A Comparative Analysis of Algorithms to Tackle the One-Dimensional Bin Packing Problem

***Abstract*— The one-dimensional Bin Packing Problem, an NP-hard combinatorial optimisation challenge, necessitates fitting variably sized items into a minimal number of bins, with applications ranging from computer memory management to cargo loading. This paper overviews and evaluates four primary algorithms—Genetic Algorithm, Ant Colony Optimisation, Cuckoo Search, and Hybrid Firefly Algorithm—focusing on their solution optimality, computational efficiency, and convergence traits. Through a comparative analysis, we aim to identify the most effective algorithmic approach considering their unique configurations and application contexts.**

***Keywords*— Combinatorial Optimisation, Metaheuristics, Bin Packing Problem, Algorithmic Efficiency, Solution Convergence**

I. Introduction

The Bin Packing Problem (BPP) is a classic combinatorial optimisation problem categorised under the NP-hard complexity class, known for its computational intractability when attempting solutions within polynomial time. At its core, the BPP asks how to efficiently pack a set of objects of various sizes into a finite number of bins or containers of a fixed capacity, aiming to minimise the number of bins used. Although seemingly straightforward, the problem encapsulates a range of complex real-world applications, including cargo loading, resource allocation, data segmentation, and memory management.

Since its introduction in the 1970s, the BPP has served as a critical benchmark for evaluating the efficacy of various optimisation algorithms, particularly in operations research and computer science. The versatility of the problem's framework has led to its adoption across diverse industrial operations, influencing sectors such as logistics, manufacturing processes, and even the optimisation of digital storage systems.

This analysis aims to provide a detailed review and classification of the various historically and currently employed methodologies to address the BPP. It will further assess and execute three distinct algorithms to explore their performance and applicability to a standard BPP instance, demonstrating the challenges and innovations in tackling this enduring optimisation puzzle.

II. Literature Review

A. History of Algorithms

Early approaches to solving the bin packing problem involved using exact algorithms, which guaranteed optimal solutions for problems with fewer items and bins. These algorithms relied on techniques such as dynamic programming or branch and bound [1].

However, these fell out of favour as they became impractical for more significant problems. Heuristics like First-Fit and Best-Fit, prioritising speed over guaranteed optimality, emerged alongside exact algorithms [2]. The late 20th century witnessed the rise of metaheuristics [3]. These advanced algorithms balanced optimality and efficiency, making them ideal tools for tackling large-scale BPPs.

These advanced algorithms offered a balance between optimality and efficiency, making them ideal tools for tackling large-scale BPPs, the most common of which are metaheuristics [1] [2].

B. Narrowing Down Algorithms

The Genetic Algorithm (GA) study by Iima in 2003 [4] offers a versatile framework that efficiently uses computational resources for diverse problem scales. It features inherent exploratory mechanisms that promote rapid convergence, often outperforming Tabu Search and Variable Neighbourhood Search in solution accuracy. This paper was chosen for its adaptability to various bin packing problems, allowing parameter tuning specific to the problem context. This flexibility is essential for addressing the varied challenges in bin packing scenarios. The GA emphasises maintaining item combinations within bins for high packing efficiency and integrates effective heuristic methods to enhance solution quality.

The 2007 paper by Zhang on the application of Ant Colony Optimisation (ACO) [5] is significant due to its approach to a well-defined item set and quantity challenges. It aims to find the optimal packing strategy that minimises bin usage under perfect constraints. This scenario demonstrates the robustness of the modified ACO approach in achieving theoretical optimality. The innovation lies in integrating pheromone summation rules with traditional heuristic methods, enhancing the algorithm's efficiency, and providing a holistic solution strategy. This approach is pivotal for extending ACO's applicability to real-world problems.

Inspired by Kanagaraj's 2014 [6] study on optimising hole-making sequences, the Cuckoo Search Genetic Algorithm (CSGA) replaces traditional cuckoo search operators with GA processes. This adaptation leverages the strengths of both metaheuristics to enhance efficiency and convergence. Integrating GA's mutation and crossover processes replaces the random walks and Lévy flights typically used in cuckoo search, providing a more directed search strategy. Mutation introduces solution diversity, while crossover facilitates information recombination, supporting systematic solution space exploration. This hybrid approach optimally utilises the complementary strengths of both algorithms, making CSGA particularly effective for complex problems like the One-Dimensional Bin Packing Problem (1DBPP).

The Hybrid Firefly Algorithm (FAH), evolving from Firefly Algorithm, an implementation of Yang's 2009

paper [7] [8] is designed to tackle NP-hard problems like bin packing by incorporating mutation mechanisms to enhance population diversity and prevent stagnation.

Munien's 2020 Analysis [7] specifically refines both the CSGA and FAH for the nuanced challenges of bin packing, a problem of broad relevance and complexity in various industries. Furthermore, it provides a robust comparative analysis against other contemporary metaheuristics, a critical aspect for comprehensive scholarly evaluation. Such detailed comparative insights are indispensable for accurately assessing the relative performance and suitability of each algorithm for bin packing scenarios. This focus not only enriches the academic discourse surrounding optimisation algorithms but also enhances practical decision-making in algorithm selection for specific industrial applications.

III. METHDOLOGY

A. Demonstrating the Chosen Algorithms

The theoretical minimum number of bins for test cases has been determined by the sum of the weight of all items and by dividing the bin's capacity. This calculation is then rounded up to find the ideal solution. Due to variability in item sizes and distribution, reaching this ideal may not be possible. These numbers are classified in Table 1. If any tests reach the optimal number of bins, as displayed in Fig. 1, the test will conclude that there is no more optimal solution. Table. 2 classifies the algorithms that will be implemented and analysed.

TABLE I. THEORETICAL MINIMUM NUMBER OF BINS

Test Cases	***Bins***
TEST0049	11
TEST0014	23
TEST0082	24
TEST0044	14
TEST0030	27

TABLE II. ALGORITHM CATEGORY

Algorithms	***Solution Strategy***	***Search Strategy***	***Type***
GA	Preserve item combinations in bins	Global Search	Metaheuristic
ACO	Uses pheromones and heuristics for solution improvement	Global Search	Metaheuristic
CSGA	Combines elements of cuckoo search and genetic algorithms to optimise solutions.	Global and Local Search	Metaheuristic
FAH	Moves towards brighter (better) solutions	Global Search	Metaheuristic

B. Genetic Algorithm

The first choice of algorithm is GA, a population-based metaheuristic that simulates the process of natural evolution. Adapted from Iima's 2003 paper [4], the paper leverages genetic principles such as selection, crossover, and mutation to generate solutions that converge towards an optimal or near-optimal solution over successive generations. In this context, individuals within the population represent distinct arrangements of items within bins, aiming to minimise the number of bins used without exceeding their capacity — the pseudocode for the implementation used in this study is given below [4].

Algorithm 1: Genetic Algorithm

Initialize population randomly using items and binCapacity
  Evaluate fitness of each individual in the population
  Repeat
    Select parent individuals based on fitness
    Perform crossover to generate offspring
    Apply mutation to offspring with a certain probability
    Evaluate fitness of new offspring
    Select for next generation from parents and offspring
    If time finishes or no diversity, exit loop
  End Repeat
  bestConfiguration = individual with highest fitness in population
  Return bestConfiguration
End Algorithm

In GA, diversity is strategically used to gauge the range of solutions within the population and to prevent premature convergence to suboptimal solutions. Diversity is quantified by assessing the uniqueness of bin configurations, where each configuration represents a different arrangement of items across bins. This is achieved by converting the contents of each bin into a sorted list of item sizes, which is then transformed into a string for simple comparison and storage. The size of this set indicates the number of distinct configurations present, serving as a measure of diversity.

The Parameters were initially chosen through a literature review. Population size and Generations were chosen directly from the referenced paper, where the operators were directly stated [4]. The offspring size was initially chosen as 500 through a literature review [9] [10]. Independent testing from the range of [200, 400] in intervals of 50 concluded 250 offsprings to efficiently balance time complexity with producing an efficient solution. The Mutation rate was chosen through a literature review, and independent testing determined that the results remained the same due to the test cases being very compact [4] [9] [10]. The runtime ranged from around 10 minutes due to the variability of the items in the tests. Diversity was introduced as.

Given the problem constraints with less than 150 items per test that are closely related in size, the impact of achieving more than 100% diversity has shown to be minimal in routine tests. This outcome suggests that

increasing diversity only significantly enhances the search for a better solution beyond a certain threshold, likely because the problem space is inherently limited due to the similarity and small number of items. Therefore, while the GA includes mechanisms to maintain genetic diversity, such as mutation and crossover, the practical benefits of highly high diversity levels in this context appear linked. This field emphasises the need for a balanced approach in managing diversity, ensuring enough variation to escape local optima without unnecessarily broadening the search space beyond what is helpful for the problem.

C. Ant Colony Optimisation Algorithm

The second algorithm is ACO, inspired by the foraging behaviour of real ants. Adapted from Zhang's 2007 paper [5], the algorithm uses a metaheuristic approach to solve complex optimisation problems by mimicking how ants find the shortest path between food sources and their colony. In the bin packing context, each ant represents a potential solution where items are packed into bins to minimise the number of bins used without exceeding their capacity. The pseudocode in this implementation is given below.

Algorithm 2: Ant Hill Colony

Initialize pheromone trails and heuristic for item and bin
  While termination criteria are not satisfied
    For each ant
      For each item
        If item is assignable
          Calculate transition probability for assigning item to bins based on pheromone and heuristic information
          Select bin for the item based on calculated probabilities
          Update local pheromone trail for the chosen item and bin
      End For
    End For
    Apply global pheromone update rule to best solution found
    Save best solution if it improves the current best
  End While
  Return the best solution found
End Algorithm

The ACO approach adapts well to the dynamic and combinatorial nature of bin packing problems, where the algorithm effectively handled precedence constraints. The flexibility of the ACO algorithm allows it to explore a vast space of possible solutions through stochastic and heuristic methods, making it robust against varying problem sizes and complexities [5].

The parameters such as the number of ants, pheromone evaporation rate, and heuristic factors are selected based on a mixture of literature suggestions and empirical tuning. For instance, Zhu's paper in 2009 [11] shows adjustments in pheromone update rules to enhance convergence speed and solution quality in large-scale applications. During initial testing, the values were all based on a literature review. After the complete implementation of the algorithm, it was determined that modifying the values would not impact the algorithm.

$$\tau 0 = 1/(n \cdot m*) \quad (1)$$

τ0 is the initial value of the pheromone trail, n is the number of items and m* is the optimal solution.

The initial pheromone trail and values are optimally set at 0.01 for a dispersed pheromone landscape with no particular route, and the decision is to encourage exploration to avoid early convergence.

ACO's performance is gauged by its ability to converge towards a minimal bin usage solution, as utilized from Eq. 1. The robustness of the ACO algorithm in maintaining high-quality solutions even as problem parameters change is aligned with findings from Levine's paper in 2004, where ACO was adapted for varied bin constraints and item sizes [11] [12].

D. Cuckoo Search Genetic Algorithm

In Munien's 2020 study on CSGA [7] starts by generating a population of "nests," each representing a different configuration of bin items. These configurations are evaluated to minimise "waste" or unutilised space. The algorithm utilises two critical genetic operations: crossover and mutation. Crossover combines elements from two-parent solutions to produce offspring, which may inherit advantageous traits for increased efficiency. When applied sparingly, mutation introduces minor changes to maintain diversity within the population and prevent premature convergence to local optima, facilitating the exploration of new solutions. Improved solutions replace less efficient ones in a selection process repeated until reaching a termination condition, such as a maximum number of generations or a specific fitness improvement. CSGA effectively manages the dynamic challenges of bin packing by optimising bin usage and reducing waste, providing scalable solutions to complex optimisation problems. The pseudocode of implementation is given below.

Algorithm 3: Cuckoo Search Genetic Algorithm

Initialize population of m host nests
while t < MaxGeneration
  Choose two solutions by k-tournament: Xi and Zi
  Crossover Xi and Zi to reproduce new egg yi
  Convert yi to item permutation Yi
  F(Yi) = Evaluate the fitness of Yi
  If Fitness (Yi) > F(Xi) or F(Yi) > F(Zi) then replace lower fitness with Yi
  Mutate cuckoo eggs and lay them in other hosts' nests
  Reject low quality eggs by host birds
  Convert new solutions to item permutations
  Evaluate fitness of the new solutions
  Rank solutions and find the current best
end while

The choice of parameters is crucial for balancing computational efficiency and the quality of results. A k-tournament size of 5, 30 nests, and a limit of 100 generations ensure diversity and sufficient iterations for refining solutions. A 15% mutation rate promotes exploring new solutions while maintaining genetic integrity, enhancing the algorithm's ability to bypass local optima and optimise search effectiveness. Strategic parameter selection is essential to maximise CSGA's performance. Going beyond these parameter values leads to no improvements in results but increases computational cost and time complexity, making it essential to adhere to these settings for optimal performance.

E. Hybrid Mutated Firefly Algorithm

In Munien's 2020 study on HFA [7], the initialisation phase generates diverse solutions, each represented by a firefly symbolising a unique item configuration created through random shuffling. This setup facilitates broad exploration of the solution space. As the algorithm progresses, each firefly's position, corresponding to a specific item arrangement, is evaluated for its efficiency in minimising bin usage. Fireflies move toward more optimal solutions, guided by an attractiveness factor that decreases with distance, based on a predefined light absorption coefficient, mimicking the natural attraction to brighter light observed in fireflies.

The optimisation also incorporates a mutation mechanism that introduces random changes to maintain genetic diversity and prevent premature convergence on local optima. The mutation rate starts high to encourage exploration and is gradually reduced to focus on exploiting promising solutions. This iterative evaluation, movement, and mutation process continues until a specified number of generations are completed, allowing the algorithm to balance exploration and exploitation effectively, adjusting as better configurations emerge and refining solutions to minimise the number of bins used, the pseudocode of which is given below.

Algorithm 4: Hybrid Mutated Firefly Algorithm

```
Initialize population of fireflies
Initialize the parameters: gamma, n, beta0, alpha, and mutation rate
    Determine light intensity I for each firefly
    while (t < MaxGeneration)
      for i = 1 to n fireflies
        for j = 1 to n fireflies
          if (i> j))
            Move ith firefly towards jth firefly from Euclidean Distance
            Mutate the ith firefly with probability mutation rate
          End If
          Evaluate new solution and compute new light intensity
          Decrease mutation rate in with current iteration
        End For j
      End For i
      Rank fireflies and identify current best
end while
```

The HFA for solving the bin packing problem meticulously sets key parameters to balance exploration and exploitation optimally, thus facilitating near-optimal solutions with minimal computational overhead. Gamma (γ), the light absorption coefficient, is set at 0.3 to control how attractiveness decreases with distance, ensuring substantial yet regulated interaction among fireflies. Initial attractiveness (β0), set at 0.3, defines the base level of attraction at zero distance, promoting an extensive search space exploration without premature convergence on local optima. The randomness component (α), maintained at 0.1, introduces a low level of randomness to make movements more deterministic and predominantly influenced by the attraction towards superior solutions. Lastly, a mutation rate of 0.25 is adopted to preserve adequate genetic diversity within the population, preventing premature convergence and enabling the circumvention of potential local optima, effectively facilitating the discovery of high-quality solutions in an efficient manner. Gamma, initial attractiveness, randomness, and mutation rate govern the algorithm's behaviour and effectiveness across iterations, guiding fireflies toward optimal solutions and ensuring diversity within the population to achieve efficient bin packing.

IV. Result and Discussion

The simulations in this study were conducted on a Ryzen 9 5900HS CPU with a clock speed of 5.2 GHz with 32GB RAM. All algorithms are developed in Java and compiled using the IntelliJ IDE.

TABLE III. Average Results Over 30 Runs

Algorithms	*Time (seconds)*	*TEST 0049*	*TEST 0014*	*TEST 0082*	*TEST 0044*	*TEST 0030*
GA	33.58	11	24	25	15	28
ACO	00.92	12	27	30.86	15	31
CSGA	07.56	12	24.40	25.43	15	28
FAH	06.19	12	24	25	15	28

- GA: Shows the longest computation time but manages to achieve optimal results for TEST 0049 and good performance across other tests. GA is suitable for scenarios where obtaining the most optimal solution is more crucial than computational speed, such as complex BPPs with a high variance in item sizes.
- Ant Colony Optimization (ACO): This algorithm shows the shortest computation time but tends to result in higher bin usage, which might not be as efficient as other algorithms. However, its rapid execution makes it suitable for scenarios that demand quick solutions, such as dynamic BPP scenarios where decisions need to be made rapidly, albeit at the cost of optimality.

- Cuckoo Search Genetic Algorithm (CSGA): Offers a balance between execution time and performance, suitable for environments where a more refined approach to bin packing is needed but within a reasonable time frame. This could be effective in moderately complex industrial applications where both time and efficiency are important.
- Firefly Algorithm (FAH): Similar to CSGA in performance and time, making it also suitable for applications needing a balance of efficiency and speed. The adaptive nature of FAH can be advantageous in applications where the problem dynamics can change, requiring the algorithm to adapt quickly.

In less time-sensitive scenarios where the number of bins used is a critical cost factor, GA might be preferred. Conversely, in high-speed, dynamic environments where quick approximations are acceptable, ACO could be the better choice. For balance, FAH is ideal.

A. Analysis of Genetic Algorithm

GA produces effective results at the cost of time. As demonstrated in Fig. 1, GA goes through the entire search space of unique solutions to find the least number of bins possible. As a result, it takes more computational time. The results in Table. 3 demonstrates that GA was able to find the most optimal bins for TEST0049 and closest to optimal for all other tests, compared to the other algorithms.

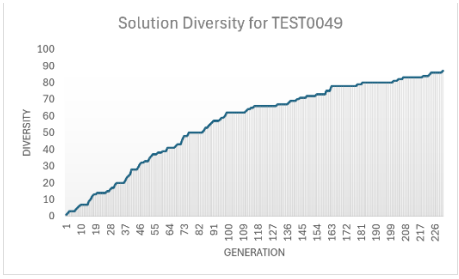

Fig. 1. Diversity for obtaining optimal solution in Genetic Algorithm

B. Analysis of Ant Colony Optimisation

The high-test case results for the ACO in the study, which focused on the one-dimensional bin packing problem with precedence constraints, can largely be attributed to the constraints of using a small and homogeneous dataset. This limited dataset curtailed the ACO's ability to showcase its full capabilities in managing complexity. The lack of diverse scenarios hindered the algorithm's capacity to adapt and optimise effectively. Consequently, the outcomes may not fully reflect the ACO's potential in more varied and complex scenarios, where inter-item relationships and packing orders are more unpredictable. This suggests that while the ACO excels under controlled conditions, its performance might wane in more dynamic settings, as illustrated by Fig. 2, where it repeatedly searches the same state space without improvement.

Fig. 2. Ant Colony Optimization Struggles in Local Search

As a result, ACO has the worst overall algorithm results, reaching an average of 30 during multiple tests. ACO will be more efficient.

C. Analysis of Cuckoo Search Genetic Algorithm

The CSGA demonstrates limitations in handling the BPP, mainly due to the variability in item sizes and distributions across test cases. Despite optimally tuned parameters, as depicted in Fig. 3, CSGA tends to converge prematurely. This characteristic significantly affects its ability to optimise packing processes, especially in non-uniform item sizes efficiently.

This early convergence, noticeable in Fig. 3. fitness visualisation over 100 generations in TEST0044, suggests that CSGA may only explore the complete solution space after settling on a near-optimal or suboptimal solution. Such behaviour is disadvantageous in real-world applications of BPP, where item size and distribution can vary widely and unpredictably.

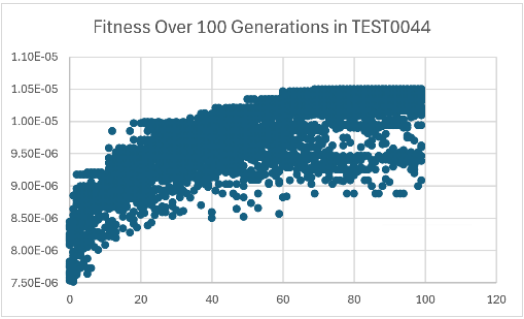

Fig. 3. Cuckoo Algorithm's Fitness Visualization

Moreover, the rigidity of CGA's parameter settings, though ideal under controlled test conditions, might adapt poorly to the dynamic nature of practical bin-packing scenarios. The algorithm's fixed approach to mutation rate and probability only accommodates the diverse range of item configurations typically encountered in practical settings.

D. Analysis of Hybrid Firefly Algorithm

The FAH displays a robust ability to refine solutions efficiently, as evidenced by its adaptive mutation rate shown in Fig. 4. This rate decreases over time, promoting an extensive initial exploration of the solution space, followed by targeted exploitation of the most promising solutions. This adaptive behaviour enhances FAH's performance, particularly in environments where the balance between exploration and exploitation is critical.

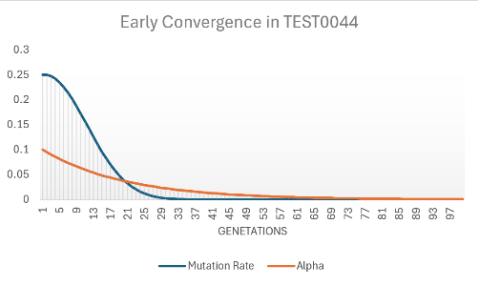

Fig. 4. Firefly Early Convergence to Refine Optimal Solution

FAH incorporates genetic algorithm concepts such as mutation processes to maintain diversity within the solution population, which is crucial for avoiding local optima and ensuring the search process remains dynamic. By introducing new genetic material continuously, FAH prevents the search from stagnating, thus remaining responsive to new optimisation opportunities, enhancing the algorithm's effectiveness. In practice, FAH's ability to dynamically adjust its parameters in response to the characteristics of the problem at hand allows it to handle the variability and complexity of the Bin Packing Problem effectively. This adaptability is particularly valuable in industrial applications where optimal packing strategies significantly impact operational efficiency and costs, making FAH a superior choice for tackling complex, real-world problems.

CONCLUSION

The GA has shown considerable robustness in optimisation algorithms, offering highly effective solutions, albeit with extended computational times. This trait makes GA particularly valuable when solution quality is crucial and time constraints are lenient. GA's ability to explore diverse solutions ensures a thorough examination of the potential solution space, which is critical for achieving near-optimal results.

On the other hand, ACO tends to operate faster but does not consistently achieve optimal solutions, especially in complex setups with many items and bins. Its quick execution makes ACO suitable for situations requiring prompt responses, even though these solutions may not be perfect. The heuristic nature of ACO promotes quick convergence, but this sometimes comes at the expense of solution quality.

Further, in the field of metaheuristics, both the CSGA and the FAH represent advanced developments, integrating various elements to boost their exploration and exploitation capabilities. This makes them particularly well-suited for complex, dynamic optimization scenarios. CSGA combines global and local search mechanisms, enhancing its ability to maintain a diverse pool of potential solutions and avoid premature convergence. In contrast, FAH benefits from adaptive mutation rates and the integration of genetic algorithm concepts, enabling it to escape local optima and explore the solution space extensively. FAH often outperforms CSGA inefficiency within the same timeframe due to its superior adaptability and dynamic response to evolving solution environments.

REFERENCES

[1] Amini, A. A., Weymouth, T. E., & Jain, R. C. (1990). Using dynamic programming for solving variational problems in Vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(9), 855–867. https://doi.org/10.1109/34.57681

[2] Kamali, S., & López-Ortiz, A. (2015). All-around near-optimal solutions for the online bin packing problem. Algorithms and Computation, 727–739. https://doi.org/10.1007/978-3-662-48971-0_61

[3] Gendreau, M., & Potvin, J.-Y. (2005). Metaheuristics in combinatorial optimization. Annals of Operations Research, 140(1), 189–213. https://doi.org/10.1007/s10479-005-3971-7

[4] Iima, H., & Yakawa, T. (2003). A new design of genetic algorithm for Bin Packing. The 2003 Congress on Evolutionary Computation, 2003. CEC '03. https://doi.org/10.1109/cec.2003.1299783

[5] Zhang, Z., Cheng, W., Tang, L., & Cheng, Y. (2007). Improved ant colony optimization for one-dimensional bin packing problem with precedence constraints. Third International Conference on Natural Computation (ICNC 2007). https://doi.org/10.1109/icnc.2007.437

[6] Kanagaraj, G., Ponnambalam, S. G., & Lim, W. C. E. (2014). Application of a hybridized cuckoo search-genetic algorithm to path optimization for PCB holes drilling process. 2014 IEEE International Conference on Automation Science and Engineering (CASE). https://doi.org/10.1109/coase.2014.6899353

[7] Munien, C., Mahabeer, S., Dzitiro, E., Singh, S., Zungu, S., & Ezugwu, A. E.-S. (2020). Metaheuristic approaches for one-dimensional bin packing problem: A Comparative Performance Study. IEEE Access, 8, 227438–227465. https://doi.org/10.1109/access.2020.3046185

[8] Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. Stochastic Algorithms: Foundations and Applications, 169–178. https://doi.org/10.1007/978-3-642-04944-6_14

[9] Bennell, J. A., Soon Lee, L., & Potts, C. N. (2013). A genetic algorithm for two-dimensional bin packing with due dates. International Journal of Production Economics, 145(2), 547–560. https://doi.org/10.1016/j.ijpe.2013.04.040

[10] Jing, X., Zhou, X., & Xu, Y. (2005). A Hybrid Genetic Algorithm for Bin Packing Problem Based on Item Sequencing.

[11] Fangjin Zhu, Xiangxu Meng, Hua Wang, & Shanwen Yi. (2009). An ant colony optimization algorithm to aggregated multicast using the idea of Bin Packing. 2009 IEEE Youth Conference on Information, Computing and Telecommunication. https://doi.org/10.1109/ycict.2009.5382392

[12] Levine, J., & Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society, 55(7), 705–716. https://doi.org/10.1057/palgrave.jors.2601771