strategies.

**References:**

[1] L. F. Plata-González, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín, and C. A. Coello Coello, “Evolutionary-based tailoring of synthetic instances for the Knapsack problem,” *Soft comput*, vol. 23, no. 23, pp. 12711–12728, Dec. 2019, doi: 10.1007/s00500-019-03822-w.

[2] Y. Li, Y. He, X. Liu, X. Guo, and Z. Li, “A novel discrete whale optimization algorithm for solving knapsack problems,” *Applied Intelligence*, vol. 50, no. 10, pp. 3350–3366, Oct. 2020, doi: 10.1007/s10489-020-01722-3.

[3] A. E. Ezugwu, V. Pillay, D. Hirasen, K. Sivanarain, and M. Govender, “A Comparative Study of Meta-Heuristic Optimization Algorithms for 0 - 1 Knapsack Problem: Some Initial Results,” *IEEE Access*, vol. 7, pp. 43979–44001, 2019, doi: 10.1109/ACCESS.2019.2908489.

[4] S. Lin, N. Meng, and W. Li, “Optimizing Constraint Solving via Dynamic Programming,” 2019. [Online]. Available: https://github.com/fzlinshu/DPSolver.

[5] B. DURMUŞ, Ö. İŞÇİ GÜNERİ, and A. İNCEKIRIK, “COMPARISON OF CLASSIC AND GREEDY HEURISTIC ALGORITHM RESULTS IN INTEGER PROGRAMMING: KNAPSACK PROBLEMS,” *Mugla Journal of Science and Technology*, vol. 5, no. 1, pp. 34–42, Jun. 2019, doi: 10.22531/muglajsci.469475.

[6] H. Assimi, “Application of Bio-inspired Algorithms to Selected Real-World Problems,” 2023.

[7] E. M. T. D. K. Ekanayake and E. M. U. S. B. Ekanayake, “Solving Triangular Fuzzy Transportation Problem Using Modified Fractional Knapsack Problem&amp;lt;br /&amp;gt; &amp;lt;div&amp;gt; &amp;lt;br /&amp;gt; &amp;lt;/div&amp;gt;,” *American Journal of Applied Mathematics*, Oct. 2023, doi: 10.11648/j.ajam.20231105.11.

[8] Y. Awasthi and A. Sharma, “Contrasting of Various Algorithmic Techniques to Solve Knapsack 0-1 Problem.”

[9] F. Della Croce, U. Pferschy, and R. Scatamacchia, “New exact approaches and approximation results for the Penalized Knapsack Problem,” *Discrete Appl Math (1979)*, vol. 253, pp. 122–135, Jan. 2019, doi: 10.1016/j.dam.2017.11.023.

[10] B. Schulze, M. Stiglmayr, L. Paquete, C. M. Fonseca, D. Willems, and S. Ruzika, “On the rectangular knapsack problem: approximation of a specific quadratic knapsack problem,” *Mathematical Methods of Operations Research*, vol. 92, no. 1, pp. 107–132, Aug. 2020, doi: 10.1007/s00186-020-00702-0.

[11] N. Ali, A. Etawi, F. Aburomman, N. A. Al Etawi, and F. T. Aburomman, “0/1 KNAPSACK PROBLEM: GREEDY VS. DYNAMIC-PROGRAMMING,” *International Journal of Advanced Engineering and Management Research*, vol. 5, no. 02, 2020, [Online]. Available: www.ijaemr.com

[12] A. Shahbandegan and M. Naderi, “A Binary Butterfly Optimization Algorithm for the Multidimensional Knapsack Problem,” in *6th Iranian Conference on Signal Processing and Intelligent Systems, ICSPIS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/ICSPIS51611.2020.9349589.

[13] M. Abdel-Basset, R. Mohamed, and S. Mirjalili, “A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems,” *Comput Ind Eng*, vol. 151, p. 106946, Jan. 2021, doi: 10.1016/j.cie.2020.106946.

[14] S. L. Benford, “Solving the binary knapsack problem using tabular and deep reinforcement learning algorithms,” Northeastern University, 2021. doi: 10.17760/D20410381.

[15] Z. Wei and J.-K. Hao, “A threshold search based memetic algorithm for the disjunctively constrained knapsack problem,” *Comput Oper Res*, vol. 136, p. 105447, Dec. 2021, doi: 10.1016/j.cor.2021.105447.

[16] A. Khan, E. Sharma, and K. V. N. Sreenivas, “Approximation Algorithms for Generalized Multidimensional Knapsack,” Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.05854

[17] W. Gálvez, F. Grandoni, A. Khan, D. Ramírez-Romero, and A. Wiese, “Improved Approximation Algorithms for 2-Dimensional Knapsack: Packing into Multiple L-Shapes, Spirals, and More,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.10406

[18] J. Yuan and Y. Li, “Solving binary multi-objective knapsack problems with novel greedy strategy,” *Memet Comput*, vol. 13, no. 4, pp. 447–458, Dec. 2021, doi: 10.1007/s12293-021-00344-7.

[19] S. Coniglio, F. Furini, and P. San Segundo, “A new combinatorial branch-and-bound algorithm for the Knapsack Problem with Conflicts,” *Eur J Oper Res*, vol. 289, no. 2, pp. 435–455, Mar. 2021, doi: 10.1016/j.ejor.2020.07.023.

[20] X. Sánchez-Díaz, J. C. Ortiz-Bayliss, I. Amaya, J. M. Cruz-Duarte, S. E. Conant-Pablos, and H. Terashima-Marín, “A feature-independent hyper-heuristic approach for solving the knapsack problem,” *Applied Sciences (Switzerland)*, vol. 11, no. 21, Nov. 2021, doi: 10.3390/app112110209.

[21] Y. He and X. Wang, “Group theory-based optimization algorithm for solving knapsack problems,” *Knowl Based Syst*, vol. 219, May 2021, doi: 10.1016/j.knosys.2018.07.045.

[22] N. Moradi, V. Kayvanfar, and M. Rafiee, “An efficient population-based simulated annealing algorithm for 0–1 knapsack problem,” *Eng Comput*, vol. 38, no. 3, pp. 2771–2790, Jun. 2022, doi: 10.1007/s00366-020-01240-3.

[23] S. V. Chebakov and L. V. Serebryanaya, “Algorithm for solving the knapsack problem with certain properties of Pareto layers,” *Journal of the Belarusian State University. Mathematics and Informatics*, no. 3, pp. 54–66, Dec. 2022, doi: 10.33581/2520-6508-2022-3-54-66.

[24] S. Kwon, H. Choi, and S. Park, “Solving Bilevel Knapsack Problem using Graph Neural Networks,” *arXiv preprint arXiv:2211.13436*, 2022.

[25] Y. Laalaoui and H. Mhalla, “An exact algorithm for large knapsack sharing problems,” *INFOR: Information Systems and Operational Research*, vol. 60, no. 3, pp. 315–341, Aug. 2022, doi: 10.1080/03155986.2022.2049567.

[26] G. Sur, S. Y. Ryu, J. Kim, and H. Lim, “A Deep Reinforcement Learning-Based Scheme for Solving Multiple Knapsack Problems,” *Applied Sciences*, vol. 12, no. 6, p. 3068, Mar. 2022, doi: 10.3390/app12063068.

[27] D. Bramantya, I. G. S. Astawa, I. W. Supriana, L. G. Astuti, N. A. Sanjaya ER, and I. G. A. G. A. Kadyanan, “Rancangan dan Analisis Model Algoritma Genetika Untuk Menyelesaikan Permasalahan Knapsack 2 Dimensi,” *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 11, no. 2, p. 395, Jul. 2022, doi: 10.24843/JLK.2022.v11.i02.p18.

[28] L. Lamanna, R. Mansini, and R. Zanotti, “A two-phase kernel search variant for the multidimensional multiple-choice knapsack problem,” *Eur J Oper Res*, vol. 297, no. 1, pp. 53–65, Feb. 2022, doi: 10.1016/j.ejor.2021.05.007.

[29] M. Aïder, O. Gacem, and M. Hifi, “Branch and solve strategies-based algorithm for the quadratic multiple knapsack problem,” *Journal of the Operational Research Society*, vol. 73, no. 3, pp. 540–557, 2022, doi: 10.1080/01605682.2020.1843982.

[30] Y. Feng and G. G. Wang, “A binary moth search algorithm based on self-learning for multidimensional knapsack problems,” *Future Generation Computer Systems*, vol. 126, pp. 48–64, Jan. 2022, doi: 10.1016/j.future.2021.07.033.

[31] E. GAZİOĞLU, “Solving Multidimensional Knapsack Problem with Bayesian Multiploid Genetic Algorithm,” *Journal of Soft Computing and Artificial Intelligence*, vol. 3, no. 2, pp. 58–64, Dec. 2022, doi: 10.55195/jscai.1216193.

[32] Y. Y. Ghadi *et al.*, “An efficient optimizer for the 0/1 knapsack problem using group counseling,” *PeerJ Comput Sci*, vol. 9, 2023, doi: 10.7717/PEERJ-CS.1315.

[33] R. Jovanovic and S. Voß, “Solving the Quadratic Knapsack Problem Using GRASP,” 2023, pp. 157–178. doi: 10.1007/978-981-19-3888-7\_6.

[34] C. Hertrich and M. Skutella, “Provably Good Solutions to the Knapsack Problem via Neural Networks of Bounded Size,” *INFORMS J Comput*, vol. 35, no. 5, pp. 1079–1097, Sep. 2023, doi: 10.1287/ijoc.2021.0225.

[35] A. P. Khandekar and A. Nargundkar, “Dynamic Programming Approach to Solve Real-World Application of Multi-Objective Unbounded Knapsack Problem,” 2023, pp. 417–422. doi: 10.1007/978-981-19-6581-4\_32.

[36] M. Abdel-Basset, R. Mohamed, I. M. Hezam, K. M. Sallam, A. M. Alshamrani, and I. A. Hameed, “A novel binary Kepler optimization algorithm for 0–1 knapsack problems: Methods and applications,” *Alexandria Engineering Journal*, vol. 82, pp. 358–376, Nov. 2023, doi: 10.1016/j.aej.2023.09.072.

[37] Z. Wei, J.-K. Hao, J. Ren, and F. Glover, “Responsive strategic oscillation for solving the disjunctively constrained knapsack problem,” *Eur J Oper Res*, vol. 309, no. 3, pp. 993–1009, Sep. 2023, doi: 10.1016/j.ejor.2023.02.009.