Institut Riset dan Publikasi Indonesia (IRPI)
MALCOM: Indonesian Journal of Machine Learning and Computer Science
Journal Homepage: https://journal.irpi.or.id/index.php/malcom
Vol. 5 Iss. 4 October 2025, pp: 1360-1366
ISSN(P): 2797-2313 | ISSN(E): 2775-8575

Leaders and Followers Algorithm for the Binary Knapsack Problem

Helen Yuliana Angmalisang[1*], Harrychoon Angmalisang[2], Nita Anggriani[3]

[1,3]Mathematics Department, Faculty of Mathematics, Natural Science and Earth Science,
Universitas Negeri Manado, Indonesia
[2]Department of Electrical Engineering, Faculty of Engineering,
Universitas Negeri Manado, Indonesia

E-Mail: [1]helenangmalisang@unima.ac.id,
[2]angmalisangharry@unima.ac.id, [3]nita_anggriani@unima.ac.id

Received Jul 08th 2025; Revised Oct 04th 2025; Accepted Oct 18th 2025; Available Online Oct 31th 2025
Corresponding Author: Helen Yuliana Angmalisang

Abstract

The Leaders and Followers (LaF) algorithm, as a relatively recent metaheuristic compared to other well-established algorithms, has demonstrated strong performance in solving continuous constrained optimization problems, the balanced transportation problem, and the traveling salesman problem. The distinctive feature of the LaF algorithm lies in its dual-population structure, where two groups operate with different roles, namely exploration and exploitation, to balance search diversity and convergence. This design effectively prevents premature convergence. In this study, the LaF algorithm is applied to address the binary knapsack problem. The proposed algorithm was evaluated using a well-established benchmark dataset for this problem. The results indicate that the LaF algorithm exhibits stable performance in solving binary knapsack problems with moderately sized capacities and outperforms several other metaheuristic algorithms.

Keyword: Knapsack Problem, Leaders and Followers Algorithm, Metaheuristic Algorithm, Optimization

1. INTRODUCTION

The Leaders and Followers (LaF) algorithm is a relatively recent metaheuristic that has its distinctive dual-population design and promising performance in optimization. Unlike traditional metaheuristics that rely on a single population for search and solution refinement, LaF employs two separate populations with complementary roles: leaders focus on exploitation to refine high-quality solutions, while followers perform exploration to maintain search diversity. This structure effectively mitigates early elitism and reduces the risk of premature convergence, which are common challenges in metaheuristic optimization.

Originally proposed by [1], the LaF algorithm has demonstrated strong performance in continuous constrained optimization problems, balanced transportation problems, and combinatorial optimization problems such as the traveling salesman problem (TSP) [2], [3], [4]. Its adaptive balance between exploration and exploitation makes it a versatile approach capable of handling both complex search landscapes and diverse problem domains. These promising results naturally raise the question of how robust and versatile the LaF algorithm is when applied to other widely studied combinatorial optimization problems, including the binary knapsack problem.

The knapsack problem is a combinatorial optimization problem that finds widespread application across various fields. Being classified as an NP-complete problem, research on finding effective and efficient ways to solve this problem is of paramount importance. The knapsack problem finds applications in various real-world scenarios, such as resource allocation in logistics [5], [6], portfolio optimization in finance [7], [8], scheduling tasks in project management [9], [10], advertising campaign optimization[11], [12], and optimizing bandwidth usage in telecommunication networks [6], [13]. In logistics, it helps in determining the optimal selection of items for transportation within constrained capacities, while in finance, it aids in constructing investment portfolios with maximum returns given limited resources. Similarly, in project management, it assists in scheduling tasks to meet deadlines and resource constraints. The versatility of the knapsack problem makes it a valuable tool across diverse domains, driving the need for innovative algorithms to efficiently tackle its complexities and deliver practical solutions.

Basically, the knapsack problem is an optimization problem where a knapsack with a capacity constraint is given, and the goal is to determine which items can be included in the knapsack without exceeding its capacity while maximizing profit. This problem was first introduced by Dantzig [14] in the mid-20th century and has since garnered significant attention in the field of optimization. Over decades, numerous research efforts have been undertaken to discover the most efficient and effective algorithms to solve the knapsack problem [15]. There are many variants of the knapsack problem developed, and the most basic version is the 0-1 knapsack problem, also known as the binary knapsack problem. In this version, items can only be either fully included or not included at all in the knapsack. It is like packing a bag for a trip where each item must either go in the bag completely or stay out.

According to [16], despite the apparent simplicity of its mathematical formulation, the binary knapsack problem remains difficult to solve due to its NP-hard nature, which poses significant computational challenges in finding optimal solutions within a reasonable time, especially as the problem size increases. Generally, the techniques for solving the binary knapsack problem can be divided into two groups, namely exact techniques and approximation or metaheuristic techniques [17]. Since the knapsack problem is NP-hard, the exact technique can only perform well where the capacity of the knapsack and the weights of the items are relatively small. For larger instances, exact techniques may not be feasible due to the exponential time complexity of exploring all possible combinations of items. Therefore, the approximation algorithms or heuristic methods are often used to find near-optimal solutions in a reasonable amount of time. There are many heuristic methods used to solve the binary knapsack problem, such as the Evolutionary Algorithm [18], Differential Evolution (DE) [19], Genetic Algorithm (GA) [20], Particle Swarm Optimization [21], Frog Leaping Algorithm [22], Binary Kepler Optimization Algorithm [17], Harmony Search (HS) Algorithm [23], Firefly Algorithm (FA) [24], and Quantum-inspired Social Evolution (QSE) algorithm [25]. Thus, in this paper, the performance of LaF is evaluated and compared to the others.

2. METHODS

2.1. Binary Knapsack Problem

The binary knapsack problem (known as 0-1 knapsack problem) is a classical combinatorial optimization problem that revolves around a scenario where a knapsack of limited capacity must be filled with a selection of items, each characterized by a weight and a value [26]. The objective is to maximize the total value of the items included in the knapsack while ensuring that the combined weight does not exceed the knapsack's capacity [27]. Crucially, items can only be selected or rejected, hence the term is "binary" knapsack.

The mathematical model of the Binary Knapsack Problem is as follows:

$$Max\ f(x) = \sum_i^n v_i x_i \tag{1}$$

Subject to

$$\sum_i^n w_i x_i \leq C \tag{2}$$

Where item set $U = \{u_1, u_2, \dots, u_n\}, x_i = 0\ or\ 1, i = 1, 2, \dots, n$. The $x_i = 1$ if item u_i is selected and loaded into the knapsack, whereas if item is not selected for loading into the knapsack. w_i denotes the weight ofu_i and v_i indicates the value or profit of u_i.

2.2. Leaders and Followers Algorithm for the Binary Knapsack Problem

LaF algorithm is a metaheuristic algorithm that employs two distinct populations: LaF. Each population comprises an equal number of elements but serves different functions. The Leaders population stores promising solutions, potentially global optima, while the Followers population explores new 'attraction basins,' which are sets of solutions containing local optima. To mitigate premature convergence, LaF refrains from directly comparing newly discovered solutions with the current best solution.

Initially, the populations are generated randomly. Given the combinatorial nature of the binary knapsack problem, each individual in the population is represented by an array of 0s and 1s of length. After that, new solutions are created through a mating process involving a randomly selected leader (an individual from the Leaders population) and a randomly selected follower (an individual from the Followers population). In this paper, one-point crossover, which is commonly employed in Genetic Algorithms (GAs) [28], is utilized as the mating technique. One point is randomly selected and the crossover is conducted based on the point. There are two new solutions created, but only the better one is selected to be the trial. These newly generated solutions (trials) are then compared with the followers, which are their parents. If a trial is superior to the follower, it replaces the follower's position within the Followers population.

The process after that has slighty change from the original LaF algorithm by [1]. The algorithm conducts elitism to store the best found solution as the first leader in its population. This is important as at

many times, there are already the optimal solution in the population, but then it is removed because it is not selected in the selection process. Another change is the comparison between median value of both population. In the original algorithm the comparison determines whether there will be update in Leaders population or not. If the median of Followers population is higher or equal than the Leaders population, the algorithm will do the replacement process. In this paper, the comparison of median values is replaced by taking a random value between 0 to 1. If the value is higher than 0.5, then the replacement in Leaders population is conducted. If it is not or the replacement process finished, the algorithm continue to randomly generate the new Follower population and repeat the process until the termination criterion.

The replacement process is basically a binary tournament selection between a random pair of leader and follower. The winners will replace the current leaders in the Leaders population, except for the first index, which is reserved for the current best solution.

2.3. Algorithm Evaluation

The dataset used for evaluating the proposed algorithm is presented in Table 1. The problem p01-p08 are from : "https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html". Each problem was evaluated in twenty-five independent runs. The population size was 100 and the iteration stop after there is no better solution in 1000 consecutive iterations. To handle the infeasible solutions, the objective function is subtracted with a penalty value of 1000×max(0, total weight – Capacity), if there was constraint violation. The algorithm was run on Windows 11 with a 12th Gen Intel® Core™ i7-1255U (1.70 GHz) processor and 16 GB of RAM, using Python 3.13.2. The evaluation results were then compared to the results reported by other algorithms in their papers: DE [19], original FA [24], FA of Attractiveness and Movement (FA-AM) [24], Attractiveness FA (A-FA) [24], Movement-FA (M-FA) [24], HS [29], Greedy Search Algorithm (GSA) [30], Dynamic Programming (DP) [30], Branch and Bound (BB) [30], GA [30], Simulated Annealing (SA) [30], Greedy Search Algorithm (GSA) [30], and Discrete Shuffled Frog Algorithms (SF) [22].

Table 1. Dataset

Dataset	Dimension	Parameter (weight, profit)	Capacity
p01	10	Weights: 23, 31, 29, 44, 53, 38, 63, 85, 89, 82 Profits: 92, 57, 49, 68, 60, 43, 67, 84, 87, 72	165
p02	5	Weights: 12, 7, 11, 8, 9 Profits: 24, 13, 23, 15, 16	26
p03	6	Weights: 56, 59, 80, 64, 75, 17 Profits: 50, 50, 64, 46, 50, 5	190
p04	7	Weights: 31, 10, 20, 19, 4, 3, 6 Profits: 70, 20, 39, 37, 7, 5, 10	50
p05	8	Weights: 25, 35, 45, 5, 25, 3, 2, 2 Profits: 350, 400, 450, 20, 70, 8, 5, 5	104
p06	7	Weights: 41, 50, 49, 59, 55, 57, 60 Profits: 442, 525, 511, 593, 546, 564, 617	170
p07	15	Weights: 70, 73, 77, 80, 82, 87, 90, 94, 98, 106, 110, 113, 115, 118, 120 Profits: 135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240	750
p08	24	Weights: 382745, 799601, 909247, 729069, 467902, 44328, 34610, 698150, 823460, 903959, 853665, 551830, 610856, 670702, 488960, 951111, 323046, 446298, 931161, 31385, 496951, 264724, 224916, 169684 Profits: 825594, 1677009, 1676628, 1523970, 943972, 97426, 69666, 1296457, 1679693, 1902996, 1844992, 1049289, 1252836, 1319836, 953277, 2067538, 675367, 853655, 1826027, 65731, 901489, 577243, 466257, 369261	6404180
f01	10	Weights: 95, 4, 60, 32, 23, 72, 80, 62, 65, 46 Profits: 55, 10, 47, 5, 4, 50, 8, 61, 85, 87	269
f02	20	Weights: 92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58 Profits: 44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63	878
f03	4	Weights: 6, 5, 9, 7 Profits: 9, 11, 13, 15	20
f04	4	Weights: 2, 4, 6, 7 Profits: 6, 10, 12, 13	11
f05	15	Weights: 56.358531, 80.87405, 47.987304, 89.59624, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575 Profits: 0.125126, 19.330424, 58.500931, 35.029145, 82.284005,	375

Dataset	Dimension	Parameter (weight, profit)	Capacity
		17.41081, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.89114, 53.166295, 60.176397	
f06	10	Weights: 30, 25, 20, 18, 17, 11, 5, 2, 1, 1 Profits: 20, 18, 17, 15, 15, 10, 5, 3, 1, 1	60
f07	7	Weights: 31, 10, 20, 19, 4, 3, 6 Profit: 70, 20, 39, 37, 7, 5, 10	50
f08	23	Weights: 983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959 Profits: 81, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857	10000
f09	5	Weights: 15, 20, 17, 8, 31 Profits: 33, 24, 36, 37, 12	80
f10	20	Weights: 84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92 Profits: 91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44	879

3. RESULTS AND DISCUSSION

The evaluation results and comparisons with other algorithms are presented in Tables 2 and 3. For p01-p07, f01-f07, f09, and f10, LaF consistently performed well, achieving the optimal solution in every repetition and outperforming other algorithms. However, in p08, LaF failed to find the optimal solution, although it still outperformed most algorithms, except HS. This discrepancy does not necessarily indicate that HS is superior to LaF, as LaF performed better than HS in f01-f08.

Moreover, LaF struggled with problems involving large capacity values, as evidenced by its slightly inferior performance compared to GA in f08, despite outperforming GA in other functions. However, this does not imply that GA is superior to LaF, especially considering GA's notably poorer performance in f05. Overall, LaF demonstrates superior performance compared to all other algorithms, particularly in solving binary knapsack problems with moderately sized capacities.

Table 2. Comparison of Evaluation Results between LaF and other algorithms on p01-p10

Problem		LaF	DE [24]	FA [10]	FA-AM [10]	A-FA [10]	M-FA [10]	HS [11]
p01	Best	309	309	309	309	309	309	309
	Mean	309	305	293	305.8	302.6	302.6	-
	Worst	309	284	277	277	277	277	-
p02	Best	51	51	51	51	51	51	51
	Mean	51	51	51	51	51	51	-
	Worst	51	51	51	51	51	51	-
p03	Best	150	150	150	150	150	150	150
	Mean	150	150	150	150	150	150	-
	Worst	150	150	150	150	150	150	-
p04	Best	107	107	107	107	107	107	107
	Mean	107	107	107	107	107	107	-
	Worst	107	107	107	107	107	107	-
p05	Best	900	900	900	900	900	900	900
	Mean	900	900	895.6	899.3	894.7	899.2	-
	Worst	900	900	883	895	883	898	-
p06	Best	1735	1735	1735	1735	1735	1735	1735
	Mean	1735	1735	1712.3	1726	1717	1712	-
	Worst	1735	1735	1682	1692	1688	1682	-
p07	Best	1458	1458	1444	1452	1454	1449	1458
	Mean	1457.6	1456	1431.4	1445.7	1437.7	1440.3	-
	Worst	1455	1448	1415	1437	1430	1434	-
p08	Best	13496672	13496748	13399518	13292007	13458609	13334101	13549094
	Mean	13436907.9	13345362	13211679	13199650	13198131	13220431	-
	Worst	13364586	13229190	13114239	13122592	13040689	13092294	-

Table 3. Comparison of Evaluation Results between LaF and other algorithms on f01-f04

Problem		LaF	BB [9]	DP [9]	GA [9]	GSA [9]	SA [9]	HS [11]	SF-1 [12]	SF-2 [12]	SF-3 [12]
f1	Best	295	280	295	295	288	294	295	295	295	295
	Mean	295	-	-	-	-	-	167.4	287.8	287.24	291.7
	Worst	295	-	-	-	-	-	-	269	253	279

Problem		LaF	BB [9]	DP [9]	GA [9]	GSA [9]	SA [9]	HS [11]	SF-1 [12]	SF-2 [12]	SF-3 [12]
f2	Best	1024	972	1024	1024	1024	972	945	955	958	950
	Mean	998.7	-	-	-	-	-	560.16	868	868,77	872,03
	Worst	963	-	-	-	-	-	-	816	815	801
f3	Best	35	24	35	35	28	35	35	35	35	35
	Mean	35	-	-	-	-	-	26.5	35	35	35
	Worst	35	-	-	-	-	-	-	35	35	35
f4	Best	23	22	23	23	19	22	23	23	23	23
	Mean	23	-	-	-	-	-	14.94	23	23	23
	Worst	23	-	-	-	-	-		23	23	23
f5	Best	481.07	469.16	477	477	481.07	481.07	481.06	454.42	466.34	481.07
	Mean	479.08	-	-	-	-	-	481.06	402.9	406.41	408.55
	Worst	469	-	-	-	-	-		362.98	356.76	352.35
f6	Best	52	49	52	52	43	52	52	52	52	52
	Mean	52	-	-	-	-	-	40.57	51.52	51.62	51.63
	Worst	52	-	-	-	-	-		49	49	50
f7	Best	107	96	107	107	107	107	107	107	107	107
	Mean	107	-	-	-	-	-	52.58	106.79	106.72	106.73
	Worst	107	-	-	-	-	-		105	105	105
f8	Best	9765	-	-	9767	-	9704	9731	9755	9752	9759
	Mean	9764	-	-	-	-	-	7226.6	9736.13	9732.93	9733.47
	Worst	9761	-	-	-	-	-	-	9718	9712	9707
f9	Best	130	-	-	130	-	-	-	130	130	130
	Mean	130	-	-	-	-	-	-	130	130	130
	Worst	130	-	-	-	-	-	-	130	130	130
f10	Best	1025	-	-	1025	-	973	-	951	952	1010
	Mean	1005.9	-	-	-	-	-	-	868.43	874.07	879.9
	Worst	973	-	-	-	-	-	-	810	822	811

4. CONCLUSION

Based on the results and discussion, it can be concluded that the LaF algorithm exhibits stability in solving binary knapsack problems with moderately sized capacities and demonstrates superior performance compared to other algorithms. However, it encounters challenges when tasked with problems of substantial capacity. It is recommended to conduct further investigation using larger populations, additional iterations, and different techniques in creating new individual (trials) and enhance the exploration.

REFERENCES

[1] Y. Gonzalez-Fernandez and S. Chen, "Leaders and followers-A new metaheuristic to avoid the bias of accumulated information," in *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Sep. 2015, pp. 776–783. doi: 10.1109/CEC.2015.7256970.

[2] H. Y. Angmalisang, H. Angmalisang, and S. J. A. Sumarauw, "Leaders and Followers Algorithm for Balanced Transportation Problem," *Computer Engineering and Applications*, vol. 12, no. 2, 2023.

[3] H. Y. Angmalisang, H. Angmalisang, and S. J. A. Sumarauw, "Leaders and Followers Algorithm for Balanced Transportation Problem," *Computer Engineering and Applications*, vol. 12, no. 2, 2023, Accessed: Oct. 01, 2023. [Online]. Available: https://comengapp.unsri.ac.id/index.php/comengapp/article/view/436/265

[4] H. Angmalisang and S. Anam, "Leaders and Followers Algorithm For Traveling Salesman Problem," *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, vol. 18, no. 1, pp. 449–456, Mar. 2024, doi: 10.30598/barekengvol18iss1pp0449-0456.

[5] N. Morimoto, "Power allocation optimization as the multiple knapsack problem with assignment restrictions," in *2017 8th International Conference on the Network of the Future (NOF)*, 2017, pp. 40–45. doi: 10.1109/NOF.2017.8251218.

[6] N. Ferdosian, M. Othman, K. Y. Lun, and B. M. Ali, "Optimal solution to the fractional knapsack problem for LTE overload-state scheduling," in *2016 IEEE 3rd International Symposium on Telecommunication Technologies (ISTT)*, 2016, pp. 97–102. doi: 10.1109/ISTT.2016.7918092.

[7] F. Vaezi, S. J. Sadjadi, and A. Makui, "A Robust Knapsack Based Constrained Portfolio Optimization," *International Journal of Engineering*, vol. 33, no. 5, pp. 841–851, 2020, doi: 10.5829/ije.2020.33.05b.16.

[8] V. Traneva, P. Petrov, and S. Tranev, "An Elliptic Intuitionistic Fuzzy Portfolio Selection Problem based on Knapsack Problem," in *Communication Papers of the 18th Conference on Computer Science and Intelligence Systems*, PTI, Oct. 2023, pp. 335–342. doi: 10.15439/2023f4882.

[9] Y. Laalaoui and R. M'Hallah, "A binary multiple knapsack model for single machine scheduling with machine unavailability," *Comput Oper Res*, vol. 72, pp. 71–82, Aug. 2016, doi: 10.1016/J.COR.2016.02.005.

[10] M. Samavati, D. Essam, M. Nehring, and R. Sarker, "A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry," *Int J Prod Econ*, vol. 193, pp. 12–20, Nov. 2017, doi: 10.1016/J.IJPE.2017.06.025.

[11] J. and M. G. Pferschy Ulrich and Schauer, "A Quadratic Knapsack Model for Optimizing the Media Mix of a Promotional Campaign," in *Operations Research and Enterprise Systems*, F. and V. B. Pinson Eric and Valente, Ed., Cham: Springer International Publishing, 2015, pp. 251–264.

[12] X. Hao *et al.*, "Dynamic Knapsack Optimization Towards Efficient Multi-Channel Sequential Advertising," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., in Proceedings of Machine Learning Research, vol. 119. PMLR, Apr. 2020, pp. 4060–4070. [Online]. Available: https://proceedings.mlr.press/v119/hao20b.html

[13] N. Ferdosian, M. Othman, B. M. Ali, and K. Y. Lun, "Greedy–knapsack algorithm for optimal downlink resource allocation in LTE networks," *Wireless Networks*, vol. 22, no. 5, pp. 1427–1440, 2016, doi: 10.1007/s11276-015-1042-9.

[14] G. B. Dantzig, "Discrete-Variable Extremum Problems," *Oper Res*, vol. 5, no. 2, pp. 266–288, Apr. 1957, doi: 10.1287/opre.5.2.266.

[15] C. Cacchiani, M. Iori, A. Locatelli, and S. Martello, "Knapsack problems-An Overview of Recent Advances. Part I: Single Knapsack Problems," 2022. [Online]. Available: https://www.elsevier.com/open-access/userlicense/1.0/

[16] N. Acevedo, C. Rey, C. Contreras-Bolton, and V. Parada, "Automatic design of specialized algorithms for the binary knapsack problem," *Expert Syst Appl*, vol. 141, p. 112908, 2020, doi: https://doi.org/10.1016/j.eswa.2019.112908.

[17] 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.

[18] J. He, B. Mitavskiy, and Y. Zhou, "A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 141–148. doi: 10.1109/CEC.2014.6900442.

[19] I. M. Ali, D. Essam, and K. Kasmarik, "An Efficient Differential Evolution Algorithm for Solving 0–1 Knapsack Problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477916.

[20] V. Jain and J. S. Prasad, "Applying greedy genetic algorithm on 0/1 multiple knapsack problem," *International Journal of Advanced Technology and Engineering Exploration*, vol. 5, no. 45, pp. 292–296, Aug. 2018, doi: 10.19101/ijatee.2018.545018.

[21] Wei-Zhong Sun, Min Zhang, Jie-Sheng Wang, Sha-Sha Guo, Min Wang, and Wen-Kuo Hao, "Binary Particle Swarm Optimization Algorithm Based on Z-shaped Probability Transfer Function to Solve 0-1 Knapsack Problem," *IAENG Int J Comput Sci*, vol. 48, no. 2, pp. 294–303, 2021, Accessed: Apr. 13, 2024. [Online]. Available: http://www.iaeng.org/IJCS/issues_v48/issue_2/IJCS_48_2_09.pdf

[22] K. K. Bhattacharjee and S. P. Sarmah, "Shuffled frog leaping algorithm and its application to 0/1 knapsack problem," *Applied Soft Computing Journal*, vol. 19, pp. 252–263, 2014, doi: 10.1016/j.asoc.2014.02.010.

[23] A. C. Adamuthe, V. N. Sale, and S. U. Mane, "Solving single and multi-objective 01 Knapsack Problem using Harmony Search Algorithm," *Journal of scientific research*, vol. 64, no. 01, pp. 160–167, 2020, doi: 10.37398/jsr.2020.640136.

[24] D. David, E. V. H. S, R. Ronny, and T. Widayanti, "Modification of Attractiveness and Movement of the Firefly Algorithm for Resolution to Knapsack Problems," in *2022 4th International Conference on Cybernetics and Intelligent System (ICORIS)*, 2022, pp. 1–5. doi: 10.1109/ICORIS56080.2022.10031553.

[25] R. S. Pavithr and Gursaran, "Quantum Inspired Social Evolution (QSE) algorithm for 0-1 knapsack problem," *Swarm Evol Comput*, vol. 29, pp. 33–46, Aug. 2016, doi: 10.1016/J.SWEVO.2016.02.006.

[26] G. Yildizdan and E. Baş, "A Novel Binary Artificial Jellyfish Search Algorithm for Solving 0–1 Knapsack Problems," *Neural Process Lett*, vol. 55, no. 7, pp. 8605–8671, 2023, doi: 10.1007/s11063-023-11171-x.

[27] 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, 2021, doi: https://doi.org/10.1016/j.cie.2020.106946.

[28] A. J. Umbarkar and P. D. Sheth, "Crossover Operators In Genetic Algorithms: A Review," *ICTACT Journal on Soft Computing*, vol. 06, no. 01, pp. 1083–1092, Oct. 2015, doi: 10.21917/ijsc.2015.0150.

[29] A. C. Adamuthe, V. N. Sale, and S. U. Mane, "Solving single and multi-objective 01 Knapsack Problem using Harmony Search Algorithm," *Journal of scientific research*, vol. 64, no. 01, pp. 160–167, 2020, doi: 10.37398/jsr.2020.640136.

[30] 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.