Journal of Ubiquitous Computing and Communication Technologies (ISSN: 2582-337X)
www.irojournals.com/jucct/

Novel Solutions to the Multidimensional Knapsack Problem Using CPLEX: New Results on ORX Benchmarks

Khelifa Meriem[1], Mezzoudj Saliha[2], Hacini Mohammed abdelaziz[3], Fenniche Mohammed Amine[4], Babasidi Mohammed Khaled[5]

[1]Associate Professor, Department of Computer Science and Information Technologies, University of Kasdi Merbah Ouargla Algeria, Artificial intelligence of information technologies, Algeria

[2]senior Researcher, Department of Computer Science, University of Algiers.

[3-5]Master Graduate, Department of Computer Science and Information Technologies, University of Kasdi Merbah Ouargla Algeria, Artificial intelligence of information technologies, Algeria

E-mail: [1]khelifa.meriem@univ-ouargla.dz, [2]s.mezzoudj@univ-alger.dz

Abstract

The Multidimensional Knapsack Problem (MKP) is a challenging combinatorial optimization problem that extends the classical knapsack problem by introducing multiple capacity constraints across various dimensions. This problem has significant practical applications, including resource allocation in supply chain management, portfolio optimization in finance, and cargo loading in logistics, where the goal is to maximize the total profit of selected items while adhering to these constraints. In this research, the CPLEX solver was applied to address the MKP using a set of complex instances from the OR-Library, specifically the ORX Benchmarks. The study focuses on 270 MKP instances characterized by varying numbers of variables (n = 100, 250, 500), constraints (m = 5), and tightness ratios ($\alpha = 0.25$). Through advanced CPLEX techniques, new results were successfully obtained by employing advanced CPLEX methods, contributing to the existing literature, and setting new benchmarks for these instances.

DOI: https://doi.org/10.36548/jucct.2024.3.007
Received: 12.08.2024, received in revised form: 02.09.2024, accepted: 19.09.2024, published: 01.10.2024

Keywords: Multidimensional Knapsack Problem (MKP), CPLEX Solver, ORX Benchmarks, OR Library, Combinatorial Optimization.

1. Introduction

Combinatorial optimization focuses on finding the best solution from a set of discrete choices, each bound by specific constraints. These problems are essential in theoretical studies and practical fields, including operations research, computing, logistics, and finance. Efficiently tackling these challenges can greatly enhance strategic decision-making and optimize how resources are managed.

Among the most recognized combinatorial optimization problems is the Traveling Salesman Problem (TSP) [1-3] which aims to determine the shortest possible route that allows a salesman to visit each city exactly once and return to the starting point. Another notable example is the Traveling Tournament Problem (TTP) [4-8] which aims to schedule games in a sports league while minimizing travel distances and meeting various constraints. The Knapsack Problem [9-11] is a well-known optimization problem where the objective is to maximize the total value of items packed into a knapsack without exceeding its capacity. The 0/1 Knapsack Problem is a specific case where each item can be either included (1) or excluded (0). This is the fundamental definition of the problem. However, several variations exist, including [12] the following

- Single Knapsack Problem: All items must be packed into a single container.
- Multidimensional Knapsack Problem (MKP): Multiple containers are available for item placement.
- Multiple-Choice Knapsack Problem: Items are grouped into subsets, with at most one item from each subset being selected.
- Bounded Knapsack Problem: The number of items available for selection is limited.

In this work, the focus is on the 0/1 MKP, a problem with practical significance due to its high computational complexity and numerous real-world applications.

The 0-1 MKP extends the classic 0-1 Knapsack Problem by introducing multiple constraints instead of just one. In the MKP, each item has various attributes, such as weight, volume, and cost, corresponding to different knapsack constraints (e.g., weight limit, volume limit, budget limit). The goal remains the same: to achieve the highest possible combined value of the selected items. However, the solution must satisfy all the constraints simultaneously.

The MKP is particularly challenging due to its increased complexity [13]. Unlike the single-constraint knapsack problem, where the solution can be more straightforward, the MKP requires balancing multiple constraints simultaneously. This makes the problem much more challenging, especially as the number of constraints and items increases. The MKP is NP-hard [14,15] meaning that finding an optimal solution can be computationally intensive, particularly for large problem instances.

The MKP has significant practical applications in various fields: Resource Allocation [16]: In scenarios where resources are limited and must be allocated across multiple projects or departments while respecting budgetary and other constraints. Portfolio Optimization [17]: Selecting a portfolio of investments that maximizes returns while adhering to risk, budget, and other constraints. Cargo Loading [18]: Determining the optimal way to load cargo into a vehicle or container while considering multiple constraints like weight, volume, and stability.

Due to the NP-hardness of the 0–1 MKP problem, several exact and heuristic optimization methods have been used to solve both small and large MKP instances. For exact or deterministic algorithms, many works have been proposed: the authors in [18] introduces a dynamic programming-based approach for the MKP using sparse data representation to reduce memory and time requirements. This study [19] explored the MKP by analyzing the gap between LP-relaxed and optimal solutions, introducing a new core concept, and evaluating collaborative approaches combining ILP and metaheuristics. Exact algorithms can efficiently solve small-scale instances of the MKP, but their computational time increases dramatically as the problem size grows, making them impractical for larger instances. In contrast, metaheuristic algorithms, while not guaranteeing optimal solutions, can quickly find high-quality approximations—often reaching near-optimal or even optimal solutions—with a significantly better approximation ratio. For example, in [21], the authors propose a new

approach using Deep Symbolic Regression and Recurrent Neural Networks to train scoring functions for the Multidimensional Knapsack Problem. It shows that it can outperform traditional human-crafted heuristics; this work [22] presents TEPSOq, an algorithm that merges Tabu Search with Essential Particle Swarm Optimization Queen to tackle large-scale MKP instances. It demonstrates superior performance compared to TEPSOq and other meta-heuristic methods. [23] This study introduces an enhanced Teaching-Learning-Based Optimization (TLBO) method for the MKP. It eliminates extensive parameter tuning and proves effective for large-scale industrial problems, with competitive results on standard benchmarks from Beasley's OR Library.

2. Multidimensional Knapsack Problem

In MKP, a set of items is given, each with a value and multiple resource consumption levels across different dimensions (e.g., weight, volume). The objective is to choose a subset of items that maximizes the total value while adhering to the capacity limits imposed on each dimension.

Consider a knapsack with M dimensions, where c_j represents the capacity of the i^{th} dimension, for i = 1, 2, ...,m. we have n items available. Each item j item requires w_{ij} units of the i^{th} dimension of the knapsack j, j = 1, 2... n *and i, =1, 2…*m. The reward of including the item j in the knapsack is p_j . The problem can be formulated as follows:

$$\max Z = \sum_{j=1}^{n} p_j x_j \qquad (1)$$

$$\text{subject to} \sum_{j=1}^{n} w_{ij} x_j < c_{j,} \quad i = 1,2,..,m, \qquad (2)$$

Equations (1) and (2) represent the objective function and the constraints, respectively, as follows:

•x_j is a binary value. Such x_j = 1 if we put the item j into the knapsack and x_j = 0 if we don't.

• j refers to the item, $j \leq n$.

3. CPLEX Solver to Solve MKP

CPLEX, developed by IBM, is a widely used commercial solver designed for large-scale optimization problems such as Linear Programming (LP), Mixed-Integer Programming (MIP), and Quadratic Programming. In this study, CPLEX was used to efficiently solve the Multidimensional Knapsack Problem (MKP) by employing its advanced Branch-and-Cut algorithm, which integrates various optimization techniques to improve computational efficiency.

The process begins with a preprocessing phase, where CPLEX applies probing and symmetry detection techniques to reduce the problem size by eliminating redundant variables and constraints. For the ORX100 instance, the original MIP model had 100 binary variables (columns), 500 nonzeros, and 5 constraints. After CPLEX applied presolve and aggregation steps, the model was reduced to 83 binary variables, 415 nonzeros, and the same number of constraints. This reduction focuses the solver on the core structure of the problem, improving computational efficiency without altering its essential characteristics.

After preprocessing, CPLEX employs the Branch-and-Cut algorithm, a combination of Branch-and-Bound and cutting-plane techniques. The first step in this process is solving the LP relaxation at the root node, where the binary constraints are relaxed to allow fractional values. For the ORX100 instance, the LP relaxation at the root node was solved in 0.18 ticks, providing an initial objective value of 24,538.2090.

Once the relaxation is complete, CPLEX generates cutting planes, which are additional constraints that tighten the LP relaxation and help eliminate fractional solutions. In the ORX100 instance, CPLEX applied several types of cuts, including 52 cover cuts, 5 mixed-integer rounding cuts, 4 zero-half cuts, and 2 Gomory fractional cuts. These cuts reduced the objective value at the root node to 24,505.5560, with the gap between the best integer solution (24,190.0000) and the LP bound narrowing to 1.30%.

Following the application of cutting planes, CPLEX proceeds with the branching phase, where the solver explores fractional variables and creates sub-problems by branching

on those variables. This divides the solution space into smaller regions. For the ORX100 instance, after the relaxation and cuts, CPLEX dynamically explored fractional variables across 4,239 nodes. At each node, further cuts were applied to tighten the solution space. CPLEX also employed symmetry detection multiple times during this phase, identifying and eliminating equivalent solutions to reduce the search space, leading to faster processing.

A critical feature of CPLEX is its ability to utilize parallel processing, which allows for simultaneous exploration of multiple nodes in the search tree. For the ORX100 instance, CPLEX used 8 threads in parallel mode, completing the Branch-and-Cut process in 0.77 seconds. This parallel processing significantly accelerates the computation for large-scale problems like the MKP by allowing the solver to explore several regions of the solution space concurrently.

The final solution for the ORX100 instance had a gap of 0.66%, indicating that the solution was near-optimal. This result demonstrates the effectiveness of the Branch-and-Cut algorithm, combined with cutting-plane generation, preprocessing, aggregation, and dynamic search methods. These techniques, along with CPLEX's parallel processing capabilities played a crucial role in efficiently solving the MKP by reducing the problem size, improving computational efficiency, and ensuring faster convergence to an optimal or near-optimal solution.

To evaluate the performance of CPLEX on the MKP, tests were conducted on various benchmark instances, including the ORX suite. For instance, in the OR5x100 instance, CPLEX reduced the problem's dimensionality by 95%, effectively solving the problem within a few seconds. The empirical analysis demonstrated CPLEX's ability to handle large-scale MKP instances with a high degree of accuracy, consistently achieving near-optimal solutions while maintaining low computational time. The detailed results from these benchmarks highlighted the solver's ability to exploit the structural properties of the MKP, validating its suitability for solving such complex combinatorial problems.

The steps of the CPLEX algorithm applied to the ORX instances are summarized as follows:

Step 1: Initialization of the MKP Model

- n: Number of items
- m: Number of knapsack constraints
- p: Profit of item i
- w_{ij}: Weight of item i in constraint j
- c_j: Capacity of knapsack j
- x_i: Binary decision variable, x_i=1 if item i is selected, otherwise x_i=0

The Objective Function: $\max Z = \sum_{j=1}^{n} p_j x_j$

The Constraints: $\sum_{j=1}^{n} w_{ij} x_j < c_{j,} \quad i = 1,2,..,m,$

$$x_i \in \{0,1\} \; \forall \; i = 1, \dots, n$$

Output:

- Optimal solution x∗
- Optimal objective value z*

Set Initial Bounds:

$$z_{best} = -\infty \; (the\ initial\ best\ solution)$$

$$z_{ub} = +\infty \; (the\ initial\ upper\ bound)$$

Step 2: Presolve and Preprocessing

- **Identifies constraints** that are not active and can be removed to simplify the problem: if any constraint j is dominated by another (i.e., if for all items i, $w_{ij} > w_{ik}$, and $c_j < c_k$, then constraint j can be removed).
- **Variable Fixing:** If an item i has a weight w_{ij} in any knapsack j that exceeds the knapsack's capacity $c_{j,}$, then x_i can be immediately fixed to 0.

- **Constraint Aggregation:** if certain items have identical weights across several knapsacks), CPLEX aggregates these constraints into one.

- **Symmetry Detection:** In MKP, symmetry arises when some items have identical profits and weights across all knapsacks. CPLEX adds lexicographical ordering constraints to ensure the problem does not explore redundant symmetric solutions (i.e., selecting the same items in different orders).

Step 3: Root Node Initialization (LP Relaxation)

- **Relaxing the Binary Constraints**: CPLEX first solves the Linear Programming (LP) relaxation of the MKP by allowing $x_i \in [0,1]$ to take continuous values between 0 and 1. This relaxation provides an initial bound for the objective.

- **Check Feasibility of Relaxed Solution**: If the LP solution is already integer feasible (i.e., all x_i values are binary), CPLEX will skip further branching and cuts.

Step 4: Cut Generation

If the LP relaxation yields a fractional solution (some x_i's are not 0 or 1), CPLEX adds cuts to exclude these fractional solutions.

- **Cover Inequality Cut**: A cover inequality is generated when a subset of items exceeds the capacity of the knapsack. Example: If the fractional solution selects items whose combined weight exceeds the capacity, CPLEX will add a constraint that excludes such a solution.

- **Gomory Fractional Cuts**: These cuts are based on the fractional values from the simplex tableau. If an MKP solution contains fractional weights, CPLEX cuts off the fractional part by generating Gomory cuts.

- **Mixed Integer Rounding (MIR) Cuts**: MIR cuts are used to round fractional coefficients in the knapsack constraints. For instance, if an item x_i contributes fractionally to the constraint but exceeds a rounding threshold, CPLEX adds a cut.

- **Zero-Half Cuts**: CPLEX identifies zero-half cuts, which apply when the sum of fractional variables is either close to 0 or 1. This helps eliminate fractional solutions in MKP.

- **Cut Pool Management**: After adding a cut, CPLEX re-solves the LP relaxation, refining the solution. If the new solution is still fractional, CPLEX continues generating cuts until the solution becomes integer feasible or no more effective cuts can be found.

Step 5: Branching Procedure

After generating cuts, CPLEX begins the branch-and-cut phase, where it explores the solution space by branching on fractional decision variables.

- **Branching on Fractional Variables:** If the solution after cuts is still fractional, CPLEX selects a decision variable x_i with the largest fractional value and creates two subproblems:

  -One subproblem with x_i (item selected).

  -Another subproblem with $x_i = 0$ (item not selected).

- **Exploring the Branch-and-Cut Tree**: CPLEX dynamically explores the branch-and-cut tree, selecting the most promising nodes to explore further

- **Dynamic Search Method**: CPLEX uses a dynamic search method for node exploration, balancing between depth-first and best-first strategies to efficiently explore nodes.

- **Subproblem Processing and Pruning**:

-Each subproblem is solved, and infeasible or non-promising nodes are pruned (cut off from the tree).

-If a subproblem yields a feasible integer solution, the incumbent is updated. If not, it is pruned from further consideration.

- **Node Logging**: As CPLEX explores the tree, the best integer solution improves.

Step 6: Restarts and Further Optimization

- **Restarts**: To improve the solution process, CPLEX can perform restarts by clearing the current search tree while retaining useful information about the problem. This strategy can lead to faster convergence after cuts are reapplied.
- **Presolve After Restarts**: After each restart, CPLEX reapplies presolve to further reduce the problem size.

Step 7: Convergence and Solution Improvement

- **Improving the Best Integer Solution**: Throughout the process, CPLEX improves the **best integer solution** as it explores new nodes and applies cuts.
- **Optimality Gap Reduction**: CPLEX continuously reduces the optimality gap between the current best integer solution and the bound obtained from the LP relaxation.

Step 8: Termination and Solution Output

- Termination Condition: The algorithm terminates when all nodes are pruned or processed, and no further improvement is possible. The final solution is provided along with the total time and ticks used

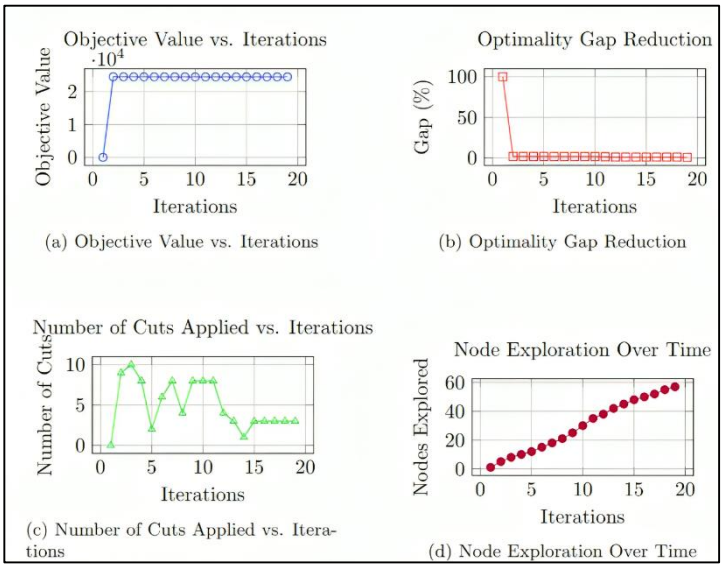

Figure 1. CPLEX Process Curves based on OR5x100-0.25_2 Dataset

Figure 1 represents the CPLEX process curves obtained by applying the solver to the OR5x100-0.25_2 data. These curves visually represent how CPLEX optimizes the solution over time through different strategies, such as applying cuts and exploring nodes.

Objective Value vs. Iterations (Figure 1a): This curve shows how the objective value improves over iterations as CPLEX progresses. The y-axis represents the objective value, and the x-axis represents the number of iterations. The curve's upward trend indicates how CPLEX refines the solution by applying cutting planes and branching strategies, gradually moving towards an optimal or near-optimal solution.

Optimality Gap Reduction (Figure 1b): This curve illustrates the reduction of the optimality gap over iterations, highlighting how close the current solution is to the optimal one. The y-axis shows the percentage gap, while the x-axis represents the iterations. A downward-sloping curve indicates that the gap is narrowing, reflecting CPLEX's effectiveness in improving the solution quality over time.

Number of Cuts Applied vs. Iterations (Figure 1c): This curve highlights the role of cutting planes, such as cover cuts and mixed integer rounding cuts, in tightening the solution space. The y-axis displays the number of cuts applied, and the x-axis represents the iterations. Peaks in the curve suggest critical moments where cuts were particularly effective in enhancing the solution quality by reducing the feasible region.

Node Exploration Over Time (Figure 1d): This curve demonstrates how CPLEX explores nodes in the search tree during the branch-and-cut process. The y-axis shows the number of nodes explored, while the x-axis represents the iterations. The upward trend reflects bursts of node exploration, which align with significant changes in the objective value and the overall progress toward an optimal solution.

4. Benchmarks

The experimental data utilized in this study are derived from the well-known Chu & Beasley benchmarks available in the OR-Library [14], which is frequently referenced in the literature. The focus was on 270 instances of the MKP, characterized by n=100, 250, and 500 variables, m = 5, 10, and 30 constraints, and tightness ratios α=0.25. For each combination of

(m_n_α). Each problem is labelled as ORmxn-α_r, where m indicates the number of constraints, n is the number of variables, α the tightness ratio and r is the instance number.

5. The Numerical Results

The research utilized CPLEX Optimization Studio 20.1.0, running on an Intel(R) Core(TM) i5-10210U CPU @ 1.60 GHz with 8 GB of RAM.

Tables 1 and 2 provide a summary of the overall measurement results. The 'Best Known Results' column lists the best-known values from the literature for the instances considered [24-27], [20]. The proposed results' column shows the outcomes obtained using the CPLEX solver, while the 'Gap' column highlights the difference between the best-known results and the new results achieved in this study.

The findings demonstrate that the CPLEX solver significantly improved the best solutions for almost all OR5x500-0.25 instances, achieving an average gap of 1.22%.

Tables 1. Numerical Results for OR5x500-0.25 Instances

Instances	The Best-Known Results	Proposed Results	Gap	Execution Time (seconds)	Memory Usage (MB)
OR5x500-0.25_1.dat	117811	120148	+1,83%	80	748.328
OR5x500-0.25_2.dat	117879	117879	0%	89	812.221
OR5x500-0.25_3.dat	119215	121131	+1,58%	94	678.456
OR5x500-0.25_4.dat	120804	120804	0%	101	987.111
OR5x500-0.25_5.dat	122319	122319	0%	75	754.345
OR5x500-0.25_6.dat	119504	122024	+2,06	104	868.409

OR5x500-0.25_7.dat	119127	119827	0%	123	711.398
OR5x500-0.25_8.dat	118329	120568	+1,85%	67	799.987
OR5x500-0.25_9.dat	121575	121649	+0,07%	55	654.432
OR5x500-0.25_10.dat	120717	121398	+0,58%	92	766.298

Table 2. Numerical Results for OR5x250-0.25 and OR30x100-0.25 Instances

Instances **OR5x250-0.25**	**1.dat**	**2.dat**	**3.dat**	**4.dat**	**5.dat**	**6.dat**	**7.dat**	**8.dat**	**9.dat**	**10.dat**
Proposed results	59243	61472	62130	59404	59463	60007	60342	61440	61870	58869
HLMS [28]	59063	61295	61767	59140	58605	-	-	-	-	-
Gap	+0.30%	+0.28%	+0.58%	+0.44%	+1.46%					
Execution Time (seconds)	50	44	32	67	23	15	36	70	22	54
OR30x100-0.25	1.dat	2.dat	3.dat	4.dat	5.dat	6.dat	7.dat	8.dat	9.dat	10.dat
Proposed results	21946	21716	20754	21464	21844	22176	21799	21397	22525	20983
SSTSA [29]	21835	21716	20675	21464	21767	22121	21699	21397	22450	20983
Gap	+0.50%	0%	+0.38	0%	+0.33%	+0.24%	+0.46%	0%	+0.33	0%

For the OR5x250-0.25 instances, the comparison in Table 2 is made with the HLMS (Hybrid Learning Moth Search Algorithm) method [28]. The proposed approach shows an

Execution Time (seconds)	88	90	99	106	102	65	44	82	30	84

improvement over HLMS, achieving an average gap reduction of 0.61%. Similarly, for the OR30x100-0.25 instances, the proposed results were compared with the SSTSA (Scatter Search and Tabu Search Algorithms) approach [29]. The proposed method outperformed SSTSA with an average gap reduction of 0.24%.

6. Conclusion

In summary, the investigation into the 0/1 Multidimensional Knapsack Problem (MKP) using the CPLEX solver has led to noteworthy advancements in solving challenging instances from the OR library, particularly within the OR5x500-0.25 benchmarks. The results demonstrate the efficacy of the CPLEX approach in navigating the MKP's complex landscape, yielding significant improvements in solution quality across a diverse range of instances.

References

[1] Pop, Petrică C., Ovidiu Cosma, Cosmin Sabo, and Corina Pop Sitar. "A comprehensive survey on the generalized traveling salesman problem." European Journal of Operational Research 314, no. 3 (2024): 819-835.

[2] Lin, Shen. "Computer solutions of the traveling salesman problem." Bell System Technical Journal 44, no. 10 (1965): 2245-2269.

[3] Min, Yimeng, Yiwei Bai, and Carla P. Gomes. "Unsupervised learning for solving the travelling salesman problem." Advances in Neural Information Processing Systems 36 (2024).

[4] Frohner, Nikolaus, Bernhard Neumann, Giulio Pace, and Günther R. Raidl. "Approaching the traveling tournament problem with randomized beam search." Evolutionary Computation 31, no. 3 (2023): 233-257.

[5] Khelifa, Meriem, and Dalila Boughaci. "Hybrid harmony search combined with variable neighborhood search for the traveling tournament problem." In Computational Collective Intelligence: 8th International Conference, ICCCI 2016,

Halkidiki, Greece, September 28-30, 2016. Proceedings, Part I 8, Springer International Publishing, 2016. 520-530.

[6] Khelifa, Meriem, and Dalila Boughaci. "A cooperative local search method for solving the traveling tournament problem." Computing and Informatics 37, no. 6 (2018): 1386-1410.

[7] Khelifa, Meriem, and Dalila Boughaci. "A variable neighborhood search method for solving the traveling tournaments problem." Electronic Notes in Discrete Mathematics 47 (2015): 157-164.

[8] Khelifa, Meriem, Dalila Boughaci, and Esma Aïmeur. "An enhanced genetic algorithm with a new crossover operator for the traveling tournament problem." In 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, 2017.1072-1077.

[9] D'Ambrosio, Ciriaco, Federica Laureana, Andrea Raiconi, and Gaetano Vitale. "The Knapsack Problem with forfeit sets." Computers & Operations Research 151 (2023): 106093.

[10] Perera, Kokila Kasuni, and Aneta Neumann. "Multi-objective evolutionary algorithms with sliding window selection for the dynamic chance-constrained knapsack problem." In Proceedings of the Genetic and Evolutionary Computation Conference, 223-231. 2024.

[11] Wang, Lina, Yichao He, Xizhao Wang, Zihang Zhou, Haibin Ouyang, and Seyedali Mirjalili. "A novel discrete differential evolution algorithm combining transfer function with modulo operation for solving the multiple knapsack problem." Information Sciences 680 (2024): 121170.

[12] Martello, Silvano, and Paolo Toth. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc., 1990.

[13] H. P. U. P. D. e. a. KELLERER, Multidimensional knapsack problems, pringer Berlin Heidelberg, 2004. 235-283

[14] M. R. D. S. J. Gary, A Guide to the Theory of NP-completeness, W.H.Freeman & Co Ltd (26 April 1979)

[15] Fréville, Arnaud. "The multidimensional 0–1 knapsack problem: An overview." European Journal of Operational Research 155, no. 1 (2004): 1-21.

[16] Jihad, Siddig, Xianqiao Chen, Bing Shi, and Solyman Aiman. "Multidimensional knapsack problem for resource allocation in a distributed competitive environment based on genetic algorithm." In 2019 international conference on computer, control, electrical, and electronics engineering (ICCCEEE), IEEE, 2019. 1-5.

[17] I. S. A. D. D. e. a. ANADANI, «Genetic Algorithm Approach for Portfolio Optimization, International Conference on Data Science and Applications, 113-124, 2023.

[18] Ramos, António G., and José Fernando Oliveira. "Cargo stability in the container loading problem-state-of-the-art and future research directions." In Operational Research: IO2017, Valença, Portugal, June 28-30 XVIII, Springer International Publishing, 2018. 339-350.

[19] Balev, Stefan, Nicola Yanev, Arnaud Fréville, and Rumen Andonov. "A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem." European journal of operational research 186, no. 1 (2008): 63-76.

[20] Puchinger, Jakob, Günther R. Raidl, and Ulrich Pferschy. "The multidimensional knapsack problem: Structure and algorithms." INFORMS Journal on Computing 22, no. 2 (2010): 250-265.

[21] Kieffer, Emmanuel, Gabriel Duflo, Grégoire Danoy, Sébastien Varrette, and Pascal Bouvry. "A RNN-Based Hyper-heuristic for Combinatorial Problems." In European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar), Cham: Springer International Publishing, 2022. 17-32.

[22] Ktari, Raida, and Habib Chabchoub. "Essential particle swarm optimization queen with tabu search for MKP resolution." Computing 95 (2013): 897-921.

[23] Z. Y. L. F. V. Kern, «An OR practitioner's solution approach to the multidimensional knapsack problem,» International Journal of Industrial Engineering Computations, 2020. 73-82.

[24] Ktari, Raida, and Habib Chabchoub. "Essential particle swarm optimization queen with tabu search for MKP resolution." Computing 95 (2013): 897-921.

[25] Chu, Paul C., and John E. Beasley. "A genetic algorithm for the multidimensional knapsack problem." Journal of heuristics 4 (1998): 63-86.

[26] Angelelli, Enrico, M. Grazia Speranza, and Martin WP Savelsbergh. "Competitive analysis for dynamic multiperiod uncapacitated routing problems." Networks: An International Journal 49, no. 4 (2007): 308-317.

[27] Vasquez, Michel, and Yannick Vimont. "Improved results on the 0–1 multidimensional knapsack problem." European Journal of Operational Research 165, no. 1 (2005): 70-81.

[28] Feng, Yanhong, Hongmei Wang, Zhaoquan Cai, Mingliang Li, and Xi Li. "Hybrid Learning Moth Search Algorithm for Solving Multidimensional Knapsack Problems." Mathematics 11, no. 8 (2023): 1811.

[29] He, Song, and Wei Li. "An Adaptive Search Algorithm with Scatter and Tabu Strategy for Multidimensional Knapsack Problem." In International Symposium on Intelligence Computation and Applications, Singapore: Springer Nature Singapore, 2021. 327-344.