Nature-Inspired Approaches for Solving the Bin-Packing Problem: An Analysis with Ant Colony Optimisation

720064987

Abstract

This report reviews nature-inspired approaches to solving the bin-packing problem and an experiment applying ant colony optimisation to solve a 1D bin-packing problem. Section 1 is a literature review of various nature-inspired approaches to the bin-packing problem. Section 2 contains the results of the ant colony optimisation experiment. Section 3 provides a detailed discussion of the experimental results and explores further work and potential experiments. Section 4 provides a conclusion of this report summarising the key findings of the experiment and possible future experiments which could lead to a better nature-inspired approach to solving the bin-packing problem. Section 5 is a list of references used throughout this report.

1 Literature Review

1.1 Introduction

The bin-packing problem (BPP) is an NP-hard combinatorial optimisation problem where a set of items with variable weights must be packed into a finite number of bins while minimising the difference between the heaviest and lightest bin. Various versions of BPP exist, but all share the challenge of finding an optimal solution. NP-hard problems, including BPP, have no polynomial-time algorithm for an exact solution, making exhaustive approaches impractical due to exponential time complexity. Nature-inspired algorithms offer a more effective alternative by exploring larger solution spaces through metaheuristics [3].

1.2 Nature-Inspired Approach(ACO) to Optimisation Problems

In *Nature-Inspired Metaheuristic Algorithms*, Yang (2010) defines nature-inspired algorithms as optimisation techniques based on natural processes, including biological evolution, animal behavior, and ecological interactions.[15] These algorithms harness the adaptability and resilience observed in nature, enabling efficient navigation through large, complex solution spaces. Among the various nature-inspired approaches, Ant Colony Optimisation (ACO), Genetic Algorithm (GA), and Particle Swarm Optimisation (PSO) have been widely applied to the BPP.

1.3 Ant Colony Optimisation and the BPP

The ACO algorithm, inspired by ant foraging behavior, utilises pheromone trails for indirect communication, allowing ants to solve optimisation problems, including BPP [5]. In BPP applications, each "ant" represents a candidate solution where items are allocated to bins, aiming to minimise bin count or balance the load. The pheromone trail intensity, proportional to the solution quality, guides future ants, while pheromone evaporation prevents premature convergence on suboptimal paths. This balance between exploration and exploitation allows ACO to generate near-optimal BPP solutions.

The most effective implementations of ACO and other nature-inspired algorithms incorporate heuristics to improve their performance. Heuristics allow for the algorithm to receive additional specialised information and shortcuts in the processes it utilises. In 2004, John Levine and Frederick Ducatelle improved the performance of a standard ACO by using ACO combined with a local search, which allowed items to be placed in the bin more effectively. [8]

1.4 Gentic Algorithms(GA) and the BPP

Genetic Algorithms (GA) mimic natural selection principles to evolve solutions. As described by Mitchell (1998), GAs use selection, crossover, and mutation to create new generations of solutions, refining the candidate population.[12] In BPP, each chromosome encodes a bin arrangement, with fitness calculated by bin load balancing. Faulkner (1996) advanced GA applications for BPP by developing a Hybrid Grouping Genetic Algorithm (HGGA), incorporating heuristics and local search to improve solution quality and performance [6].

In 2005, José Fernando Gonçalves designed a hybrid genetic algorithm heuristic for a two-dimensional orthogonal packing problem. The packing problem addressed in this paper shares many similarities to the BPP as they are both NP-hard combinatorial optimisation problems where a potential solution is an arrangement of items placed. In this problem, rectangles of different sizes must be placed onto a larger stock rectangle with the aim of minimising the surface area of the larger stock rectangle while minimising the number of unused rectangles.

This paper used heuristics to create a heuristic placement strategy for the rectangles and a difference process to calculate the remaining space after placing a rectangle on the stock rectangle. These heuristics, when combined with a GA, proved to be an effective approach to solving this problem and, most importantly, outperformed a standard GA that used no heuristics to aid with its performance.

1.5 Particle Swarm Optimisation(PSO) and the BPP

Particle Swarm Optimisation (PSO), inspired by swarming behaviors in nature, represents solutions as particles that move through the solution space, adjusting based on their own best positions and those of their neighbors. PSO is particularly suited for real-time optimisation due to its rapid convergence, but in complex spaces, it may require hybrid techniques, such as combining with local search, to avoid local optima issues [16]. Practical applications of PSO use heuristics to improve performance and avoid the shortfalls in a standard PSO algorithm. In 2008, Liu, Tan, Huang, Goh, and Ho (2008) applied PSO techniques to a 2D BPP with the aim of arranging items in bins while minimising the number of bins used and deviation of the weight of the bin from a target centre of gravity to ensure the stability of the bin.[9]

Their proposed algorithm, Multiobjective Evolutionary Particle Swarm Optimization(MOEPSO), uses a Bottom-Left-Fill(BFL) heuristic that tracks the most optimal position for the next item.

MOEPSO also uses mutation operators, which are more frequently present in GAs. The mutation operators were tailored to this specific BPP, allowing the algorithm to merge the two lightest bins to ensure fewer bins were used. A rotation and shuffle mutation operator was used to randomly shuffle the order of items within a bin to find arrangements that improve the bin's stability.

This hybrid approach of combining features of two different nature-inspired algorithms can effectively address the shortfalls of either algorithm, thus creating a more optimal solution.

1.6 Comparison of Algorithms and Conclusion

Although ACO, GAs and PSO can be effective at solving the BPP and other complex optimisation problems, a hybrid approach of combining the approaches with heuristics increases performance. ACO excels at balancing exploration and exploitation, which is essential when performing combinatorial optimisation. However, its performance relies on careful tuning of the number of ants and pheromone evaporate used. This reliance on parameter tuning is amplified when tackling complex problems. If the parameters are not selected carefully, ACOs can be prone to converging prematurely or performing unnecessary exploration. This is further explored in section 3.1.4 of this report. Parameter tuning can be time-consuming and computationally expensive, which can limit the performance of the ACO when tackling the BPP. [5]

A standard GA can effectively search large solution spaces but struggles to capture the complexities and nuance of optimisation problems with a large solution space, such as the BPP (Faulkener 1996). Simple GAs tend to get stuck in local optima. Therefore, several heuristics must be introduced to the GA to increase performance, such as those used in HGGA designed by Faulkenaur. The shortfalls of PSO, such as convergence speed and quality of solution, have been improved since it was first created. This improvement relies on a number of control parameters, namely the dimension of the problem, the number of particles (swarm size), acceleration coefficients (The acceleration coefficient, together with random vectors r1 and r2, control the stochastic influence), inertia weight, neighbourhood size, number of iteration, and the random values which scale the contribution of the cognitive and social component.[13] The number of possible combinations of the parameters that need to be tuned for the PSO to perform effectively is rather large, and therefore, implementing an effective PSO algorithm for a complex optimisation problem would be a laborious task.

After highlighting the limitations of the ACO, GA, and PSO, a hybrid approach, as seen in the HGGA designed by Faulkner, or the hybrid heuristic GA created by José Fernando Gonçalves would be a more effective approach to tackling the BPP due to the complex nature of the problem and the limitations of each algorithm when applied without the aid of heuristics.

2 Description of Results

An implementation of ACO was used to address two different Bin-Packing Problems, BPP1 and BPP2. In each, there are 500 items to be packed into a number of (b) bins.In problem BPP1, b = 10 and the weight of item i (where i starts at 1 and finishes at 500) is i. In problem BPP2, b = 50 and the weight of item i is $(i)^2/2$. The parameters of the ACO which were changed per experiment were the number of ant paths (p) generated in each iteration of the ACO and the pheromone evaporation rate (e) which determines the rate at which pheromone trails evaporate. The algorithm's termination criteria was 10000 fitness evaluations. Five trials of each set of differing parameters were performed and the results are represented in the following tables for both BPP1 and BPP2. I decided to record the best average fitness over the five trials because it is a key quantitative indicator of the algorithm's ability to find a near-optimal (a fitness closer to zero) solution across all five trials of a pair of a given p value and e value.

Table 1 and Table 2 show the best fitness achieved in each trial of the different combinations of parameters after ten thousand fitness evaluations for BPP1 and BPP2, respectively. The best average across the five trials is recorded in the last column.

In Table 1 and Table 2, f_n^* denotes the best fitness of trial n, and $\overline{f^*}$ denotes the average best fitness across the five trials. The average best fitness is calculated as $\overline{f^*} = \frac{f_1^*+f_2^*+f_3^*+f_4^*+f_5^*}{5}$.

Figure 1 and Figure 2 on page 3, show the progression of the average best fitness($\overline{f^*}$) achieved across the five trials of each set of parameters over the number of fitness evaluations performed, until the algorithm terminates, for BPP1 and BPP2, respectively.

Table 1: BPP1 Experimental Results with Various Parameter Combinations

p	e	f_1^*	f_2^*	f_3^*	f_4^*	f_5^*	$\overline{f^x}$
100	0.9	1806	1258	1611	1360	1781	1563.2
100	0.6	1865	1735	1652	1648	1820	1744
10	0.9	1204	1245	1226	917	1117	1141.8
10	0.6	161	742	390	594	263	430

Table 2: BPP2 Experimental Results with Various Parameter Combinations

p	e	f_1^*	f_2^*	f_3^*	f_4^*	f_5^*	$\overline{f^*}$
100	0.9	$4.57e5$	$4.69e5$	$4.56e5$	$4.58e5$	$4.51e5$	$4.58e5$
100	0.6	$4.61e5$	$4.20e5$	$4.41e5$	$4.23e5$	$4.18e5$	$4.33e5$
10	0.9	$3.82e5$	$3.89e5$	$4.10e5$	$4.14e5$	$3.49e5$	$3.89e5$
10	0.6	$4.92e5$	$4.52e5$	$4.31e5$	$4.33e5$	$4.22e5$	$4.46e5$

3 Discussion and Further Work

3.1 Discussion

3.1.1 Question 1 - Which Combination of Parameters Produced the Best Results? The best-performing combination of parameters for BPP1 was p=10 and e = 0.6, which achieved an average best fitness

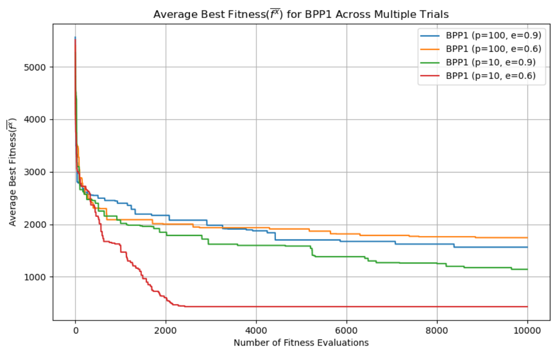

Figure 1: Average Best Fitness($\overline{f^*}$) for BPP1 Across Multiple Trials

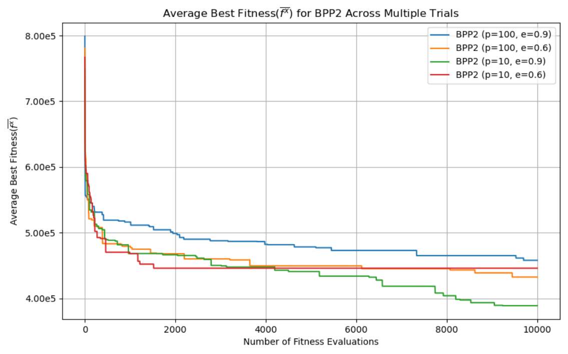

Figure 2: Average Best Fitness($\overline{f^*}$) for BPP2 Across Multiple Trials

of 430. The best-performing combination of parameters for BPP2 was p=10 and e = 0.9, which achieved an average best fitness of 3.89e5.

3.1.2 Question 2 - Why did these Combinations of Parameters Produce these Results? For BPP1, this combination of parameters allowed the algorithm to balance exploration and exploitation more effectively than the other combinations. Exploration effectively avoids premature convergence by searching for new solutions, and exploitation is used to refine known global solutions. [4]

In BPP1, the moderate pheromone evaporation rate (e=0.6) enabled the algorithm to retain successful paths while allowing new paths to be explored. If the pheromone evaporated at a higher rate(e = 0.9), successful paths would fade quickly, creating a more random search for solutions, which may have prevented the ants from converging on high-quality solutions. This is shown in Figure 1, as the higher evaporation rate (e=0.9) generally leads to worse performance across both values of p, while the high number of ants introduces excessive exploration that hinders convergence.

However, in BPP 2, a higher evaporation rate leads to better algorithm performance. A rate of e = 0.9 did not cause the excessive exploration seen in BPP1 but stopped the algorithm from converging on a less optimal solution. A high evaporation rate was ineffective for BPP1 and effective for BPP2 because BPP2's solution space is significantly larger than BPP1's solution. There are more possible arrangements of items placed in bins due to the greater number of items and bins used in BPP2. Therefore, increased importance on exploration proved effective, as seen in Figure 2, when p = 10 and e = 0.9, the solution continues to improve over the number of fitness evaluations because of this increased exploration of the larger solution space, which resulted in the best solution being found. The other combinations of parameters all performed similarly but did not find the best solution due to either a lower evaporation rate or a higher number of ant paths.

The combination that used a p of 100 and e of 0.9, did not perform as well due to the number of ant paths it used. The smaller number of ant paths (p = 10) contributed to the best performance for both BPP problems.

3.1.3 Question 3: How do each of the parameter settings influence the performance of the algorithm? Both the number of ant paths(p) and the rate of pheromone evaporation (e) balance the algorithm's exploration and exploitation and, therefore, its performance. As for any metaheuristic algorithm, such as ACO, a good balance of intensive local search and an efficient exploration of the whole search space will usually lead to a more efficient algorithm. [15]

A higher emphasis on exploration and generating diverse solutions is useful early in optimisation. However, too many ants dilute pheromone trails, slowing convergence as promising paths are less reinforced. A lower p enhances exploitation, concentrating pheromones on fewer paths, which can lead to faster convergence around strong solutions. Yet, fewer ants reduce diversity, raising the risk of getting stuck in local optima. [11]

A higher e (e.g. 0.9) causes faster pheromone decay, reducing the impact of older paths and promoting the exploration of new paths. This high evaporation prevents premature convergence and encourages the algorithm to explore the solution space more widely, making it particularly useful in the early stages of the search. However, if it is too high, it may hinder convergence by preventing good solutions from being adequately refined. In contrast, a lower e (e.g., 0.6) slows pheromone decay, allowing pheromones to persist and reinforcing successful paths, favouring exploitation. This approach is beneficial in later stages, helping to refine solutions, but it risks premature convergence if the algorithm reinforces suboptimal paths too early.[2]

3.1.4 Question 4: Do you think that one of the algorithms in your literature review might have provided better results? As mentioned in Section 1, combining heuristics with a genetic algorithm would be a better approach to solving the bin-packing problem addressed in this report. Due to its effectiveness, this hybrid approach is commonly used in current literature when addressing the bin-packing problem.

A hybrid approach involving a GA or PSO could achieve better results than ACO for the bin-packing problem addressed in this report. Falkenauer's design of an HGGA shows that GAs can be effective when solving optimisation problems such as BPP, as they maintain diversity within the population. This diversity can help avoid the premature convergence that can occur when employing an ACO. The HGGA that Faulkenauer designed uses specialised encoding to represent bins as genes instead of individual items. Faulkenaur also designed a specialised crossover operator that allows offspring to inherit and retain promising genes. Furthermore, a specialised mutation operator was designed to create or eliminate entire groups of genes. The experiment results show that the HGGA outperformed contemporary techniques such as the branch-and-bound technique.

Additionally, when PSO is combined with local search methods, solutions can be refined more intensely and quickly, thus achieving better results after fewer iterations than the standard ACO.

Additionally, PSO is known for its rapid convergence, which is useful in real-time or high-dimensional optimisation contexts. When PSO is combined with local search methods, it can refine solutions more quickly, potentially achieving higher-quality results in fewer iterations. PSO's flexibility to incorporate hybrid techniques allows it to address complex solution spaces effectively, which is particularly valuable in NP-hard problems like BPP.

3.2 Further Work and Experiments

Although I believe my implementation effectively demonstrated using an ACO to solve the bin-packing problem, it can be improved. Monitoring the performance of the ACO over a greater number of fitness evaluations could provide more insight into where this current implementation falls short. I further tuned the number of ants used and the rate of pheromone evaporation to see if I could improve the algorithm's performance and I managed to achieve better results than I did when I used the combinations of parameters in this report.

Parallelisation of the ACO by allowing multiple ants to traverse paths and explore the solution space simultaneously would reduce the computational time taken to find the best solution. This would be beneficial for the real-world implementations of ACO, such as water resource management and vehicle routing. [1][14]

Additionally, applying heuristics such to my implementation of the ACO would likely improve the algorithm's performance. A common heuristic used to aid the ACO in solving more effectively the BPP is the First Fit Decreasing(FFD) heuristic. FFD sorts all the items in descending order of size which helps decide where larger items should be placed as they are harder to place effectively. After sorting the items, each item is placed in the first available bin with enough capacity to hold it. [7][8]

Another heuristic that would have likely improved the ACO's performance is a local search with dominance criteria. In each iteration of the ACO, the local search attempts to remove items from bins or swap items between bins to create better future arrangements of items. The dominance criteria analyses which items should be replaced or swapped. For example, if a bin can be filled more optimally by replacing a larger item with several smaller ones that add up to the same weight or volume, the algorithm makes this change. [10][8]

4 Summary of Experiment and Future Experiments

When solving BPP1, the best-performing combination of parameters for the ACO was p=10 and e = 0.6, which achieved an average best fitness of 430. This considerably outperformed the worst-performing combination of parameters, p = 100 and e = 0.6, which achieved an average fitness of 1744. When solving BPP2, the best-performing combination of parameters O was p=10 and e = 0.6, which achieved an average best fitness of 3.895. The worst-performing combination of parameters was p = 100 and e = 0.6, resulting in an average best fitness of 4.585.

These results highlight the importance of balancing the exploration and exploitation performed by the ACO in order to achieve a near-optimal solution and the impact that the number of ant paths and pheromone evaporation rate can have on this balance between exploration and exploitation. The ACO implemented in this experiment performed well. An analysis of the ACO's performance over a greater number of fitness evaluations could be beneficial in discovering the shortfalls of the current set of parameter combinations used in this experiment. However, I do believe that further improvements can be made to it, such as the introduction of heuristics such as a local search and the creation of dominance criteria amongst the items being arranged. The implementation of ACO in this experiment does not use parallelisation, which would reduce the computational time taken by the ACO to provide its best solution. This would allow for the ACO to be more applicable to real-world applications where similar NP-hard combinatorial optimisation problems require quick and effective solutions.

The use of a hybrid metaheuristic approach would improve performance. AN HGGA that utilises mutation operators tailored to this specific BPP would prove to be an effective approach. Additionally, a MOEPSO would prove to be beneficial due to its capability to solve problems where a possible solution has multiple metrics that need to be taken into consideration.

The implementation of these approaches would be beneficial in achieving a more optimal solution to the BPP. However, I do believe that the ACO implemented in this experiment performed well and adequately solved the two different BPPs addressed in this report.

References

[1] Ramadan Abdelaziz. 2023. Application of Ant Colony Optimization in Water Resource Management. *IntechOpen* (2023). https://doi.org/10.5772/intechopen.112895 Submitted: 06 June 2023, Reviewed: 16 August 2023, Published: 02 November 2023.

[2] Christian Blum. 2005. Ant Colony Optimization: Introduction and Recent Trends. *Physics of Life Reviews* 2, 4 (2005), 353–373. https://doi.org/10.1016/j.plrev.2005.10.001

[3] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. 2013. A survey on optimization metaheuristics. *Information Sciences* 237 (July 2013), 82–117. https://doi.org/10.1016/j.ins.2013.02.041

[4] Erik Cuevas, Primitivo Diaz, and Octavio Camarena. 2021. *Metaheuristic Computation: A Performance Perspective.* Springer.

[5] Marco Dorigo and Thomas Stützle. 2004. *Ant Colony Optimization.* MIT Press, Cambridge, MA.

[6] Emmanuel Falkenauer. 1996. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics* 2, 1 (June 1996), 5–30. https://doi.org/10.1007/BF00226291

[7] David S. Johnson, Alan Demers, Jeffrey Ullman, Michael R. Garey, and Ronald L. Graham. 1974. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* 3, 4 (1974), 299–325. https://doi.org/10.1137/0203025

[8] John Levine and Frederick Ducatelle. 2003. Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems. In *Proceedings of the 2003*

Congress on Evolutionary Computation (CEC). IEEE, 35–42. https://doi.org/10.1109/CEC.2003.1299565

[9] David Sihai Liu, Kay Chen Tan, Soon Yin Huang, Chee Keong Goh, and Weng Kin Ho. 2008. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research* 190, 2 (2008), 357–382. https://doi.org/10.1016/j.ejor.2007.06.032

[10] Silvano Martello and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley Sons, Inc., New York, NY.

[11] Daniel Merkle and Martin Middendorf. 2002. On the impact of the number of ants in Ant Colony Optimization. In *Proceedings of the 2002 International Conference on Artificial Evolution*. 91–102.

[12] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA. 221 pages. Not for sale on the Indian subcontinent.

[13] Dian Palupi Rini, Siti Mariyam Shamsuddin, and Siti Yuhaniz. 2011. Particle Swarm Optimization: Technique, System and Challenges. *International Journal of Artificial Intelligence & Applications (IJAIA)* (September 2011). https://doi.org/10.5120/ijais-3651

[14] Andrea-Emilio Rizzoli, Roberto Montemanni, Enzo Lucibello, and Luca Maria Gambardella. 2007. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence* 1, 2 (November 2007), 135–151. https://doi.org/10.1007/s11721-007-0005-x

[15] Xin-She Yang. 2010. *Nature-Inspired Metaheuristic Algorithms* (2nd ed.). Luniver Press, Middlesex University, UK.

[16] Shengzhe Zhao, Jingjing Liang, Ponnuthurai Nagaratnam Suganthan, and Mehmet Fatih Tasgetiren. 2008. Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 3845–3852. https://doi.org/10.1109/CEC.2008.4631339