



A Plant Propagation Algorithm for the Bin Packing Problem


Rewayda Razaq Abo-Alsabeh

(Faculty of Computer Science and Mathematics, University of Kufa, Iraq)
 <https://orcid.org/0000-0002-2572-8845>, ruwaida.mohsin@uokufa.edu.iq

Meryem Cheraitia

(LabSTIC, 8 Mai 1945 University, Guelma, Algeria)
 <https://orcid.org/0000-0003-3642-5572>, meryem.cheraitia@hotmail.fr

Abdellah Salhi

(School of Mathematics, Statistics and Actuarial Science, University of Essex, UK)
 <https://orcid.org/0000-0003-2433-2627>, as@essex.ac.uk

Abstract: We consider the one-dimensional Bin Packing Problem (BPP) and its solution using a novel heuristic approach namely the Plant Propagation Algorithm (PPA). The problem can be stated as follows: given a set of objects of various weights, sizes, or any measure, and a set of bins each with fixed capacity, find the minimum number of bins needed to pack all the items such that the sum of the elements packed inside each bin does not exceed its capacity. BPP is a well-studied problem, however, being NP-Hard, the search for a computationally viable solution approach for it continues. In this paper, we report on our implementation of PPA for BPP and its performance against other algorithms on number of non-trivial instances. Computational results and their discussion are included.

Keywords: Algorithms, Approximate Solution, Bin Packing, Heuristic, Plant Propagation Algorithm

Categories: G

DOI: 10.3897/jucs.102470

1 Introduction

The Bin-Packing Problem (BPP) concerns packing a finite number of objects with different sizes (weights) into the minimum number of fixed size bins. It is a classical problem that goes back to the 1970s and earlier. In fact, versions of it have been addressed as early as the thirties. However, the lack of computational facilities then limited progress in its practical solution. It is NP-hard meaning that no polynomial-time deterministic algorithm has yet been discovered for the solution of a large and practical instance of the problem, [Garey and Johnson 1979, Coffman Jr et al. 2007].

The problem can be seen as a special case of the one-dimensional cutting-stock problem, the container loading problem, and some scheduling problems, [De Carvalho 2002].

In practice, many variations on the problem can be specified based on the bins sizes, the placement conditions, the priority of packing items etc... BPP has important applications in the real world particularly in cutting industries, warehousing, supply chain, and transportation, [Munien and Ezugwu 2021].

Several specific cases have been provided in the literature. In [Coffman et al. 1978] was put forward the idea of finding the maximum number of items to store in a bin with

respect to storage allocation problems. Krause et al. illustrated a different case of the problem that puts a condition on the maximum number of items permitted to be stored in each bin, [Krause et al. 1975].

Others worked on higher dimensional variants such as the two dimensional rectangular bin packing problem which concerns fitting a set of rectangles into a minimum number of rectangular bins, [Gass and Harris 1997]. Martello suggested a three dimensional version that also demands storing a set of $3d$ rectangular items into a minimum number of $3d$ rectangular bins, [Martello et al. 2000]. Both problems should have bins with similar sizes and the items are not permitted to be overlapping. Many surveys have comprehensively reviewed the problem, the most recent being by [Munien and Ezugwu 2021], [Christensen et al. 2017], and [Delorme et al. 2016]. Previous ones are by [Garey and Johnson 1981] and [Coffman et al. 1984].

Over the years, a wide variety of strategies have been suggested for BPP. Yet, none effective on all instances due to the complexity of the problem as stated earlier, [Carmona et al. 2022]. Heuristics and algorithms of both deterministic and stochastic natures have been used. These include Tabu Search, a semi-deterministic approach, Simulated Annealing a stochastic heuristic, [Munien et al. 2020], and Branch-and-Bound which is deterministic and exact. Also, approximate approaches, of the metaheuristic and hybrid metaheuristic types, were employed. Online and offline approximating algorithms were the first proposed procedures for BPP, the online algorithms being based on heuristic rules, such as First Fit (FF), Next Fit (NF), and Best Fit (BF). Others used hybrid approximating rules, such as FF Decreasing (FFD) and BF Decreasing (BFD) rules, [Munien and Ezugwu 2021, Sgall 2014, Ojeyinka 2015, Johnson 1973].

Metaheuristics have shown their robustness and strength on optimization problems. Moreover, they are simple to apply and often reach good approximate solutions in reasonable computational times. They are often nature-inspired, simulating biological, physical, or sociological systems (phenomena), [Abo-Alsabeh and Salhi 2020, Abo-Alsabeh and Salhi 2021, Abo-Alsabeh and Salhi 2022, Abo Alsabeh 2017].

Algorithms applied to BPP include the genetic algorithm (GA) [Luo et al. 2017], Tabu Search (TS) method [Crainic et al. 2009], Cuckoo Search Algorithm (CSA) [Layeb 2011], Firefly Algorithm (FA) [Yang 2009], Particle Swarm Optimization (PSO) [Clerc and Kennedy 2002, Garcia et al. 2012], Ant Colony Optimization (ACO) [Dorigo and Gambardella 1997], and Artificial Bee Colony algorithm (ABC) [Karaboga and Basturk 2007].

Zendaoui and Layeb provided an Adaptive Cuckoo Search (ACS) algorithm to solve the BPP by proposing an important mechanism, called Ranked Order Value (ROV). It is a technique employed to transform a continuous solution space into a discrete one. The concept of finding discrete solutions relies on integer permutations based on Levy flight and on a decoding mechanism, [Zendaoui and Layeb 2016].

The FA is a population-based algorithm that is inspired by the flashing behavior of fireflies and their bio-luminescent communication. Layeb and Benayad illustrated a new FA algorithm called the Firefly Colony Optimization algorithm (FCO) that is inspired by the ACO algorithm. It merges distributed computation with a constructive greedy procedure to get near-optimal solutions to the problems, [Layeb and Benayad 2014].

Abdullah et al. introduced a swarm intelligent algorithm called the Fitness Dependent Optimizer (FDO). It is inspired by the collective intelligence of bees, throughout the reproductive process of the colony when they are looking for new nest sites (hives). It is based on the PSO algorithm where the search agent location is updated by adding velocity. The difference between them is that the FDO computes the velocity in a different way, [Abdullah and Ahmed 2019].

Abdul-Minaam et al. adapted the FDO for solving the bin packing problem. They provided an Adaptive Fitness Dependent Optimizer (AFDO) using an optimized swarm algorithm and the FDO. Their algorithm relies on generating a feasible random initial population by the FF algorithm and on updating the algorithm procedure. They compared their results with three other well-known algorithms, namely the PSO algorithm, Crow Search Algorithm (CSA), and Jaya Algorithm, [Abdul et al. 2020].

The Plant Propagation Algorithm (PPA) is an optimization algorithm developed by Salhi and Fraga in 2011. It is inspired by the way strawberry plants propagate using runners, [Salhi and Fraga. 2011]. This strategy seems to be effective as is shown widely by the performance of PPA on a variety of problems. Although originally designed for continuous problems over normed spaces for the purpose of distinguishing between short and long runners, it has been successfully adapted for discrete optimization as in TSP, [Selamoglu and Salhi 2016], exam timetabling problems, [Cheraitia et al. 2019] and others [Geleijn et al. 2019, Slegers and van den Berg 2020, Paauw and Van den Berg 2019]. For continuous optimization implementation, it works well such as the Seed-based Plant Propagation Algorithm (SbPPA) [Sulaiman and Salhi 2015] and the hybridization of PPA and SbPPA [Sulaiman and Salhi 2016] which combines the runner-based strategy for exploitation and the seed-based strategy for exploration.

Here, we consider applying PPA to BPP. The paper is organized as follows: We briefly introduce PPA in Section 2. A model of BPP is described in Section 3. The main algorithm is provided in Section 4. Computational results are reported in Section 5, finally. Section 6 is the conclusion.

2 The Plant Propagation Algorithm

In nature, hybrids, including plants such as the strawberry plant do not have the ability to reproduce by seed; cuttings and runners are usually used as a strategy for propagation. The plant strives hard for its offspring to grow in a land rich in the necessary nutrients and growth ingredients such as light and moisture. For this, the plant sends runners in all directions searching for the right place to optimize its growth. Once these runners touch the ground, roots are put down by the plant. If the strawberry plant is in a good place, it sends short runners in large numbers, but if it is in a poor place, it sends a small number of long runners to far places looking for more suitable sites for growth.

This problem can be seen as an optimization one solved by using the runners strategy. Exploiting this approach involves de facto mapping our optimization problem onto a survival optimization one.

Consider the generic optimization problem

$$\max_{x \in K} z = f(x) \quad (1)$$

where K is the search space and the aim is to detect the best point x in K , i.e. one that gives $f(x)$ the largest value or an approximate of it. A population of h solutions $x \in K$ is first generated randomly. $f(x)$ is then computed for each one of them. They are ranked in descending order according to these values of $f(x)$. From each of the top α ranking solutions (α to be decided) a number $N(x)$ of runners is generated from each of the α high-ranking solutions. A suitable $N(x)$ of runners is computed in a certain way described in [Salhi and Fraga. 2011] for continuous problems. From the rest of the solutions one single runner and, therefore, a new solution is generated. As already

mentioned, short runners implement exploitation while long ones implement exploration and diversification.

PPA relies on a fitness function which is no other than the objective function of the problem as well as parameters such as the number of generations, the number of runners to generate for each solution, and the length of each runner. These parameters are on the whole set arbitrarily, although the knowledge of the problem may provide some guidance as to how to choose them favorably. It is, however, important to note the relatively low number of parameters compared to what is needed in, say the Genetic Algorithm, [Abo-Alsabeh et al. 2020, Pratama et al. 2023].

For completeness, we describe the original algorithm meant for constrained continuous optimization. The algorithm is iterated q_{max} generations. This parameter is used in the stopping criteria. As the algorithm sorts the plants/solutions according to their fitness values, these are normalized to be such that $f(x) \in [0, 1]$. The function used to calculate the number of runners for a given solution requires that the fitness values $f(x)$ in $[0, 1]$ be mapped onto $(0, 1)$. This is guaranteed by the following equation.

$$N(x) = \frac{1}{2}(\tanh(4f(x) - 2) + 1) \quad (2)$$

where the tanh function is used to assure that the length of the runner will not cross the boundaries of the search space, which could be just those of a unit hypercube. It also guarantees that the number of runners created for a plant is proportional to its fitness. The following formula determines this number:

$$n_k = \lceil n_{max} N_j k \rceil \quad (3)$$

where n_k is the runners' number created for shoot j in the 'current' sorted population, n_{max} is the maximum number of runners allowed to be generated, N_j is the mapped fitness in Equation (2) of a shoot j , and $k \in [0, 1]$ is a selected random number for each shoot at each generation. The length of each runner is suggested as follows:

$$D_{k,i} = 2(1 - N_j)(k - 0.5), \quad i = 1, \dots, n \quad (4)$$

where n is the space dimension, $D_{k,i}$ will be in $(-1, 1)$, and N_j here guarantees that the best shoots will have the possibility of sending runners for distance > 0 even if $f_j(x) \equiv 1$. The shoot j is updated via this function using defining bounds $[a_i, b_i]$ on x_i .

$$x_i^* = x_i + (b_i - a_i)D_{k,i} \quad (5)$$

The new population of individuals is evaluated and added to the initial one, it is sorted in descending order, then the best solutions are moved to the next generation. To keep the population constant in size, low ranked individuals are omitted. This process is repeated until the optimal or near-optimal solution is found, in other words, the stopping criterion is satisfied. The algorithm is stated as follows:

3 The Bin Packing Problem

The classic mathematical formulation of the problem can be stated as given a set of n items $J = (J_1, J_2, \dots, J_n)$, where J_i has a value (weight, size, or any measure), capacity $C \in N$, ($0 < J_i < C, i = 1, 2, \dots, n$). The demand is to find the smallest integer, m , such that there is a partition of $J = (K_1, K_2, \dots, K_m)$ where the sum of the items'

Algorithm 1: Original Plant Propagation Algorithm

```

1 Require objective  $f(x)$ ,  $x \in R^n$ ;
2 Create a population  $P = \{p_j, j = 1, \dots, h\}$ ;
3  $q \leftarrow 1$ ;
4 for  $q \leftarrow 1$  to  $q_{max}$  do
5 Evaluate  $N_j = f(p_j)$ ,  $\forall p_j \in P$ ;
6 Rank  $P$  in descend order of  $N$ ;
7 Generate new population  $\phi$ ;
8 for each  $p_j, j = 1, \dots, h$  do {best  $h$  only};
9  $k_j \leftarrow$  set of runners such that the set size and each runner distance is
   proportional to the fitness  $N_j$ ;
10  $\phi \leftarrow \phi \cup k_j$  {append to population; death occurs by omission above};
11 end for
12  $P \leftarrow \phi$  {new population};
13 end for
14 Return  $P$ ;

```

weights, $J_i \in K_j$, needed not to exceed C . Each K_j can be viewed as the contents of a bin with capacity, C , [Munien et al. 2020, Gourgand et al. 2014].

The problem can be formulated as follows [Munien et al. 2020]: Let U be an upper bound on the minimum number of bins requested to store all the items in, such that they are numbered as $1, 2, \dots, U$. Let n be the number of items to be stored.

$$\text{Min} \quad \sum_{i=1}^U y_i \quad (6)$$

subject to

$$\sum_{j=1}^n K_j x_{ij} \leq C y_i \quad \forall i \in \{1, \dots, U\} \quad (7)$$

$$\sum_{i=1}^U x_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \quad (8)$$

$$x_{ij} \in \{0, 1\}, \quad y_i \in \{0, 1\}, \quad \forall i \in \{1, \dots, U\}, \quad \forall j \in \{1, \dots, n\} \quad (9)$$

where,

$$y_i = \begin{cases} 1, & \text{if bin } i \text{ is used in the solution } (1 \leq i \leq U) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

$$x_{ij} = \begin{cases} 1, & \text{if items } j \text{ is stored into bin } i \quad (1 \leq i \leq U; 1 \leq j \leq n) \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

In this model, the objective function minimizes the number of bins, N , wanted to

store the items where all the bins have similar capacity C .

$$N \geq \lceil (\sum_{i=1}^n S_i) / C \rceil \quad (12)$$

where S_i is the weight of each item in the list J . Equation. (7) ensures that the weights of the items do not exceed the capacity and Equation. (8) makes sure that each item is stored in only one bin. Equations (10) and (11) are two 0-1 decision variables with equation (10) ensuring that each item is placed in precisely one bin.

BPP has been shown to be NP-complete in [Karp, 1972] and it is known to be NP-Hard even in its simplest form, i.e. binary and one-dimensional [Garey and Johnson 1979]. This is why heuristic and approximate algorithms are the most viable solution approaches for BPP, and generally other NP-Hard problems such as Knapsack, Set-Packing etc...; they generally provide satisfactory results in reasonable computational times, [Munien et al. 2020]. Having said that, not all heuristics are as good as each other. Hence, we investigate for the first time the suitability of PPA for BPP. Its implementation is given below.

4 PPA for the bin packing problem

In any solution approach, it is essential to have a suitable way to represent the solutions to the problem. This allows for navigation in the search space and sifting through the potential candidates for the optimum when it exists.

4.1 Solution representation

As mentioned earlier, PPA was initially suggested for continuous optimization problems. Here, an Integer Representation is used instead, i.e. the solution candidate will be represented as an integer permutation vector of items. Each item is assigned to a specific bin by using the indices and the value of the item, see Table 1.

Items	1	2	3	4	5	6	7	8	9	10
weight	7	3	6	2	4	3	5	10	4	9
Bin	1	1	2	2	2	3	3	3	4	4

Table 1: Integer representation

4.2 Objective function

For the problem at hand, using only the number of bins as the objective function can produce various arrangements of solutions with the same number of bins. This can lead to a stagnating search. Because of this, combining this information with another criterion such as the fullness of the bins can provide better function values.

$$\text{Min } f = 1 - \frac{\sum_{k=1}^N (F_k/C)^k}{N} \quad (13)$$

$$F_k = \sum_{i=1}^n S_i \tag{14}$$

where N is the total number of bins, F_k is the fill of bin i , C is the capacity of bin, k is a constant (best $k = 2$) which defines equilibrium of the filled bin. This fitness function minimizes the number of bins used for packing and reduces the remaining empty spaces in each bin after sorting the items.

Runner1	2	1	8	4	5	6	7	3	9	10
Runner2	6	2	3	4	5	1	9	8	7	10
Runner3	1	2	3	5	4	6	7	10	9	8

Table 2: Three runners are generated from one plant

4.3 Searching the space

In the absence of a norm in the traditional sense to go with the discrete space of interest here, we consider the dissimilarity of permutations as a surrogate for the distance between them. In this way, short runners can be implemented by changing a permutation slightly (one or two indices), and long runners by changing 3 or more. Note that 'runner' and 'solution' are used interchangeably. In the present implementation, we used three types of alterations of solutions to generate new ones. The first type of new solution is obtained by randomly changing only two places of the items of the permutation. The second type is obtained by changing two positions to other ones, and the third type is by changing more than two positions.

Suggest that the number of runners to be generated for a plant from the second type is three. These runners can be represented as in table 2, where the color numbers are the items indexes that are already selected and their positions changed: The parameters are set as follows: Initial population $h = 20$, generations number $q_{max} = 5$. The number of runners is computed by Equation (2), for example, suppose a shoot/plant j has a fitness function $f(x) = 21$ and we set $n_{max} = 10$, $k = 0.4$, then $N(x) = \frac{1}{2}(\tanh(4 * 21 - 2) + 1) = 1$ and $n_k = \lceil 10 * 1 * 0.4 \rceil = 4$ runners. In this problem, there is no need to compute the runners' distance as the solution is an integer.

In this implementation, to select the solutions randomly, we chose μ as the selection parameter. We set $\mu = 60\%$ to create short runners from the first and second types and $\mu = 40\%$ for long runners from the third type. The initial population is set between (40-100) and the generation number is between (20-100). A solution is not allowed to create different types of runners within a generation. Algorithm 2 displays the suggested method's pseudo-code. Lines 10 and 11 show the creation of the runners. Lines 12 and 13 describe the updating process.

To fill the bins, we open them one by one and fill them with the items randomly. The fitness function recognizes the solution with the best set of bins containing all the items with fewer empty spaces and without exceeding the capacity of the bins. In other words, we check the feasibility of solutions before appending them to the population. This is how the constraints of the problem are mapped into the implementation of PPA. All the solutions that are generated by the plants are added to the current population to create a

Algorithm 2: The adaptive Plant Propagation Algorithm

```

1 Require objective  $f(x)$ ,  $x \in R^n$ ;
2 Create a population  $P = \{p_j, j = 1, \dots, h\}$ ;
3  $q \leftarrow 1$ ;
4 for  $q \leftarrow 1$  to  $q_{max}$  do;
5 Evaluate  $N_j = f(p_j)$ ,  $\forall p_j \in P$ ;
6 Rank  $P$  in descending order of  $N$ ;
7 Generate new population  $\phi$ ;
8 Input  $\mu$  the rate of selection;
9 for each  $p_j, j = 1, \dots, h$ , do {best individuals only};
10  $k1_j \leftarrow$  create a set of short runners from the first and second types;
11  $k2_j \leftarrow$  create a set of long runners from the third type;
12  $k_j = k1_j \cup k2_j$ , such that the set size is proportional to the fitness  $N_j$ ;
13  $\phi \leftarrow \phi \cup k_j$  {append to population; death of weak individuals is by omission
    from  $\phi$ };
14 end for;
15  $P \leftarrow \phi$  {new population};
16 end for;
17 Return  $P$ ;

```

new one. The latter is sorted in descending order and the best solutions are selected to go into the next generation. Repeating this process improves the number of bins. To keep the population constant, unwanted individuals are omitted from the new population.

5 Computation Results

To test our procedure, we consider a set of well-known benchmark BPP instances found at <https://site.unibo.it/operations-research/en/research/bpplib-a-bin-packing-problem-library>, [Delorme et al. 2018].

These instances are in three categories: easy, medium, and hard. The procedure is implemented in Matlab 2014a under the 32 bits operating system. The processor is (Intel(R) Core(TM) i5 CPU, Duo 2.53 GHz, and 4. GB of RAM). We decided to choose the instances that are solved by [Abdul et al. 2020] to compare their results and the ones from the Best first heuristic with those from Algorithm 2. The tables display the instance name in the first column, then the number of items, the bin's capacity, the best known result, BF, AFDO, and PPA results respectively. The Friedman test has been used to compare the obtained outcomes statistically.

Table 3 shows that our algorithm provides a similar number of bins for all fourteen instances except for the N2C1W1-C and N2C2W1-H as it uses an extra bin. The results are not different from the best known generated by other algorithms; the difference is in one bin only as is clear in Figure 1.

Table 4 gives similar outcomes too, except for N4W4B2-R7 which needs an extra bin, while N2W2B1-R2 needs one less bin than the BF algorithm. It can be seen in Figure 2 that PPA outperforms BF in that it solves most of the instances.

Instance	N	C	Best known	BF	AFDO	PPA
N1C1W1 – B	50	100	31	31	31	31
N1C1W1 – E	50	100	26	26	26	26
N1C1W1 – I	50	100	25	25	25	25
N1C1W1 – M	50	100	30	30	30	30
N1C1W1 – Q	50	100	28	28	28	28
N1C1W2 – D	50	100	31	31	31	31
N1C2W1 – P	50	120	21	21	21	21
N1C2W2 – R	50	120	25	25	25	25
N1C3W2 – A	50	150	19	19	19	19
N2C1W1 – B	100	100	49	49	49	49
N2C1W1 – C	100	100	46	46	46	47
N2C1W4 – F	100	100	77	77	77	77
N2C2W1 – H	100	120	46	46	46	47
N3C1W4 – N	200	100	148	148	148	148

Table 3: Results of Algorithm for the easy type

Instance	N	C	Best known	BF	AFDO	PPA
N1W1B1 – R2	50	1000	19	20	19	19
N1W1B1 – R5	50	1000	17	19	17	17
N1W1B2 – R7	50	1000	18	19	18	18
N1W2B2 – R9	50	1000	11	11	11	11
N1W4B3 – R8	50	1000	6	6	6	6
N2W2B1 – R2	100	1000	21	22	21	21
N2W2B3 – R9	100	1000	20	20	20	20
N3W1B3 – R5	200	1000	65	65	65	65
N3W4B2 – R9	200	1000	22	22	22	22
N4W4B2 – R7	500	1000	57	57	57	58

Table 4: Results of Algorithm for the medium type

For the hard category, the AFDO solved only five instances, but we decided to test PPA on some other instances already solved by other methods. Table 5 shows that our results are similar in two sets of their instances and better than the others. Results from the other five extra sets are better than those of the BF heuristic in four sets and similar in one. The sign (...) in the table means no result has been returned. It can be seen in Figure 3 that PPA outperforms the other two algorithms in all the instances.

Overall, PPA succeeds in determining near optimal solutions in comparison to BF and AFDO methods.

To give more details about PPA convergence, we plotted the objective function values over the number of iterations for the instance N1C1W1-M. Also, maximum and minimum values are compared with the average value along the graph, see Figure 4.

Instance	N	C	Best known	BF	AFDO	PPA
Hard0	200	100000	56	59	59	59
Hard1	200	100000	57	60	...	59
Hard2	200	100000	56	60	...	59
Hard3	200	100000	55	59	59	58
Hard4	200	100000	57	60	60	59
Hard5	200	100000	56	59	60	59
Hard6	200	100000	57	60	...	60
Hard7	200	100000	55	59	58	58
Hard8	200	100000	57	60	...	59
Hard9	200	100000	56	60	...	59

Table 5: Results of Algorithm for the hard type

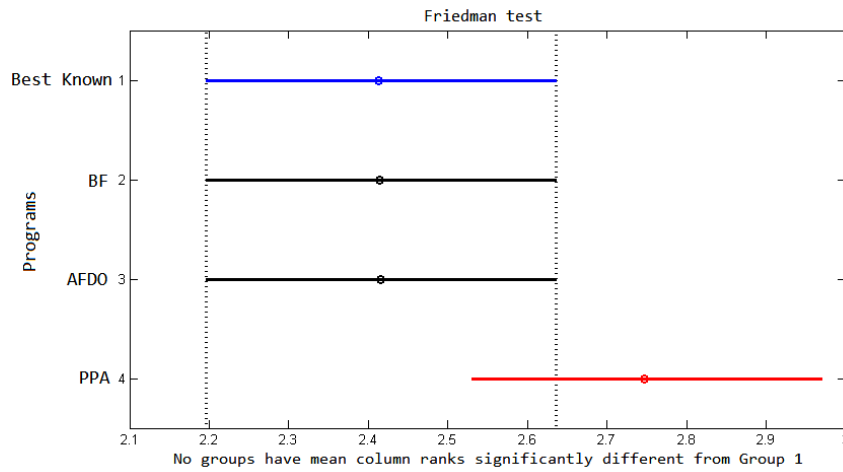


Figure 1: Friedman test for easy datasets

In graph (a), we can see that the value of the objective function drops sharply after about five iterations. Then it tails off smoothly over the next 40 generations. Convergence is then achieved to a local optimum with little improvement observed after the 50th iteration. Figure (b) shows the maximum and minimum fitness values with their average. Figure (c) plots the graph using a base 10 logarithmic scale for the iteration number axis and a linear scale for the objective function axis.

Finally, Figure 5 displays different packing schemes for the instance N2W2B3-R9 to pack 100 items into 20 bins, Figure (a) shows the scheme of PPA and (b) that of the AFDO method.

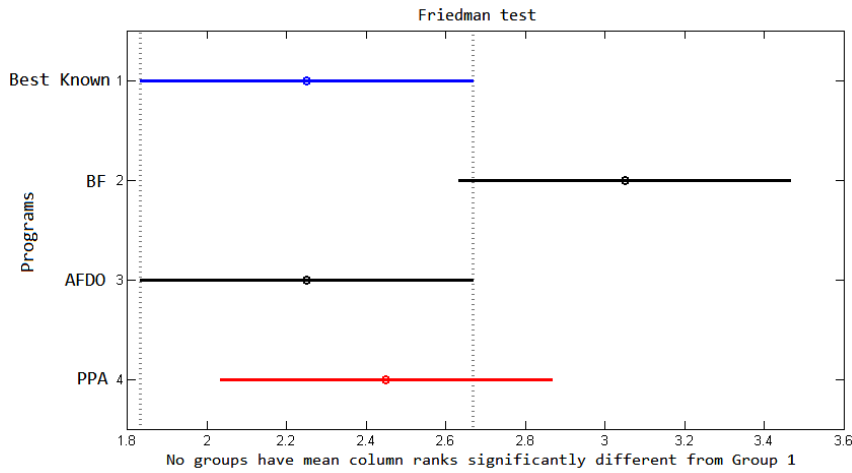


Figure 2: Friedman test for medium datasets

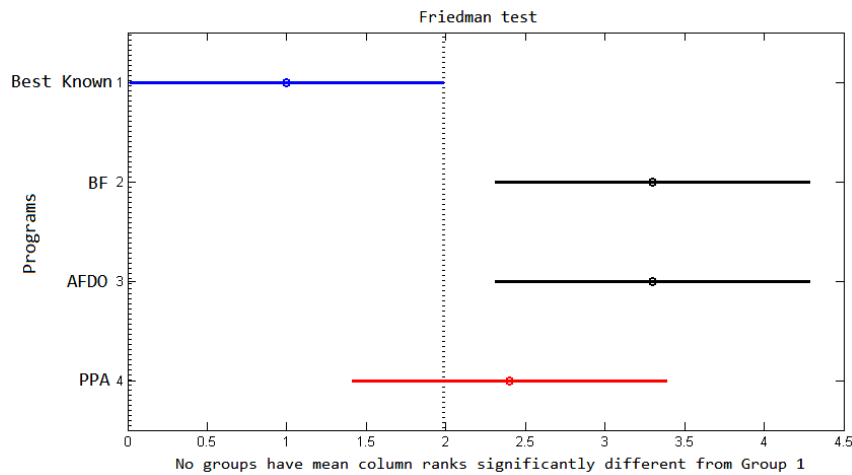


Figure 3: Friedman test for hard datasets

6 Conclusion

An alternative method, namely PPA, initially designed for continuous optimization has been implemented for Bin Packing. For this implementation, the discrete nature of the problem considered has been taken into account, required a different and suitable representation of solutions, necessary for navigating in the search space. The performance of our implementation of PPA is evaluated on different benchmark instances. The implementation has been explained and experimental results are provided. The outcomes are good particularly when the algorithm is run long enough. The complexity of the problem arises from the demand to optimally pack items into a minimum number of bins. This

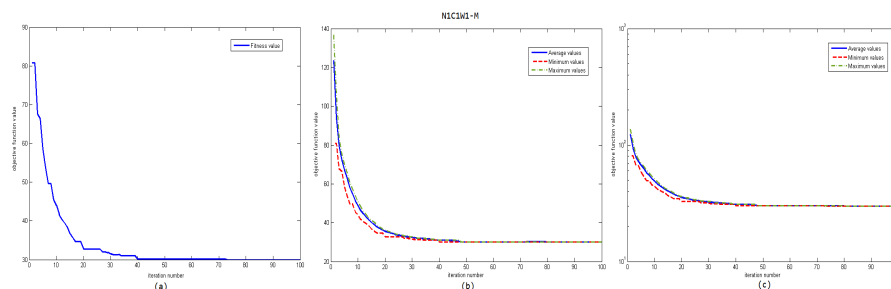


Figure 4: (a) displays a convergence of the algorithm; (b) minimum, maximum, and average fitness values; (c) the graph using a base 10 logarithm for iterations and a linear scale for the objective

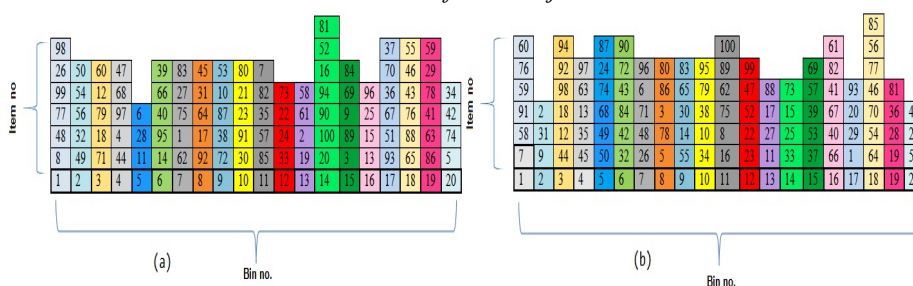


Figure 5: Packing solution, for instance, N2W2B3-R9 according to (a) PPA, (b) AFDO

algorithm has a run time proportional to the number of items and the number of bins, making it feasible for reasonably sized instances.

We are confident that our approach can tackle this problem in two and three dimensions. This is the subject of future work.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[Abo-Alsabeh and Salhi 2020] Abo-Alsabeh, R., Salhi, A.: “An evolutionary approach to constructing the minimum volume ellipsoid containing a set of points and the maximum volume ellipsoid embedded in a set of points”, in Journal of Physics: Conf. Series, 1530 (2020) 012087.

[Abo-Alsabeh and Salhi 2021] Abo-Alsabeh, R., Salhi, A.: “A metaheuristic approach to the c1s problem”, Iraqi Journal of Science, 62, 1 (2021) 218-227.

[Abo-Alsabeh and Salhi 2022] Abo-Alsabeh, R., Salhi, A.: “The genetic algorithm: A study survey”, Iraqi Journal of Science, 63, 3 (2022) 1215-1231.

[Abo Alsabeh 2017] Abo Alsabeh, R.: “An Evolutionary Approach to Solving the Maximum Size Consecutive Ones Submatrix and Related Problems”, Ph.D. thesis, University of Essex, 2017.

- [Abo-Alsabeh et al. 2020] Abo-Alsabeh, R., Daham, H.A., Salhi, A.: "An Evolutionary Approach for Solving the Minimum Volume Ellipsoid Estimator Problem"; *Int. Conf. on Innovations in Bio-Inspired Comput. and Appl.*, Springer, (2020) 23-31.
- [Abdullah and Ahmed 2019] Abdullah, J. M., Ahmed, T.: "Fitness dependent optimizer: inspired by the bee swarming reproductive process"; *IEEE Access*, 7 (2019) 43473-43486.
- [Abdul et al. 2020] Abdul-Minaam, D. S., Al-Mutairi, W. M. E. S., Awad, M. A., El-Ashmawi, W. H.: "An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem"; *IEEE Access*, IEEE, 8 (2020) 97959-97974.
- [Carmona et al. 2022] Carmona-Arroyo, G., Vázquez-Aguirre, J. B., Quiroz-Castellanos, M.: "Onedimensional bin packing problem: An experimental study of instances difficulty and algorithms performance"; in *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications*, Springer, (2021) 171-201.
- [Christensen et al. 2017] Christensen, H. I., Khan, A., Pokutta, S., Tetali, P.: "Approximation and online algorithms for multidimensional bin packing: A survey"; *Computer Science Review*, Elsevier, 24 (2017) 63-79.
- [Cheraitia et al. 2019] Cheraitia, M., Haddadi, S., Salhi, A.: "Hybridising plant propagation and local search for uncapacitated exam scheduling problems"; *Int. J. of Services and Operations Management*, Inderscience Publishers (IEL), 32, 4 (2019) 450-467.
- [Clerc and Kennedy 2002] Clerc, M., Kennedy, J.: "The particle swarm-explosion, stability, and convergence in a multidimensional complex space"; *IEEE transactions on Evolutionary Computation*, IEEE, 6, 1 (2002) 58-73.
- [Coffman et al. 1984] Coffman, E. G., Garey, M. R., Johnson, D. S.: "Approximation algorithms for bin-packing, an updated survey"; in *Algorithm design for computer system design*, Springer, (1984) 49-106.
- [Coffman et al. 1978] Coffman, E. G., Leung, J.-T., Ting, D.: "Bin packing: Maximizing the number of pieces packed"; *Acta Informatica*, 9, 3 (1978), 263-271
- [Coffman Jr et al. 2007] Coffman Jr, E. G., Leung, J. Y.-T., Csirik, J.: "Variants of classical one-dimensional bin packing"; 2007.
- [Crainic et al. 2009] Crainic, T. G., Perboli, G., Tadei, R.: "Ts2pack: A two-level tabu search for the three-dimensional bin packing problem"; *European Journal of Operational Research*, Elsevier, 195, 3 (2009) 744-760.
- [De Carvalho 2002] De Carvalho, J. V.: "Lp models for bin packing and cutting stock problems"; *European Journal of Operational Research*, 141, 2 (2002), 253-273.
- [Delorme et al. 2016] Delorme, M., Iori, M., Martello, S.: "Bin packing and cutting stock problems: Mathematical models and exact algorithms"; *European Journal of Operational Research*, Elsevier, 255, 1 (2016) 1-20.
- [Delorme et al. 2018] Delorme, M., Iori, M., Martello, S.: "Bpplib: a library for bin packing and cutting stock problems"; *Optimization Letters*, Springer, 12 (2018) 235-250.
- [Dorigo and Gambardella 1997] Dorigo, M., Gambardella, L. M.: "Ant colony system: a cooperative learning approach to the traveling salesman problem"; *IEEE Transactions on evolutionary computation*, American Scientific Publishers, IEEE, 1, 1 (1997) 53-66.
- [Garey and Johnson 1979] Garey, M. R., Johnson, D. S.: "Computers and intractability"; *freeman San Francisco.*, 174 (1979).
- [Garey and Johnson 1981] Garey, M. R., Johnson, D. S.: "Approximation algorithms for bin packing problems: A survey"; in *Analysis and design of algorithms in combinatorial optimization*, Springer, (1981) 147-172.

- [Garcia et al. 2012] Garcia-Gonzalo, E., Fernandez-Martinez, J. L., Juan Luis.: "A brief historical review of particle swarm optimization (ps0)"; *Journal of Bioinformatics and Intelligent Control*, American Scientific Publishers, 1, 1 (2012) 3-16.
- [Gass and Harris 1997] Gass, S. I., Harris, C. M.: "Encyclopedia of operations research and management science"; *Journal of the Operational Research Society*, Taylor & Francis, 48, 7 (1997) 759-760.
- [Geleijn et al. 2019] Geleijn, R., van der Meer, M., van der Post, Q. van den Berg, D.: "The plant propagation algorithm on timetables: First results"; *EVO*2019*, (2019) 2.
- [Gourgand et al. 2014] Gourgand, M., Grangeon, N., Klement, N.: "An analogy between bin packing problem and permutation problem: A new encoding scheme"; in *IFIP Int. Conf. on Advances in Production Management Sys.*, Springer, (2014) 572-579.
- [Johnson 1973] Johnson, D. S.: "Near-optimal bin packing algorithms"; PhD thesis, Massachusetts Institute of Technology, 1973.
- [Karaboga and Basturk 2007] Karaboga, D., Basturk, B.: "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm"; *Journal of global optimization*, Springer, 39, 3 (2007) 459-471.
- [Krause et al. 1975] Krause, K. L., Shen, V. Y., Schwetman, H. D.: "Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems"; *Journal of the ACM (JACM)*, ACM New York, NY, USA, 22, 4 (1975), 522-550.
- [Layeb 2011] A.: "A novel quantum inspired cuckoo search for knapsack problems.: *International Journal of bio-inspired Computation*, Inderscience Publishers, 3, 5 (2011) 297-305.
- [Layeb and Benayad 2014] Layeb, A., Benayad, Z.: "A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems"; *Int. J. Comput. Sci. Appl.*, 11, 2 (2014) 19-37.
- [Luo et al. 2017] Luo, F., Scherson, I. D., Fuentes, J.: "A novel genetic algorithm for bin packing problem in jmetal"; in *2017 IEEE Int. Conf. on Cognitive Comput. (ICCC)*, IEEE, (2017) 17-23.
- [Martello et al. 2000] Martello, S., Pisinger, D., Vigo, D.: "The three-dimensional bin packing problem"; *Operations research*, INFORMS, 48, 2 (2000) 256-267.
- [Munien and Ezugwu 2021] Munien, C., Ezugwu, A. E.: "Metaheuristic algorithms for one dimensional binpacking problems: A survey of recent advances and applications"; *Journal of Intelligent Systems*, 30, 1 (2021), 636-663.
- [Munien et al. 2020] Munien, C., Mahabeer, S., Dzitiro, E., Singh, S., Zungu, S., Ezugwu, A. E.-S.: "Metaheuristic approaches for one-dimensional bin packing problem: A comparative performance study"; *IEEE Access*, 8 (2020) 227438-227465.
- [Ojeyinka 2015] Ojeyinka, T. O.: "Bin packing algorithms with applications to passenger bus loading and multiprocessor scheduling problems"; *Communications on Applied Electronics*, 2, 8 (2015) 38-44.
- [Paauw and Van den Berg 2019] Paauw, M., Van den Berg, D.: "Paintings, polygons and plant propagation"; in *Int. Conf. on Comp. Int. in Music, Sound, Art and Design (Part of EvoStar)*, Springer, (2019) 84-97.
- [Pratama et al. 2023] Pratama, D. A., Abo-Alsabeh, R. R., Bakar, M. A., Salhi, A., Ibrahim, N. F.: "Solving partial differential equations with hybridized physic-informed neural network and optimization approach"; *Incorporating genetic algorithms and L-BFGS for improved accuracy*. *Alexandria Engineering Journal*, Elsevier, 77, (2023) 205-226.
- [Salhi and Fraga 2011] Salhi, A., Fraga, E. S.: "Nature-inspired optimisation approaches and the new plant propagation algorithm"; 2011.

[Selamoglu and Salhi 2016] Selamoglu, B. I., Salhi, A.: “The plant propagation algorithm for discrete optimisation: The case of the travelling salesman problem”; in *Nature-inspired computation in engineering*, Springer, (2016) 43-61.

[Sulaiman and Salhi 2015] Sulaiman, M., Salhi, A.: “A seed-based plant propagation algorithm: the feeding station model”; *The Scientific World Journal*, Hindawi, 2015 (2015).

[Sulaiman and Salhi 2016] Sulaiman, M., Salhi, A.: “A hybridisation of runner-based and seed-based plant propagation algorithms”; in *Nature-inspired computation in engineering*, Springer, (2016) 195-215.

[Sgall 2014] Sgall, J.; “Online bin packing: Old algorithms and new results”; in *Conf. on Computability in Euro.*, Springer, (2014) 362-372.

[Sleegers and van den Berg 2020] Sleegers, J., van den Berg, D.: “Plant propagation & hard hamiltonian graphs”; *Evo*2020*, (2020) 10.

[Yang 2009] Yang, X.-S.: “Firefly algorithms for multimodal optimization”; in *International symposium on stochastic algorithms*, Springer, (2009) 169-178.

[Zendaoui and Layeb 2016] Zendaoui, Z., Layeb, A.: “Adaptive cuckoo search algorithm for the bin packing problem”; in *Modelling and Implementation of Complex Systems*, Springer, (2016) 107-120.