IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2020, accepted March 22, 2020, date of publication April 6, 2020, date of current version June 5, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2985752*

# An Adaptive Fitness-Dependent Optimizer for the One-Dimensional Bin Packing Problem

**DIAA SALAMA ABDUL-MINAAM[1,2], WADHA MOHAMMED EDKHEEL SAQAR AL-MUTAIRI[1], MOHAMED A. AWAD[1], AND WALAA H. EL-ASHMAWI[2,3]**

[1]Information Systems Department, Faculty of Computers and Artificial Intelligence, Benha University, Benha 12311, Egypt
[2]Computer Science Department, Faculty of Computer Science, Misr International University, Cairo 19648, Egypt
[3]Computer Science Department, Faculty of Computers and Informatics, Suez Canal University, Ismailia 41522, Egypt

Corresponding author: Diaa Salama Abdul-Minaam (diaa.salama@fci.bu.edu.eg)

This work was supported by the DSA Laboratory, Benha University, under Grant 28211231302952.

**ABSTRACT** In recent years, the one-dimensional bin packing problem (1D-BPP) has become one of the most famous combinatorial optimization problems. The 1D-BPP is a robust NP-hard problem that can be solved through optimization algorithms. This paper proposes an adaptive procedure using a recently optimized swarm algorithm and fitness-dependent optimizer (FDO), named the AFDO, to solve the BPP. The proposed algorithm is based on the generation of a feasible initial population through a modified well-known first fit (FF) heuristic approach. To obtain a final optimized solution, the most critical parameters of the algorithm are adapted for the problem. To the best of our knowledge, this is the first study to apply the FDO algorithm in a discrete optimization problem, especially for solving the BPP. The adaptive algorithm was tested on 30 instances obtained from benchmark datasets. The performance and evaluation results of this algorithm were compared with those of other popular algorithms, such as the particle swarm optimization (PSO) algorithm, crow search algorithm (CSA), and Jaya algorithm. The AFDO algorithm obtained the smallest fitness values and outperformed the PSO, CS, and Jaya algorithms by 16%, 17%, and 11%, respectively. Moreover, the AFDO shows superiority in terms of execution time with improvements over the execution times of the PSO, CS, and Jaya algorithms by up to 46%, 54%, and 43%, respectively. The experimental results illustrate the effectiveness of the proposed adaptive algorithm for solving the 1D-BPP.

**INDEX TERMS** Bin packing, first fit heuristic, fitness-dependent optimizer, swarm intelligent algorithms.

## I. INTRODUCTION

The bin packing problem (BPP) is a commonly studied combinatorial optimization problem; it can be defined as a finite collection of items with varying specifications to be packed into several bins or containers [1] without exceeding the capacity of each bin. The BPP can be viewed as a particular case of the one-dimensional (1D) cutting-stock problem [2], [3]. The BPP focuses on minimizing the number of used bins or the amount of wasted space inside a bin. Therefore, solving the BPP can help solve many real-world problems, such as cutting-stock packaging design in supply chain management [4] and industrial applications [5], [6]. The 1D-BPP allows items to be packed according to a fixed dimension. Although this problem seems simple to define, it is a well-known NP-hard problem [7], and achieving an optimal solution is time consuming, especially with the increasing magnitude of the problem. Therefore, different approximation solutions have been proposed for solving the 1D-BPP based on heuristic approaches, such as next fit (NF) and first fit (FF) [8], or through hybrid approximation algorithms, such as FF decreasing (FFD) and best fit decreasing (BFD) methods [9]. In [9], items were not separated onto shelves. Although the algorithms, as mentioned earlier, provide a solution, they are bound to the proposed approximation ratio. Coffman Jr. *et al.* [10] provided an overview of the approximation algorithms used for solving the 1D-BPP.

Another new and popular solution of this problem involves the use of metaheuristic approaches naturally inspired by biological, physical, or sociological phenomena; these include the genetic algorithm (GA) [11], particle swarm optimization (PSO) [12], and the tabu search (TS) method [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

Cui [14] used the NF heuristic with the GA to solve combinatorial optimization problems. One of the main advantages of metaheuristic approaches for solving various optimization problems is obtaining high-quality solutions through fewer iterations (i.e., shorter computational time).

Additionally, metaheuristic approaches are flexible and straightforward [15], [16]. However, there is no universal algorithm that addresses all common optimization problems well. Therefore, solving NP-hard problems is still an open challenge; thus, in the current study, the 1D-BPP was approached practically and effectively by proposing an adaptive version of the fitness-dependent optimizer (FDO) algorithm called the AFDO. The FDO algorithm is a newly proposed swarm intelligence algorithm that is stable in both exploration and exploitation phases compared with other algorithms based on benchmark test functions and statistical analysis [17].

Therefore, this study adapted the FDO algorithm to solve the 1D-BPP. The adaptation occurred in two main phases: a random initial population is used, and an update to the algorithm procedure was added. In this study, the FF heuristic was modified to obtain a better initial solution (i.e., exploration phase), and an adaptive procedure was then embedded through operators in the updating stage during solution improvement to obtain the final optimal solution (i.e., exploitation phase). A set of experiments was used to test the feasibility of the proposed algorithm, and the results show superiority over those of previously used algorithms. To the best of our knowledge, this is the first study to propose the use of the FDO algorithm to solve the 1D-BPP in a discrete domain.

The remainder of this paper is organized as follows. Section 2 provides an overview of the most recent research focused on solving the BPP. The mathematical model and objective function of the BPP are discussed in section 3. Furthermore, the general structure of the FDO algorithm is illustrated in section 4. In section 5, we propose the modelling of the 1D-BPP by using the AFDO algorithm. Section 6 presents the testing of the proposed algorithm on benchmark datasets and a performance analysis. Finally, section 7 draws the conclusions.

## II. LITERATURE REVIEW

As the BPP is considered an NP-hard problem, its solution is generally contingent upon the availability of efficient methods. Various exact and heuristic methods have been proposed to solve the BPP in past years. For example, Martello and Toth [18] presented a branch-and-bound procedure, where the procedure of Scholl *et al.* [19] was combined with TS to develop a new branching schema. A new lower bound based on the cutting-stock problem was designed in [20] for BPP. Although exact methods find the best solution, they are unable to solve the problem when constraints are added. In addition, these methods are affected by the size of the problem. Delorme *et al.* [21] reviewed the most important exact methods for solving the 1D-BPP.

Some heuristic methods have also been proposed for solving the 1D-BPP; they provide approximate solutions ranging from the FF to the best fit (BF) or worst fit and other variations [22], [23]. Each of these heuristic methods assigns several varying objects to the first, best, or worst bins according to the bin capacity. Although each method displayed good results in each case, the results are not necessarily optimal. Heuristic methods cannot be treated as a general approach because they depend on the nature of the problem. Therefore, the best solution is achieved when the time of execution is considered an essential factor, and the worst solution is achieved when the quality is considered an essential factor.

Recently, researchers have discussed improving the current algorithms to produce an optimal solution based on increasing the number of parameters in and constraints of the BPP. Korf [24] proposed an improved algorithm for packing cartons into a minimum number of identical containers based on a local guided search process.

Based on the advantages of metaheuristic algorithms over exact methods, one of the most popular metaheuristic algorithms, GA, has been extensively applied to solve the BPP [25], [26]. In [27], a new island parallel grouping GA (IPGG) was developed to solve the 1D-BPP based on a modified class of the GA designed for solving complex grouping problems. Ross *et al.* [28], [29] solved the BPP through combinations of GAs and hyperheuristics. Another solution for solving the 1D-BPP was based on the greedy randomized adaptive search (GRASP) procedure [30], and it was built in two main phases: random initiation based on the integration of the FF and BF as a first phase and the application of a TS as the second phase to enhance the initial solution. In contrast, in the current study, two main stages are considered; in the first phase, a randomized FF heuristic is applied to generate initial feasible solutions, and an AFDO is applied in the second phase to achieve the optimal solution. Stakic *et al.* [31] presented different greedy randomized adaptive search procedure (GRASP) algorithms for solving a vector BPP and its variants.

Hemmelmayr *et al.* [32] introduced a solution for the BPP through a variable neighbourhood search (VNS), and Fleszar and Hindi [33] combined a minimal bin slack heuristic [34] with a VNS to combine the advantages of both methods for solving the 1D-BPP. Loh *et al.* [35] proposed a simple heuristic based on weight annealing (WA) to solve the BPP and improve the solution quality. An FFD heuristic was applied as a first step in [36] to obtain the initial solution, which was improved through a simulated annealing algorithm. The FFD heuristic requires a decreasing order of items before processing, whereas the proposed method does not require item sorting. With more constraints on packing items, a VNS can be used to solve the BPP with conflicts [37]. The Firefly algorithm was presented to solve the 1D-BPP with fixed-sized bins [38].

A hybrid improvement procedure utilizing TS was proposed in [39] to improve the results when the current solution was not feasible for the BPP. Similarly, in [40], a multistep

TS was proposed to solve the BPP, where a partial solution was first derived based on the dynamic programming of the item set. An adaptive cuckoo search algorithm was proposed in [41], which combines the standard cuckoo search with the decoding mechanism to solve the BPP. Moreover, a quantum cuckoo search was presented in [42], where the solution was represented as a binary representation, and the obtained results were compared with the FFD results. In this paper, the AFDO solution for the BPP is represented as a 1D discrete value, where the index identifies the item and the corresponding values represent the bins. The experimental results were compared with the results of other popular algorithms, such as PSO, the crow search algorithm (CSA), and the Jaya algorithm. Although these algorithms are successful in solving various well-known problems, the proposed algorithm has a crucial difference in improving the quality of the solution.

Variants of other algorithms [43], [44] have been proposed for solving the 1D-BPP, and their results were evaluated through various benchmark datasets. In this paper, the proposed adaptive algorithm was tested using three benchmark datasets with various item weights and bin capacities. The results of the AFDO algorithm for solving the 1D-BPP were better than those of other algorithms in terms of achieving the optimal solution (i.e., the minimum number of bins) within a reasonable execution time.

## III. MATHEMATICAL MODEL OF THE 1D-BPP

Solving the BPP can be considered practical for many real-life problems, such as collecting musical pieces to be stored on audio compact discs at maximal capacity, filling up containers, job scheduling, and passenger bus allocation [8]. All of these problems can be modelled as BPPs, with the primary constraint being that the set of items assigned to a bin should not exceed a specific capacity. Therefore, the aim is to use as few bins as possible. An extended view of the 1D-BPP mathematical model can be found in [21], [45], where items had a single dimension (size, weight, or any other measure). The general mathematical formula of the 1D-BPP can be described as follows.

Given a set of n items, $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_n$, capacity $\mathcal{C} \in \mathbb{N}$, and size function $\mathcal{S} : \mathcal{I} \rightarrow \mathbb{N}$, the objective is to find a feasible assignment $f$ that minimizes the number of used bins. The feasible assignment of items into $N$ bins can be considered a partitioning process for the set of items, $P = \{P_1, \ldots, P_N\}$. For each $P_k$, the sum of the item sizes or weights in $P_k$ must not exceed the capacity $C$, which can be modelled as follows. The objective function is minimized as

$$f(P) = \sum_{k=1}^{N} P_k$$

subject to the following constraints.

Constraint (1):

$$P_k \in \{0, 1\} \quad \forall k \in P$$

Constraint (2):

$$\sum_{i \in P_k} \mathcal{S}_i \leq C \quad \forall k \in \{1, \ldots, N\}$$

Constraint (3):

$$\sum_{k \in P} I_{i,k} = 1 \quad \forall i \in \{1, \ldots, n\}$$

Constraint (4):

$$I_{i,k} \in \{0, 1\} \quad \forall i \in \mathcal{I} \; and \; \forall k \in P$$

Constraint (1) is a decision variable related to the number of partitions used (i.e., bins), which is *one* if bin $k$ is used and 0 otherwise. Capacity constraint (2) guarantees that the total number of items in each bin does not exceed the bin capacity (assuming that all bin capacities are identical). Constraints (3) and (4) guarantee that each item is assigned to precisely one partition (i.e., bin). However, the 1D-BPP is an NP-hard problem that requires an efficient algorithm for its solution.

## IV. FDO ALGORITHM

In recent years, many researchers working on optimization problems have tried to develop new algorithms to apply to real-world optimization problems. One of the most recent swarm intelligent algorithms that is inspired by nature is the FDO developed in 2019 [17]. The FDO algorithm encompasses the specific characteristics of bee swarms in the reproductive process and their collective decision-making behaviours. However, the FDO algorithm differs from the honeybee algorithm or artificial bee colony algorithm. Furthermore, FDO somewhat mimics the particle position updating process of the PSO algorithm.

The PSO algorithm requires more computations in terms of updating both the velocity and the particle positions, and the FDO requires fewer computations for updating the positions. Moreover, the FDO algorithm was tested on a group of 19 classic benchmark test functions, and the results showed that its efficiency was comparable to the PSO efficiency. Bees are social insects that live in hollow trees or small caves. They work in groups in colonies called hives [46]. There are three kinds of bees in a natural colony: queen bees, worker bees, and drones or scout bees. Each one has a distinct role and task according to its characteristics. The FDO focuses on the task of the scout bees, i.e., the exploration of the environment to find and locate a new place for the colony to build a hive (i.e., exploit preferable hives). Once a suitable location is found, the bees perform a "dance" to interact with the swarm [47]. For algorithmic representation, each hive exploited by a scout bee (i.e., the artificial search agent) represents a possible solution, and the best hive represents the global optimum solution according to the fitness weight. The main steps in the FDO algorithm are discussed as follows, and the overall process is shown in Fig. 1.

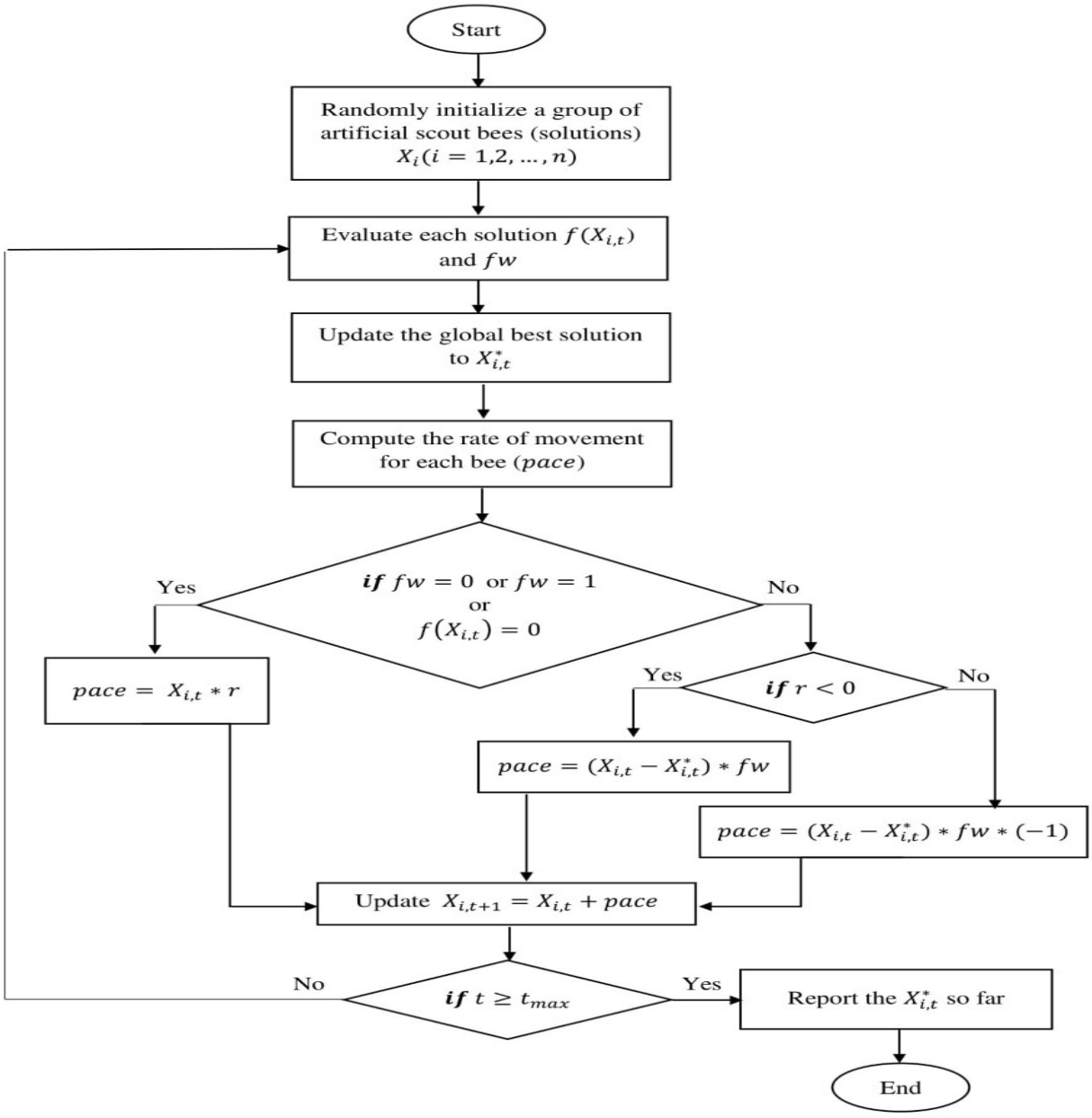


**FIGURE 1.** Flowchart of the FDO algorithm.

### A. STEP 1 (INITIAL POPULATION)

The initial population consists of a random set of artificial scout bees in the search space $X_i(i = 1, 2, \dots, n)$. Each scout bee position represents a solution. These bees try to find a better hive by searching randomly for more positions and evaluating the locations to allocate resources for the best one.

### B. STEP 2 (SCOUT BEE FITNESS WEIGHT)

Each scout bee position is evaluated according to the fitness weight, as represented by the following equation:

$$fw = \left| \frac{f(X^*_{i,t})}{f(X_{i,t})} \right| * wf \tag{1}$$

where $f(X^*_{i,t})$ is the fitness value of the global best solution, $f(X_{i,t})$ is the fitness value of the current solution at iteration $t$, and $wf$ is the weight factor. Considering $wf = \{0, 1\}$, in the case of zero (i.e., neglected), the algorithm provides a stable search, whereas $wf = 1$ represents a high level of convergence and a low chance of coverage. Considering that $fw \in [0, 1]$,

$$fw = 1 \; if \begin{cases} f\left(X^*_{i,t}\right) \; is \; the \; current \; soluion \\ f\left(X^*_{i,t}\right) = f\left(X_{i,t}\right) \\ f\left(X^*_{i,t}\right) \; and \; f\left(X_{i,t}\right) \; have \; the \; same \; fitness \; value \end{cases} \tag{2}$$

Furthermore, $fw = 0$ when $f\left(X^*_{i,t}\right)$ dramatically affects the movement of a bee to a new position, as described in step 3.

### C. STEP 3 (SCOUT BEE MOVEMENT)

The scout bees move from their current position at iteration t to a new position by adding "*pace*" to find a better position, as shown in equation (3).

$$X_{i,t+1} = X_{i,t} + pace \tag{3}$$

where $X_i$ represents the current position of the scout bee at iteration t and pace is the movement rate, which depends on the value of the fitness weight *fw*, as shown in equation (4), for different cases. The direction of *pace* is based on a random mechanism.

$$pace = \begin{cases} X_{i,t} * r & \textbf{\textit{if}}\ fw = 1 or\ 0\ or f\left(X_{i,t}\right) = 0 \\ (X_{i,t} - X^*_{i,t}) * fw * (-1) & \textbf{\textit{if}}\ 0 < fw < 1\ and\ r < 0, \\ (X_{i,t} - X^*_{i,t}) * fw & \textbf{\textit{if}}\ 0 < fw < 1\ and\ r \geq 0 \end{cases} \tag{4}$$

where $r$ is a random number in the range of $[-1, 1]$. Unlike any other similar algorithm, updating the FDO is dependent only on the fitness weight *fw* and random number $r$.

### D. STEP 4 (TERMINATION CONDITION)

The fitness value of each scout bee is computed for each iteration, and the locations of the new hives are updated. This process is repeated until a termination condition is satisfied (i.e., the maximum number of iterations $t_{max}$ is reached). At the end of the iterations, the global best solution is produced as the final solution.

## V. AFDO FOR THE 1D-BPP

In this section, we propose an AFDO algorithm to improve the BPP solution quality. This adaptive version is based on the generation of a feasible random initial population through a randomized FF heuristic. The FF heuristic is a well-known and simple heuristic for the 1D-BPP. It is a greedy algorithm that packs each item into the lowest-indexed bin and ensures that the capacity constraint is not violated. Furthermore, at least one solution representing the optimal solution is guaranteed to exist [48], based on which set of random movements is applied to update the bee location, generate a new position and obtain a global best solution. The steps of the proposed AFDO method are discussed in detail as follows.

### A. STEP 1 (SCOUT BEE REPRESENTATION)

Each artificial scout bee represents a candidate solution to the BPP. The presentation scheme for the solution is a critical part of solving any optimization problem. In this paper, each assignment $P_k$ can be represented by a vector of $n$ dimensions of integer numbers, and indices and values are used to identify the item and represent the bin, respectively, as shown in Fig. 2. This representation helps minimize the number of bins used and allows a group of items to be packed into the same bin without violating the bin capacity constraint.

| Item $I_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bin $k$ | 1 | 2 | 1 | 1 | 3 | 4 | 2 | 4 | 3 | 5 |

**FIGURE 2.** Scout bee candidate solution ($P_k$).

Fig. 2 represents the packing of 10 items into five bins, where items 1, 3, and 4 are packed into bin 1; items 2 and 7 are packed into bin 2; and so on, according to the capacity constraints.

### B. STEP 2 (INITIAL POPULATION GENERATION)

Each run of the AFDO algorithm starts with the generation of an initial population. The quality of the initial population considerably affects both the execution time and result quality. In this study, the FF heuristic [49] was modified to allow for the random generation of different initial solutions. The basic concept of the modified FF heuristic based on the packing of a randomly selected item from a shuffled list of items to the first suitable bin is that if the item does not fit any available bin, then a new bin is created to pack the item. To illustrate the modified FF heuristic, consider a set of 10 items (i.e., $\mathcal{I} = I_1, \ldots, I_{10}$); the corresponding weighted items are packed into bins with identical capacities of 10 ($C = 10$). Fig. 2 shows the standard representation of an FF heuristic that packs the items into five bins. Fig. 3 shows the list of items and three randomly generated candidate solutions from the modified FF.

| Item | weight |
|---|---|
| 1 | 4 |
| 2 | 8 |
| 3 | 5 |
| 4 | 1 |
| 5 | 7 |
| 6 | 6 |
| 7 | 1 |
| 8 | 4 |
| 9 | 2 |
| 10 | 2 |

| Item $I_i$ | 1 | 10 | 4 | 9 | 7 | 5 | 2 | 3 | 8 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bin $k$ | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 5 |

| Item $I_i$ | 4 | 7 | 2 | 10 | 8 | 1 | 6 | 9 | 5 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bin $k$ | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 5 |

| Item $I_i$ | 10 | 5 | 7 | 2 | 9 | 3 | 4 | 1 | 6 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bin $k$ | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 4 |

**FIGURE 3.** Illustration of a random population.

Each solution $P_k$ in the population is formed by packing a number of items into a number of bins. The characteristics of the modified FF are twofold. First, the modified FF allows for the generation of different solutions with different packing schemes; this approach is suitable and practical for cases where the target number of bins is not known. As shown in Fig. 3, one of the solutions allows the items to be packed into four bins. Second, the modified FF heuristic allows the generation of diversified feasible solutions that can be used as the starting stage for the proposed algorithm.

### C. STEP 3 (FITNESS VALUE CALCULATION)

At each iteration, every scout bee in the population is evaluated according to the fitness function $f$, and the bee with the minimum fitness value is assigned the best candidate solution $X^*_{i,t}$. The goal of the 1D-BPP is to minimize the number of used bins $N$ to pack all n items as follows:

$$N \geq \left\lceil (\sum_{i=1}^{n} \mathcal{S}_i)/\mathcal{C} \right\rceil \tag{5}$$

With the number of bins as the objective function, the algorithm experiences stagnation due to having many solutions with different representations and the same number of bins. However, it is better to integrate this criterion with another that is focused on bin fullness as follows:

$$Minmize\, f\,(P) = 1 - \frac{\sum_{k=1}^{N} (fill_k/C)^k}{N} \tag{6}$$

$$fill_k = \sum_{i \in P_k} \mathcal{S}_i, \tag{7}$$

where $fill_k$ is the full capacity of bin $k$, $C$ is the bin capacity, $N$ is the total number of bins, and $k$ is a constant that defines the equilibrium of the filled bin (usually 2).

In addition to minimizing the number of used bins, the fitness function aims to minimize the number of empty spaces for each bin remaining after packing the items. Therefore, the candidate with the lowest fitness value (i.e., lowest unused space) is considered the best solution. The fitness value represents how close the solution is to the objective.

### D. STEP 4 (SCOUT BEE FITNESS WEIGHT)

The fitness weight $fw$ of each scout bee in the current population is computed using equation (1). For a more stable search, we set $wf = 0$, which implies that the factor can be neglected. The computation of $fw$ dramatically affects the movement of scout bees towards a new position.

### E. STEP 5 (SCOUT BEE MOVEMENT)

Updating the position of an artificial scout bee depends on the movement rate, *pace*, and direction. In the proposed AFDO algorithm, to solve the 1D-BPP, only a positive direction is considered, where $r$ is a random number generated in the uniform distribution [0, 1] because of the nature of the problem. Equation (4) can be rewritten in the format of the AFDO algorithm as follows:

$$pace = \begin{cases} X_{i,t} \otimes r & \textbf{if } fw = 1 or\ 0\ or f\ (X_{i,t}) = 0 \\ (X_{i,t} \ominus X^*_{i,t}) \otimes fw & \textbf{if } 0 < fw < 1 \end{cases} \tag{8}$$

Equation (8) represents two cases that influence the direction of movement. In the first case, the result of $(X_{i,t} \otimes r)$ is a random sequence of $\mathcal{V}$s generated from the current solution depending on the number of items, $\mathcal{I}$, and the number of bins, N, in a given instance of the BPP dataset. Here, $\mathcal{V}$ is a raw vector (3 tuple) and is represented as V $=<a, b, c>$ with three indices: $a$ represents the item_id, $b$ represents the current bin_no., and $c$ represents the new bin_no. The variable $r$ indicates the probability of a given number of tasks being selected from $\mathcal{I}$, as illustrated in Fig. 4.

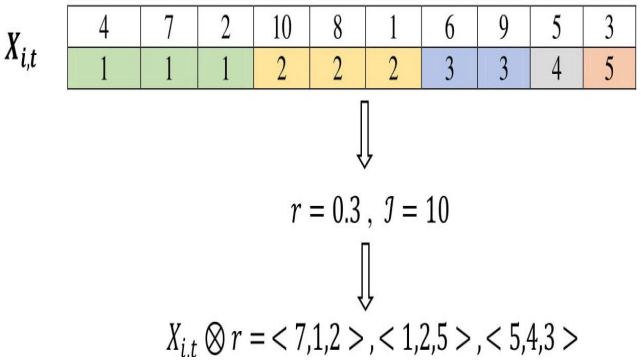


**FIGURE 4.** Illustration of $X_{i,t} \otimes r$.

In the example in Fig. 4, we assume that there are 10 items and $r = 0.3$; then, the number of raw vectors, $\mathcal{V} = 3$, is randomly chosen from the current candidate solution as $< 7, 1, 2 >$, which indicates that 7 items are currently in bin 1. In this case, the new bin is 2 (i.e., the new bin is chosen randomly from a list of $N$ bins). The modification of this parameter in the algorithm allows for the possibility of obtaining multiple solutions through randomization.

To compute the result of the operator $\ominus$, the difference between the two candidate solutions is computed in terms of the sequence of raw vectors $\mathcal{V}$; $\otimes$ is the probability of $fw$ selecting all raw vectors $\mathcal{V}$ from $(X_{i,t} \ominus X^*_{i,t})$. Fig. 5 illustrates $(X_{i,t} \ominus X^*_{i,t}) \otimes fw$, where $(X_{i,t} \ominus X^*_{i,t})$ returns the value representing how far the current candidate is from the global best candidate in the form of a sequence of raw vectors $\mathcal{V}$. For example, if item $I_1$ is assigned to the same bin in both solutions, then the item remains unchanged; otherwise, it is assigned to the new bin. According to Fig. 5, if item 4 is assigned to bin 1 in $X_{i,t}$ and bin 3 in $X^*_{i,t}$, then $\mathcal{V} =< 4, 1, 3 >$.

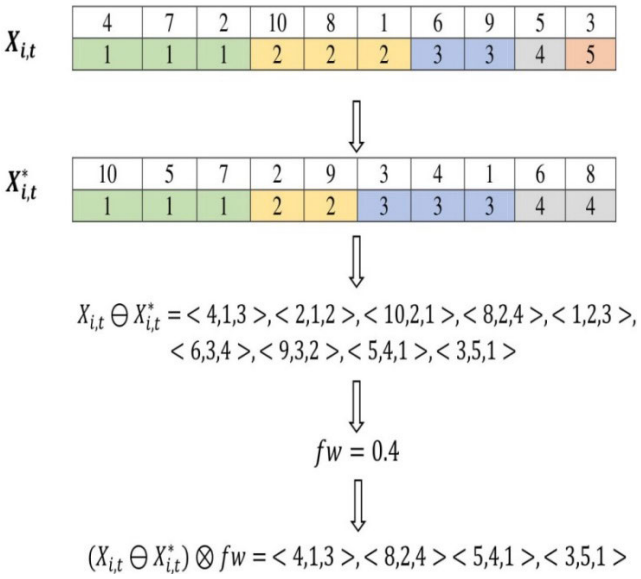


**FIGURE 5.** Illustration of $(X_{i,t} \ominus X^*_{i,t}) \otimes fw$.

From the two cases mentioned earlier, pace can be viewed as a sequence of raw vectors $\mathcal{V}$ sequentially applied to the current candidate solution in the population to obtain a new solution according to equation (9) (Fig. 6).

$$X_{i,t+1} = X_{i,t} \oplus pace. \tag{9}$$

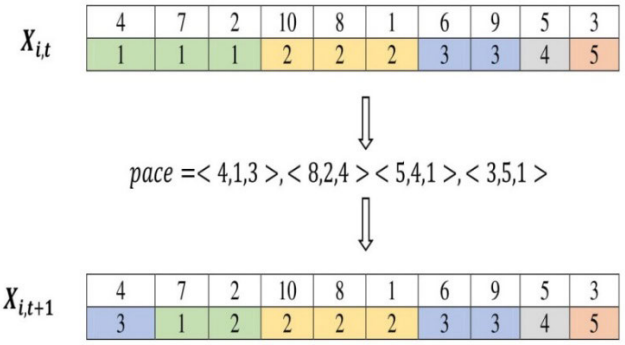


**FIGURE 6.** Illustration of $X_{i,t+1} = X_{i,t} \oplus pace$.

The example in Fig. 6 shows that item 4 is currently assigned to bin 1 and can be assigned to the new bin 3, which implies that $\mathcal{V}$ is equivalent to $< 4, 1, 3 >$. This reassignment can occur if and only if the solution does not violate the capacity constraint (i.e., the weight of item 4 is less than or equal to the remaining capacity of bin 3). $\mathcal{V} =< 8, 2, 4 >$ is not applied according to the capacity constraints. Generally, the successful application of $\mathcal{V}$ implies the removal of item $I_i$ from bin O and its packaging into bin P if the remaining capacity of bin P is greater than or equal to $S_i$. Then, the free capacities of both bins O and P are updated.

Thus, two critical parameters $r$ and $fw$ are modified through the adaptive algorithm to update the positions of artificial scout bees and improve the range of exploration of the search space to solve the 1D-BPP.

### F. STEP 6 (TERMINATION CONDITION)

Steps 3–5 are repeated until a predefined number of iterations, $t_{max}$, is reached or the number of distinct bins in the current best solution reaches the optimal solution according to equation (5). Once the termination condition is satisfied, the best global candidate solution represents the optimal assignment of items for the BPP.

## VI. EXPERIMENTAL EVALUATION

The proposed algorithm is implemented in Java, and the integrated development environment is Eclipse Java Neon V-1.8, which encompasses the benefits of object-oriented programming and runs on Intel(R) Core i7 2.80 GHz CPU with 8 GB RAM and the Windows 10 operating system. The proposed adaptive algorithm (i.e., the AFDO) was evaluated against other popular algorithms, such as PSO [50], the CSA [51], and the Jaya algorithm [52]. All of the algorithms were tested on a set of publicly available benchmark datasets [53]. The benchmark datasets are divided into three datasets, each comprising various items and bin capacities. The algorithms were applied to some instances of the three datasets, and the qualities of the solutions were evaluated by achieving the largest (Max.), smallest (Min.), and average (Avg.) fitness values out of ten runs for each instance. Furthermore, the performance of the proposed algorithm was compared to the performance of other algorithms according to several evaluation metrics, such as the minimum number of used bins ($b^*$), average fitness value ($AvgF$), average execution time ($AvgT$), and performance percentage ($PP$), as illustrated below.

- The number of used bins ($b^*$) for each algorithm is compared with the optimal number for each instance, $m^*$.
- The average fitness value ($AvgF$) is achieved by computing a fitness function [i.e., equation (6)] over ten runs:

$$AvgF = \sum_{i=1}^{10} f_i/10 \tag{10}$$

where $f_i$ is the fitness value of the $i^{th}$ run.

- The average execution time ($AvgT$) is computed in milliseconds according to equation (11). The smaller the execution time is, the more efficient the algorithm.

$$AvgT = \sum_{i=1}^{10} T_i/10, \tag{11}$$

where $T_i$ is the time need to reach a feasible solution in the $i^{th}$ run.

- The performance percentage ($PP$) is computed for the proposed algorithm and compared against the values of other algorithms in terms of fitness value minimization and the execution time as

$$PP\,(\%) = \frac{\left(f_{PSO,CSA,Jaya} - f_{AFDO}\right)}{f_{PSO,CSA,Jaya}} * 100 \tag{12}$$

where the variations in $f_{PSO,CSA,Jaya}$ are the fitness value results of the PSO, CS, and Jaya algorithms, respectively.

To achieve a fair comparison between the AFDO and other algorithms, the population size was initiated with *ten* individuals for all algorithms, where the maximum number of iterations ranged from 50 to 100 for all the compared algorithms. The characteristic parameters and different parameters associated with each algorithm are listed in Table 1.

The main parameters of the most popular swarm algorithm PSO are set to two (i.e., $C_1 = C_2 = 2$), and CSA has fewer computation parameters and is affected mainly by the flight length, which is set to one (i.e., $fl = 1$), and a randomly generated awarness probability. Although the Jaya algorithm is considered a parameterless algorithm that is affected only by the two random numbers as learning factors, it requires more computational steps. According to the parameters of the proposed adaptive algorithm AFDO, the weight factor is set to zero (i.e., $wf = 0$) to achieve search stability with high coverage of the search space. Moreover, the random walk parameter is generated within the range [0,1] to obtain the learning factors for Jaya and the awareness probability for the CSA.

**Algorithm 1** AFDO for solving the 1D-BPP

**Input:** 1D-BPP occurrence
**Output:** Feasible BPP solution

**Processing:**
**Step 1. Initialization**
Number of items, $\mathcal{I}$; bin capacity, $C$; the number of iterations, $t_{max}$; population size, *psize*
**Step 2. Generate initial population*pop*through the modified FF heuristic**
For $p = 1$ to *psize*
Shuffle the list of items
While (the list is not empty)
Select item $I_i$ from the list
Assign an item to the first available bin $k$
If the item does not fit in any available bin, open a new bin $k + 1$.
Update the residual capacity of the bin $re_k$ (i.e., $re_k = C - \mathcal{S}_i$).
End for
**Step 2. Improve the initial solution**
While ($t \leq t_{max}$)
/// artificial scout bee evaluation
For each candidate $X_{i,t}$ in the population *pop*
Evaluate it according to equation (6)
Set the candidate with the minimum fitness value to the global best value $X^*_{i,t}$
End for
Set $b^*$ to the current best number of bins
/// artificial scout bee movement
For each candidate $X_{i,t}$ in the population *pop*
Compute *fw* according to equation (1).
If ($fw == 1$ *or* $fw == 0$ *or* $f\left(X_{i,t}\right) = 0$), then
$r = rand()$;
$pace = X_{i,t} \otimes r$
$X_{i,t+1} = X_{i,t} \oplus pace$
Else
$pace = (X_{i,t} \ominus X^*_{i,t}) \otimes fw$
$X_{i,t+1} = X_{i,t} \oplus pace$
End if
End for
Set $t = t + 1$;
If ($b^* \leq N$) then stop // according to equation (5)
End while
**Step 3. Output the results**
The final assignment of items to bins and the minimum number of used bins

### A. COMPUTATIONAL RESULTS FOR DATASET 1

The proposed algorithm was tested on 15 instances from Dataset 1 with various numbers of items $n = \{50, 100, 200, 500\}$; bin capacities $C = \{100, 120, 150\}$; weights (sizes) of items $w_j$ from [1, 100], [20, 100], and [30, 100] for $j = 1..., n$; and the minimal number of bins $m^*$ for each instance. These instances are named "NxCyWz_v", and the corresponding parameters are listed in Table 2.

Table 3 shows the optimal number of bins ($m^*$) in each instance and the results of all the algorithms in terms of the number of used bins, $b^*$. Moreover, the AFDO algorithm is shown to have reached the optimal number of bins in all test instances (i.e., $b^* = m^*$). In the table, the numbers in bold represent the best results obtained throughout this study.

Moreover, the AFDO algorithm displays high efficiency in solving the 1D-BPP in terms of the average fitness value, as illustrated in Table 4.

As shown in Table 4, the adaptive FDO algorithm yielded the smallest average fitness value for all instances. For example, as shown in instance no. 3 (N1C1W1_I), the *AvgF* of the proposed algorithm is 0.1662, and those of PSO, the CSA, and the Jaya algorithm are 0.1846, 0.1803, and 0.1872, respectively. For other instances, the AFDO obtained a lower minimum fitness value (Min.) than did other algorithms; for example, for instance no. 11 (N2C1W1_C), the AFDO algorithm achieved a minimum fitness value of 0.158, and the PSO, CS and Jaya algorithm achieved values of 0.194, 0.186 and 0.171, respectively.

**TABLE 1.** List of parameters.

| Algorithm | Parameter | Meaning | Value |
|---|---|---|---|
| | $pop$ | Population size | 10 |
| | $t_{max}$ | Maximum number of iterations | 50–100 |
| PSO | $C_1$ and $C_2$ | Local and global learning factors | 2 |
| CSA | $AP$ | Awareness probability | A random number between 0 and 1 |
| | $fl$ | Flight length | 1 |
| AFDO | $wf$ | Weight factor | 0 |
| | $r$ | Random walk | A random number between 0 and 1 |

**TABLE 2.** Dataset 1 instances denoted as "NxCyWz_v".

| Parameter | Notation | | | |
|---|---|---|---|---|
| x | x=1 for $n$ =50 | x=2 for $n$ =100 | x=3 for $n$ =200 | x=4 for $n$ =500 |
| y | y=1 for $C$ =100 | y=2 for $C$ =120 | y=3 for $C$ =150 | |
| z | z=1 for $w_j$ from [1,100] | z=2 for $w_j$ from [20,100] | z=4 for $w_j$ from [30,100] | |
| v | v=A…T for the 20 instances of each class | | | |

**TABLE 3.** Instances and the minimum number of bins (dataset 1).

| | Dataset 1 | | | | Algorithms | | | |
|---|---|---|---|---|---|---|---|---|
| | Instance characteristics | | | | PSO | CSA | Jaya | AFDO |
| No. | Instance_id | $N$ | $C$ | $m^*$ | $b^{*2}$ | $b^*$ | $b^*$ | $b^*$ |
| **1** | N1C1W1_B | 50 | 100 | 31 | 31 | 31 | 31 | 31 |
| **2** | N1C1W1_E | 50 | 100 | 26 | 27 | 27 | 27 | **26** |
| **3** | N1C1W1_I | 50 | 100 | 25 | 25 | 25 | 25 | 25 |
| **4** | N1C1W1_M | 50 | 100 | 30 | 30 | 30 | 30 | 30 |
| **5** | N1C1W1_Q | 50 | 100 | 28 | 28 | 28 | 28 | 28 |
| **6** | N1C1W2_D | 50 | 100 | 31 | 32 | 32 | 31 | 31 |
| **7** | N1C2W1_P | 50 | 120 | 21 | 22 | 22 | 22 | **21** |
| **8** | N1C2W2_R | 50 | 120 | 25 | 27 | 26 | 26 | **25** |
| **9** | N1C3W2_A | 50 | 150 | 19 | 20 | 20 | 20 | **19** |
| **10** | N2C1W1_B | 100 | 100 | 49 | 50 | 50 | 50 | **49** |
| **11** | N2C1W1_C | 100 | 100 | 46 | 47 | 46 | 47 | 46 |
| **12** | N2C1W4_F | 100 | 100 | 77 | 77 | 77 | 77 | 77 |
| **13** | N2C2W1_H | 100 | 120 | 46 | 46 | 46 | 46 | 46 |
| **14** | N3C1W4_N | 200 | 100 | 148 | 148 | 148 | 148 | 148 |
| **15** | N3C2W2_S | 200 | 120 | 107 | 108 | 108 | 108 | **107** |

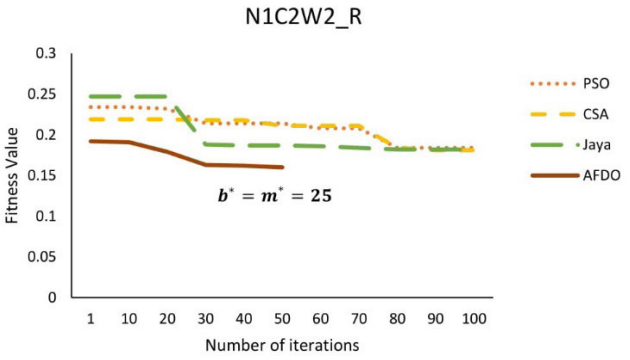


**FIGURE 7.** Fitness value and number of bins for the N1C2W2_R instance.

Additionally, the superior capability of the proposed algorithm is evident from the results obtained through comparisons with other algorithms; this superiority was achieved through the feasible improvement of the initial population and the effective exploration approach to obtain the optimal solution. According to Table 3, for many instances, such as instances 2, 7, 8, 9, 10, and 15, the proposed algorithm reached m* (i.e., b*= m*), but the other algorithms did not reach the optimal number of bins. For example, in instance 8 (N1C2W2_R), the AFDO reached the optimal number of bins ($m^* = 25$) for packing 50 items with a bin capacity of 120 for each bin in the smallest number of iterations, while CSA and Jaya yielded 26 and PSO reached 27 at the end of the iterative process, as shown in Fig. 7.

Moreover, the quality of the initial population of the AFDO solution was improved, and the maximum fitness value of the scout bees was 0.192, which is smaller than the values in the other algorithms. During the iterations, the solution improved due to the set of operators used to explore the search space and achieve a final optimal solution with a minimum fitness value equal to 0.16.

Although the compared algorithms reached the minimal number of bins in some instances, such as instances 3, 4, 5 and 13, as shown in Table 4, the AFDO showed superiority based on the performance percentage in terms of the average fitness value, as shown in Fig. 8.

For example, in instance 3 (N1C1W1_I), the proposed adaptive algorithm achieved a better average fitness value than PSO (by 10%), the CSA (by 8%), and the Jaya algorithm (by 11%). For other instances, the improvement achieved by the proposed algorithm ranged from 4% to 10% when

**TABLE 4.** The results for dataset 1 (average fitness value, *AvgF*).

| | PSO | | | CSA | | | Jaya | | | AFDO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. |
| 1 | 0.228 | 0.199 | 0.2081 | 0.227 | 0.199 | 0.2079 | 0.224 | 0.198 | 0.2053 | 0.224 | 0.198 | **0.205** |
| 2 | 0.203 | 0.194 | 0.1971 | 0.224 | 0.192 | 0.2003 | 0.228 | 0.194 | 0.208 | 0.177 | **0.175** | **0.1759** |
| 3 | 0.209 | 0.174 | 0.1846 | 0.199 | 0.165 | 0.1803 | 0.21 | 0.172 | 0.1872 | 0.178 | **0.142** | **0.1662** |
| 4 | 0.221 | 0.189 | 0.2004 | 0.219 | 0.192 | 0.1988 | 0.218 | 0.191 | 0.1977 | 0.195 | 0.189 | **0.192** |
| 5 | 0.232 | 0.199 | 0.2181 | 0.236 | 0.201 | 0.2258 | 0.232 | 0.199 | 0.2128 | 0.226 | **0.174** | **0.1992** |
| 6 | 0.246 | 0.228 | 0.2405 | 0.266 | 0.227 | 0.2421 | 0.23 | 0.208 | 0.2253 | 0.226 | **0.203** | **0.2095** |
| 7 | 0.213 | 0.175 | 0.2082 | 0.213 | 0.18 | 0.1941 | 0.216 | 0.178 | 0.2085 | 0.211 | **0.148** | **0.1829** |
| 8 | 0.234 | 0.184 | 0.2137 | 0.219 | 0.181 | 0.201 | 0.247 | 0.182 | 0.1993 | 0.192 | **0.16** | **0.1767** |
| 9 | 0.179 | 0.164 | 0.1714 | 0.216 | 0.164 | 0.1803 | 0.248 | 0.129 | 0.1723 | 0.184 | **0.104** | **0.1413** |
| 10 | 0.151 | 0.13 | 0.1383 | 0.151 | 0.114 | 0.1337 | 0.151 | 0.129 | 0.1354 | 0.138 | **0.102** | **0.1187** |
| 11 | 0.26 | 0.194 | 0.2247 | 0.26 | 0.186 | 0.2225 | 0.23 | 0.171 | 0.2112 | 0.214 | **0.158** | **0.1811** |
| 12 | 0.293 | 0.274 | 0.2826 | 0.294 | 0.275 | 0.2836 | 0.307 | 0.282 | 0.2889 | 0.291 | **0.273** | **0.2811** |
| 13 | 0.181 | 0.145 | 0.1701 | 0.174 | 0.145 | 0.1631 | 0.193 | 0.157 | 0.1707 | 0.163 | 0.145 | **0.1524** |
| 14 | 0.266 | 0.253 | 0.2608 | 0.268 | 0.252 | 0.2598 | 0.274 | 0.257 | 0.2615 | 0.264 | 0.252 | **0.2576** |
| 15 | 0.171 | 0.145 | 0.1572 | 0.169 | 0.159 | 0.162 | 0.167 | 0.146 | 0.1548 | 0.158 | **0.141** | **0.1491** |

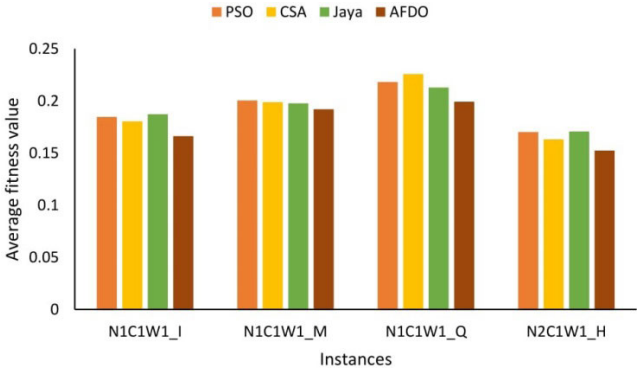


**FIGURE 8.** Average fitness value (*AvgF*) of some instances.

compared with PSO and from 3% to 12% and 11% when compared with the CSA and Jaya algorithm, respectively.

In terms of the average execution time, *AvgT*, the proposed adaptive FDO algorithm is superior to other compared algorithms for achieving a packing solution, as listed in Table 5.

As shown in the table, the AFDO algorithm has the smallest *AvgT* when compared with the values of the other algorithms in reaching an optimal solution. For example, in instance 3 (N1C1W1_I), the *AvgT* of the proposed algorithm was **94.6 ms,** while those of PSO, the CSA, and the Jaya algorithm were 143.2, 122.9, and 174.2 ms, respectively.

Fig. 9 shows the average execution times of the compared algorithms for solving some BPP instances and achieving the optimal number of bins.

The AFDO algorithm outperforms PSO in minimizing the average execution time for packing a set of items into the minimum number of bins by up to 49% for different instances. Furthermore, the proposed algorithm achieved improvements within ranges of 11%–44% and 13%–46% when compared to the CS and Jaya algorithms, respectively.

### B. COMPUTATIONAL RESULTS FOR DATASET 2

All instances from Dataset 2 are characterized by the same bin capacity $C = 1000$ and various numbers of

**TABLE 5.** Dataset 1 (average execution time, *AvgT*).

| No. | PSO | CSA | Jaya | AFDO |
|---|---|---|---|---|
| 1 | 162.3 | 116.3 | 120.1 | **104.4** |
| 2 | 171.5 | 167.6 | 514 | **119.1** |
| 3 | 143.2 | 122.9 | 174.2 | **94.6** |
| 4 | 118.3 | 116.3 | 184.5 | **103.2** |
| 5 | 115.5 | 157.4 | 107.9 | **88.2** |
| 6 | 159 | 104.7 | 126.2 | **82.8** |
| 7 | 100.4 | 95.5 | 107.3 | **79.7** |
| 8 | 97 | 90.3 | 104.6 | **76** |
| 9 | 150.5 | 176.1 | 181.2 | **125.5** |
| 10 | 244.4 | 210.4 | 233.1 | **160.9** |
| 11 | 124.5 | 211.7 | 152.5 | **113.6** |
| 12 | 375.2 | 375.5 | 342.5 | **257** |
| 13 | 477.4 | 299.8 | 276.8 | **241.7** |
| 14 | 646.9 | 726 | 606.8 | **509.6** |
| 15 | 620.7273 | 527.2 | 627.7 | **346.4** |

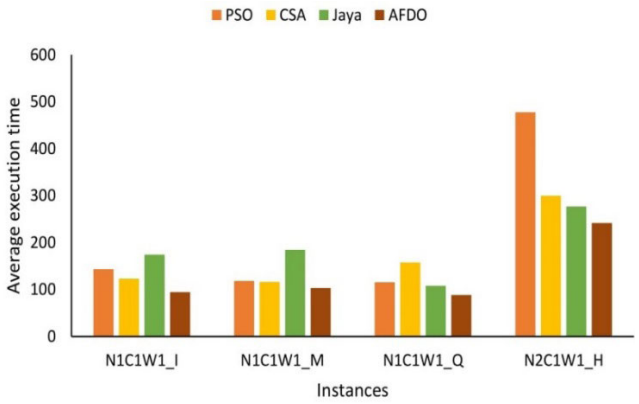


**FIGURE 9.** Average execution time (*AvgT*) of some instances.

items $n = \{50, 100, 200, 500\}$. In Dataset 2, the weight of an item is determined by two parameters: the average weight and delta. The average weight $avgWeight = \{C/3, C/5, C/7, C/9\}$, and *delta* represents the maximum deviation of each $w_j$ from a*vgWeight*, where $delta = \{20\%, 50\%, 90\%\}$. The proposed algorithm and other compared algorithms were tested on ten instances from this dataset, and the instances were named "NxWyBzRv", with parameters defined in Table 6.

The solutions of the BPP instances using different algorithms in terms of the number of used bins and the optimal number of bins for each instance are listed in Table 7.

As shown in the above table, the AFDO algorithm reached the optimal number of bins in all test instances (i.e., $b^* = m^*$). Furthermore, the AFDO gained superiority in terms of the average fitness value when compared with the other algorithms (PSO, CSA, and Jaya algorithm) over ten runs, as illustrated in Table 8.

For all test instances, the proposed adaptive algorithm achieves the smallest average fitness value. For example,

**TABLE 6.** Dataset 2 instances denoted as "NxWyBzRv".

| Parameter | Notation | | | |
|---|---|---|---|---|
| **x** | x=1 for $n$ =50 | x=2 for $n$ =100 | x=3 for $n$ =200 | x=4 for $n$ =500 |
| **y** | y=1 for $avgWeigh = C/3$ | y=2 for $vgWeight = C/5$ | y=3 for $vgWeight = C/7$ | y=4 for $vgWeight = C/9$ |
| **z** | z=1 for $delta =$ 20% | z=2 for $delta =$ 50% | z=3 for $delta = 90\%$ | |
| **V** | v=0…9 for the ten instances of each class | | | |

**TABLE 7.** Instances and the optimal number of bins (dataset 2).

| | Dataset 2 | | | | Algorithms | | | |
|---|---|---|---|---|---|---|---|---|
| | Instance characteristics | | | | PSO | CSA | Jaya | AFDO |
| No. | Instance_id | $N$ | $C$ | $m^*$ | $b^*$ | $b^*$ | $b^*$ | $b^*$ |
| 1 | N1W1B1R2 | 50 | 1000 | 19 | 20 | 20 | 20 | **19** |
| 2 | N1W1B1R5 | 50 | 1000 | 17 | 18 | 18 | 18 | **17** |
| 3 | N1W1B2R7 | 50 | 1000 | 18 | 19 | 19 | 18 | 18 |
| 4 | N1W2B2R9 | 50 | 1000 | 11 | 11 | 11 | 11 | 11 |
| 5 | N1W4B3R8 | 50 | 1000 | 6 | 6 | 6 | 6 | 6 |
| 6 | N2W2B1R2 | 100 | 1000 | 21 | 22 | 22 | 22 | **21** |
| 7 | N2W2B3R9 | 100 | 1000 | 20 | 20 | 20 | 20 | 20 |
| 8 | N3W1B3R5 | 200 | 1000 | 65 | 66 | 66 | 66 | **65** |
| 9 | N3W4B2R9 | 200 | 1000 | 22 | 22 | 22 | 22 | 22 |
| 10 | N4W4B2R7 | 500 | 1000 | 57 | 58 | 58 | 58 | **57** |

in instance 3 (N1W1B2R7), the *AvgF* of the AFDO is **0.1229,** the *AvgF* of the PSO algorithm is 0.169, the *AvgF* of the CSA is 0.1536, and the *AvgF* of the Jaya algorithm is 0.1536.

Although all the algorithms reached the optimal number of bins as a solution for some instances (e.g., instances 4, 5, 7 and 9) for Dataset 2, differences occurred. For example, when solving instance 4 (N1W2B2R9), all the algorithms reached the optimal number of bins (i.e., $m^* = 11$) but with different *AvgF* values. The *AvgF* values of the AFDO, PSO, CS, and Jaya algorithms are **0.118**, 0.1205, 0.1219, and 0.1236, respectively.

In some instances, such as instance 7 (N2W2B3R9), the proposed algorithm achieves the minimum fitness value (**Min.= 0.075**), which was better than the minimum fitness values of the other algorithms (**Min.= 0.076**) with different packing schema for packing 100 items into 20 bins, as shown in Figs. 10 and 11.

The AFDO algorithm has shown its efficiency not only in terms of the average fitness value but also in terms of the average execution time, as illustrated in Table 9.

The efficiency of the proposed algorithm in terms of the execution time is illustrated in Table 9, as it required the shortest time to pack a set of items into the minimum number of bins.

**TABLE 8.** The results of dataset 2 (average fitness value, *AvgF*).

| | PSO | | | CSA | | | Jaya | | | AFDO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. |
| 1 | 0.274 | 0.232 | 0.256 | 0.272 | 0.233 | 0.2642 | 0.273 | 0.268 | 0.2707 | 0.248 | **0.211** | **0.2297** |
| 2 | 0.195 | 0.138 | 0.1797 | 0.184 | 0.179 | 0.1819 | 0.186 | 0.18 | 0.1836 | 0.158 | **0.111** | **0.1484** |
| 3 | 0.199 | 0.151 | 0.169 | 0.158 | 0.15 | 0.1536 | 0.193 | 0.152 | 0.1667 | 0.131 | **0.108** | **0.1229** |
| 4 | 0.129 | 0.113 | 0.1205 | 0.128 | 0.117 | 0.1219 | 0.129 | 0.118 | 0.1236 | 0.122 | 0.113 | **0.118** |
| 5 | 0.061 | 0.06 | 0.0605 | 0.196 | 0.06 | 0.074 | 0.195 | 0.06 | 0.0738 | 0.061 | 0.06 | **0.0602** |
| 6 | 0.177 | 0.129 | 0.150364 | 0.18 | 0.125 | 0.1507 | 0.175 | 0.127 | 0.1473 | 0.149 | **0.11** | **0.1222** |
| 7 | 0.076 | 0.076 | 0.076 | 0.086 | 0.076 | 0.077 | 0.08 | 0.076 | 0.0764 | 0.076 | **0.075** | **0.0758** |
| 8 | 0.085 | 0.065 | 0.0759 | 0.085 | 0.066 | 0.0724 | 0.082 | 0.069 | 0.073 | 0.073 | **0.059** | **0.0646** |
| 9 | 0.065 | 0.062 | 0.0635 | 0.065 | 0.062 | 0.0633 | 0.064 | 0.06 | 0.0629 | 0.064 | **0.059** | **0.0624** |
| 10 | 0.056 | 0.051 | 0.0534 | 0.053 | 0.051 | 0.0519 | 0.056 | 0.052 | 0.053 | 0.05 | **0.048** | **0.0482** |

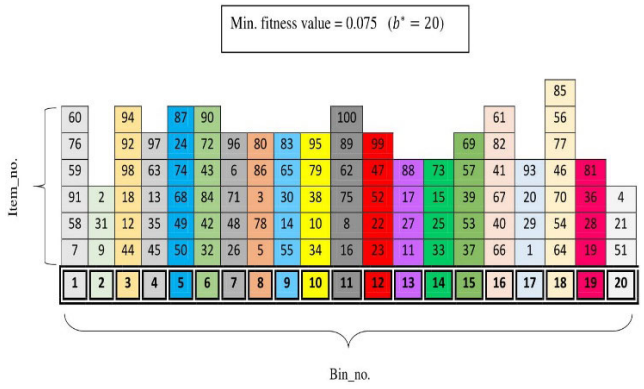


**FIGURE 10.** Packing solution for instance N2W2B3R9 according to the AFDO algorithm.

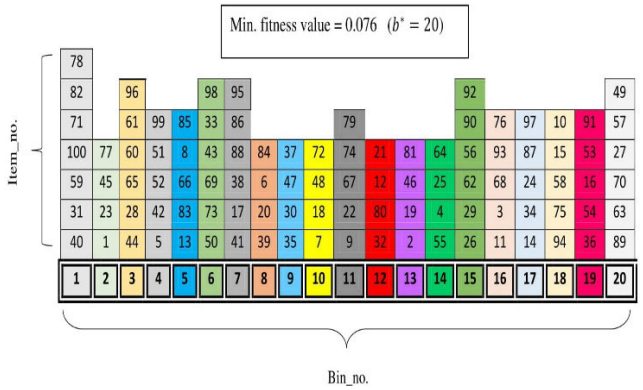


**FIGURE 11.** Packing solution for instance N2W2B3R9 according to the PSO, CS, and Jaya algorithms.

For example, in instance, 6 (N2W2B1R2), the *AvgT* value of the AFDO algorithm was **182.1 ms,** whereas those of the PSO, CS, and Jaya algorithms were 228.0909, 309.6, and 306.9 ms, respectively. The *AvgT* of the adaptive algorithm throughout different test instances improved by up to 56% when compared with that of PSO, 51% when compared with that of the CSA and 48% when compared with that of the Jaya algorithm.

The capability of the adaptive algorithm to solve the 1D-BPP was reflected not only by obtaining the optimal number of bins (e.g., instances 1, 2, 6, 9 and 10) but also in requiring fewer iterations, as illustrated in Fig. 12.

The figure shows the effectiveness of the AFDO solution for solving instance N4W4B2R7 in the fewest number of iterations. The solution improved during the iterative process (e.g.,, the number of bins is 58 for the first 20 iterations) to reach the optimal number of bins ($b^* = m^* = 57$) at iteration 22, and the other algorithms did not reach the optimal number of bins through the maximum number of iterations.

All of the above results of test instances from Dataset 2 unquestionably demonstrate the effectiveness and efficiency of the proposed algorithm for solving the 1D-BPP with improvements in the quality of the solution.

### C. COMPUTATIONAL RESULTS FOR DATASET 3

Five instances from Dataset 3 are tested with the number of items $n = 200$ and the largest bin capacity $C = 100,000$.

**TABLE 9.** Dataset 2 (average execution time, *AvgT*).

| No. | PSO | CSA | Jaya | AFDO |
|---|---|---|---|---|
| 1 | 98.8 | 123.8 | 119.9 | **89.2** |
| 2 | 111.3 | 101.8 | 116.7 | **81.7** |
| 3 | 104 | 89 | 99.9 | **71** |
| 4 | 179.5 | 123 | 124.3 | **117.2** |
| 5 | 109.8 | 156.3 | 129.6 | **76.7** |
| 6 | 228.0909 | 309.6 | 306.9 | **182.1** |
| 7 | 207.9 | 215.9 | 215.3 | **176.4** |
| 8 | 555.5 | 538 | 533.1 | **493.6** |
| 9 | 517.9 | 455.2 | 496.2 | **409.9** |
| 10 | 2463 | 1998.7 | 2067.6 | **1084.6** |

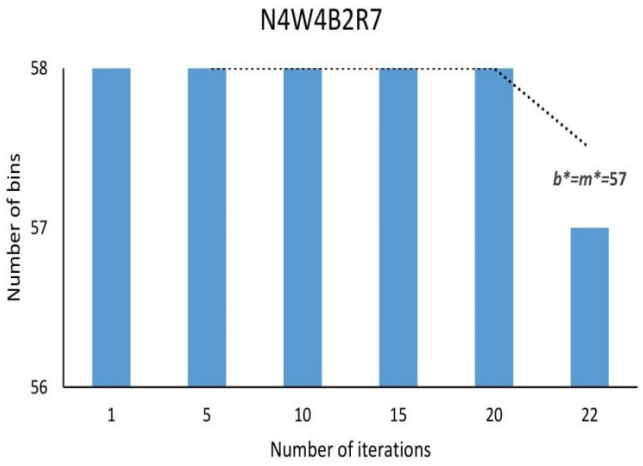


**FIGURE 12.** $b^*$ of N4W4B2R7 (AFDO algorithm).

The weight of each item $w_j$ is within the range [20, 000, 35, 000] for $j = 1..., n$ to guarantee the diversity of the packing problem. These instances are named ''HARDv,'' where v=0…9. Table 10 lists the results of the compared algorithms in terms of the number of used bins $b^*$ and the optimal number of bins for each instance $m^*$.

Although the solution of the AFDO algorithm for this dataset did not reach the optimal number of bins, the solution was better than those of the PSO, CS and Jaya algorithms. For example, in instance 5 (HARD7), the proposed adaptive algorithm packed items into 58 bins (i.e., $b^*= 58$), while other algorithms packed the same number of items into 59 bins (i.e., $b^*= 59$).

In addition, the AFDO algorithm achieved the smallest average fitness value for all test instances over ten runs, as shown in Table 11.

Regarding instance 3 (HARD4), the proposed algorithm obtained an average fitness value of $AvgF= \mathbf{0.1814}$, while the PSO, CS, and Jaya algorithms obtained values of 0.1939, 0.1934 and 0.1941, respectively. The improvement in the AFDO in terms of the average fitness value for different test instances ranged from 5% to 7% when compared with the PSO and CSA results and reached up to 10% when compared with the value for the Jaya algorithm. In addition, the proposed algorithm recorded the smallest minimum fitness value via the adaptive procedure with different operators when searching for an optimal solution. For example, as in instance 2 (HARD3), the **Min.** value of the AFDO was **0.171,** which improved by 4%, 6%, and 5% when compared with that of PSO (Min.= 0.179), the CSA and the Jaya algorithm, respectively.

**TABLE 10.** Instances and the optimal number of bins (dataset 3).

| | Dataset 3 | | | | Algorithms | | | |
|---|---|---|---|---|---|---|---|---|
| | Instance characteristics | | | | PSO | CSA | Jaya | AFDO |
| No | | $N$ | $C$ | $m^*$ | $b^*$ | $b^*$ | $b^*$ | $b^*$ |
| 1 | HARD0 | 200 | 100,000 | 56 | 59 | 60 | 59 | 59 |
| 2 | HARD3 | 200 | 100,000 | 55 | 60 | 60 | 60 | **59** |
| 3 | HARD4 | 200 | 100,000 | 57 | 61 | 61 | 61 | **60** |
| 4 | HARD5 | 200 | 100,000 | 56 | 60 | 60 | 60 | 60 |
| 5 | HARD7 | 200 | 100,000 | 55 | 59 | 59 | 59 | **58** |

Obtaining a high-quality solution within a reasonable execution time is an essential factor, especially when there are substantial scaling problems. The AFDO algorithm outperforms other comparison algorithms in solving the 1D-BPP, as shown in Table 12, in terms of the average execution time to achieve a feasible packing solution for a broad set of items.

As stated in Table 10, as an example, when solving instance 4 (HARD5) for packing 200 items, all of the comparison algorithms reached the same number of bins ($b^*= 60$) but at different times. The proposed adaptive algorithm (AFDO) requires an average execution time of **486 ms**, and the PSO, CS, and Jaya algorithms require execution times of 718.6, 589.5, and 571.8 ms, respectively.

Fig. 13 shows the performance percentage (*PP%*) of the proposed AFDO algorithm compared to that of the other algorithms in obtaining solutions for the five instances in terms of the average execution time.

The above figure shows the performance of the proposed algorithm compared with that of the other algorithms. The reduction in *AvgT* via the FDO algorithm ranged from 17% to 34% when compared with that of PSO, from 16% to 38% when compared with that of the CSA and reached up to 34% when compared with that of the Jaya algorithm for different instances.

As mentioned above, all of the tests show that the best-performing algorithm is the AFDO, as it has the smallest average fitness value and shortest average execution time and achieves the optimal number of bins in all instances for Datasets 1 and 2. For Dataset 3, the AFDO achieved the lowest number of bins with the shortest execution time compared with the other algorithms.

**TABLE 11.** The results of dataset 3 (average fitness value, *AvgF*).

| | PSO | | | CSA | | | Jaya | | | AFDO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. | Max. | Min. | Avg. |
| 1 | 0.2 | 0.173 | 0.186 | 0.197 | 0.182 | 0.1903 | 0.2 | 0.187 | 0.1958 | 0.185 | **0.171** | **0.1761** |
| 2 | 0.199 | 0.179 | 0.187 | 0.198 | 0.181 | 0.1871 | 0.192 | 0.18 | 0.1845 | 0.182 | **0.171** | **0.1732** |
| 3 | 0.202 | 0.186 | 0.1939 | 0.201 | 0.182 | 0.1934 | 0.195 | 0.192 | 0.1941 | 0.182 | **0.181** | **0.1814** |
| 4 | 0.2 | 0.188 | 0.1923 | 0.196 | 0.187 | 0.1902 | 0.196 | 0.186 | 0.1916 | 0.186 | **0.174** | **0.1816** |
| 5 | 0.191 | 0.165 | 0.1771 | 0.186 | 0.165 | 0.1762 | 0.187 | 0.166 | 0.1787 | 0.173 | **0.158** | **0.1655** |

**TABLE 12.** The results of dataset 3 (average execution time, *AvgT*).

| No. | PSO | CSA | Jaya | AFDO |
|---|---|---|---|---|
| 1 | 527.1 | 604.5 | 583.9 | **382.6** |
| 2 | 587.9 | 548.7 | 567 | **393.5** |
| 3 | 583.2 | 619.7 | 564.6 | **383.2** |
| 4 | 718.6 | 589.5 | 571.8 | **486** |
| 5 | 570.7 | 563.2 | 612.5 | **474.9** |

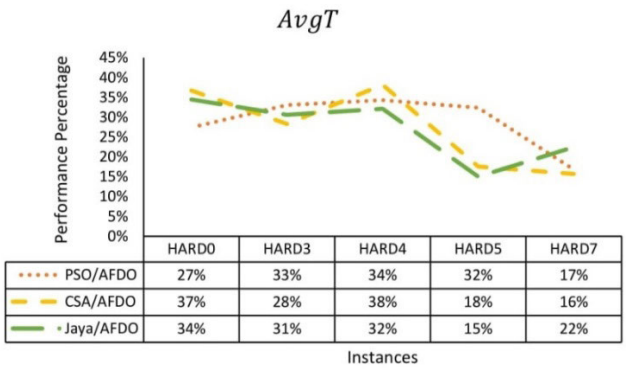


**FIGURE 13.** Performance percentage of AFDO in terms of the average execution time.

## VII. CONCLUSION

The BPP is one of the most famous combinatorial optimization problems, and it has a surprisingly large number of applications, especially in logistics and supply chain management. Solving this problem in a reasonable time requires an efficient optimization method. This study is based on the adaptive version of the most recent swarm algorithm called FDO. The proposed AFDO algorithm is based on a feasible random initial population through a modified FF heuristic and improves the BPP solution by adjusting the parameters to search for the final optimal solution. In this study, the proposed algorithm and other popular algorithms, including the PSO, CS, and Jaya algorithms, were tested on three standard benchmark datasets to demonstrate the efficiency and effectiveness of the AFDO in solving the BPP. These datasets were characterized by various numbers of items and different bin capacities to guarantee a large number of packing solutions. The proposed algorithm and other algorithms were tested on 15 instances in the first dataset, and the results demonstrated the effectiveness of the AFDO algorithm in terms of achieving the optimal solution (i.e., the optimal number of bins) within a reasonable time compared to the other algorithms. At the same time, the proposed adaptive algorithm achieved an average fitness value that was up to 19% better than the PSO average fitness value, 22% better than the CSA average fitness value, and 18% better than the Jaya algorithm average fitness value when obtaining solutions for the different instances in Dataset 1.

Moreover, updating the phase of the adaptive procedure via different operator strategies led to the optimal solution with fewer iterations. With a larger bin capacity than in the previous case, all compared algorithms were tested on ten instances from the second dataset, where each instance was associated with various item weights and packing schemes. In all test instances, the AFDO algorithm reached the minimal number of used bins, with an average execution time that was 10% to 56% better than that of PSO, 5% to 46% better than that of the CSA and 48% better than that of the Jaya algorithm. For a large-scale 1D-BPP, the proposed algorithm displayed high efficiency in obtaining solutions for instances from the third dataset with the smallest minimum and average fitness values when compared to those of other algorithms. In the last dataset, the AFDO did not reach the optimal number of bins but achieved a better item packing schema with a lower number of bins compared to the results of the PSO, CS, and Jaya algorithms. Based on the experimental results and a performance analysis of various test instances (i.e., 30 instances), the results showed that the proposed algorithm can be used to effectively explore the search space and locate an optimal solution with the smallest fitness value within a reasonable time when compared with other popular algorithms. In the future, we plan to improve this algorithm in two ways. The first is by comparing the proposed algorithm

with other recent metaheuristic algorithms and testing the methods with the remaining instances from each dataset. The second is by applying the proposed algorithm to real-life packing problems in a specific domain, such as the logistic and transportation sectors.

## ACKNOWLEDGMENT

They authors would like to thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

## REFERENCES

[1] D. Pisinger, "Heuristics for the container loading problem," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 382–392, Sep. 2002, doi: 10.1016/S0377-2217(02)00132-7.

[2] J. M. V. de Carvalho, "LP models for bin packing and cutting stock problems," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 253–273, Sep. 2002, doi: 10.1016/S0377-2217(02)00124-8.

[3] A. Aliano Filho, A. C. Moretti, and M. V. Pato, "A comparative study of exact methods for the bi-objective integer one-dimensional cutting stock problem," *J. Oper. Res. Soc.*, vol. 69, no. 1, pp. 91–107, Jan. 2018, doi: 10.1057/s41274-017-0214-7.

[4] U. Eliiyi and D. T. Deniz, "Applications of bin packing models through the supply chain," *Int. J. Bus. Manag. Stud.*, vol. 1, no. 1, pp. 11–19, Jan. 2009.

[5] E. López-Camacho, H. Terashima-Marín, G. Ochoa, and S. E. Conant-Pablos, "Understanding the structure of bin packing problems through principal component analysis," *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 488–499, Oct. 2013, doi: 10.1016/j.ijpe.2013.04.041.

[6] K. Fleszar and C. Charalambous, "Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem," *Eur. J. Oper. Res.*, vol. 210, no. 2, pp. 176–184, Apr. 2011, doi: 10.1016/j.ejor.2010.11.004.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1990.

[8] T. Ojeyinka, "Bin packing algorithms with applications to passenger bus loading and multiprocessor scheduling problems," *Commun. Appl. Electron.*, vol. 2, no. 8, pp. 38–44, Sep. 2015, doi: 10.5120/cae2015651851.

[9] E. C. Xavier and F. K. Miyazawa, "A one-dimensional bin packing problem with shelf divisions," *Discrete Appl. Math.*, vol. 156, no. 7, pp. 1083–1096, Apr. 2008, doi: 10.1016/j.dam.2007.05.053.

[10] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: Survey and classification," in *Handbook of Combinatorial Optimization*, P. M. Pardalos, D.-Z. Du, R. L. Graham, Eds. New York, NY, USA: Springer, 2013, pp. 455–531.

[11] F. Luo, I. D. Scherson, and J. Fuentes, "A novel genetic algorithm for bin packing problem in jMetal," in *Proc. IEEE Int. Conf. Cognit. Comput. (ICCC)*, Honolulu, HI, USA, Jun. 2017, pp. 17–23.

[12] T. Tlili and S. Krichen, "On solving the double loading problem using a modified particle swarm optimization," *Theor. Comput. Sci.*, vol. 598, pp. 118–128, Sep. 2015, doi: 10.1016/j.tcs.2015.05.037.

[13] T. G. Crainic, G. Perboli, and R. Tadei, "TS2PACK: A two-level tabu search for the three-dimensional bin packing problem," *Eur. J. Oper. Res.*, vol. 195, no. 3, pp. 744–760, Jun. 2009, doi: 10.1016/j.ejor.2007.06.063.

[14] X. Cui, "Using genetic algorithms to solve combinatorial optimization problems," in *The Application Handbook of Genetic Algorithms*, L. Chambers, Ed. Boca Raton, FL, USA: CRC Press, 1995, pp. 143–172.

[15] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, *Metaheuristic Applications in Structures and Infrastructures*. Waltham, MA, USA: Elsevier, 2013.

[16] X. S. Yang, *Nature-inspired Metaheuristic Algorithms*. Bristol, U.K.: Luniver Press, 2010.

[17] J. M. Abdullah and T. Ahmed, "Fitness dependent optimizer: Inspired by the bee swarming reproductive process," *IEEE Access*, vol. 7, pp. 43473–43486, Mar. 2019, doi: 10.1109/ACCESS.2019.2907012.

[18] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Chichester, U.K.: Wiley, 1990.

[19] 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.

[20] P. Schwerin and G. Wäscher, "A new lower bound for the bin-packing problem and its integration into MTP and BISON," *Pesquisa Operacional*, vol. 19, pp. 111–129, Nov. 1999.

[21] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *Eur. J. Oper. Res.*, vol. 255, no. 1, pp. 1–20, Nov. 2016, doi: 10.1016/j.ejor.2016.04.030.

[22] J. E. G. Coffman, M. R. Carey, and D. S. Johnson, "Approximation algorithms for bin packing," in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed. Boston, MA, USA: Pws Publishing Company, 1997, pp. 46–93.

[23] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM J. Comput.*, vol. 3, no. 4, pp. 299–325, Dec. 1974, doi: 10.1137/0203025.

[24] R. E. Korf, "An improved algorithm for optimal bin packing," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, Aug. 2003, pp. 1252–1258.

[25] J. Thomas and N. S. Chaudhari, "Design of efficient packing system using genetic algorithm based on hyper heuristic approach," *Adv. Eng. Softw.*, vol. 73, pp. 45–52, Jul. 2014, doi: 10.1016/j.advengsoft.2014.03.003.

[26] A. Stawowy, "Evolutionary based heuristic for bin packing problem," *Comput. Ind. Eng.*, vol. 55, no. 2, pp. 465–474, Sep. 2008, doi: 10.1016/j.cie.2008.01.007.

[27] 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.

[28] P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart, "Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics," in *Genetic and Evolutionary Computation—GECCO*, E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, and U.-M. O'Reilly, Eds. Berlin, Germany: Springer, 2003, pp. 1295–1306.

[29] P. Ross, S. Schulenburg, J. G. Marín-Bläzquez, and E. Hart, "Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems," in *Proc. 4th Annu. Conf. Genetic Evol. Comput.*, New York, NY, USA, Jul. 2002, pp. 942–948.

[30] A. Layeb and S. Chenche, "A novel GRASP algorithm for solving the bin packing problem," *Int. J. Inf. Eng. Electron. Bus.*, vol. 4, no. 2, pp. 8–14, Apr. 2012, doi: 10.5815/ijieeb.2012.02.02.

[31] D. Stakic, A. Anokic, and R. Jovanovic, "Comparison of different grasp algorithms for the heterogeneous vector bin packing problem," in *Proc. China-Qatar Int. Workshop Artif. Intell. Appl. Intell. Manuf. (AIAIM)*, Doha, Qatar, Jan. 2019, pp. 63–70.

[32] V. Hemmelmayr, V. Schmid, and C. Blum, "Variable neighbourhood search for the variable sized bin packing problem," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 1097–1108, May 2012, doi: 10.1016/j.cor.2011.07.003.

[33] 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.

[34] J. N. D. Gupta and J. C. Ho, "A new heuristic algorithm for the one-dimensional bin-packing problem," *Prod. Planning Control*, vol. 10, no. 6, pp. 598–603, Jan. 1999, doi: 10.1080/095372899232894.

[35] K.-H. Loh, B. Golden, and E. Wasil, "Solving the one-dimensional bin packing problem with a weight annealing heuristic," *Comput. Oper. Res.*, vol. 35, no. 7, pp. 2283–2291, Jul. 2008, doi: 10.1016/j.cor.2006.10.021.

[36] E. Sonuc, B. Sen, and S. Bayir, "Solving bin packing problem using simulated annealing," in *Proc. 65th Int. Conf. (ISERD)*, Mecca, Saudi Arabia, Jan. 2017, pp. 83–85.

[37] L. F. O. M. Santos, R. S. Iwayama, L. B. Cavalcanti, L. M. Turi, F. E. D. S. Morais, G. Mormilho, and C. B. Cunha, "A variable neighborhood search algorithm for the bin packing problem with compatible categories," *Expert Syst. Appl.*, vol. 124, pp. 209–225, Jun. 2019, doi: 10.1016/j.eswa.2019.01.052.

[38] R. Yesodha and T. Amudha, "Effectiveness of firefly algorithm in solving bin packing problem," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 5, pp. 1003–1010, May 2013.

[39] A. C. F. Alvim, C. C. Ribeiro, F. Glover, and D. J. Aloise, "A hybrid improvement heuristic for the one-dimensional bin packing problem," *J. Heuristics*, vol. 10, no. 2, pp. 205–229, Mar. 2004, doi: 10.1023/B:HEUR.0000026267.44673.ed.

[40] M. Buljubašić and M. Vasquez, "Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing," *Comput. Oper. Res.*, vol. 76, pp. 12–21, Dec. 2016, doi: 10.1016/j.cor.2016.06.009.

[41] Z. Zendaoui and A. Layeb, "Adaptive cuckoo search algorithm for the bin packing problem," in *Modelling and Implementation of Complex Systems*, S. Chikhi, A. Amine, A. Chaoui, M. K. Kholladi, and D. E. Saidouni, Eds. Cham, Switzerland: Springer, 2016, pp. 107–120.
[42] A. Layeb and S. R. Boussalia, "A novel quantum inspired cuckoo search algorithm for bin packing problem," *Int. J. Inf. Technol. Comput. Sci.*, vol. 4, no. 5, pp. 58–67, May 2012, doi: 10.5815/ijitcs.2012.05.08.
[43] W. H. El-Ashmawi and D. S. A. Elminaam, "A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem," *Appl. Soft Comput.*, vol. 82, Sep. 2019, Art. no. 105565, doi: 10.1016/j.asoc.2019.105565.
[44] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, and S. Mirjalili, "An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems," *Pers. Ubiquitous Comput.*, vol. 22, nos. 5–6, pp. 1117–1132, Oct. 2018, doi: 10.1007/s00779-018-1132-7.
[45] G. Scheithauer, "One-dimensional bin packing," in *Introduction to Cutting and Packing Optimization: Problems, Modelling Approaches, Solution Methods*, G. Scheithauer, Ed. Cham, Switzerland: Springer, 2018, pp. 47–72.
[46] J. D. Villa, "Swarming behavior of honey bees (Hymenoptera: Apidae) in Southeastern Louisiana," *Ann. Entomol. Soc. Amer.*, vol. 97, no. 1, pp. 111–116, Jan. 2004, doi: 10.1603/0013-8746(2004)097[0111:SBOHBH]2.0.CO;2.
[47] K. M. Schultz, K. M. Passino, and T. D. Seeley, "The mechanism of flight guidance in honeybee swarms: Subtle guides or streaker bees?" *J. Experim. Biol.*, vol. 211, no. 20, pp. 3287–3295, Oct. 2008, doi: 10.1242/jeb.018994.
[48] R. Lewis, "A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing," *Comput. Oper. Res.*, vol. 36, no. 7, pp. 2295–2310, Jul. 2009, doi: 10.1016/j.cor.2008.09.004.
[49] B. Rieck, *Basic Analysis of Bin-Packing Heuristics*. Heidelberg, Germany: Publicado por Interdisciplinary Center for Scientific Computing, Heildelberg Univ., 2010.
[50] J. Yan, W. He, X. Jiang, and Z. Zhang, "A novel phase performance evaluation method for particle swarm optimization algorithms using velocity-based state estimation," *Appl. Soft Comput.*, vol. 57, pp. 517–525, Aug. 2017, doi: 10.1016/j.asoc.2017.04.035.
[51] A. Askarzadeh, "A novel Metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016, doi: 10.1016/j.compstruc.2016.03.001.
[52] R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, pp. 19–34, 2016, doi: 10.5267/j.ijiec.2015.8.004.
[53] *Bin Backing Problem Benchmark Datasets*. Accessed: Nov. 23, 2019. [Online]. Available: https://www2.wiwi.unijena.de/Entscheidung/binpp/index.htm

**DIAA SALAMA ABDUL-MINAAM** was born in Kafr Saqr, Sharqia, Egypt, in November 1982. He received the B.Sc. degree (Hons.) from the Faculty of Computers and Informatics, Zagazig University, Egypt, in 2004, the master's degree in information systems from the Faculty of Computers and Information, Menoufia University, Egypt, in 2009, specializing in cryptography and network security, and the Ph.D. degree in information systems from the Faculty of Computers and Information, Menoufia University, Egypt, in 2015. He has been an Associate Professor at the Information Systems Department, Faculty of Computers and Artificial Intelligence, Benha University, Egypt, since February 2020. He has worked on several research topics and contributed to more than 40 technical articles in the areas of wireless networks, wireless network security, information security and internet applications, cloud computing, mobile cloud computing, the Internet of Things, and machine learning in international journals, international conferences, local journals, and regional conferences. His primary interests are cryptography, network security, the IoT, big data, cloud computing, and deep learning.

**WADHA MOHAMMED EDKHEEL SAQAR AL-MUTAIRI** was born in Kuwait, in May 1977. She received the bachelor's degree in computer and information systems from Arab Open University—Kuwait Branch, in 2009. She has been working as a Computer Teacher at the Middle School of the Ministry of Education in Kuwait, since 2010. She has also been a Trainer with the Public Authority for Applied Education and Training in Kuwait, since 2016.

**MOHAMED A. AWAD** was born in Shibin Al Qanater, Qalyubia Governorate, Egypt, in November 1977. He received the B.S. degree from the Faculty of Computers and Informatics, Helwan University, Egypt, in 2000, attaining a grade of very good, the master's degree in information systems from the Faculty of Computers and Information, Helwan University, Egypt, in 2005, specializing in distributed databases, and the Ph.D. degree in information systems from the Faculty of Computers and Information, Helwan University, Egypt. He has been an Associate Professor with the Information Systems Department, Faculty of Computers and Information, Benha University, Egypt, since 2015. He has worked on many research projects and contributed to more than 20 technical articles in various areas of decision support systems.

**WALAA H. EL-ASHMAWI** received the B.Sc. and M.Sc. degrees in Egypt, in 2001 and 2008, respectively, and the Ph.D. degree from the Computer Science Department, College of Information Science and Engineering, Hunan University, China, in 2013. She is currently an Assistant Professor with the Computer Science Department, Faculty of Computers and Informatics, Suez Canal University, Egypt. She has authored or coauthored many journal publications and conferences in the areas of artificial intelligence, team formation problems, multiagent systems, expert systems, and neural networks.