A Comparative Analysis of Optimisation Algorithms in Solving the Bin Packing Problem

Patricia Valerie Santoso
School of Computer Science
University of Nottingham Malaysia
hcyps1@nottingham.edu.my

Naufal Hardiansyah
School of Computer Science
University of Nottingham Malaysia
hcynh1@nottingham.edu.my

Bryan Chandra
School of Computer Science
University of Nottingham Malaysia
hcybc2@nottingham.edu.my

Devan Lucian
School of Computer Science
University of Nottingham Malaysia
hcydl6@nottingham.edu.my

***Abstract*—The Bin Packing Problem (BPP) is a well-known NP-hard combinatorial optimisation problem that seeks to find a possible arrangement of items that results in the least number of bins used. Approximate methods can find near-optimal solutions within excellent computational time for small scale instances, whereas Metaheuristic approaches utilise stochastic-based techniques and parameters that heavily effect exploration and exploitation of strategies alongside being more robust to dataset size compared to traditional heuristics but may not always find the most optimal solution. This research aims to conduct a literature overview regarding algorithms used to solve the BPP and implements four algorithms to solve the BPP dataset: Modified First Fit Decreasing, Genetic Algorithm, Firefly Algorithm, and Tabu Search. The performance of each algorithm will be tested based on the solution optimality, computation time, and other metrics that helps in determining which algorithm is the most optimal for handling the BPP.**

Keywords—Bin packing problem, optimisation problem, approximate methods, genetic algorithm, firefly algorithm, tabu search.

I. Introduction

A difficult optimisation challenge, known as the bin-packing problem (BPP), involves the effective distribution of a finite number of objects with different weights into a finite number of bins while making sure that the bins aren't overflowing to capacity [1]. Reducing the overall number of bins used is the goal of BPP, a deceptively simple sounding task that belies its inherent complexity. The structure and uses of the classical BPP have been investigated since the 1930s [2], while the first studies of it date to the early 1970s. According to computational complexity theory, the BPP is NP-hard [3], meaning that it might take a lot of computation to discover the best solution, especially when dealing with big number of items. The traditional one-dimensional bin-packing problem (1D BPP) can mathematically be represented as:

$$Minimise\ t$$

subject to:

$$\sum_{a_i \in W_j} s(a_i) \leq C, \forall j = 1,2, \dots, t$$

Here, each set $W_j$represents the contents of a bin with capacity C. The goal is to find the smallest integer t given a non-negative bin capacity, denoted as C, and a list of n items, represented as $L = (a_1, a_2, \dots, a_n)$, where each item ai has a size (or weight) $s(a_i)$ such that $0 \leq s(a_i) < C$ (i.e., each item's size cannot exceed a bin's capacity). This number denotes the bare minimum of bins needed to hold the items, subject to the restriction that the total of all the item sizes $(a_i \in W_j)$ in every bin (W_j) cannot be greater than the maximum capacity (C). It is possible to think of each set W_j as the contents of a bin of capacity C[4].

In this research, we conduct a comparative study of four different approaches to the Offline One-Dimensional Bin Packing Problem. Our main goal is to thoroughly examine, assess, and contrast these algorithms to determine how well they perform in terms of producing packing solutions that are either optimum or nearly ideal. The algorithms that are being examined include more sophisticated techniques like Genetic Algorithms, Firefly Algorithm, and Tabu Search in addition to other method like Modified First Fit Decreasing.

Moreover, we will offer a thorough representation of these algorithms' packing results to give a thorough understanding of their performance. Our goal is to determine and suggest the best algorithm to be used in real-world applications, adding to the current discussion on optimisation techniques for bin packing issues.

II. Literature Review

At its core, The BPP involves choosing n elements from an n-dimensional array to arrange items in a manner that minimises the number of bins used. As a result, Offline one-dimensional BPP can be thought of as a deterministic combinatorial problem, with *n!* being the number of configurations. As it is for most NP-Hard optimisation problems, numerous studies have tackled the BPP through application of diverse algorithms, all of which strive to apply an efficient algorithm that can attain optimal performance while maintaining computational time. These algorithms come in a variety of forms, ranging from simple heuristic techniques to increasingly sophisticated metaheuristic approaches, each with their own unique set of benefits and drawbacks.

Format adapted and modified from ©2023 IEEE.

A. Approximate Methods

Historically speaking, research has relied on using heuristic techniques for tackling NP-hard problems not limited to one-dimensional BPP, as they provide near-optimal performance within reasonable computation time. BPP especially utilises sequential heuristics, namely the First Fit (FF), Best Fit (BF), and Next Fit (NF) heuristics, where each orders individual items by placing each item based on an algorithmic approach. The use of metaheuristic algorithms to solve the one-dimensional BPP has become increasingly popular in recent years.

Among these heuristics, FF is the simplest, where items are added to bins solely into the first one it fits into [7]. New bins are created if an item does not fit into the current available bins. BF differs slightly by analysing all current available bins to find the best fit, resulting in least wastage at the expense of computational time. Finally, NF functions as a modification to FF by placing items into the same bin until it exceeds capacity.

While these techniques performed competently in bin packing, noticeable problems arise which are especially apparent with the FF heuristic, in cases where the list of items are sorted in ascending order [5]. As a result, a modification was made to the FF heuristic that addresses issues regarding order of items by first sorting the list using other methods before applying the sequential heuristic process, referred to as the First-Fit Decreasing (FFD).

Although FFD may not be able to always find the best solution, it performs well in smaller datasets alongside having lower computational time compared to more complex heuristics that take longer to reach the same level of optimality as FF in smaller datasets [6]. In addition to First Fit Decreasing, there is also Modified First Fit Decreasing (MFFD), which separates items into four different sizes and allocates different bins before using the normal FFD approach [7].

B. Metaheuristic Approaches

The use of metaheuristic algorithms to solve the one-dimensional BPP has become increasingly popular in recent years. This is because of metaheuristic approaches having the ability to handle complex problems effectively while requiring less computational time to produce near-optimal results and are generally more robust to dataset size compared to traditional sequential heuristic methods. One such study conducted by Kucukyilmaz, T. [8], presented an island parallel model to solve the one-dimensional bin packing issue by using parallelisation approaches such Grouping Genetic Algorithms (GGA). In a separate study, a novel representation scheme for Genetic Algorithms (GA) was proposed by [9], presenting a unique representation scheme for Genetic Algorithms (GA), which differs from previous models in that it is compact, fixed length, and devoid of redundancies.

A comparative Study Conducted by Munien, C. et al [10], compared the performance of GA with other algorithms and found that although GA produces more accurate results for bigger datasets, it takes longer to compute than simpler heuristics like best fit and better fit. [11] used the Heuristic Genetic Algorithm (HGA) to optimise the vacant volume within bins using a genetic technique to solve the Three-Dimensional bin packing issue. By meeting several restrictions, such as stack priority, weight restrictions, box orientation, container stability, shipping placement, and overlapping limits, the study produced workable packing designs. Additionally, [12] developed an effective GGA that demonstrated a considerable performance boost, running 66 times quicker than its counterparts, by utilising the processing capacity of Graphics Processing Units (GPUs) via CUDA.

An additional study conducted by R. Yesodha applied the Firefly Algorithm (FA), a bio-Inspired algorithm to assess its performance on solving the classical one-dimensional BPP [13]. FA showed superior performance compared to traditional heuristic techniques such as FF, only being slightly outperformed by BF on Bigger datasets where both execution times are identical. According to these findings, FA may hypothetically perform better than BF in certain scenarios if it is given more computing time, displaying similar characteristics exerted by GA. Delving deeper into the effectiveness of FA, Kratika. C applied FA to solve the two-dimensional BPP [14], achieving efficient results compared to traditional heuristics and arriving to the same conclusion as the one-dimensional BPP.

To solve the BPP and other NP-Hard Problems, the two previously stated algorithms mostly rely on stochastic based approaches to locate solutions inside a search space. Tabu Search (TS) is another metaheuristic approach that sets itself apart from both GA and FA, by utilising a semi-deterministic approach for optimisation [15]. An initial study by Armin S et al [16], proposed a new solution called BISON, a Hybridised technique that combines both tabu search with branch-and-bound procedures. The mix of several bounds and heuristics used to determine suitable initial lower and upper limits is responsible for BISON's exceptional effectiveness on larger datasets. A further Study conducted by Adriana C.F aet al [17], proposed a hybridised improvement procedure for solving the one-dimensional BPP, where Tabu Search was implemented to minimise the number of solutions generated by lower bound methods that exceeds bin capacity, demonstrating satisfactory performance and robustness compared to other methods.

III. METHODOLOGY

Our proposed workflow consists of three main stages: dataset preprocessing, algorithm implementation, and performance evaluation. Each algorithm is executed and evaluated until it reaches a convergence state with no changes in 30 iterations. The detailed steps are outlined in Fig 2. The system specifications that were used to do the experiment is described as follows:

TABLE I. SYSTEM SPECIFICATIONS

Component	***Details***
CPU	Intel Core i7-1265U @1.3Ghz
RAM	8GB @3200Mhz

[a.] The system has no graphical processing unit.

b.

For this research, all the algorithms were developed using Java and compiled in the IntelliJ IDEA 2023.3.3.

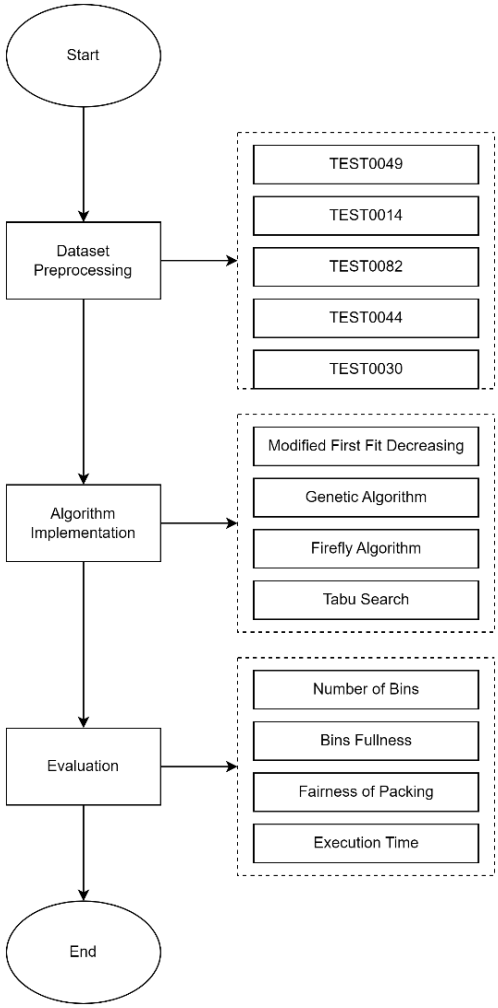

Fig. 1. Research workflow process

A. *Dataset preparation*

As shown in Fig. 1, there are 5 different types of problems found in this dataset that contains bin packing problem instances organised as follows: each instance has a unique problem name, indicating its distinctiveness; it includes the number of different item weights present in the problem and specifies the capacity of the bins, indicating the maximum weight each bin can hold. Next, for each of the m item weights, the dataset lists the item weight and the number of items that match that weight. Each of these problems will be used for evaluation.

B. *Algorithms implementations*

The choice of algorithms for solving the BPP boils down to choose from sequential heuristics and the wide variety of metaheuristic algorithms. For this research, each metaheuristic implements an underlying FF heuristic as a measure to provide better results, further supported by previous studies conducted by C. Munien, et al [10], which utilises BF and Better Fit as underlying heuristics. The algorithms chosen for this research are MFFD, GA, FA, and TS.

a) *Modified First Fit Decreasing*

First Fit Decreasing (FFD) is an approximate method that is often used to solve the bin packing problem [18]. Modified First Fit Decreasing improves upon the FFD by having an additional processing to the items by splitting them into large, medium, small, and tiny items [19]. Large items are those that take up over half of the bin's capacity, medium items are those smaller than or equal to half of the bin's capacity but larger than one third of it. Small items are smaller than or equal to one third of the bin's capacity but larger than a quarter of it. Tiny items are those smaller than or equal to a quarter of the bin's capacity. After splitting them into different sizes, they will all be sorted from largest to smallest. Then, starting from the items on the 'large' item size to the 'tiny' item size, try to put them into a bin that fits, and if there are none, create a new bin and put the item inside. The pseudocode for the implementation is given below.

Algorithm 1 – Modified First Fit Decreasing

```
Split items into four different sizes
Sort each different size from largest to lowest
for items in sizes [large to tiny]:
    place item in the first bin available
    if no available bin: create new bin and place item

return bins
```

In small datasets, FFD can perform as well as more complex methods such as GA [18]. The same applies to MFFD as MFFD is an improvement to FFD which is proven by [19]. The time complexity of this method would be O(n) since it scales linearly according to the size of the dataset, which makes it excellent in term of CPU time required to solve the problem.

b) *Genetic Algorithm*

A genetic algorithm (GA) is a metaheuristic optimisation method used to solve the bin packing issue that draws inspiration from the principles of natural selection and evolution [10]. It works by mimicking natural selection to repeatedly provide better answers to the bin packing problem. A population of candidate solutions (chromosomes) representing various methods of packing objects into bins is generated over numerous generations in a conventional genetic algorithm for bin packing. Every chromosome in the population correlates to a potential configuration for how things are packed into bins. Chromosomes are chosen, joined (crossover), and altered (mutation) to create new candidate solutions throughout evolution. Every chromosome's fitness, which is frequently determined by how many bins it uses or how much space it wastes, dictates how likely it is to be chosen for reproduction. Genetic algorithms search the solution space iteratively, utilising selection, crossover, and mutation operators to identify high-

quality packing solutions that maximise bin capacity and minimise waste. The pseudocode for implementation of the algorithm is given below.

Algorithm 2 – Genetic Algorithm

```
initialise random population of individuals
evaluate fitness of each individual
do
    Select parents using tournament selection
    Perform crossover operation
    Mutate offspring
    Evaluate population
    Replace old population with new population
while stop criteria not satisfied
return best/fittest solution
```

When compared to a set covering-based heuristic, GA yields better results for large-scale examples within short CPU times [10]. GA is a good option for handling the one-dimensional bin-packing problem as it may produce excellent results in a short amount of CPU time compared to other alternative heuristics in terms of quality and efficiency of the solution, which makes it a practical strategy for large-scale situations [20]. We opt for the conventional genetic algorithm over the grouped genetic algorithm for tackling the bin packing problem. This choice is based on the understanding that the conventional approach provides sufficient accuracy for the given problem while demanding less computational power than the alternative genetic algorithm.

c) Firefly Algorithm

Introduced by Xin-She Yang in 2008 [21], The Firefly Algorithm (FA) is a natured-inspired, stochastic-based, metaheuristic algorithm inspired by the flashing behaviour of fireflies, represented by individual fireflies within a population. This algorithm works by utilising the primary function of a firefly's flashing light, speculated to be a form of communication for finding potential mating partners. Each firefly represents a possible configuration on how objects are packed into bins with a corresponding fitness value determined by the number of bins used and fairness in packing.

Additionally, three standardised assumptions have been implemented to simplify the algorithmic process of FA. First, all fireflies are unisex to ensure that fireflies are attracted to all possible fireflies regardless of gender. Second, a firefly's attractiveness is directly and inversely proportional to brightness and distance respectively. Finally, Fireflies are attracted to other fireflies that exhibit higher attractiveness and the brightest firefly moves randomly since it is not attracted to any other fireflies [22]. The pseudocode for implementation of Firefly Algorithm is described in Algorithm 3.

FA is a promising choice for handling one-dimensional bin packing problem, as its stochastic based global search nature allows it to have excellent exploration properties useful for deterministic environments, while reducing the likelihood of getting stuck within local optima. Results from experiments done by Adidtya. P [23], shows FA outperforming GA in terms of performance by finding combinations of bin packing arrangement that results in less amounts of bins used, perfectly aligning with one of the metrics chosen for this comparative study. The pseudocode for FA is given below.

Algorithm 3 – Firefly Algorithm

```
initialise random population of fireflies
evaluate fitness of the fireflies
do
    for i=1 to n (all n fireflies)
      for j=1 to n (all n fireflies)
      calculate light intensity
          If Ij > Ii then
              move firefly i towards firefly j
          end if
      end for j
    end for i
    evaluate fireflies
While stop criteria not satisfied
return best/fittest firefly
```

d) Tabu Search

Tabu Search is a semi-deterministic metaheuristic algorithm developed by Fred. G in 1986 [23]. It utilises local search techniques by taking promising potential solutions and exploring its adjacent neighbours in the hopes of finding better solutions. Additionally, Tabu search introduces memory structures that stores solutions that have either been recently visited or violate certain rules, where the algorithm uses to avoid exploring these forbidden solutions. The pseudocode for TS is given below.

Algorithm 4 – Tabu Search

```
initialise solutions with first fit heuristic
evaluate solutions
do
    Find Best Solution
    Search Neighbouring Best Solutions
    Generate Candidate Solutions
    Evaluate Candidate Solutions
    Update Tabu List
while stop criteria not satisfied
return best/fittest solution
```

As Tabu search is a semi-deterministic metaheuristic algorithm, it provides us more diversity

for comparative analysis as the other two metaheuristic methods previously discussed are stochastic based, while MFFD is a sequential heuristic.

C. *Evaluation metrics*

The program evaluates the performance of each algorithm based on these several key metrics.

1) Number of bins: The total number of bins used by each algorithm to pack all the items. A lower bin count suggests better optimisation and resource utilisation.

2) Bin fullness: The degree of fullness of each bin, calculated as the percentage of capacity utilised in each bin. Higher bin fullness indicates more efficient packing and reduced wastage of space.

3) Fairness of packing: This metric assesses the fairness in the distribution of items among the bins using standard deviation calculation. It evaluates how evenly items are distributed across bins, aiming to minimise disparities in bin usage.

4) Computational time: The evaluation metric for computational time assess how long it takes for an algorithm to converge, meaning it achieves the same result consistently over 30 consecutive iterations.

By evaluating these metrics, our program provides insights into the performance and effectiveness of each algorithm in solving the bin packing problem. Users can compare the results obtained by different algorithms and select the most suitable approach based on their specific requirements and constraints.

IV. RESULT AND DISCUSSION

A. *Number of bins and bin fullness*

Each algorithm is executed at least 30 times to obtain the total amount of bins used for packing items into individual bins and determining the fullness of each used bin. Fig. 2, shows the performance results of all algorithms based on this metric.

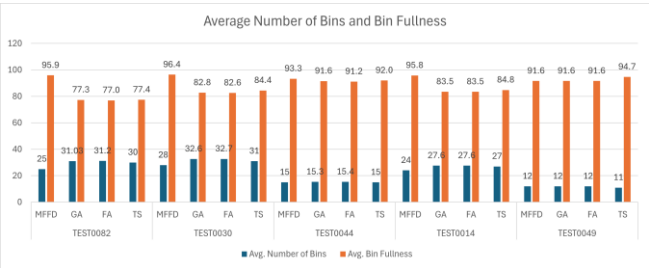

Fig. 2. Total bin counts and bin fullness

The results indicate MFFD's superior performance on packing items and maximising bin capacity, managing to find optimal solutions in both total number of bins used, and average bin fullness compared to the other algorithms on 3 out of 5 test cases. TS closely followed by managing to tie with MMFD on one test case and beating MMFD on another. Finally, the performance of GA and FA are almost identical, with GA only being marginally better than FA on 3 test cases.

MFFD managing to beat all the metaheuristic algorithms likely stems from the size of the BPP dataset. Since MFFD is a sequential heuristic, it is more likely to find near-optimal solutions in small to medium test sizes compared to most metaheuristic approaches that utilises stochastic-based search methods, further supported by sequential heuristics not exhibiting strong exploration behaviour compared to algorithms like FA, where solutions are compared with each other which may not always provide good results for smaller scale datasets.

B. *Fairness of packing*

For the fairness of packing results, Fig. 3. and Fig. 4. presents Tabu Search as the best overall algorithm, outperforming GA, FA, and especially MFFD. Both GA and FA perform almost identically once again, with GA slightly having an edge compared to FA. An interesting finding is MFFD's performance, as it is by far the worst performing algorithm when compared in fairness of packing.

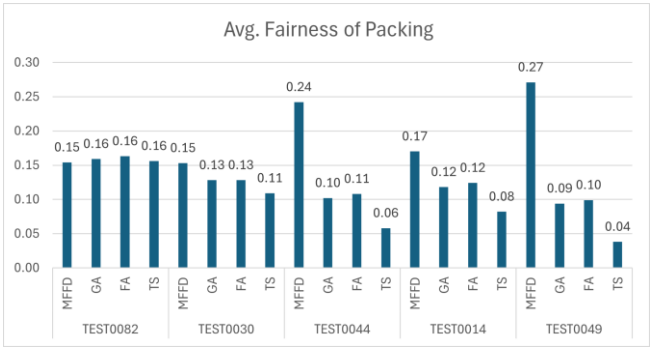

Fig. 3. Average fairness of packing

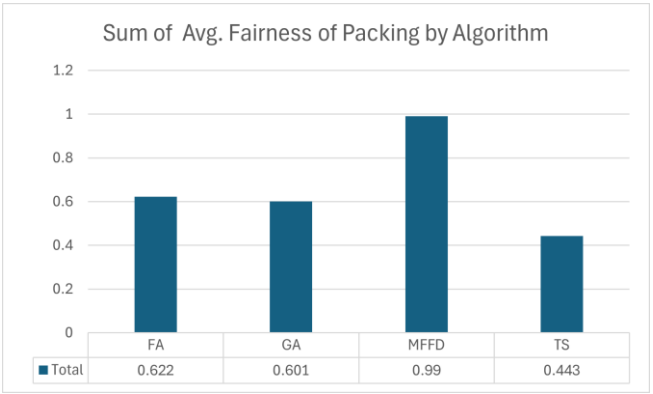

Fig. 4. Sum of average fairness of packing

MFFD's utilisation of FF heuristic, essentially causes the algorithm to always pack items without exploring other packing combinations that results in overall better distribution of item weights. In contrast, all three metaheuristic algorithms managed to find combinations of items that results in better distribution of items, likely caused by the explorative configurations allowing these algorithms to explore more possible combinations which results in better distribution of items.

C. Computational time

Figure 4 shows the execution time of each algorithm for every test case. modified first fit decreasing managed to run faster than every other algorithm in all tests, while firefly algorithm is considered the longest, as it has the highest computation runtime on all 5 test sets. tabu search is generally faster than genetic algorithm, making it overall the second fastest algorithm.

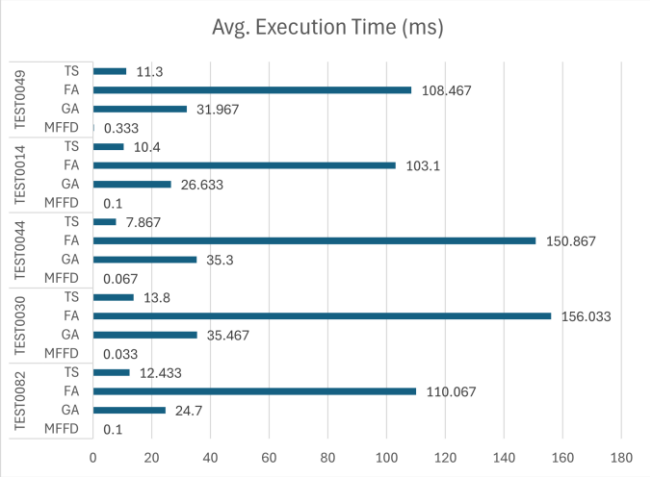

Fig. 5. Computational Runtime

MFFD is faster than the other algorithms as it iterates over every item once and puts them into a bin and will iterate only once. TS will iterate multiple times, searching for a better solution. The algorithm starts by using first fit decreasing then searches other potential solutions for improvements. Because it iterates several times after using first fit decreasing, it will take longer than modified first fit decreasing which only iterates once.

Since GA will have multiple initial solutions and tries to combine them with each other to create new and potentially better solution, as well as needing to compute mutation, the computation for each iteration will be longer than tabu search. Finally, FA has the longest execution time due to each individual solution being compared to the entire population of solutions, effectively making the time complexity $O(n^2)$.

V. Conclusion

This study conducted a comparative analysis of four distinct algorithms for solving the classical one-dimensional BPP, where the FF heuristic was utilised as underlying heuristics for all metaheuristic approaches. Out of all the algorithms, MFFD is the most optimal algorithm by being able to find optimal solutions within all test sets on small to medium scale datasets within excellent computational time. However, MFFD does not perform well for fairness of packing due to its sole reliance on FF heuristic. Tabu search is the second-best performing algorithm, as it combines both stochastic and semi-deterministic methods to balance exploration and exploitation, making it perform above average on all metrics. GA and FA are both the least optimally performing algorithm, as they lack mechanism of properly exploiting promising solutions. Therefore, the optimal algorithm choice for solving the BPP problem narrows down into either MFFD and TS, where MFFD should be prioritised on instances where performance optimality and computation time is heavily valued on small to medium scale datasets, while TS is preferred for instances where even distribution of items is important without sacrificing performance optimality and computation time.

VI. References

[1] K. Fleszar and K. S. Hindi, “New heuristics for one-dimensional bin-packing,” *Comput Oper Res*, vol. 29, no. 7, pp. 821–839, Jun. 2002, doi: 10.1016/S0305-0548(00)00082-4.

[2] E. G. Coffman Jr, J. Y.-T. Leung, and J. Csirik, “Variants of Classical One-Dimensional Bin Packing.” 2007.

[3] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 174. freeman San Francisco, 1979.

[4] S. I. Gass and C. M. Harris, “Encyclopedia of operations research and management science,” *Journal of the Operational Research Society*, vol. 48, no. 7, pp. 759–760, 1997.

[5] E. G. Coffman, M. R. Garey, and D. S. Johnson, “Approximation Algorithms for Bin-Packing — An Updated Survey,” 1984, pp. 49–106. doi: 10.1007/978-3-7091-4338-4_3.

[6] G. Carmona-Arroyo, J. B. Vázquez-Aguirre, and M. Quiroz-Castellanos, “One-Dimensional Bin Packing Problem: An Experimental Study of Instances Difficulty and Algorithms Performance,” in *Studies in Computational Intelligence*, vol. 940, Springer Science and Business Media Deutschland GmbH, 2021, pp. 171–201. doi: 10.1007/978-3-030-68776-2_10.

[7] D. S. Johnson and M. R. Garey, “A 71/60 Theorem for Bin Packing*,” 1985.

[8] T. Kucukyilmaz and H. E. Kiziloz, “Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem,” *Comput Ind Eng*, vol. 125, pp. 157–170, Nov. 2018, doi: 10.1016/j.cie.2018.08.021.

[9] N. Mohamadi, “Application of Genetic Algorithm for the Bin Packing Problem with a New Representation Scheme,” 2010. [Online]. Available: www.SID.ir

[10] C. Munien, S. Mahabeer, E. Dzitiro, S. Singh, S. Zungu, and A. E.-S. Ezugwu, “Metaheuristic Approaches for One-Dimensional Bin Packing Problem: A Comparative Performance Study,” *IEEE Access*, vol. 8, pp. 227438–227465, 2020, doi: 10.1109/ACCESS.2020.3046185.

[11] R. Sridhar, M. Chandrasekaran, C. Sriramya, and T. Page, “Optimization of heterogeneous Bin packing using adaptive genetic algorithm,” *IOP Conf Ser Mater Sci Eng*, vol. 183, p. 012026, Mar. 2017, doi: 10.1088/1757-899X/183/1/012026.

[12] S. O. Ozcan, T. Dokeroglu, A. Cosar, and A. Yazici, “A Novel Grouping Genetic Algorithm for the One-Dimensional Bin Packing Problem on GPU,” 2016, pp. 52–60. doi: 10.1007/978-3-319-47217-1_6.

[13] R. Yesodha, “Effectiveness of FireFly Algorithm in solving Bin Packing Problem,” *ijarcsse*, 2013.

[14] Kratika. C., “Firefly Algorithm to Solve Two Dimensional Bin Packing Problem,” 2014.

[15] A. M. Connor and K. Shea, “A comparison of semi-deterministic and stochastic search techniques,” May 2016.

[16] A. Scholl, R. Klein, and C. Jürgens, “Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem,” *Comput Oper Res*, vol. 24, no. 7, pp. 627–645, Jul. 1997, doi: 10.1016/S0305-0548(96)00082-2.

[17] A. C. F. Alvim, C. C. Ribeiro, F. Glover, and D. J. Aloise, “A Hybrid Improvement Heuristic for the One-Dimensional Bin Packing Problem,” *Journal of Heuristics*, vol. 10, no. 2, pp. 205–229, Mar. 2004, doi: 10.1023/B:HEUR.0000026267.44673.ed.

[18] G. Carmona-Arroyo, J. B. Vázquez-Aguirre, and M. Quiroz-Castellanos, “One-Dimensional Bin Packing Problem: An Experimental Study of Instances Difficulty and Algorithms Performance,” in *Studies in Computational Intelligence*, vol. 940, Springer Science and Business Media Deutschland GmbH, 2021, pp. 171–201. doi: 10.1007/978-3-030-68776-2_10.

[19] D. S. Johnson and M. R. Garey, “A 71/60 Theorem for Bin Packing*,” 1985.

[20] C. Munien and A. E. Ezugwu, “Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications,” *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 636–663, Apr. 2021, doi: 10.1515/jisys-2020-0117.

[21] X.-S. Yang, “Firefly Algorithms for Multimodal Optimization,” Mar. 2010.

[22] P. A. and P. L. N. Chai-ead, “Bees and Firefly Algorithms for Noisy Non-Linear Optimisation Problems ,” 2011.

[23] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Comput Oper Res*, vol. 13, no. 5, pp. 533–549, Jan. 1986, doi: 10.1016/0305-0548(86)90048-1.