Nature-Inspired Computing Coursework for Bank Text Problem using Ant Colony Optimization

Student Id: 740089416
University of Exeter
ECMM409 ACO

ABSTRACT

This paper addresses a solution for solving a 0/1 knapsack problem variant called Bank text problem. Knapsack problems are a type of NP- hard problem. It leverages the use of nature inspired algorithms, called Ant Colony Optimization algorithm (ACO). It also explores the use of hybrid variants of this algorithm using differential pheromone grouping. It also explores prior research on multi-dimensional hybrid colonies to enhance performance. It also tries to enhance ACO results by implementing bat algorithms to tackle the balance between exploration and exploitation. The ACO and bat algorithms developed was applied to the Bank Text problem, its optimal parameters were studied along with variations in population, evaporation rate and deposition rate. Future work also explores progress in reinforcement learning, hybrid bat algorithms as well as exploratory analysis of modified genetic algorithms.

Keywords: Nature-inspired optimization, Combinatorial problems, Knapsack Problem, Bin Packing Problem, Multiple Knapsack Problem, Ant Colony Optimization, Genetic Algorithm, Hybrid Ant Colony algorithm, Bat Algorithms.

LITERATURE REVIEW

The given question is a very well-known problem called the Knapsack problem which is a fundamental optimization problem with extensive research and numerous real-world applications. Ever since the pathbreaking work of by Martello, Toth, Kellerer, Pferschy, and Pisinger Knapsack problems have emerged as a very popular combinational optimization problem [1,2]. Building on the foundational work of Martello et.al, Pan and Zhang applied rough set theory to refine solutions to 0/1 Knapsack problems [3]. In the aforementioned research paper, the authors use rough set theory to approximate uncertain weights or values. Ant colony optimization (ACO) emerged as a popular algorithm for combinatorial optimization, where the foraging behaviour of ants is simulated. The time complexity of the proposed algorithm was also analysed [4]. In another notable addition in [5,6], authors introduce differential and weight-based pheromone ant colony optimization (ACO) algorithms by developing DpG-ACO a Differential pheromone grouping Ant Colony Optimization approach.

1. A gradual weight-based ant colony approach for solving knapsack problems.

This paper uses a modified ant colony algorithm to solve multidimensional knapsack problem, which is an extension of the traditional knapsack problem. In this literature, the authors introduce a novel approach of gradually introducing weights along the ant path. The authors term this method as Gw-ACO. The weight vectors are based on the fact that they change relatively based on the optimization process. This modification, allows the ant in the cycle to selectively target different regions of the graph to cover a much wider range of solutions found along the knapsack vector. The gradual weight vector (Ğ(g)) is given in equation 1 and 2 where g is the generation vector and FQ is the path length:

$$\check{G}(g)1 = \ln\left[\left(\frac{4\times g\times e}{FQ}\right) + \cos\left(\frac{2\times g\times \Pi}{FQ}\right)\right] \quad (1)$$

$$\check{G}(g)2 = 1.0 - \check{G}(g)1 \quad (2)$$

ACO algorithms have proved to be an efficient solution to solve optimization problems such as knapsack problems as explored by modifications to the standard ACO algorithm by introducing multi-dimensional search termed as the Multi-Dimensional Hybrid Ant Colony Optimization (MD-HACO) which showed a good balance between convergence and diversity [7].

2. An Effective Hybrid Ant Colony Optimization for the Knapsack Problem Using Multi Directional Search

This literature seeks to effectively tackle the trade-off between local search and global search, which is a common problem faced by knapsack problems. The resultant algorithm proposed, is called the Multi-Dimensional Hybrid Ant Colony Optimization (MD-HACO) by utilizing different vectors to utilize small subsections of the data. Afterwards, local search is applied to each subset. With this multi stage approach, the algorithm effectively addresses the exploration and exploitation trade-off. However, this approach proves to be quite complexity heavy as a two-stage approach it used.

In a larger context, hybrid heuristic two stage memetic approach is chosen to solve multidimensional knapsack problems [8]. Feng et al. introduce a novel monarch butterfly optimization technique to solve such problems by simulating the behaviour of monarch butterflies by assigning two tuples of real valued vectors and binary vectors [9] while genetic mutation bat algorithms are used to solve 0/1 Knapsack problems by combining Binary Bat Algorithm (BBA) and Local Search Scheme (LSS) [10]. BBA works by simulating echolocation by bats by updating two metrics, velocity and position by creating probabilistic distributions. LSS explores nearby solutions to check for improvements on the solution. Paired with BBA, it creates a powerful optimization algorithm.

3. Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization

The paper presented develops a novel binary butterfly optimizer algorithm (BMBO) which is based on the behaviour of a pair of individual monarch butterflies. The authors demonstrate two real valued and binary vectors which represent the two monarch butterflies in vector space. It also introduces a method to penalize infeasible solutions while optimizing efficient and higher valued solutions, with an algorithm based on greedy matching. It also introduces three types of population strategies based on population initialization. The one-to-one mapping as represented by the monarch butterflies is given by g(x) in **equation 3:**

G(x) = 1 if sig(x) >= 0.5

G(x) = 0 else. Where sig(x) = $1/(1+e^{-x})$ (3)

4. A Complex-valued Encoding Bat Algorithm for Solving 0–1 Knapsack Problem

The paper introduces a novel algorithm based on the natural behaviour of bats. Bats use echolocation to move from one place to another. It uses the real and complex parts of a number to effectively diversify bat population. This diversification vastly improves algorithm convergence. The authors introduce this algorithm as a Complex-Valued Encoding Bat Algorithm (CPBA) to effectively tackle the 0/1 knapsack problem. It uses complex valued encoding as follows:

Step 1: Initialize complex 2M modulus and phase angles.

X_{Rk}+ iX_{Ik} = ρ_k(cosφ + I sinφ) k = 1,2,3…...2M

Step 2: Update the real parts.

V_R(t+1) = V_R(t) + (Xr (t) – best1) Q(t)

X_R(t+1) = X_R(t) + V_R(t+1)

Step 3: Update the imaginary parts.

V_i(t+1) = V_i(t) + (X_i(t) – best2) Q2(t)

X_i(t+1) = X_i(t) + V_i(t+1)

Many real-life applications such as budget allocation, cargo loading and investment decision making are based on knapsack problems and genetic algorithms are explored to solve these issues since these are all optimization problems which ACO is well poised to handle [11,12,13,14]. Pisinger et.al, explore heuristics to solve the container loading problem while Renier and Sorensen explore the knapsack problem for optimal allocation of resources [13,14]. Hence, evolutionary algorithms in general, and hybrids of ACO, BBA and LSS in particular are highly efficient algorithms to solve 0/1 knapsack problems.

METHOD

Algorithm Design

Let us first explore the Algorithm design for the proposed ACO algorithm:

Step 1: Parameter Initialization:

1. Read bank sack data from "BankProblem.txt" file.

2. Sort the sacks available in the data according to their weights.

3. Initialize the pheromone bank with random values between 0 and 1.

4. Create arrays to store fitness, best fitness value (most money) and the entire optimum path.

Step 2: For each generation:

1. Create array to store individual path of each ant.

2. Construct ant path.

3. For each ant:

• Create an array to store selected sacks.

• Initialize the path starting at the first sack.

• Create a variable to track the total weight of selected sacks.

• **While ant!= solution:**

Calculate the probability for each sack as $P_{ab} = \frac{\tau_{ab}^{\alpha} \times \eta_{ab}^{\beta}}{\Sigma_k \tau_{ak}^{\alpha} \eta_{ak}^{\beta}}$

P_{ab}= probability of selecting the edge (a,b).

τ_{ab}= pheromone level on edge (a,b).

η_{ab}= inverse of weight (heuristic value)

α= pheromone influence

β= heuristic influence

Randomly select sack based on calculated probabilities.

Add the sack to path length.

Update the total weight and remove any excess weight.

4. Calculate Fitness:

- Fitness[ant] = $\Sigma_{a \epsilon ant\_path}$ value(a)

5. Update best fit solution:

- Find maximum fitness value
- If this value is greater than previous fitness value, update the variable.

6. Pheromone Update:

- Evaporate the pheromone level by the evaporation rate: $\tau_{ab} \cdot \rho$
- Deposit fresh pheromone: $\tau_{ab} = \tau_{ab} + \frac{m}{fitness}$

7. Return Final result after finishing all the generations.

The flowchart and construction graph for the algorithm is given in **Figure 1** and **Figure 2** respectively.

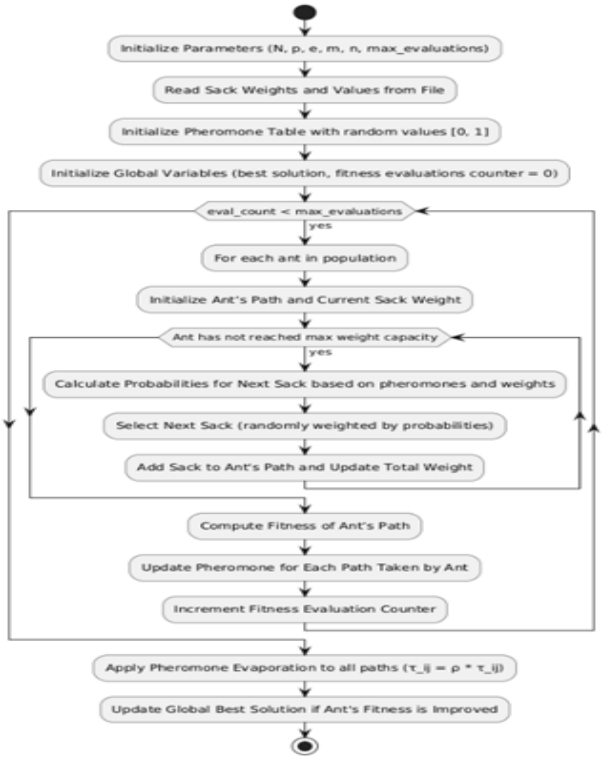

Figure 1: Flowchart for the proposed algorithm.

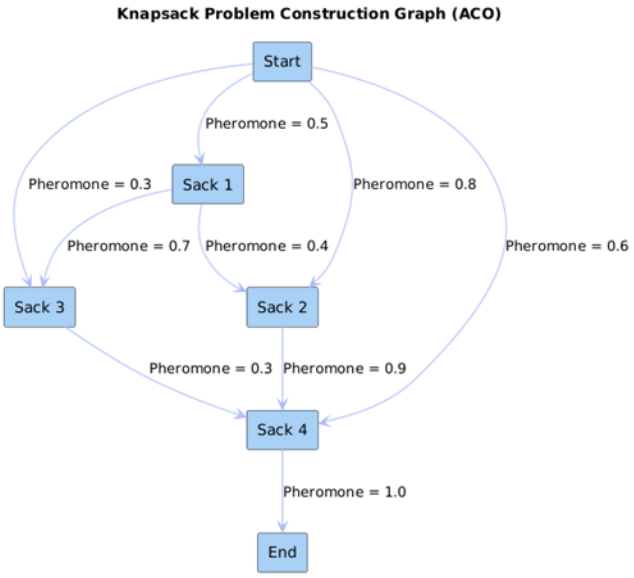

Figure 2: Construction graph for the given algorithm

DESCRIPTION OF RESULTS

The algorithm is tested for different hyperparameters and the rate of convergence for the proposed ACO is tested against these hyperparameters.

1. For different ant population:

The proposed algorithm was tested against various populations of the ant colony, and its results were observed as shown in **Figure 3**. As seen in **Table 1**, most amount collected by the security van is reflected when ant population was n= 20. Reason: There is a common trade-off between faster convergence and computational cost. The data suggests that there is a sweet spot at n=20, where the higher number of ants provide negligible increase in convergence while still increasing computational costs. Besides, a large number of ants may have caused the algorithm to prematurely converge, not maintaining enough diversity to explore new solutions. This also shows an exploration vs exploitation trade-off.

2. For different evaporation rates:

The algorithm is also compared for various evaporation rates to estimate an optimum evaporation rate for algorithm convergence. The resultant result is displayed in **Figure 4**, and the best results are observed when evaporation rate e=0.92 as shown in **Table 2**. Reason: Evaporation rates control the level of persistence of pheromone in the ant trail. Lower evaporation rate therefore, indicate more pheromone persistence. This lower pheromone evaporation indicates a much faster convergence rate hence a lower value since exploration is traded off for exploitation. An evaporation rate of 0.92 indicates a strong trade-off where optimum paths are converged while still allowing the algorithm to maintain flexibility to explore new paths and disincentivises premature convergence solutions.

3**. For different Deposition rates:**

Finally, sack values are also calculated for different deposition rates where the best results obtained were at m = 0.00001. These observations are clearly reflected in **Figure 5** and the corresponding table as shown in **Table 3**. Reason: In the aforementioned knapsack problem, the observed deposition rate provides a good trade-off to prevent any local minima that may arise when excessive deposition rates are used. At the same time, the deposition rates are high enough that an efficient convergence can be obtained. Hence, the table explains our prior assumptions regarding the trade-off between flexibility of an ant colony to take alternative paths while still providing it with enough convergence where ants can reinforce any efficient solution that the algorithm may derive.

Ant Population	Value (£)
5	4222
10	4411
15	4475
20	4520
50	4517
100	4515

Table 1: Population variation and value

Evaporation Rate	Value (£)
0.75	4515
0.82	4500
0.85	4515
0.9	4514
0.92	4524
0.95	4499

Table 2: Evaporation rate variation and value

Pheromone Deposition	Value (£)
0.000005	4517
0.00001	4527
0.00005	4522
0.0001	4525
0.00015	4522
0.0002	4521

Table 3: Variation of Pheromone Deposition rate and Value

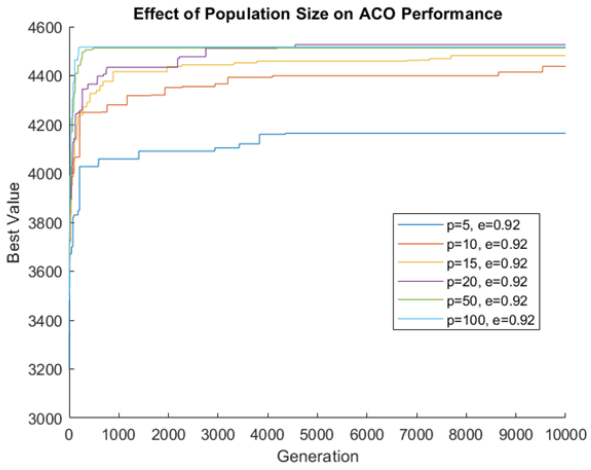

Figure 3: Population variation

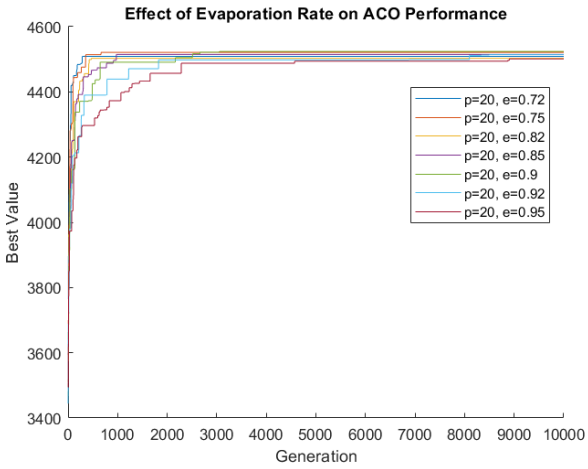

Figure 4: Evaporation rate variation

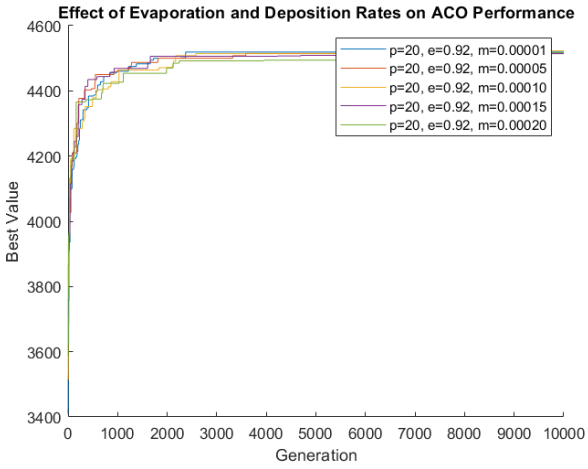

Figure 5: Deposition rate variation

4. Best parameters obtained:

Based on the parameter variations discussed above, **Figure 6** shows the graph for the fitness of 10 iterations at the best obtained parameters. **Table 4** shows the specifics of the answer obtained. The proposed algorithm shows that the maximum value that can be carried by the security van is 4527 units while the weight carried by the van is 294.7 kgs, which is under the 295 kg limit.

Proposed Algorithm	Hyperparameter
Ant population	20
Evaporation Rate	0.92
Pheromone Deposition	0.00001
Number of generations	10000
Value carried by van	294.7
Weight carried by van	4527

Table 4: Proposed Algorithm optimum results.

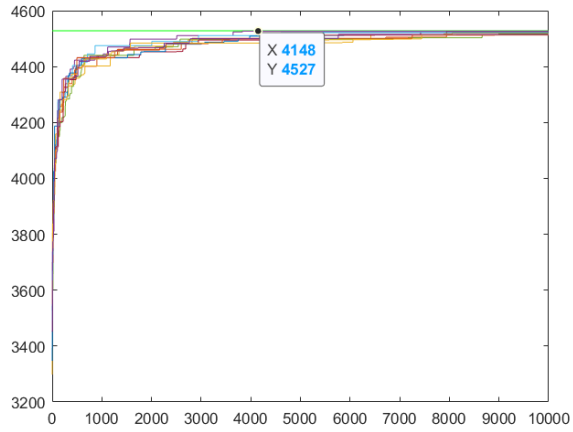

Figure 6: Best fit of the ACO algorithm

DISCUSSION

Question1: Which combination of parameters produces the best results?

Ans: Based on the parameter variations discussed above, Figure 4 shows the graph for the fitness of 5 iterations at the best obtained parameters. Table 5 shows the specifics of the answer obtained. The proposed algorithm shows that the maximum value that can be carried by the security van is 4527 units while the weight carried by the van is 294.7 kgs, which is under the 295 kg limit. Population = 20 ants. Pheromone deposition rate = 0.00001. Evaporation rate = 0.92.

Question 2: What do you think is the reason for your findings in Question 1?

Ans: The algorithm uses a very high number of iterations since knapsack problems are combinatorial problems and may require a high number of iterations. The pheromone deposition rate is relatively moderate since it makes sure that the ants contribute to the pheromone trail, which still searches for the most efficient trail while still letting the ants have the freedom to explore other viable paths. The evaporation rate of 0.92 encourages fresh pathways while still letting the colony to not abandon seemingly good paths. The number of ants is a healthy 20 as it allows the ants numbers to explore different solution combinations while not being so high that computational complexity manageable.

Question 3: How do each of the parameter settings influence the performance of the algorithm?

Ans: Influence of ant population- There is a common trade-off between faster convergence and computational cost. The data suggests that there is a sweet spot at n=20, where the higher number of ants provide negligible increase in convergence while still increasing computational costs. Besides, a large number of ants may have caused the algorithm to prematurely converge, not maintaining enough diversity to explore new solutions. This also shows a exploration vs exploitation trade-off.

Influence of deposition rate- In the aforementioned knapsack problem, the observed deposition rate provides a good trade-off to prevent any local minima that may arise when excessive deposition rates are used. At the same time, the deposition rates are high enough that an efficient convergence can be obtained. Hence, the table explains our prior assumptions regarding the trade-off between flexibility of an ant colony to take alternative paths while

still providing it with enough convergence where ants can reinforce any efficient solution that the algorithm may derive.

Influence of evaporation rate- Evaporation rates control the level of persistence of pheromone in the ant trail. Lower evaporation rate therefore, indicate more pheromone persistence. This lower pheromone evaporation indicates a much faster convergence rate hence a lower value since exploration is traded off for exploitation. An evaporation rate of 0.92 indicates a strong trade-off where optimum paths are converged while still allowing the algorithm to maintain flexibility to explore new paths and disincentivises premature convergence solutions.

Question 4: Do you think that one of the algorithms in your literature review might have provided better results? Explain your answer.

Ans: Based on the literature review, it is hypothesized that the bat algorithm as proposed in [10] will show promising results when applied on Bank Text problem. In addition, genetic algorithms as previously mentioned can also show promising results when tested on the problem in question. Bat algorithms have shown great promise in solving NP-hard problems as demonstrated in [15]. This was hypothesized as bat algorithms tackle many of the problems highlighted by ant optimization algorithms. A big feature of ant algorithms was the constant struggle between balancing exploration and exploitation. Bat algorithms tackle this by using local search for exploitation and global search for exploration. Loudness is responsible for exploration while bat algorithms use pulse frequency for exploitation. Furthermore, bat algorithms simply penalize solutions that yield results which exceed bag capacity. As discussed above, further experiments were performed on Bat algorithms to check its performance on the BankText problem. The algorithm for this proposed algorithm is discussed below.

ADDITIONAL EXPERIMENTATION

Results of the algorithm: As hypothesized, the implemented bat colony performs marginally better than the ACO algorithm on the total value carried. The main difference is that there is a noticeable decrease in weight that is utilized by the security van. **Figure 8** clearly shows that there is a decrease of nearly 0.5 over the ACO algorithm while still holding the total value carried. **Table 5** specifies the parameters of the algorithm used. Value obtained: 4528, Weight utilized: 294.1.

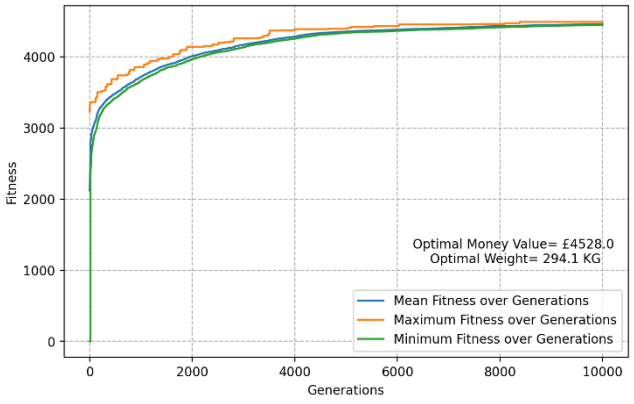

Figure 8: Algorithm performance

Parameter Setting	Parameter Value
Bat Population	50
Maximum Iterations	10000
Loudness	1.0 initially
Pulse Rate	0.5
Frequency	Between 0 and 1
Convergence Rate	0.99
Velocity	Between -1 and 1

Table 5: Algorithm parameters

Further work: While bat algorithms showed promising results, there is still scope for improvement on results. Significant improvements might be studied with the application of novel reinforcement learning techniques like Deep Reinforcement Learning to solve the knapsack problem, in [16], Sur et.al propose an 'Actor Critic' algorithmic system to improve model performance. Since results of bat algorithm showed promising results, hybrid bat algorithms can be further explored as demonstrated by [17]. Early tests were also performed on genetic algorithms. Formula for the evaluation fitness is given in formula 1. Tested results are shown in figure 1 and 2. Modifications to this algorithm may show promising future results.

CONCLUSION

In this paper, we explore a traditional 0/1 knapsack problem and used nature inspired algorithms to solve the given NP hard problem. The proposed paper analyses the algorithms used in previously proposed research. The paper studies using Ant Colony Optimization algorithms with modifications of using differential pheromone grouping which showed high efficiency in solving the knapsack problem. Building on these results, another approach was explored which used multi-dimensional search which was termed as Multi-Dimensional Hybrid Ant Colony Algorithm (MD-HACO). Among other results that were studied, bat algorithms showed promising results and better efficiency in solving knapsack problems since they address the trade-off between exploration and exploitation.

The ACO algorithm was developed in this paper which was applied to solve the Bank Text problem. The algorithm was back tested against various permutations of algorithm parameters. It was observed that the best parameters were seen when ant population was 20, evaporation rate was 0.92, deposition rate was 0.00001. Based on these parameters, it was found that the maximum value that the van could carry was 4527 while still not exceeding the van capacity by peaking at a weight of 294.9.

It was also hypothesized that the Bat algorithm might outperform the proposed ACO in this Bank Text problem. Bat algorithm is well equipped to handle certain limitations of the Ant Colony Algorithm. The chief concern with the ACO algorithm was managing exploration of new paths while still reinforcing and converging on the efficient path in a certain time. The bat algorithm utilizes bat echolocation and utilizes pulse rate, velocity and position to tackle this trade-off. As hypothesized earlier, the implemented bat algorithm outperforms earlier implementation by returning a peak value of 4528 which is a slight improvement while still maintaining a lighter weight of 294.1. Lastly, the paper proposes further work by implementing reinforcement learning techniques, hybrid bat algorithm and use of modified genetic algorithms to improve results even further.

REFERENCES

[1] Martello, S., Pisinger, D., Toth, P., 1999. Dynamic programming and strong bounds for the 0-1 knapsack problem. Manage. Sci. 45, 414–424.

[2] Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack Problems. In: Springer Nature Book Archives Millennium, Springer.

[3] Xiaohui Pan and Tao Zhang 2018 J. Phys.: Conf. Ser. 1069 012024. doi :10.1088/1742-6596/1069/1/012024.

[4] L. He and Y. Huang, "Research of ant colony algorithm and the application of 0–1 knapsack," 2011 6th International Conference on Computer Science & Education (ICCSE), Singapore, 2011, pp. 464-467, doi: 10.1109/ICCSE.2011.6028680.

[5] Ben Mansour, I., Alaya, I. & Tagina, M. A gradual weight-based ant colony approach for solving the Mult objective multidimensional knapsack problem. Evol. Intel. 12, 253–272 (2019). https://doi.org/10.1007/s12065-019-00222-9.

[6] Aseel Ismael Ali, Edward Keedwell, and Ayah Helal. 2024. A Differential Pheromone Grouping Ant Colony Optimization Algorithm for the 1-D Bin Packing Problem. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24). Association for Computing Machinery, New York, NY, USA, 1463–1469. https://doi.org/10.1145/3638529.3654074.

[7] Imen BenMansour. 2023. An Effective Hybrid Ant Colony Optimization for the Knapsack Problem Using Multi-Directional Search. SN Comput. Sci. 4, 2 (Feb 2023). https://doi.org/10.1007/s42979-022-01564-5.

[8] Rezoug, A., Bader-El-Den, M. & Boughaci, D. Guided genetic algorithm for the multidimensional knapsack problem. Memetic Comp. 10, 29–42 (2018). https://doi.org/10.1007/s12293-017-0232-7.

[9] Feng, Y., Wang, GG., Deb, S. et al. Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. Neural Comput & Applic 28, 1619–1634 (2017). https://doi.org/10.1007/s00521-015-2135-1.

[10] Z. Li, L. Ma, H.Z. Zhang, Genetic mutation bat algorithm for 0–1 Knapsack problem, Comput. Eng. Appl. 48 (4) (2012) 50–53.

[11] E. Bas, A capital budgeting problem for preventing workplace mobbing by using analytic hierarchy process and fuzzy 0–1 bidimensional Knapsack model, Expert Syst. Appl. 38 (10) (2011) 12415–12422.

[12] F. Taillandier, C. Fernandez, A. Ndiaye, Real estate property maintenance optimization based on Mult objective multidimensional Knapsack problem, Comput.-Aided Civil Infrastruct. Eng. 32 (3) (2017) 227–251.

[13] D. Pisinger, Heuristics for the container loading problem, Eur. J. Oper. Res. 141 (2) (2002) 382–392.

[14] G.L. Reniers, K. Sorensen, An approach for optimal allocation of safety resources: Using the Knapsack problem to take aggregated cost efficient preventive measures, Risk Anal. 33 (11) (2013) 2056–2067.

[15] Shehab, M., Abu-Hashem, M.A., Shambour, M.K.Y. et al. A Comprehensive Review of Bat Inspired Algorithm: Variants, Applications, and Hybridization. Arch Computat Methods Eng 30, 765–797 (2023). https://doi.org/10.1007/s11831-022-09817-5

[16] Sur, G.; Ryu, S.Y.; Kim, J.; Lim, H. A Deep Reinforcement Learning-Based Scheme for Solving Multiple Knapsack Problems. Appl. Sci. 2022, 12, 3068. https://doi.org/10.3390/app12063068.

[17] Li, S., Cai, S., Sun, R., Yuan, G., Chen, Z., Shi, X. (2019). Improved Bat Algorithm for Multiple Knapsack Problems. In: Sun, Y., Lu, T., Yu, Z., Fan, H., Gao, L. (eds) Computer Supported Cooperative Work and Social Computing. ChineseCSCW 2019. Communications in Computer and Information Science, vol 1042. Springer, Singapore. https://doi.org/10.1007/978-981-15-1377-0_11

[18] Evolutionary Algorithms Implementation Guide- https://github.com/NiklasLundstrom/Knapsack

[19] Evolutionary Algorithms Implementation Guide- https://github.com/Girish-03/Knapsack-Probelm

[20] Evolutionary Algorithms Implementation Guide- https://github.com/PadmanabhanAshwin/AntColonyOptim_Knapsack