Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2023.00061

Research Article

Optimization of one-dimensional bin packing problem using a hybrid flower pollination algorithm

Harun GEZİCİ[1,*], Haydar LİVATYALI[2]

[1]Department of Electronics and Automation, Kırklareli University, Kırklareli, 39100, Türkiye
[2]Department of Mechatronics Engineering, Yıldız Technical University, İstanbul, 34349, Türkiye

ARTICLE INFO

Article history
Received: 03 May 2021
Revised: 08 July 2021
Accepted: 28 September 2021

Keywords:
Bin Packing; Flower Pollination Algorithm; Genetic Algorithm; Hybridization

ABSTRACT

The bin packing problem (BPP) is one of the most elaborated combinatorial optimization problems, yet there is still need and room for improvement. An improved flower pollination algorithm (FPA) is proposed for the solution of one-dimensional BPP (1DBPP). To increase efficiency, global and local pollination procedures are modified and hybridized with the genetic algorithm (GA). An elimination strategy that increases the quality of the solution set in each iteration is also included and the proposed algorithm is tested on the Scholl dataset. It is compared with the adaptive fitness-dependent optimizer (AFDO), the improved Lévy-based whale optimization (ILWOA), and the modified squirrel search BPP (MSBPP) algorithms. The comparison is made in terms of metrics including the container number, minimum and average fitness values, and minimum and average percentage performances. In terms of the container number, the proposed algorithm yielded results equal to or better than competing algorithms. In terms of minimum fitness value, the proposed algorithm achieved 88% more successful results than its competitors. It achieved 92% more successful results in terms of average fitness value. In terms of minimum percentage performance, the proposed algorithm is more successful in 93.3% of the samples compared to AFDO. Compared to MSBPP, the proposed algorithm is 84.6% more successful. In terms of average percentage performance metric, the proposed algorithm has better results in 90% of the samples than AFDO; and compared to MSBPP, it is more successful in 96.1% of the samples. These results show the effectiveness of the proposed algorithm to solve the 1DBPP problem.

Cite this article as: Gezici H, Livatyalı H. Optimization of one-dimensional bin packing problem using a hybrid flower pollination algorithm. Sigma J Eng Nat Sci 2023;41(3):545−564.

INTRODUCTION

The one-dimensional bin packing problem (1BPP) is defined as the optimal placement of boxes in certain sizes inside of containers in certain sizes. The depth (a) and width (b) dimensions of the boxes are considered to be the same ($a_1 = a_2 = \dots = a_n$ and $b_1 = b_2 = \dots = b_n$) and their height (h) are different ($h_1 \neq h_2 \neq \dots \neq h_n$) (Figure 1). The depth (D) and width (W) dimensions of the containers are whole numbers which are exact times of the box dimensions

***Corresponding author.**
*E-mail address: harun.gezici@klu.edu.tr
This paper was recommended for publication in revised form by Regional Editor Oscar Castillo

(D = *k*.a and W = *l*.b, *k* and $l \in Z^+$). The container height (H) dimension does not have to be exact times of the box height dimension. The total height of the boxes placed in the containers should not exceed the height of the container. The aim is to maximize the container occupancy rate and use a minimum number of containers. The process of placing boxes in containers is done by humans or robots. According to the availability of the working area, a loading direction is determined for loading the boxes into the containers. The number and dimensions of the boxes to be loaded are already known. Based on these parameters, the optimum box order is tried to be determined. Some of the containers may remain empty after loading. The constraints and decision structure of 1BPP are defined as follows [1,2].

$$Min \sum_{j=1}^{m} y_j \tag{1}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} h_i \le H y_j \quad \forall j \in B = \{1, \ldots, m\} \tag{3}$$

$$y_j \in \{0,1\} \quad \forall j \in B \tag{4}$$

$$x_{ij} \in \{0,1\} \quad \forall j \in B, \forall i \in n \tag{5}$$

$$x_{ij} = \begin{cases} 1 \; if \; item \; i \; is \; packed \; into \; bin \; j \\ 0 \quad otherwise \end{cases} \tag{6}$$

$$y_j = \begin{cases} 1 \; if \; bin \; j \; is \; used \\ 0 \quad otherwise \end{cases} \tag{7}$$

where n shows the number of boxes, H indicates the container capacity, h_j shows the height of the jth box, m is the number of containers, and B is the set of containers.

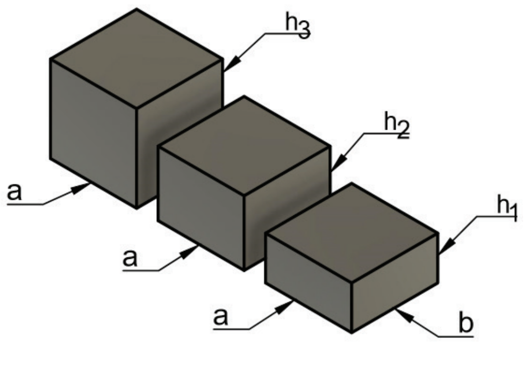


Figure 1. Box dimensions.

BPP is in the Np-hard problem class [3]. It can be adapted to real-world problems in areas such as logistics and manufacturing [4,5]. 1DBPP is used in memory management in the computer industry. In the literature, exact, heuristic, and meta-heuristic methods are suggested for the solution of BPP [6,7]. Although the exact methods give favorable results in the solution of small-scale problems, they need a long calculation time in large-scale problems [8]. Calculation time with meta-heuristic methods is short, but it does not always guarantee the correct result. Studies on meta-heuristic methods are ongoing due to short calculation times and successful results [9,10].

Meta-heuristic algorithms such as ant colony algorithm [11], particle swarm optimization algorithm [12], GA [13,14], tabu search [15], ILWOA [6] have been used in the solution of BPP. It is known that there is no single algorithm that can solve all optimization problems [16,17]. Applying meta-heuristic algorithms in pure form to BPP may not yield optimum results. For this reason, in some studies, it is preferred to transform the meta-heuristic algorithms into a hybrid structure while they are applied to BPP [18–20].

In recent years, Levy-based meta-heuristic optimization algorithms have been used for the solution of the BPP, and obtained results were successful. Studies on ILWOA and Adaptive Cuckoo Search based on levy distribution, which provide excellent results in the solution of the BPP, have been published previously [6,20]. In these studies, it is seen that the Levy distribution improves the solution quality of the algorithm. Levy distribution transforms the algorithms into the continuous form while bin packing is a problem in discrete form. Algorithms in continuous form can be converted into discrete forms with proven methods [20–23].

In this study, the problem of placing different-sized boxes into containers of the same size will be emphasized [24]. For the solution of the 1D-BPP, a hybrid algorithm was created by reinforcing the Meta-heuristic algorithm by combining the flower pollination algorithm (FPA) with GA. The proposed algorithm is called *Improved Hybrid Flower Pollination Genetic Algorithm* (IHFPGA). Global searching ability is increased by making changes in the mathematical model of FPA. Mutation operators and the elimination process have been added to the IHFPGA to increase the efficiency of the algorithm. The existence of mutation operators increases the local searching ability of the proposed algorithm. With these changes, the proposed algorithm is aimed to have the ability to search for a better solution space compared to other algorithms in the literature. The proposed algorithm has been compared with other meta-heuristic algorithms with the best results from the literature. The obtained results have confirmed the validity of the proposed algorithm. In Section 2, the literature on 1BPP is summarized. In Section 3, İnformation about the developed algorithm is given. In Section 4, the test parameters are explained and comparative results are given. These results are discussed in Section 5.

Related Work on the Bin Packing Problem

For the solution of 1DBPP, researchers have proposed various heuristic and meta-heuristic methods. The first suggested heuristics are Next-Fit (NF), First-Fit (FF), Best-Fit (BF) methods, and their derivatives [4,22]. In the NF method, the incoming box is placed in the last opened container, if the container is not available, a new bin is opened. In the FF method, the next box is placed in the first available space for it. In BF, the next box is placed in the smallest space it can fit.

Among the meta-heuristic algorithms, GA is the most known and has been used many times to solve 1DBPP [25,26]. In [3], a new island-parallel grouping GA was developed by modifying the GA for the solution of grouping problems. In [15], the Greedy Randomized Adaptive Search (GRASP) method has been proposed. This method consists of 2 steps. The first stage was created by hybridizing the FF and BF heuristic methods. In the second stage, the taboo search procedure was applied to improve the results obtained in the first stage. A modified GRASP method is presented in [27]. In [28], an iterative local search algorithm based on gradually reducing the number of divisions has been proposed. In [29] First-Fit Decreasing (FFD) algorithm has been modified to shorten the resolution time of 1DBPP. In [30], a heuristic weight annealing algorithm is presented to increase the quality of the results obtained. In [31]', a multi-step tabu search algorithm based on dynamic programming of object sets has been proposed for the solution of 1DBPP. In [32], a new algorithm is presented, which is a modified version of the Grouping Genetic Algorithm (GGA-CGT). In this study, a new grouping genetic operator is proposed to increase the probability of finding the best genes on chromosomes. In addition, a new gene derivation technique that enables the identification of the search area and the FF procedure to produce a quality starting population are also presented. In [6]', Whale Optimization Algorithm (WOA) is presented for the solution of 1DBPP. WOA is a meta-heuristic algorithm that mimics whales' unique hunting strategy. In this study, Lévy flight was adapted to the algorithm to improve the original algorithm. In addition, the algorithm includes an additional mutation stage and a logistic map. In [33], the AFDO method, which is a modified version of the Fitness-Dependent Optimizer (FDO) procedure, is presented. This algorithm is inspired by the characteristics of the breeding process of bee swarms and collective decision-making behavior. This algorithm also includes the FF procedure for optimization of the starting population. In [34], the adaptive African Buffalo Optimization algorithm (ABO) is presented for the solution of 1DBPP. The proposed algorithm is combined with the ranked order value method used for discretization. In [4]', a modified version of the Squirrel Search Algorithm, inspired by the behavior of squirrels during foraging, is proposed. The proposed algorithm has a procedure that controls the random generation of the starting population. It also has various strategies to improve the quality of the solution in each iteration. In [35], firefly algorithm, GA, adaptive cuckoo search algorithm, and artificial bee colony algorithm are presented for the solution of 1DBPP. In addition, these algorithms include best fit and better fit heuristics.

While solving optimization problems like 1D-BPP, some meta-heuristic algorithms might not result successfully. The reason for it is, as it is explained in the free-lunch theorem, that an optimization algorithm cannot solve all optimization problems [16]. Moreover, definite solutions to some optimization problems like 1D-BPP are not known. Therefore, researchers keep developing more useful optimization algorithms.

In this paper, a hybrid meta-heuristic algorithm is proposed for the solution of 1D-BPP. The basis of the algorithm is FPA based on the Levy distribution. To increase the performance of FPA, its mathematical model has been changed. Three different mutation operators of the GA have been added to increase the 1D-BPP solution performance of the algorithm. There is also an elimination operator in the algorithm. This operator removes the boxes that maximize the occupancy rate of the containers in each iteration. This elimination reduces the size of the solution space. Thus, the efficiency of the algorithm is increased in terms of the iteration number, iteration time, and occupancy rate. This study contributes to the literature in 3 ways. First, FPA's mathematical model has been modified. Second, FPA is hybridized with GA's mutation operators. Third, the hybrid FPA is applied to 1D-BPP and a competitive algorithm is developed.

Implementation of IHFPGA to Bin Packing Problem

The proposed algorithm has five main stages. These stages are FPA, discretization, fitness function, mutation, and elimination. A flow chart of the proposed algorithm is given in Figure 2.

Flower pollination algorithm

The flower pollination algorithm is an optimization algorithm inspired by the pollen transfer in nature. Pollination takes place in two different forms, biotic and abiotic. Biotic pollination is often caused by insects, birds, bees, etc. supplied with pollinators. These pollinators can carry pollen over long distances and show Levy flight behavior. Water and wind are exemplary pollinators for abiotic pollination. Pollination is divided into two as self-pollination and cross-pollination. Self-pollination occurs in the same flower or between different flowers of the same plant. Therefore, self-pollination can be considered as local pollination. Cross-pollination occurs between the flowers of two different plants. Biotic, cross-pollination can occur over long distances. Therefore, cross-pollination can be regarded as global pollination. Due to other factors such as physical proximity and wind, local pollination has an essential share in general pollination activities.

FPA is designed based on the features characterized above. There are two critical steps in this algorithm, and these are global pollination and local pollination. Global pollination is modeled by the following equation [36,37].

$$x_i^{t+1} = x_i^t + L(x_i^t - g) \tag{8}$$

$$L \approx \frac{\lambda \Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{s^{1+\lambda}} \quad s \gg s_0 > 0 \tag{9}$$

In the formula, t is the number of iterations, i is the pollen bundle or solution vector index, X_i is the solution vector, and g is the best of all solution vectors in the current generation. L is the Levy flight, the mathematical model of pollinators' flight routes. Local pollination can be formulated as follows.

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{10}$$

In the equation, x_j^t and x_k^t represent pollen bunches of different flowers of the same plant or different solutions of the solution set. ε represents a random local pollination distance and has a normal distribution between 0-1 [38]. The switching probability of local pollination or global pollination is controlled by p ∈ [0, 1].

Improved flower pollination algorithm

Global pollination and local pollination procedures have been modified to improve the performance of the flower pollination algorithm. New mathematical expressions of the IHFPGA are given in equation 11 (global pollination) and equation 12 (local pollination).

$$x_i^{t+1} = x_i^t + Lfw(x_i^t - g) \tag{11}$$

$$x_i^{t+1} = x_i^t + \varepsilon fw\left(x_j^t - x_k^t\right) \tag{12}$$

fw in the equations is the fit weight and is an array of n elements (where n is the number of elements in the solution set). *fw* is calculated using equation 13 [6].

$$fw_{n+1} = afw_n(1 - fw_n), \; 0 < a \leq 4 \tag{13}$$

where fw_n is the initial fitness weight (Equation 14).

$$fw_n = f(x_i^t)((f(x_i^t) - f_{best}) \tag{14}$$

where f_{best} is the fitness value of the best solution. $f(x_i^t)$ is the fitness value of the ith solution in the tth iteration.

Objective Function

Using the container number as an objective function causes algorithm stagnation in the solution of BPP because

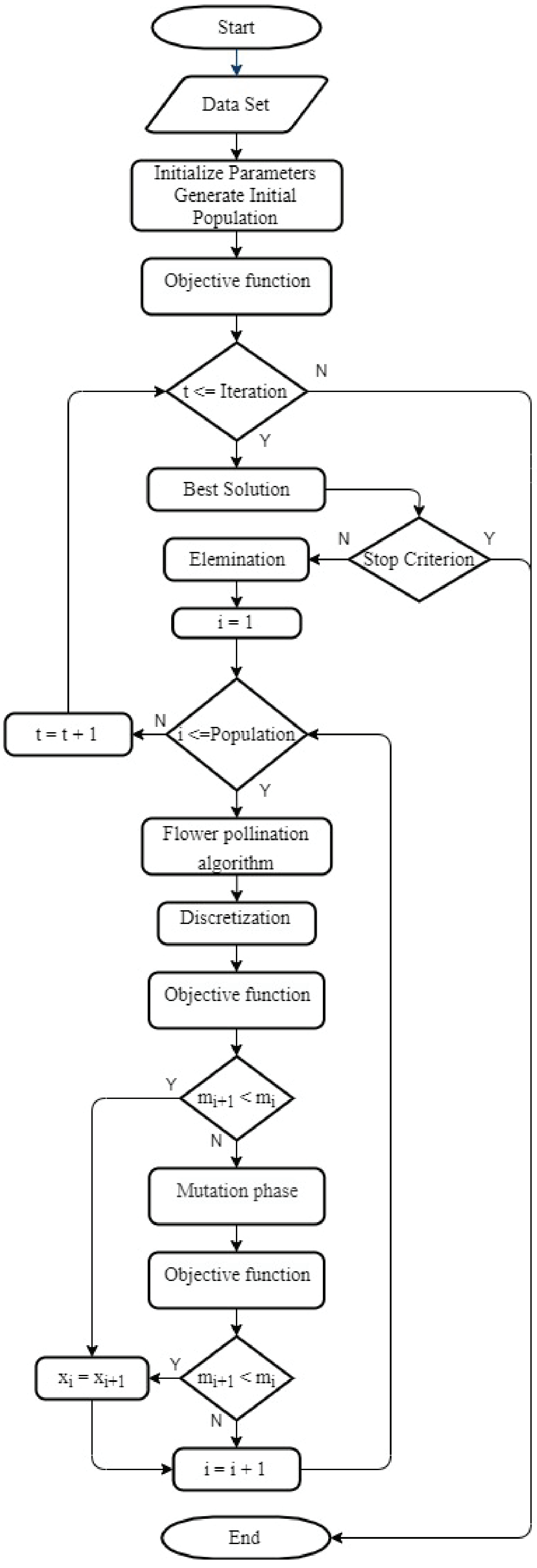


Figure 2. IHFPGA flowchart.

multiple sequences can give the same container number. For this reason, instead of container number using the container occupancy rate as the objective function increases the efficiency of the algorithm (Equation 15).

$$min.fitness\ value = 1 - \left(\frac{\sum_{i=1}^{m} \left(\frac{\sum_{items=1}^{n} h}{H} \right)^{k}}{m} \right) \tag{15}$$

Where m is the number of containers that are used, H is the container capacity, n is the box number of the ith container, h is the height of the box and k is the equation constant and is usually taken as 2.

Discretization

FPA is an optimization algorithm in continuous form. The solution vector must be converted into a discrete form to apply the FPA method to the BPP. In BPP, four different ways are used to transform the solution set into a discrete form. These are “the largest ranked value, (LRV)” [23], “the smallest position value, (SPV)” [22], “the largest order value, (LOV)” [21], and “the rank order value, (ROV) ” [20]. In Table 2, an example of converting a 6-element continuous form array into a discrete form is given. In the ROV method, the smallest element of the array is assigned 1, while the highest one is assigned 6. The ROV method converts the continuous form set into the nearest numbers when converting it into discrete form, which does not disrupt the numerical sequence of the set. For this reason, the ROV method was preferred as the method of converting into discrete form.

Mutation phase

The mutation process in IHFPGA is performed when FPA cannot improve the result. The mutation process can be achieved with three different methods. These are swap, displacement, and reversion methods. In the swap process, two randomly determined bins are moved [38]. In displacement, a random subarray is selected and transferred randomly to another location [39]. In reversion, a randomly selected subarray is reversed [6]. It is decided by a random number in which method to use. Examples of mutation methods are given in Table 3.

Elimination phase

The elimination phase is added to the proposed algorithm to increase efficiency. Each iteration of the algorithm determines the best individual of the population, which is the solution set. This solution set is placed in containers using the FF placement strategy. If the number of iterations is less than half of the total number of iterations, the boxes that fill up the containers are removed from the population. If the iteration number is more than half of the total number of iterations, containers that are as full as the theoretical minimum container occupancy rate (TMCOR) or more are removed from the population. TMCOR is calculated using the theoretical minimum container number (TMCN) (Equation 16 and 17). After this elimination process, the best sequence continues to be searched among the remaining boxes (Figure 3).

$$\text{TMCN} = \frac{\sum h}{H} \tag{16}$$

$$\text{TMCOR} = \frac{TMCN \in Q}{TMCN \in Z^{+}} \tag{17}$$

Table 1. Discretization

Continuous solution		**3.52**	**0.80**	**4.01**	**4.89**	**2**	**5.68**
Discrete solution	**LRV**	4	6	3	2	5	1
	SPV	2	5	1	3	4	6
	LOV	6	4	3	1	5	2
	ROV	3	1	4	5	2	6

Table 2. Mutation operators

Swap	Old	1	**2**	3	4	**5**	6
	New	1	**5**	3	4	**2**	6
Displacement	Old	1	2	**3**	**4**	**5**	6
	New	**3**	**4**	**5**	1	2	6
Reversion	Old	1	**2**	**3**	**4**	5	6
	New	1	**4**	**3**	**2**	5	6

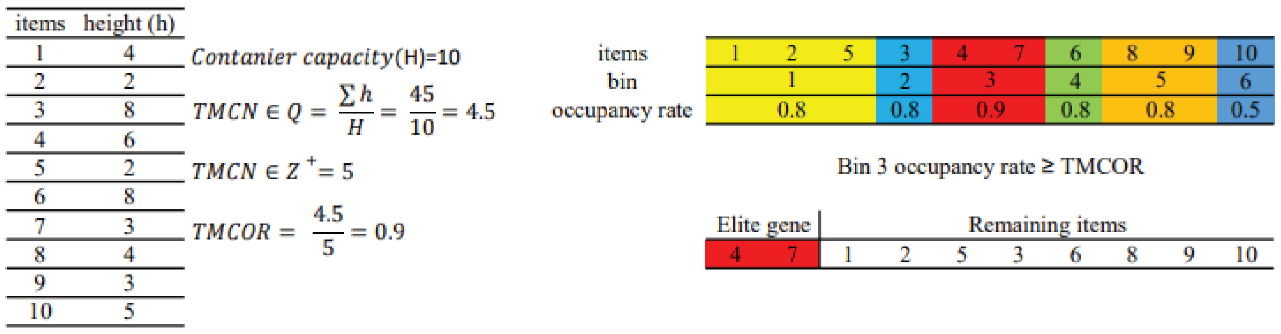


Figure 3. Elimination strategy.

Table 3. Scholl easy class container numbers

Ins_no	Ins_id	n	H	m*	AFDO	ILWOA	MSBPP	IHFPGA
1	N1C1W1_A	50	100	25	-	25	25	**25**
2	N1C1W1_B	50	100	31	31	31	31	**31**
3	N1C1W1_C	50	100	20	-	-	20	**20**
4	N1C1W1_D	50	100	28	-	28	28	**28**
5	N1C1W1_E	50	100	26	26	26	26	**26**
6	N1C1W1_F	50	100	27	-	27	27	**27**
7	N1C1W1_G	50	100	25	-	25	-	**25**
8	N1C1W1_I	50	100	25	25	25	-	**25**
9	N1C1W1_M	50	100	30	30	-	-	**30**
10	N1C1W1_Q	50	100	28	28	-	-	**28**
11	N1C1W2_D	50	100	31	31	-	-	**31**
12	N1C2W1_P	50	120	21	21	-	-	**21**
13	N1C2W2_R	50	120	25	25	-	-	**25**
14	N1C3W2_A	50	150	19	19	-	-	**19**
15	N2C1W1_A	100	100	48	-	-	48	**48**
16	N2C1W1_B	100	100	49	49	-	49	**49**
17	N2C1W1_C	100	100	46	46	-	-	**46**
18	N2C1W2_C	100	100	68	-	-	68	**68**
19	N2C1W2_D	100	100	74	-	-	74	**74**
20	N2C1W2_N	100	100	64	-	64	-	**64**
21	N2C1W2_O	100	100	64	-	64	-	**64**
22	N2C1W2_P	100	100	68	-	68	-	**68**
23	N2C1W2_R	100	100	67	-	67	-	**67**
24	N2C1W4_F	100	100	77	77	-	-	**77**
25	N2C2W1_H	100	120	46	46	-	-	**46**
26	N3C1W4_N	200	100	148	148	-	-	**148**
27	N3C2W2_S	200	120	107	107	-	-	**107**
28	N3C2W4_A	200	120	113	-	-	113	**113**
29	N3C2W4_T	200	120	119	-	-	119	**119**
30	N4C1W2_T	500	100	323	-	323	-	**323**
31	N4C1W4_A	500	100	368	-	368	-	**368**
32	N4C1W4_B	500	100	349	-	349	-	**349**
33	N4C1W4_C	500	100	365	-	365	-	**365**
34	N4C1W4_D	500	100	359	-	359	-	**359**

EXPERIMENTAL RESULTS

The proposed algorithm is coded in Python. Simulations (experiments) are performed on a 64-bit operating system with a 2.4 GHz CPU and 6 GB RAM. The proposed algorithm has been tested using the publicly available Scholl dataset [1] (http://or.dei.unibo.it/library/bpplib). The Scholl data set consists of 3 classes that include easy, medium, and difficult examples. The optimum container numbers of the Scholl dataset are known and these values are shown in the tables with m *. The proposed algorithm is compared with AFDO [33], ILWOA [6], and MSBPP [4] algorithms. The studies selected for comparison use different samples from each other and the results of some samples of the Scholl dataset are not available in these studies. Unavailable samples are shown with '-' in the tables. The container number, minimum fitness value, average fitness value, minimum percentage performance, and average percentage performance metrics are used for comparison. The proposed algorithm is run 10 times to be compared with competing algorithms and the results are recorded. The proposed algorithm has a population of 50 and an iteration number of 50. Besides, TMCN is determined as the stop limit of the algorithm (Equation 16).

In Table 3, Table 4, and Table 5, the number of containers obtained in previous studies using the Scholl dataset and in this study are given. In the tables, the first column shows

Table 4. Scholl medium class container numbers

Ins_no	**Ins_id**	**n**	**H**	**m***	**AFDO**	**ILWOA**	**MSBPP**	**IHFPGA**
1	N1W1B1R1	50	1000	18	-	-	18	**18**
2	N1W1B1R2	50	1000	19	19	19	-	**19**
3	N1W1B1R5	50	1000	17	17	-	17	**17**
4	N1W1B1R9	50	1000	17	-	17	17	**17**
5	N1W1B2R0	50	1000	17	-	17	-	**17**
6	N1W1B2R1	50	1000	17	-	17	-	**17**
7	N1W1B2R3	50	1000	16	-	17	-	**16**
8	N1W1B2R6	50	1000	17	-	-	17	**17**
9	N1W1B2R7	50	1000	18	18	-	18	**18**
10	N1W1B2R9	50	1000	18	-	-	18	**18**
11	N1W2B2R2	50	1000	10	-	-	10	**10**
12	N1W2B2R7	50	1000	10	-	-	10	**10**
13	N1W2B2R9	50	1000	11	11	-	-	**11**
14	N1W4B3R8	50	1000	6	6	-	-	**6**
15	N2W1B1R0	100	1000	34	-	34	-	**34**
16	N2W1B1R1	100	1000	34	-	35	-	**34**
17	N2W1B1R3	100	1000	34	-	35	-	**34**
18	N2W1B1R4	100	1000	34	-	34	-	**34**
19	N2W2B1R2	100	1000	21	21	-	-	**21**
20	N2W2B1R6	100	1000	21	-	-	21	**21**
21	N2W2B3R9	100	1000	20	20	-	-	**20**
22	N2W3B3R7	100	1000	13	-	13	-	**13**
23	N2W4B1R0	100	1000	12	-	12	-	**12**
24	N3W2B1R4	200	1000	40	-	-	41	**40**
25	N3W1B3R5	200	1000	65	65	-	-	**65**
26	N3W4B2R9	200	1000	22	22	-	-	**22**
27	N4W2B1R0	500	1000	101	-	105	-	**102**
28	N4W2B1R3	500	1000	100	-	104	-	**101**
29	N4W3B3R2	500	1000	72	-	-	72	**72**
30	N4W3B3R7	500	1000	74	-	74	-	**74**
31	N4W4B1R0	500	1000	56	-	57	-	**56**
32	N4W4B1R1	500	1000	56	-	57	-	**56**
33	N4W4B2R7	500	1000	57	57	-	-	**57**

Table 5. Scholl hard class container numbers

Ins_no	Ins_id	n	H	m*	AFDO	ILWOA	MSBPP	IHFPGA
1	HARD0	200	100000	56	59	58	-	**56**
2	HARD1	200	100000	57	-	59	60	**57**
3	HARD2	200	100000	56	-	59	-	**57**
4	HARD3	200	100000	55	59	58	59	**56**
5	HARD4	200	100000	57	60	59	-	**57**
6	HARD5	200	100000	56	60	58	-	**56**
7	HARD6	200	100000	57	-	59	-	**57**
8	HARD7	200	100000	55	58	57	-	**57**
9	HARD8	200	100000	57	-	59	-	**57**
10	HARD9	200	100000	56	-	59	59	**56**

the sample number, the second column shows the name of the sample, the third column shows the number of boxes in the sample, the third column shows the container capacity, the fifth column shows the best-known container numbers, the sixth, seventh and eighth columns show the container numbers of the rival algorithms and the last column shows the container numbers obtained by the proposed algorithm.

Table 3 shows the easy class of the Scholl dataset. In this table, it is seen that the proposed algorithm in all examples achieves the best container numbers similar to other algorithms.

Table 4 examines the medium class of the Scholl dataset. The proposed algorithm has achieved equal or better results than competing algorithms in all samples. The proposed algorithm has achieved better results than the ILWOA algorithm in examples 7, 16, 17, 27, 28, 31, and 32. Also, the proposed algorithm gave a better result than the MSBPP algorithm in example 24. The proposed algorithm achieved the best-known results in 31 of 33 samples. In 2 samples (27, 28) it approached the best-known results with a difference of 1 box.

Table 5 shows the hard class of the Scholl dataset. The proposed algorithm gave equal or better results than competing algorithms in all examples. The best-known results are obtained in samples 1, 2, 5, 6, 7, 9, and 10. In examples 3 and 4, the best-known results are approached with a difference of 1 box, and in example number 8, it is approached with 2 boxes.

Table 6 shows the minimum fitness values (Equation 15) and average fitness values (Equation 18) obtained from previous studies and this study for three classes of the Scholl dataset. Since minimum fitness values and average fitness values are not given in the study using the ILWOA algorithm, they are not included in table 6. The sample number is given in the first column of Table 6 and the sample name is given in the second column. In the third, fifth, and seventh columns, the minimum fitness values obtained from the studies are given, average fitness values are given in the fourth, sixth, and eighth columns.

Table 6. Scholl dataset fitness value

Scholl Easy Class							
ins_no	**ins_id**	**AFDO**		**MSBPP**		**IHFPGA**	
		Min.	**Avg.**	**Min.**	**Avg.**	**Min.**	**Avg.**
1	N1C1W1_A	-	-	0.12	0.128	**0.0498**	**0.0502**
2	N1C1W1_B	0.1980	0.2050	0.2	0.204	**0.1724**	**0.1743**
3	N1C1W1_C	-	-	0.12	0.131	**0.0157**	**0.0319**
4	N1C1W1_D	-	-	0.19	0.195	**0.1586**	**0.1608**
5	N1C1W1_E	0.1750	0.1759	**0.1**	0.157	0.1110	**0.1115**
6	N1C1W1_F	-	-	0.12	0.163	**0.1004**	**0.1037**
7	N1C1W1_I	0.1420	0.1662	-	-	**0.1013**	**0.1046**
8	N1C1W1_M	0.1890	0.1920	-	-	**0.1622**	**0.1636**
9	N1C1W1_Q	0.1740	0.1992	-	-	**0.1408**	**0.1430**
10	N1C1W2_D	0.2030	0.2095	-	-	**0.1507**	**0.1519**

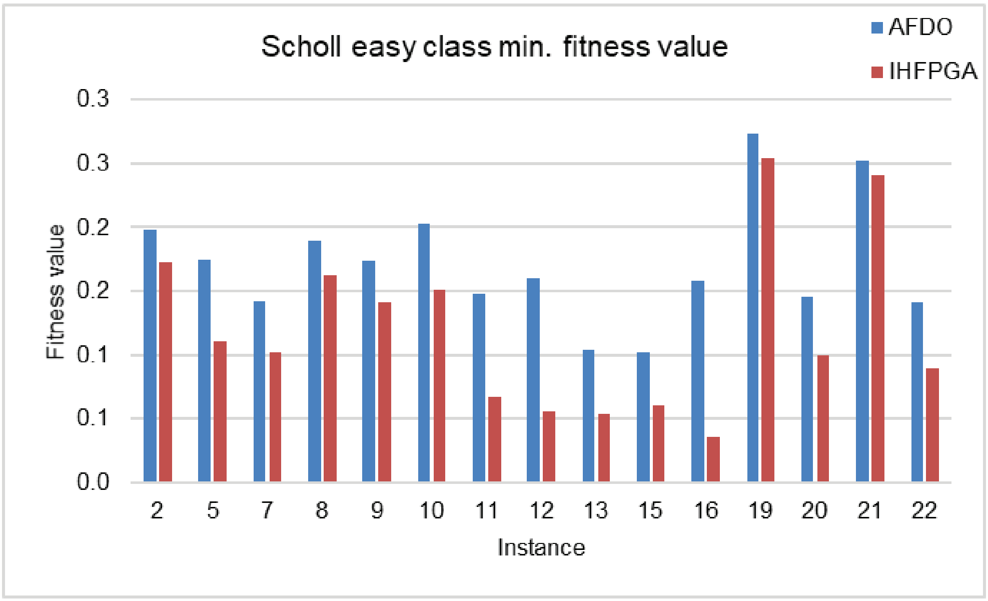
11	N1C2W1_P	0.1480	0.1829	-	-	**0.0666**	**0.0676**
12	N1C2W2_R	0.1600	0.1767	-	-	**0.0563**	**0.0593**
13	N1C3W2_A	0.1040	0.1413	-	-	**0.0538**	**0.0548**
14	N2C1W1_A	-	-	0.1	0.108	**0.0455**	**0.0485**
15	N2C1W1_B	0.1020	0.1187	0.1	0.107	**0.0606**	**0.0635**
16	N2C1W1_C	0.1580	0.1811	-	-	**0.0354**	**0.0359**
17	N2C1W2_C	-	-	0.24	0.253	**0.2247**	**0.2267**
18	N2C1W2_D	-	-	**0.17**	0.226	0.2116	**0.2126**
19	N2C1W4_F	0.2730	0.2811	-	-	**0.2537**	**0.2542**
20	N2C2W1_H	0.1450	0.1524	-	-	**0.0990**	**0.1013**
21	N3C1W4_N	0.2520	0.2576	-	-	**0.2411**	**0.2414**
22	N3C2W2_S	0.1410	0.1491	-	-	**0.0898**	**0.0907**
23	N3C2W4_A	-	-	0.17	0.177	**0.1149**	**0.1157**
24	N3C2W4_T	-	-	0.19	0.204	**0.1448**	**0.1458**
Scholl Medium Class							
ins_no	**ins_id**	**AFDO**		**MSBPP**		**IHFPGA**	
		Min.	**Avg.**	**Min.**	**Avg.**	**Min.**	**Avg.**
25	N1W1B1R1	-	-	0.17	0,218	**0.1038**	**0.1222**
26	N1W1B1R2	0.211	0.2297	-	-	**0.1683**	**0.2061**
27	N1W1B1R5	0.111	0.1484	**0.03**	0.088	0.0548	**0.0569**
28	N1W1B1R9	-	-	0.09	0.125	**0.0605**	**0.0610**
29	N1W1B2R6	-	-	0.13	0.149	**0.0832**	**0.0840**
30	N1W1B2R7	0.108	0.1229	0.13	0.14	**0.0809**	**0.0819**
31	N1W1B2R9	-	-	0.12	0.159	**0.0711**	**0.0982**
32	N1W2B2R2	-	-	**0.12**	**0.124**	0.1182	0.1272
33	N1W2B2R7	-	-	0.13	0.134	**0.0464**	**0.0469**
34	N1W2B2R9	0.113	**0.118**	-	-	**0.1128**	0.1261
35	N1W4B3R8	**0.06**	**0.0602**	-	-	0.0605	0.0615
36	N2W2B1R2	0.11	0.1222	-	-	**0.0682**	**0.0684**
37	N2W2B1R6	-	-	0.11	0.12	**0.0706**	**0.0714**
38	N2W2B3R9	0.076	0.075	-	-	**0.0295**	**0.0298**
39	N3WIB3R5	0.073	0.059	-	-	**0.0244**	**0.0279**
40	N3W2B1R4			0.11	0.109	**0.0204**	**0.0205**
41	N3W4B2R9	**0.059**	**0.0624**	-	-	0.0592	0.0634
42	N4W3B3R2	-	-	0.03	0.029	**0.0156**	**0.0161**
43	N4W4B2R7	0.048	0.0482	-	-	**0.0296**	**0.0306**
Scholl Hard Class							
ins_no	**ins_id**	**AFDO**		**MSBPP**		**IHFPGA**	
		Min.	**Avg.**	**Min.**	**Avg.**	**Min.**	**Avg.**
44	HARD0	0.171	0.1761	-	-	**0.0522**	**0.0533**
45	HARD1	-	-	0.17	0.179	**0.0518**	**0.0529**
46	HARD3	0.171	0.1732	0.17	0.175	**0.0517**	**0.0527**
47	HARD4	0.181	0.1814	-	-	**0.0501**	**0.0748**
48	HARD5	0.174	0.1816	-	-	**0.0405**	**0.0628**
49	HARD7	0.158	0.1655	-	-	**0.0960**	**0.0990**
50	HARD9	-	-	0.17	0.175	**0.0341**	**0.0555**

$$Avg.fitness\ value = \frac{\sum_{i=1}^{10} f_i}{10} \qquad (18)$$

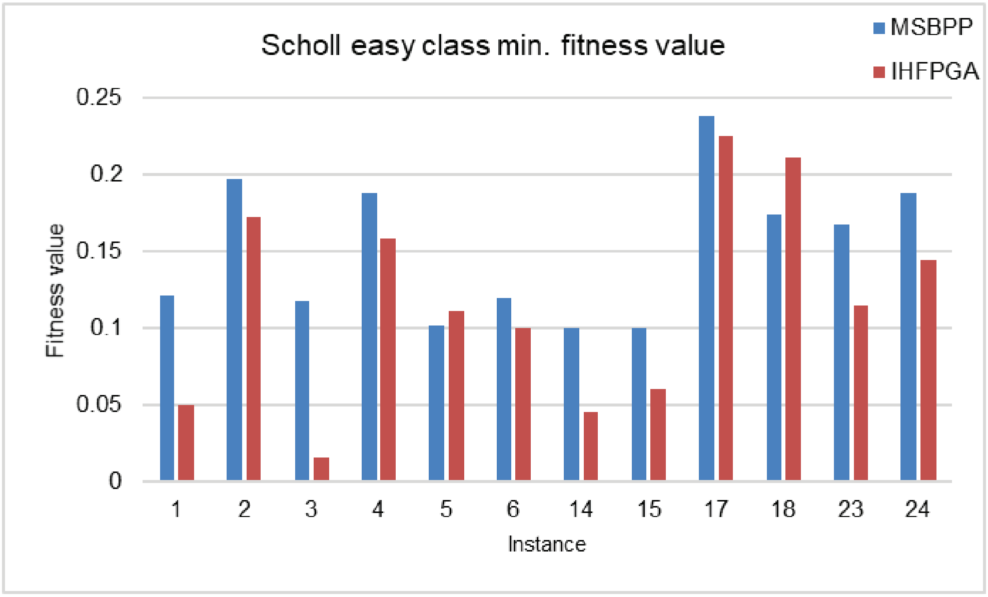
where f_i is the minimum fitness value obtained in the ith run of the proposed algorithm.

When Table 6 is examined, the proposed algorithm achieved better results than its competitors in 44 of 50 samples in terms of minimum fitness value metric. In 6 samples (5, 18, 27, 32, 35, 41), competing algorithms achieved better results. In the average fitness value metric, the proposed algorithm achieved better results in 46 of 50 samples than its competitors. Competitive studies obtained better results in 4 samples (32, 34, 35, 41).

The fact that the studies selected for comparison use different samples of the dataset makes it difficult to compare the information given in Table 6. Therefore, the proposed algorithm has been compared separately with each competing algorithm in Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9. Minimum and average fitness values are given in Figure 4 and Figure 5 for the easy class of the Scholl dataset, in Figure 6 and Figure 7 for the middle class, and in Figure 8 and Figure 9 for the hard class.

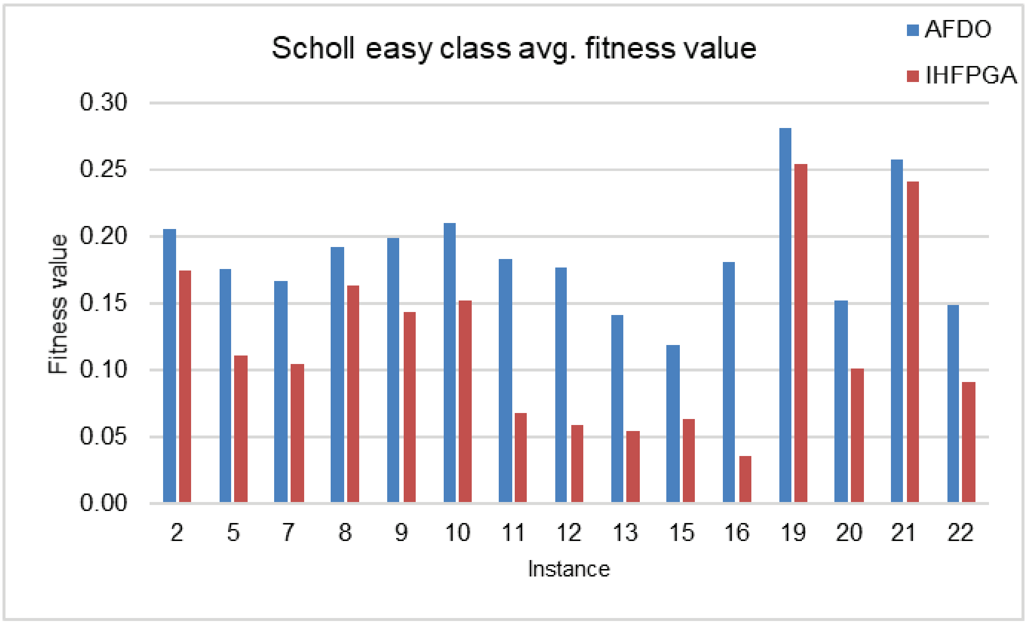


a)

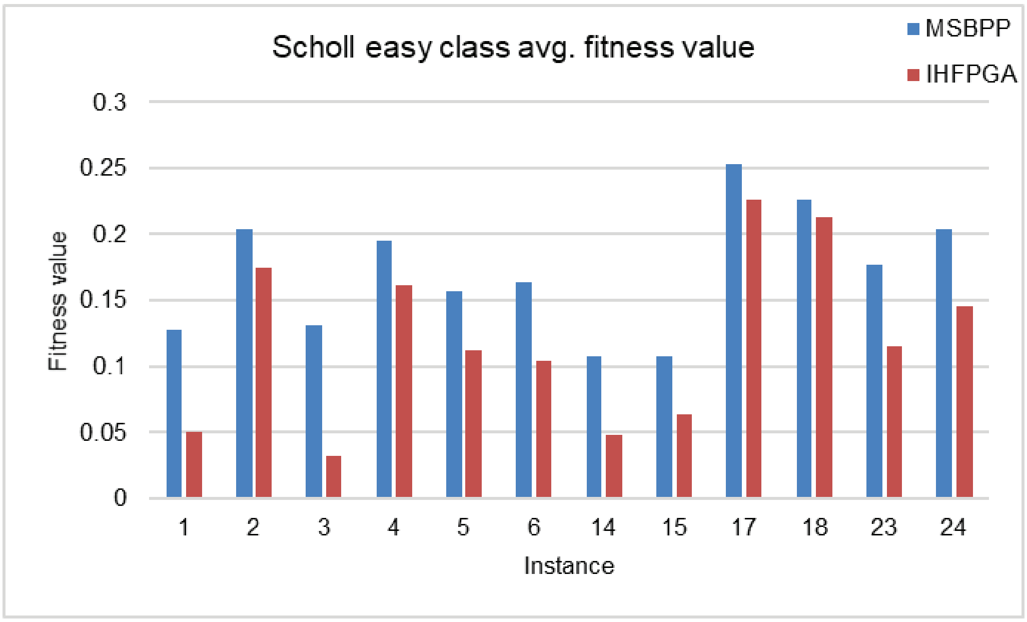


b)

Figure 4. Scholl easy class minimum fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA.



a)



b)

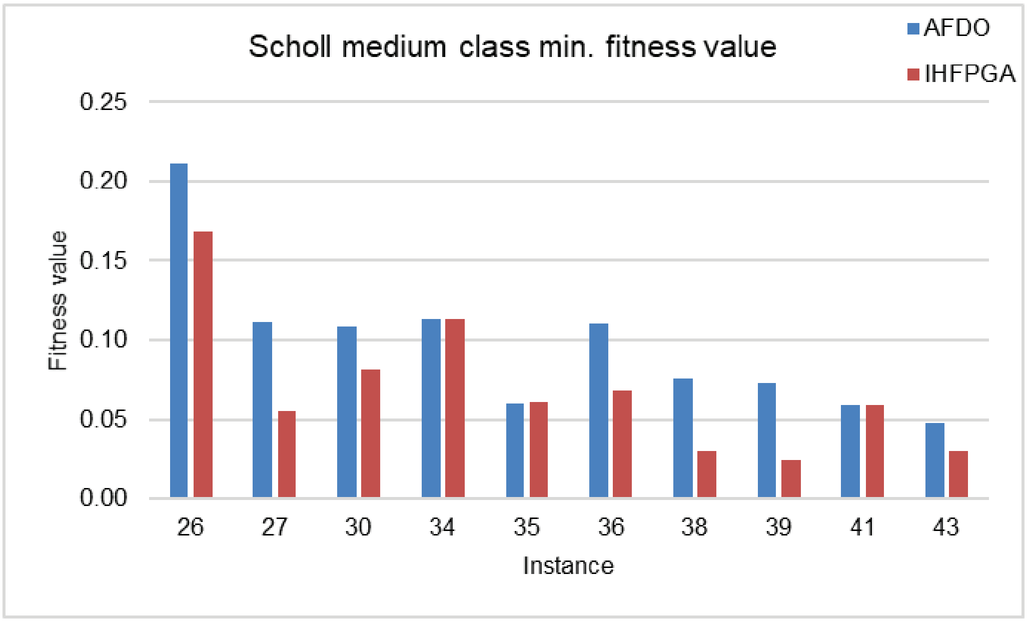
Figure 5. Scholl easy class average fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA.

Other metrics used to compare the proposed algorithm with previous studies are minimum percentage performance (Equation 19) and average percentage performance (Equation 20). The minimum fitness value and the average fitness value are used in the calculation of these metrics.

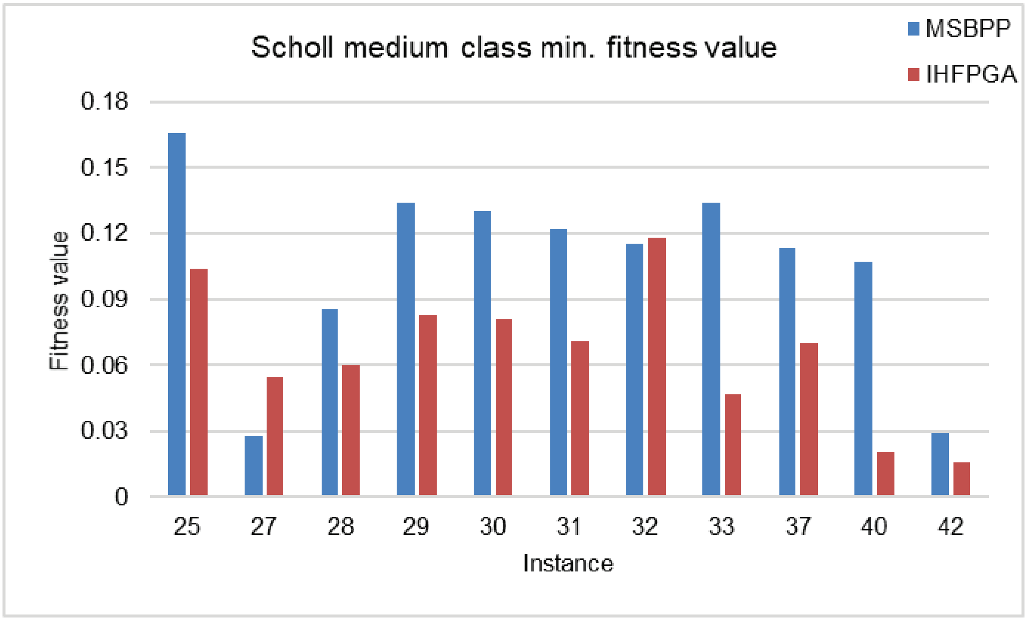
$$Min.\%f = \frac{(Min.f_{AFDO,MSBPP} - Min.f_{IHFPGA})}{Min.f_{AFDO,MSBPP}} * 100 \quad (19)$$

$$Avg.\%f = \frac{(Avg.f_{AFDO,MSBPP} - Avg.f_{IHFPGA})}{Avg.f_{AFDO,MSBPP}} * 100 \quad (20)$$

where, $Min.f_{AFDO,MSBPP,IHFPGA}$ is the minimum fitness value and $Avg.f_{AFDO,MSBPP,IHFPGA}$ is the average fitness value of algorithms. This comparison shows the performance of the proposed algorithm against its competitors as a percentage. Percentage performance values are given in Figure



a)



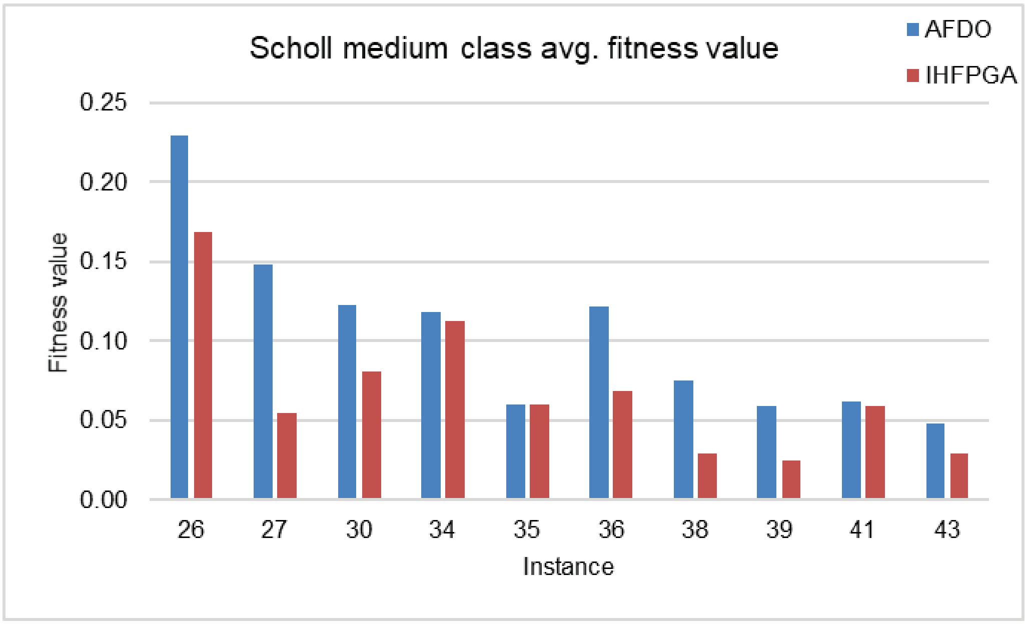
b)

Figure 6. Scholl medium-class minimum fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA.

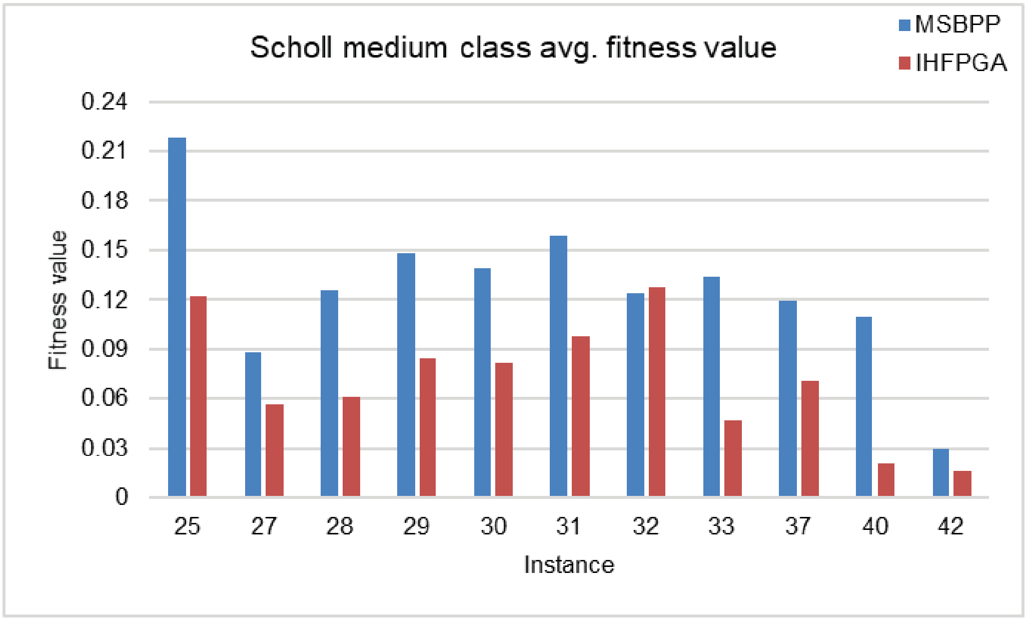
10 for the easy class of the Scholl dataset, in Figure 11 for the medium class, and in Figure 12 for the hard class.

In Figure 10a, the proposed algorithm is compared with the AFDO algorithm. The proposed algorithm performed better than the AFDO algorithm in all examples in the minimum percentage performance and average percentage performance metric. In Figure 10b, the proposed algorithm is compared with the MSBPP algorithm. The proposed algorithm performed better in 10 out of 12 samples in the minimum percentage performance metric, while MSBPP achieved better results in 2 samples (instance 5, 18). In the average percentage performance metric, the proposed algorithm performed better than the MSBPP algorithm in all samples.

In Figure 11a, the proposed algorithm is compared with the AFDO algorithm for medium-class samples. While the proposed algorithm performed better in 8 out of 10 samples in the minimum percentage performance



a)

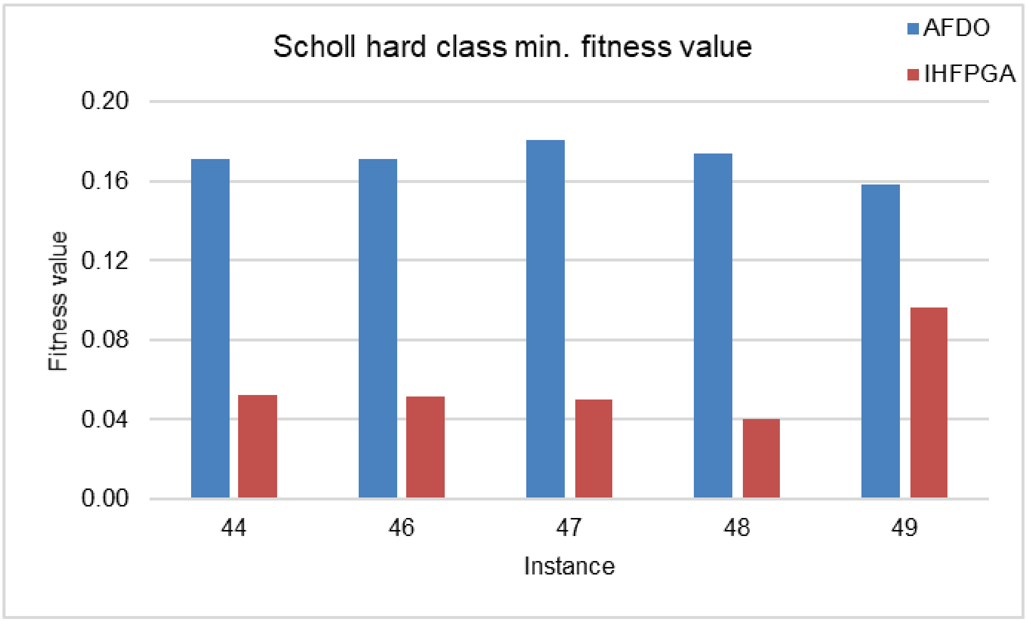


b)

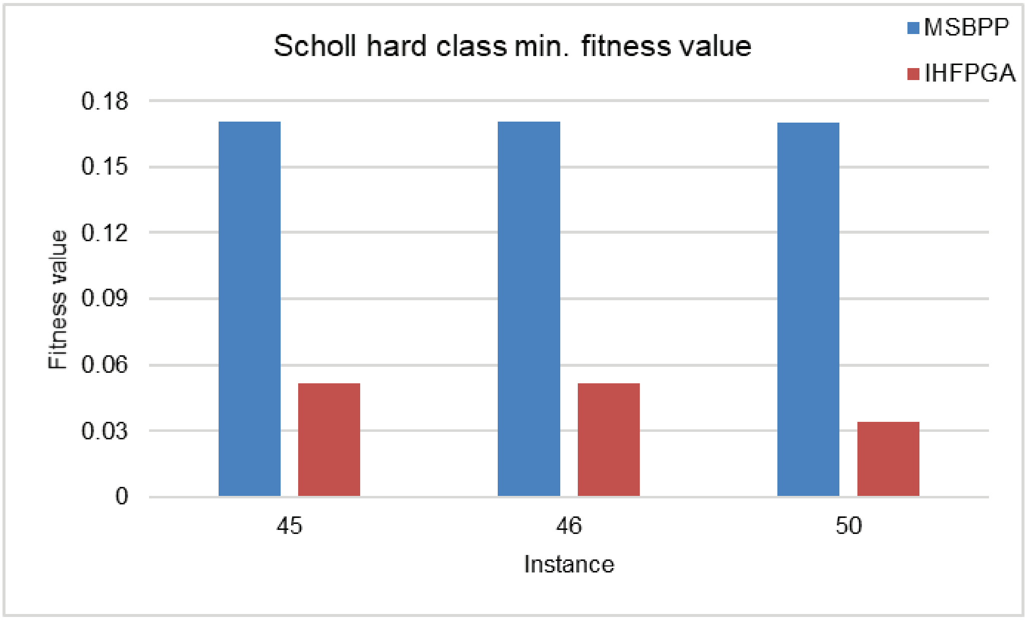
Figure 7. Scholl medium-class average fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA

metric, the AFDO algorithm achieved better results in 2 samples (instance 35,41). In the average percentage performance metric, the proposed algorithm performed better in 7 out of 10 samples, while the AFDO algorithm achieved better results in 3 samples (instance 34, 35, 41). In Figure 11b, the proposed algorithm is compared with the MSBPP algorithm. While the proposed algorithm performed better in 9 out of 11 samples in the minimum percentage performance metric, the MSBPP algorithm achieved better results in 2 samples (instance 27, 32). In the average performance metric, the proposed algorithm performed better in 10 of 11 samples, while the MSBPP algorithm achieved better results in 1 sample (instance 32).

In Figure 12a, the proposed algorithm is compared with the AFDO algorithm for hard class examples. The proposed algorithm gave better results than the AFDO algorithm in all samples in the minimum percentage performance and average percentage performance metric. In Figure 12b, the proposed algorithm is compared with the



a)



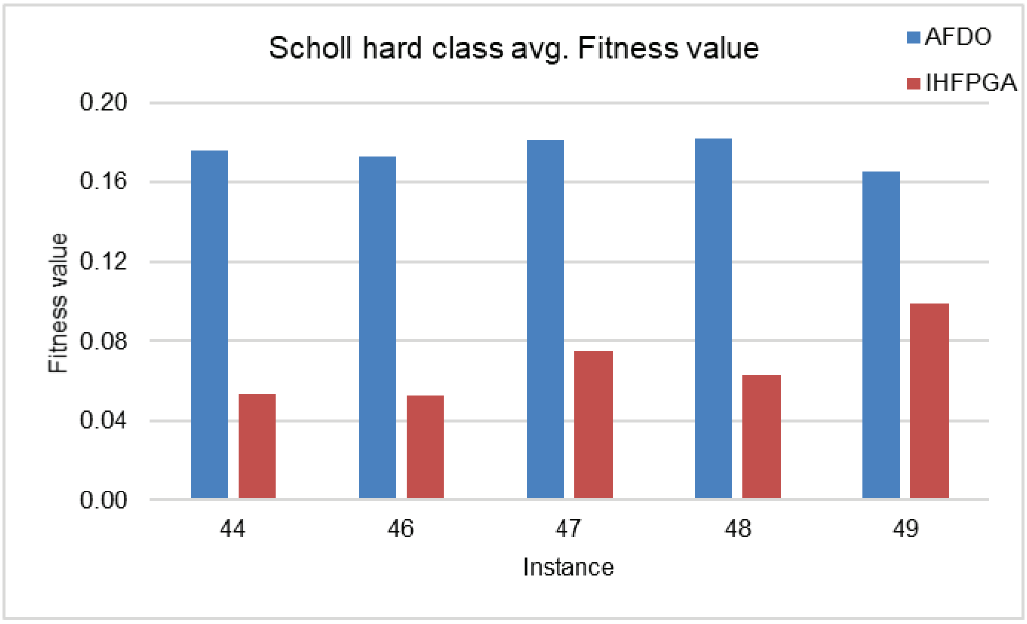
b)

Figure 8. Scholl hard-class minimum fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA.

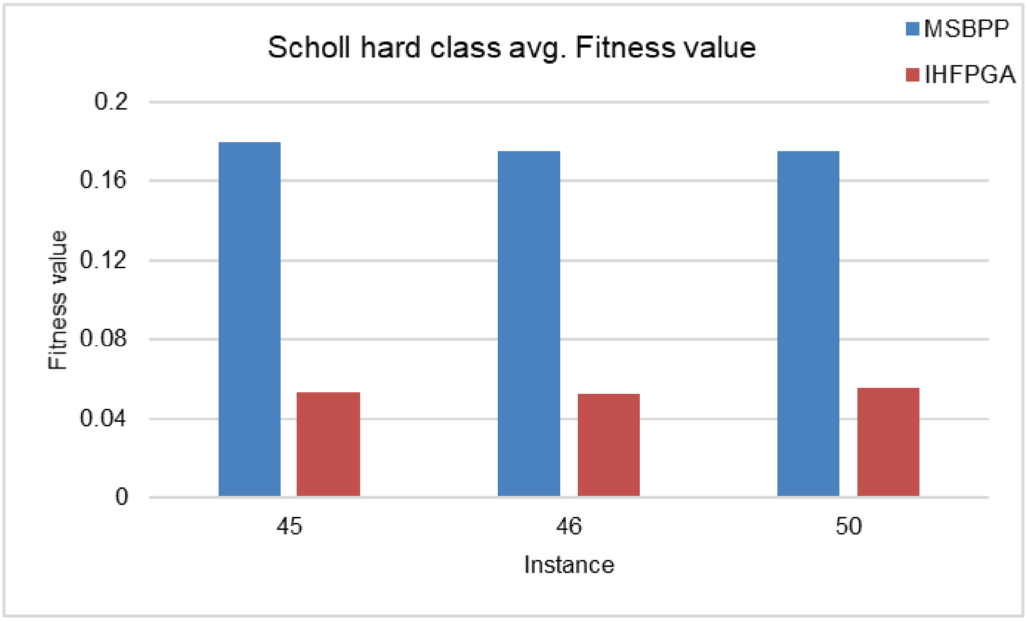
MSBPP algorithm. The proposed algorithm gave better results than the MSBPP algorithm in all samples in the minimum percent performance and average percent performance metric.

CONCLUSION

In this paper, the proposed IHFPGA algorithm is compared with the AFDO, MSBPP, and ILWOA algorithms for the 1DBPP problem. For comparison, metrics including container number, minimum fitness value, average fitness value, minimum percentage performance, and average percentage performance values are used. In terms of the container number metric, the proposed algorithm has achieved equal or better results than competing algorithms in all tests. The proposed algorithm obtained the best-known results in 72 of 77 samples (93.5%) used for comparison. In the minimum fitness metric, the proposed



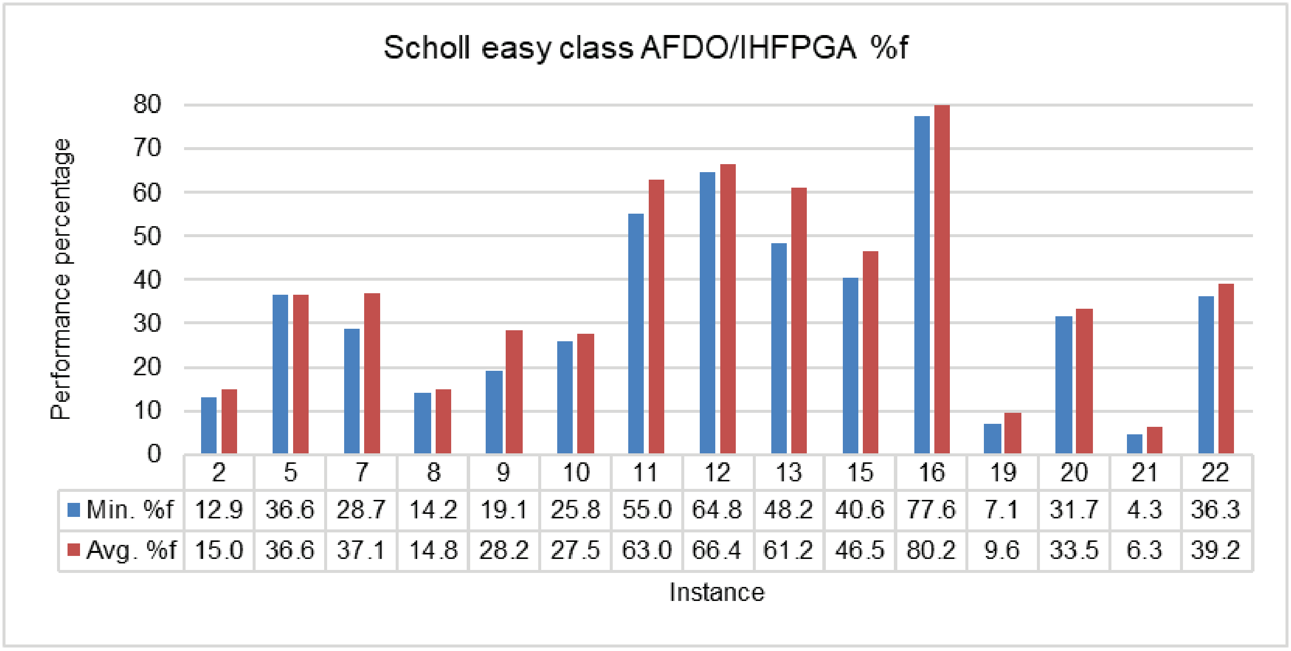
a)



b)

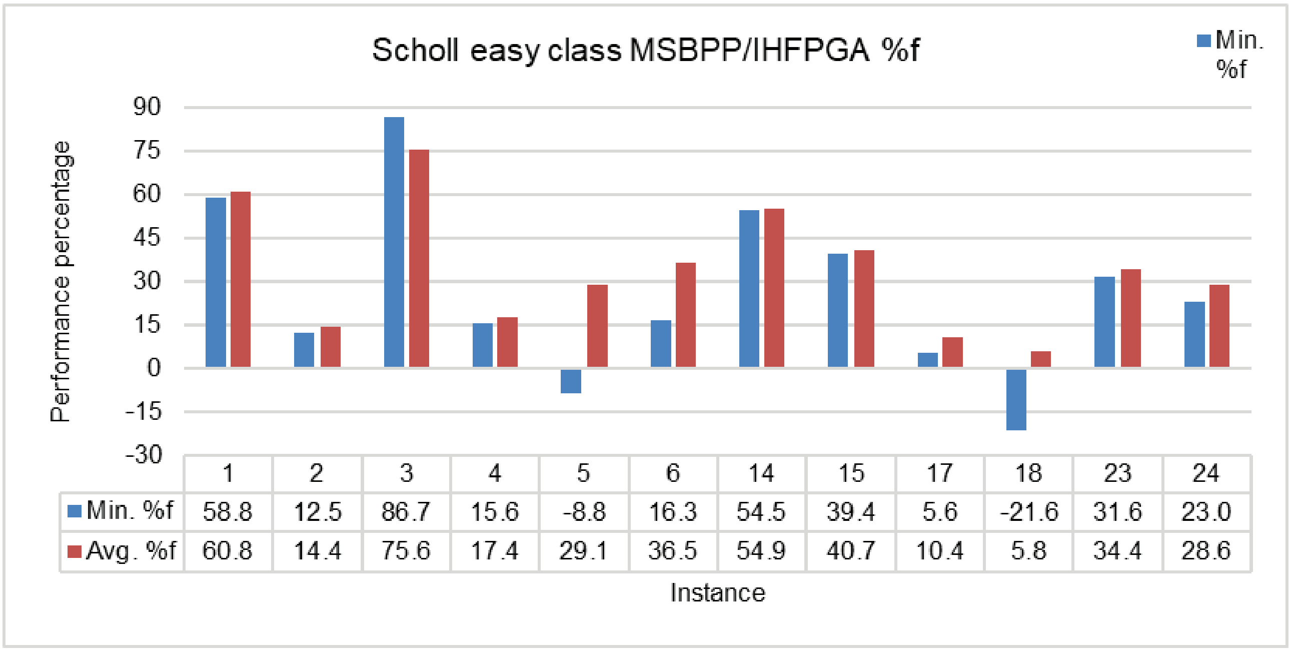
Figure 9. Scholl hard-class average fitness value **a)** AFDO - IHFPGA **b)** MSBPP - IHFPGA.

algorithm achieved more successful results than its competitors in 44 of the 50 samples (88%). In the average fitness metric, it achieved better results than its competitors in 46 of the 50 samples (92%). In the minimum percentage performance metric, it is compared with the AFDO algorithm in 30 samples, and it is more successful in 93.3% of the samples. In general, the proposed algorithm showed an average of 37.6% better performance than the AFDO algorithm. In the comparison made with the MSBPP algorithm, 26 samples are used and the proposed algorithm is more successful in 84.6% of them. In general, the proposed algorithm outperformed the MSBPP algorithm by an average of 32.7%. In the average percentage performance metric, the proposed algorithm is compared with the AFDO algorithm in 30 samples and is more successful in 90% of the samples. In general, the proposed algorithm outperformed the AFDO algorithm by an average of 38.6%. In the comparison made with the MSBPP



	2	5	7	8	9	10	11	12	13	15	16	19	20	21	22
🟦 Min. %f	12.9	36.6	28.7	14.2	19.1	25.8	55.0	64.8	48.2	40.6	77.6	7.1	31.7	4.3	36.3
🟥 Avg. %f	15.0	36.6	37.1	14.8	28.2	27.5	63.0	66.4	61.2	46.5	80.2	9.6	33.5	6.3	39.2

a)



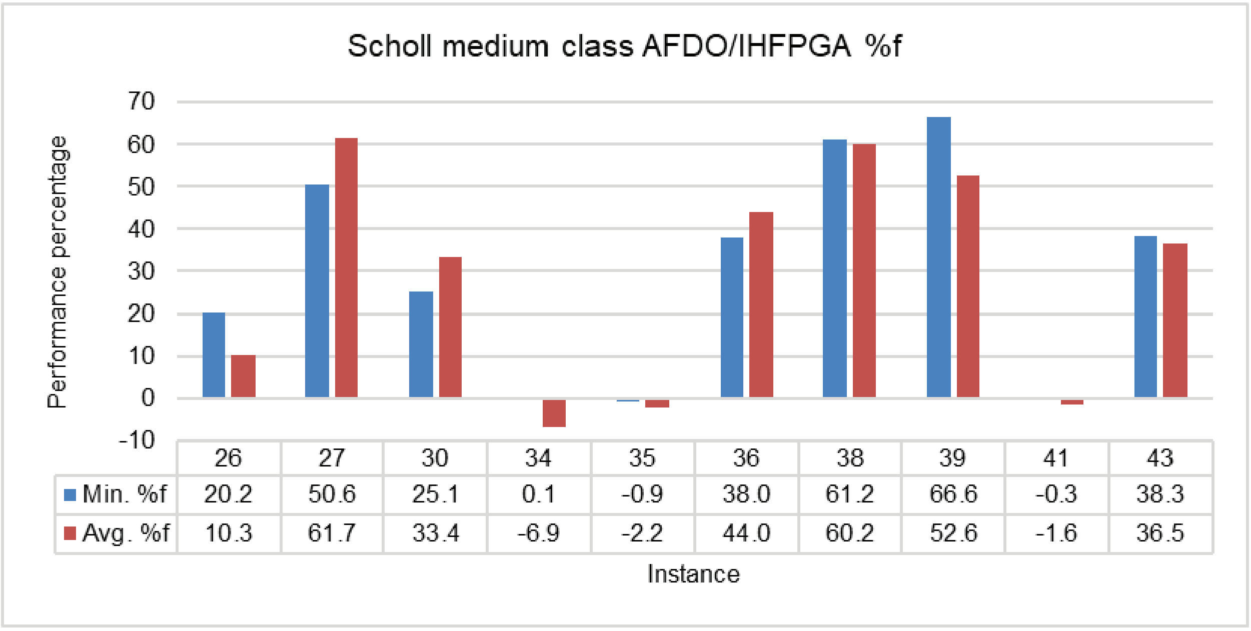
	1	2	3	4	5	6	14	15	17	18	23	24
🟦 Min. %f	58.8	12.5	86.7	15.6	-8.8	16.3	54.5	39.4	5.6	-21.6	31.6	23.0
🟥 Avg. %f	60.8	14.4	75.6	17.4	29.1	36.5	54.9	40.7	10.4	5.8	34.4	28.6

b)

Figure 10. Performance percentage of IHFPGA in terms of the fitness value (Scholl easy class) **a)** AFDO/IHFPGA **b)** MSBPP/IHFPGA.

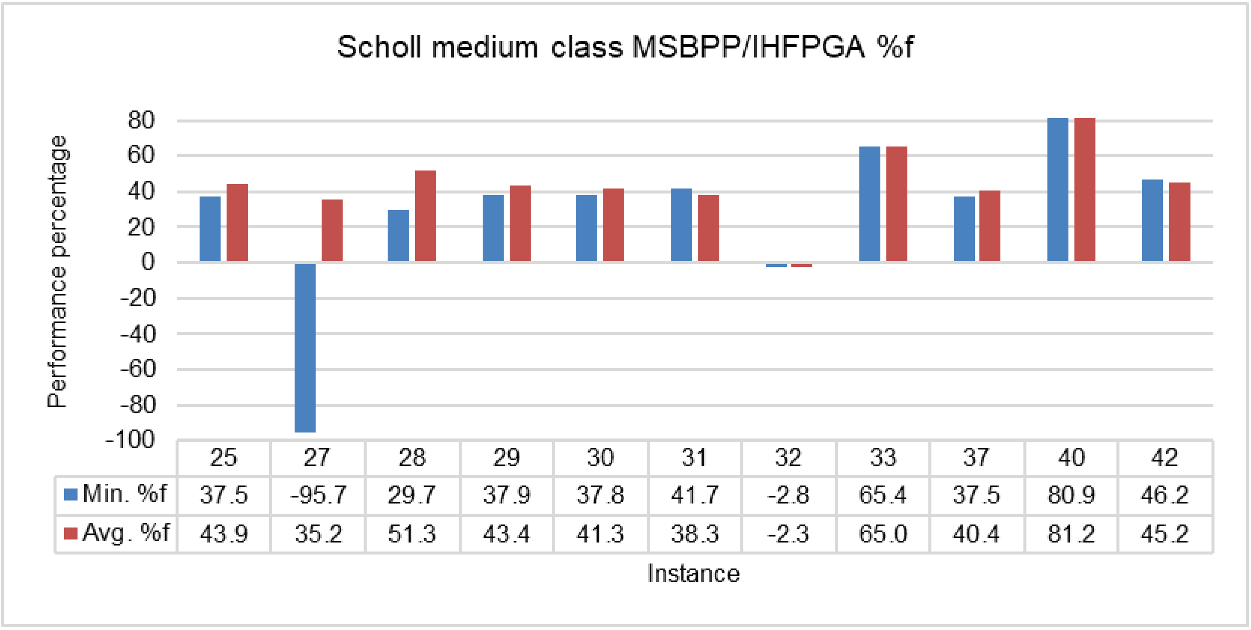
algorithm, 26 samples are used and the proposed algorithm is more successful in 96.1% of the samples. In general, the proposed algorithm outperformed the MSBPP algorithm on average 42.3%.

These data reveal the superiority of the proposed algorithm over the competing ones. Thanks to the elimination phase, the proposed algorithm achieves better results on average fitness and average percentage performance parameters. This strategy ensures that candidate solutions are close to each other. Besides, the results of the algorithm are reproducible. The proposed algorithm will be compared with more algorithms in the future. It will also be applied to two-dimensional and three-dimensional BBP.



	26	27	30	34	35	36	38	39	41	43
🟦 Min. %f	20.2	50.6	25.1	0.1	-0.9	38.0	61.2	66.6	-0.3	38.3
🟥 Avg. %f	10.3	61.7	33.4	-6.9	-2.2	44.0	60.2	52.6	-1.6	36.5

a)



	25	27	28	29	30	31	32	33	37	40	42
🟦 Min. %f	37.5	-95.7	29.7	37.9	37.8	41.7	-2.8	65.4	37.5	80.9	46.2
🟥 Avg. %f	43.9	35.2	51.3	43.4	41.3	38.3	-2.3	65.0	40.4	81.2	45.2

b)

Figure 11. Performance percentage of IHFPGA in terms of the fitness value (Scholl medium class) **a)** AFDO/IHFPGA **b)** MSBPP/IHFPGA.

AUTHORSHIP CONTRIBUTIONS

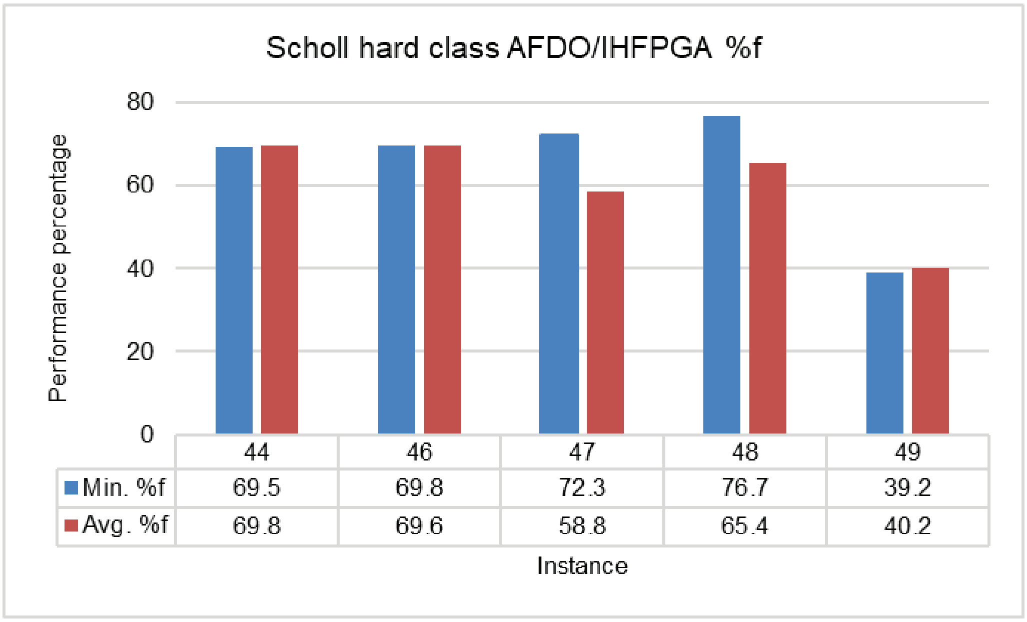
Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

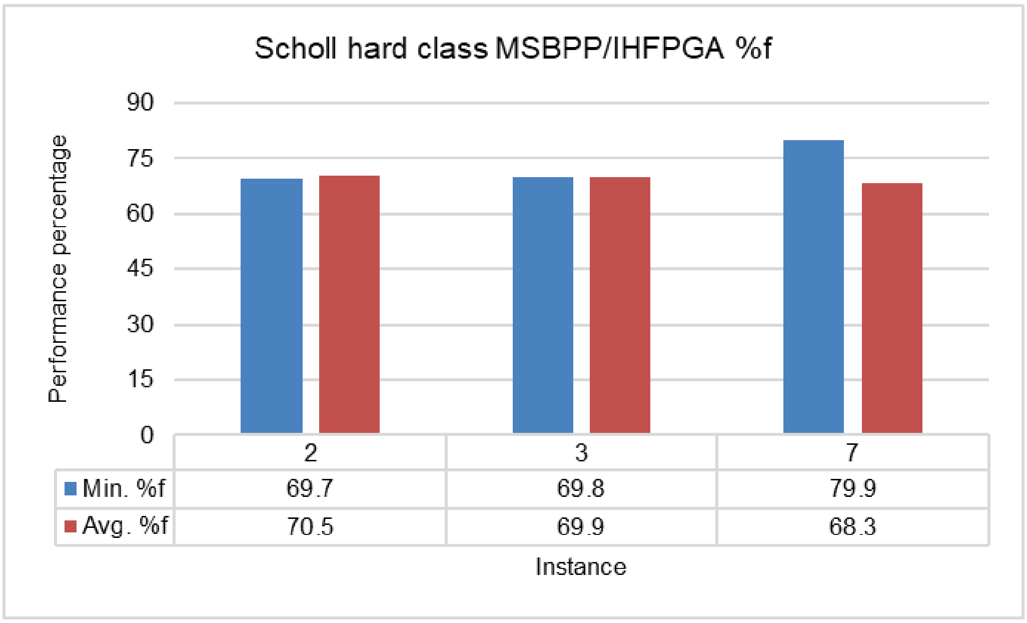
CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.



	44	46	47	48	49
🟦 Min. %f	69.5	69.8	72.3	76.7	39.2
🟥 Avg. %f	69.8	69.6	58.8	65.4	40.2

a)



	2	3	7
🟦 Min. %f	69.7	69.8	79.9
🟥 Avg. %f	70.5	69.9	68.3

b)

Figure 12. Performance percentage of IHFPGA in terms of the fitness value (Scholl hard class) **a)** AFDO/IHFPGA **b)** MSBPP/IHFPGA.

ETHICS

There are no ethical issues with the publication of this manuscript.

REFERENCES

[1] Delorme M, Iori M, Martello S. BPPLIB: a library for bin packing and cutting stock problems. Optim Lett 2018;12:235–250. [CrossRef]

[2] Delorme M, Iori M, Martello S. Bin packing and cutting stock problems: Mathematical models and exact algorithms. Eur J Oper Res 2016;255:1–20. [CrossRef]

[3] Kucukyilmaz T, Kiziloz HE. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. Comput Ind Eng 2018;125:157–170. [CrossRef]

[4] El-Ashmawi WH, Elminaam DSA. A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem. Appl Soft Comput 2019;82:105565. [CrossRef]
[5] López-Camacho E, Terashima-Marín H, Ochoa G, Conant-Pablos SE. Understanding the structure of bin packing problems through principal component analysis. Int J Prod Econ 2013;145:488−499. [CrossRef]
[6] Abdel-Basset M, Manogaran G, Abdel-Fatah L, Mirjalili S. An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. Pers Ubiquitous Comput 2018;22:1117−1132. [CrossRef]
[7] Aliano Filho A, Moretti AC, Pato MV. A comparative study of exact methods for the bi-objective integer one-dimensional cutting stock problem. J Oper Res Soc 2018;69:91−107. [CrossRef]
[8] Romanycia MHJ, Pelletier FJ. What is a heuristic? Comput Intell 1985;1:47−58. [CrossRef]
[9] Blum C, Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv 2003;35:268−308. [CrossRef]
[10] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13:533−549. [CrossRef]
[11] Wang S, Shi R, Wang L, Ge M. Study on improved ant colony optimization for bin-packing problem. In: Wang J, editor. 2010 International Conference on Computer Design and Applications, ICCDA 2010; 2010 Jun 25-27; Qinhuangdao, China: IEEE; 2010.
[12] Liu DS, Tan KC, Goh CK, Ho WK. On solving multiobjective bin packing problems using particle swarm optimization, 2006 IEEE Congress on Evolutionary Computation, CEC 2006; 2006 Jul 16-21; Vancouver, Canada: IEEE; 2006. pp. 2095−2102.
[13] Falkenauer E. A hybrid grouping genetic algorithm for bin packing. J Heuristics 1996;2:5−30. [CrossRef]
[14] Luo F, Scherson ID, Fuentes J. A novel genetic algorithm for bin packing problem in jMetal. In: Howard N, editor. Proceedings - 2017 IEEE 1st International Conference on Cognitive Computing, ICCC 2017; 2017 Jun 25-30; Honolulu, USA: IEEE; 2017. pp. 17−23. [CrossRef]
[15] Layeb A, Chenche S. A novel GRASP algorithm for solving the bin packing problem. Int J Inf Eng Electron Bus 2012;4:8. [CrossRef]
[16] Wolpert DH, Macready WG. No free lunch theorems for optimization, IEEE Trans Evol Comput 1997;1:67−82. [CrossRef]
[17] Ho YC, Pepyne DL. Simple explanation of the no free lunch theorem of optimization. Cybern Syst Anal 2002;38:292−298. [CrossRef]
[18] Layeb A, Boussalia SR. A novel quantum ınspired cuckoo search algorithm for bin packing problem. Int J Inf Technol Comput Sci 2012;4:58−67. [CrossRef]
[19] Yang XS, Deb S. Cuckoo search via Lévy flights. In: Abraham A, editor. 2009 World Congress on Nature & Biologically Inspired Computing; 2009 Dec 9-11; Coimbatore, India: IEEE; 2009. pp. 210−214. [CrossRef]
[20] Zendaoui Z, Layeb A. Adaptive cuckoo search algorithm for the bin packing problem, modelling and ımplementation of complex systems. In: Chikhi S, Amine A, Chaoui A, Kholladi MK, Saidouni DE, editors. Proceedings of the 4th International Symposium, MISC 2016; 2016 May 7-8; Constantine, Algeria: IEEE; 2016. pp. 107−120.
[21] Qian B, Wang L, Hu R, Wang WL, Huang DX, Wang X. A hybrid differential evolution method for permutation flow-shop scheduling. Int J Adv Manuf Technol 2008;38:757−777. [CrossRef]
[22] Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res 2006;44:4737−4754. [CrossRef]
[23] Liang JJ, Pan QK, Tiejun C, Wang L. Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int J Adv Manuf Technol 2011;55:755−762. [CrossRef]
[24] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. Eur J Oper Res 2007;183:1109−1130. [CrossRef]
[25] Thomas J, Chaudhari NS. Design of efficient packing system using genetic algorithm based on hyper heuristic approach. Adv Eng Softw 2014;73:45−52. [CrossRef]
[26] Stawowy A. Evolutionary based heuristic for bin packing problem. Comput Ind Eng 2008;55:465−474. [CrossRef]
[27] Stakic D, Anokic A, Jovanovic R. Comparison of different grasp algorithms for the heterogeneous vector bin packing problem. 2019International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing, AIAIM 2019; Doha, Qatar: IEEE; 2019. pp. 63−70. [CrossRef]
[28] Kramer R, Dell'Amico M, Iori M. A batching-move iterated local search algorithm for the bin packing problem with generalized precedence constraints. Int J Prod Res 2017;55:6288−6304. [CrossRef]
[29] Zehmakan AN. Bin packing problem: Two approximation algorithms. Int J Found Comput Sci Technol 2015;5:1−10. [CrossRef]
[30] Loh KH, Golden B, Wasil E. Solving the one-dimensional bin packing problem with a weight annealing heuristic. Comput Oper Res 2008;35:2283−2291. [CrossRef]
[31] Buljubašić M, Vasquez M. Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. Comput Oper Res 2016;76:12−21. [CrossRef]

[32] Quiroz-Castellanos M, Cruz-Reyes L, Torres-Jimenez J, Gómez SC, Huacuja HJF, Alvim ACF. A grouping genetic algorithm with controlled gene transmission for the bin packing problem, Comput Oper Res 2015;55:52–64. [CrossRef]

[33] Abdul-Minaam DS, Al-Mutairi WMES, Awad MA, El-Ashmawi WH. An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem. IEEE Access 2020;8:97959–97974. [CrossRef]

[34] Gherboudj A. African Bufalo optimization for one dimensional bin packing problem. Int J Swarm Intell Res 2019;10:38–52. [CrossRef]

[35] Munien C, Mahabeer S, Dzitiro E, Singh S, Zungu S, Ezugwu AE. Metaheuristic approaches for one-dimensional bin packing problem: A comparative performance study. IEEE Access 2020;8. [CrossRef]

[36] Yang XS. Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N, editors. UCNC 2012: Unconventional Computation and Natural Computation; 2012 Sept 3-7; Orleans, France: Springer; 2012. pp. 240–249. [CrossRef]

[37] Yang XS, Karamanoglu M, He X. Flower pollination algorithm: A novel approach for multiobjective optimization. Eng Optim 2014;46:1222–1237. [CrossRef]

[38] Banzhaf W. The "molecular" traveling salesman. Biol Cybern 1990;64:7–14. [CrossRef]

[39] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs, Genetic Algorithms + Data Structures = Evolution Programs. 1st ed. Berlin: Springer; 1996. [CrossRef]