IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2020, accepted June 19, 2020, date of publication June 23, 2020, date of current version July 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004380

An Effective Discrete Grey Wolf Optimization Algorithm for Solving the Packing Problem

PENG WANG, YUNQING RAO, AND QIANG LUO

State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Peng Wang (huogewang@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51975231, in part by the Foundational Research Funds for the Central Universities under Grant 2019kfyXKJC043, and in part by the Intelligent Manufacturing Foundation of Ministry of Industry and Information Technology of China under Grant [2017] 468.

ABSTRACT This article proposes a novel discrete grey wolf optimization for the packing problem, called the two-dimensional strip packing (2DSP) problem without guillotine constraint. The 2DSP involves cutting pieces from a stock sheet with the objective of minimizing waste. To solve the 2DSP problem by the discrete grey wolf algorithm, many strategies are originally proposed. The searching and attacking operators in the algorithm are redesigned to guarantee coding effectiveness. A novel approach to measure the distance between the wolves is presented. In addition, an improved best-fit strategy is developed to solve this packing problem. The best-fit strategy divides the situation into five cases based on the width and length of the rectangle. Computational results on widely used benchmark instances show that the novel discrete grey wolf algorithm can solve the 2DSP problem effectively, and surpasses most of the previously reported meta-heuristic algorithms.

INDEX TERMS Packing problem, discrete grey wolf optimization, swarm intelligence, meta-heuristic algorithm.

I. INTRODUCTION

Globally, iron and steel production cause approximately 6% of anthropogenic CO_2 emissions each year, and one ton of steel manufacturing emits approximately 1.8 t of CO_2 [1]. Reducing material waste contributes to the fight against climate change.

Specifically, the 2DSP problem is to orthogonally pack a given set of rectangles without overlap into a large rectangular strip of fixed width and infinite height. The objective is to minimize the height of the strip. Under the improved typology of cutting and packing problems by Wäscher *et al.* [2], the 2DSP is a two-dimensional open dimension problem. Furthermore, it can be divided into four subtypes based on the following two constraints: (a) whether the items are allowed to rotate by 90°, and (b) guillotine cut. In this article, the solved subtype is that the rotation of the items is not allowed and the guillotine cut is not required.

Various approaches have been proposed to solve the 2DSP problem. Three main types are summarized from the published literature: exact algorithms, heuristic algorithms, and meta-heuristic algorithms.

Some exact algorithms have been presented over the past years by Beasley [3], Martello *et al.* [4], Hifi and M'Hallah [5], Cui *et al.* [6], and Kenmochi *et al.* [7]. Exact algorithms can obtain optimal solutions in an acceptable time when dealing with the small size problem. However, if the size of the problem is large, exact algorithms might not be practical methods because the exact algorithms require a great deal of running time. Thus, most researchers have focused on the method that provides an approximately optimal solution in a reasonable time.

The advantage of the heuristic algorithm is that it can obtain good approximate solutions in an acceptable time for large scale problems. The well-known algorithm is the bottom-left (BL) method presented by Baker *et al.* [8]. The key of the method is that each rectangle is sequentially packed into the sheet by putting it downward and then leftward as much as possible. Chazelle [9] proposed the bottom-left fill (BLF) method to further improve the quality of the solution and generalized the concept of BL. Another study conducted by Hopper and Turton [10] found that the BL heuristic algorithm with different sequences of the rectangle

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

obtained different results. Hence, they proposed the bottom-left-decreasing (BLD) which can find the best result.

These BL methods select the rectangle that is packed first and then find the best position on the sheet. In contrast, Burke *et al.* [11] proposed a heuristic algorithm called best fit (BF). This algorithm selects the lowest available "gap" first and then seeks the best-fit rectangle under priority rules. Aşık and Özcan [12] extended BF by introducing a novel bidirectional best-fit heuristic. Leung and Zhang [13] proposed a simple scoring rule to select a best-fit rectangle instead of priority rules. Verstichel *et al.* [14] added new item orderings and item placement strategies to the BF. The experiments indicated that it could obtain further improved solutions compared to the original BF.

The published literature shows that the solution can be further improved if a meta-heuristic algorithm is hybridized with a heuristic placement algorithm. Hopper and Turton [15] attempted to employ a genetic algorithm (GA) hybridized with BL to solve the 2D industrial packing problem. Bortfeldt [16] suggested a different GA that works without any encoding of solutions, instead of using encoded solutions and standard operators. Ant colony optimization (ACO) was adopted by Yuan and Liu [17] to deal with 2DSP. Leung *et al.* [18] also presented a hybrid SA meta-heuristic algorithm Omar and Ramakrishnan [19] proposed an evolutionary particle swarm optimization algorithm (EPSO) for solving the cutting and packing problem.

The best-fit heuristic is a simple and powerful approach for the 2DSP problem. Furthermore, the meta-heuristic algorithm hybridizes with BF can improve the quality of solutions to some degree. Burke *et al.* [20] studied a novel simulated annealing (SA) algorithm together with BF and compared the SA with other approaches in terms of execution times and the quality of the solutions. Recently, Leung *et al.* [21] proposed a two-stage intelligent search algorithm (ISA). In the first stage, a BF heuristic based on a simple scoring rule was used. In the second stage, a local search algorithm was implemented to further improve the quality of solutions. Yang *et al.* [22] introduced the least waste priority strategy and the improved scoring rule, and then presented a randomized algorithm (SRA). Wei *et al.* [23] proposed an efficient intelligent algorithm (IA), which involves three stages named greedy selection, local improvement, and randomized improvement. Another recent study conducted by Wei *et al.* [24] presented an improved skyline based best-fit heuristic (ISH) and a random local search algorithm. These algorithms adopt scoring rule based BF (SR-BF) which is more precise than the original best-fit approach proposed by Burke *et al.* [11].

For the 2DSP problem, the heuristic algorithm and the meta-heuristic algorithm are more practical than the exact algorithms. Compared to heuristic algorithms, the meta-heuristic algorithm can improve the solution of the 2DSP problem to some degree. To enhance the material utilization, the research trend of the 2DSP problem is to improve the current algorithms and implement an efficient meta-heuristic algorithm. In this article, a novel algorithm—discrete grey wolf optimization (DGWO) is proposed to solve the 2DSP problem.

There are several contributions in this article. First, a novel algorithm DGWO is proposed in terms of the characteristics of the solved problem and implemented in 2DSP. Second, some experiments are performed to determine the appropriate parameter values for the DGWO, and the proposed algorithm is tested on all existing benchmark instances. Third, a simple improvement for the scoring rule is proposed, which is much simpler and easier to implement compared with the previous best-fit strategy. Finally, from the computational results in all instances, the DGWO algorithm has a better or similar performance compared to most of the meta-heuristic algorithms, which verifies the effectiveness of the proposed method.

The organization of this article is arranged as follows. The details of the improved scoring rule are given in Section II. In Section III, the grey wolf optimization (GWO) is concisely introduced first, and then the DGWO algorithm is presented in detail. The computational experiments and results are reported in Section IV. Finally, Section V gives the conclusions.

II. AN IMPROVED BEST-FIT HEURISTIC ALGORITHM

All of the BF heuristic algorithms have the same framework as presented by Burke *et al.* [11]. In this article, the horizontal line (HL) is used to represent the packing pattern. Given a sequence of rectangles, the BF method repeats the following steps until all rectangles are packed: first, find the lowest and left-most HL; second, search for rectangle by employing the best-fit rules; third, place the best-fit rectangle into the selected HL; finally, update the horizontal lines.

A. BF HEURISTIC BASED ON THE HORIZONTAL LINE

In this section, HL is introduced first. It is assumed that some rectangles have packed into the sheet as shown in Figure 1. The current packing pattern is represented by the five horizontal lines (S_1, S_2, S_3, S_4, S_5). The sequence numbers of horizontal lines have two properties ($j = 1, 2, 3, \ldots, k$): 1) the y-coordinate of S_j is not equal to the y-coordinate of S_{j+1}, and 2) the x-coordinate of the right endpoint of S_j is the same as the x-coordinate of the left endpoint of S_{j+1}.

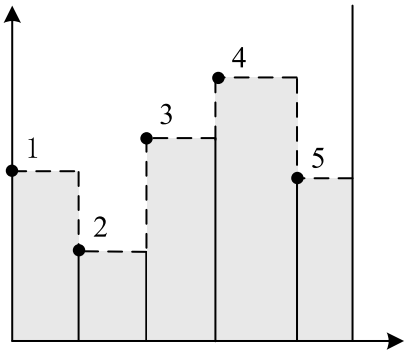


FIGURE 1. Example of the horizontal line.

In the procedure, each line segment S_j is described by the following characteristics: 1) x, y—the coordinates of the left endpoint of S_j, and 2) w—the width of the S_j. The sequence of HL is sorted by the value of the x-coordinate of the line segment. Figure 1 shows a packing sheet. The next step in the BF heuristic algorithm is to determine the lowest and left-most HL and then select a best-fit rectangle to place. Two results can be expected when searching for the best-fit rectangle. Either there are rectangles that fit the bottom-left segment (S_2 in the example), or there are not rectangles. If no available rectangles are placed on the selected segment, the operation for HL is to raise segment S_2 to align with its lower adjacent segment S_1 and merge them, as shown in Figure 2(a). The shaded area is regarded as the wasted material.

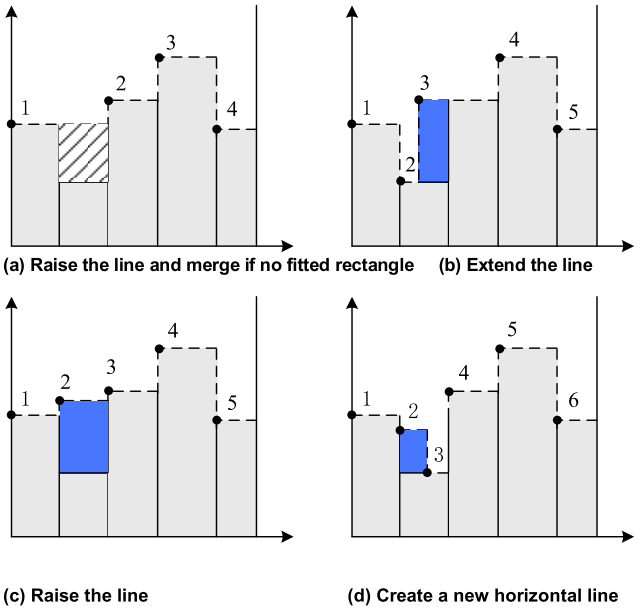


FIGURE 2. The operation for horizontal line.

If the best-fit rectangle can be found, there are three cases for the operation for HL in terms of the dimensions of the rectangle. If the height of the placed rectangle is equal to the difference between the y-coordinate of the selected line and its adjacent segment (S_1 or S_3 in the example), shortening the selected segment S_2 is shortened, and its adjacent segment S_3 is extending; Figure 2(b) shows the result. If the width of the placed rectangle is equal to the width of the selected segment only, the y-coordinate of segment S_2 is updated, as shown in Figure 2(c). The last case, similar to Figure 2(d), is required to create a new HL and insert it into the sequence of the HL.

The procedure of the BF heuristic is given as Algorithm 1, called the BestFitHeuristic. The input is the sequence of rectangles *Seq* and the width of the strip W. After initializing the horizontal line, the heuristic always selects the bottom-left segment in the sheet and then searches and places the best-fit rectangle under scoring rules. The operation for the HL is different according to the dimensions of the selected rectangle. These steps are repeated until all the rectangles are packed. The BestFitHeuristic constructs solutions by placing the rectangle one by one and finally returns the material utilization and the height of the strip used.

Algorithm 1 BestFitHeuristic Procedure

```
BestFitHeuristic(Seq, W)
1  Initialize the horizontal line
2  While all rectangles aren't packed
3      Select the lowest and left-most horizontal line S
4      Search for the best-fit rectangle r from Seq that
       can be placed at S
5      If can't find such an r
6       Raise the line and merge
7      else
8       Place r at S and update the horizontal lines
9  return the utilization of material and the height of the
used strip
```

B. IMPROVED SCORING RULES

The major difference among all the BF heuristics is the strategy for selecting the best-fit rectangle. The original BF proposed by Burke *et al.* [11] chooses the rectangle to fit the gap by the largest width. Later, a new scoring rule based on BF was proposed by Leung and Zhang [13], Yang *et al.* [22], and Wei *et al.* [23]. Adopting the scoring rule can obtain a better solution than the original BF. However, the complex scoring rule increases the time complexity of the BF heuristics. This article proposes a simplified scoring rule and only five cases are classified.

Assuming that the lowest and left-most segment is S, some cases are derived by three factors: dimensions of the rectangle, the width of S, and the height of the adjacent segments. After analyzing all possible fitting situations, four cases are initially sorted by a fitness score. The fitness score is the number of sides of the rectangle that are matched with the segments it is touching. Once a selected rectangle is placed at S, the left, right, and bottom sides of the rectangle might touch the other three segments. Thus, the fitness score can be 0, 1, 2, or 3 as shown in Figure 3. In particular, there are no sides of the rectangle that are matched with its touching segment, and the fitness score should be 0, but the quality of the solution will be better if the taller rectangle is placed first. Thus, this article divides this situation into two cases as shown in Figure 3(g) and 3(h), so five cases are classified in total.

The proposed heuristic always chooses the rectangle with the maximum score, and the first hit unpacked rectangle is selected if it has the same score. When the bottom-left segment S is determined, then the BestFitHeuristic searches the whole sequence of unpacked rectangles one after another.

III. DISCRETE GREY WOLF OPTIMIZER

The grey wolf optimization (GWO) was originally proposed by Mirjalili *et al.* [25]. After being reported, this algorithm has attracted a good deal of attention due to its intrinsic

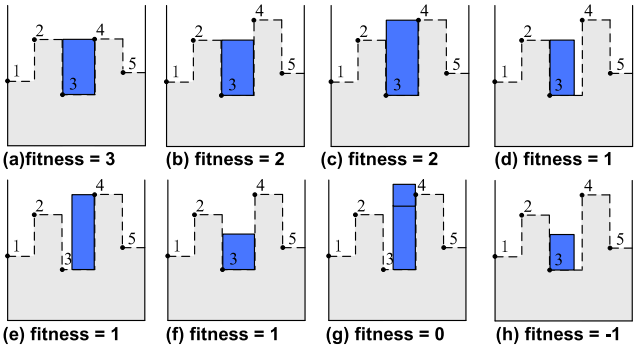


FIGURE 3. The score of fitness.

advantages, and has been applied to different fields, such as shop scheduling, and parameter optimization.

Emary *et al.* [26] proposed binary grey wolf optimization to solve the problem of feature selection in machine learning. The GWO was also applied to optimize unmanned air vehicle path planning by Radmanesh and Kumar [27]. As noted by Komaki and Kayvanfar [28], the two-stage assembly flow shop scheduling problem can be addressed successfully by the GWO, which outperforms some other well-known meta-heuristic algorithms. In the optics domain, Chaman-Motlagh [29] formulated the optical filter design problem as an optimization problem and solved it by the GWO. Additionally, Sultana *et al.* [30] and Sulaiman *et al.* [31] employed the GWO to optimize the reactive power dispatch problem and the multiple distributed generation allocation problems in the distribution system. Kaleka *et al.* [32] presented a conceptual comparison among spotted hyena optimizer, grey wolf optimizer, particle swarm optimization, ant colony optimization, gravitational search algorithm, bat algorithm, moth flame optimization, and whale optimization algorithm. The experimental results show that the grey wolf optimizer gives optimal solutions compared to the other algorithms. To further improve the performance of GWO, Kumar and Kumar [33] developed three novel strategies and applied them well to seven well-known constrained engineering design problems. There were other variants of the GWO including discrete, multi-objective. Chen *et al.* [34] explored the GWO application to the disassembly sequencing problem and proposed a discrete sequencing GWO algorithm. Qin *et al.* [35] proposed an effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint. These works proved that the GWO is an available algorithm for solving practical problems, and verified its performance of exploration, exploitation, local optima avoidance, and convergence.

In this section, the framework of the GWO is introduced first. Then, the DGWO algorithm for 2DSP is provided in detail.

A. THE FRAMEWORK OF THE GWO

The GWO mimics the social hierarchy and hunting behavior of grey wolves. The most intriguing thing is that they have a very strict social hierarchy as shown in Figure 4.

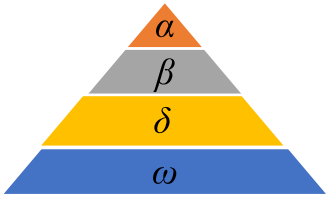


FIGURE 4. Hierarchy of the grey wolf (dominance decreases from the top-down).

The grey wolves have four levels, called alpha, beta, delta, and omega in proper order. The lower level has to obey the dominant wolves. The GWO considers the best, second-best, and third-best solutions as alpha(α), beta(β), and delta(δ) respectively. The rest of the candidate solutions are assumed to be omega(ω). The purpose of this manner is to simulate the social hierarchy of grey wolves.

In addition to the social hierarchy of the wolves, another interesting social behavior of grey wolves is group hunting. The phases of hunting grey wolves include three main steps:

- Tracking and approaching the prey.
- Pursuing, encircling, and harassing the prey until it stops moving.
- Attacking the prey

To mathematically model the group-hunting behavior of the grey wolves, the GWO assumed that the first three best solutions have more knowledge about the potential location of the optimum (prey). The rest of the search agents update their positions according to the best search agents. Attacking and searching behavior corresponding to exploitation and exploration are the main behaviors of the search agent in the algorithm. These behaviors are controlled by the parameters of the algorithm and can be switched smoothly based on the adaptive values of the parameter. The framework of the GWO is shown in Figure 5.

B. THE DGWO ALGORITHM FOR THE 2DSP

The GWO is designed for solving the continuous function of optimization problems, while the 2DSP belongs to the discrete combinatorial optimization problem. Therefore, many issues need to be resolved regarding the implementation of the algorithm.

Decimal coding is adopted in the DGWO algorithm, which means that each rectangle is solely labeled by a decimal integer. The position of the search agent is $X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ij}, \ldots, x_{in})$, n is the number of rectangles, i is the number of feasible solutions, and the value of x_{ij} is an unsigned decimal integer that represents a rectangle. When updating the position of the search agent, guaranteeing the effectiveness of the coding (the same number and lack of any number cannot exist in the sequence) is the key to adopt a pattern of coding.

Compared to the GWO, the DGWO only considers the best solution as alpha(α) and the rest of the candidate solutions are omega(ω). When the grey wolf approaches the prey, it will stop at a certain distance. Thus, in the algorithm, measuring

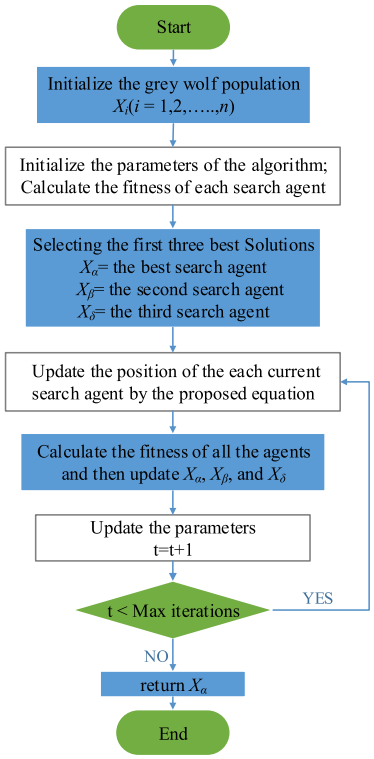


FIGURE 5. The GWO algorithm.

the distance between the search agent and the prey (alpha) is the primary problem. This problem can be successfully resolved by the proposed method. Assuming that the positions of two search agents are X_p and X_q, the distance D between them is defined as follows:

$$D = \sum_{j=1}^{n} c_j,\ c_j = \begin{cases} 1, & |x_{pj} - x_{qj}| \neq 0 \\ 0, & |x_{pj} - x_{qj}| = 0 \end{cases},$$
$$p, q \in \{1, 2, \ldots, M\}, \quad p \neq q \qquad (1)$$

M is the parameter of the DGWO and represents the number of grey wolves. Equation (1) is used to calculate the quantity of the unequal x_{pj} & x_{qj} pairs. For example, as shown in Figure 6, the value of D between X_p and X_q is five. The search agents become nearer to each other as the D decreases. This method is a simple but effective method for calculating the distance between the search agents.

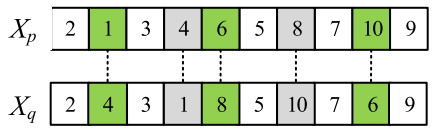


FIGURE 6. Example of calculating the distance *D*.

One of the main hunting behaviors of grey wolves is searching. Therefore, a method is proposed under decimal coding to mimic searching behavior. In the proposed method, $X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ij}, \ldots, x_{in})$ is not only the position of the search agent but also the rectangular sequence. The method randomly selects a part of the sequence first, which has $n^*\theta_1$ numbers and then moves the chosen sequence to the front of the original sequence, θ_1 is another parameter of the DGWO algorithm. Figure 7 gives an example of simulating the searching behavior in the algorithm. This method is vital because the rectangles that are packed first will dramatically affect the results of packing, and it is also good for ensuring the effectiveness of the coding.

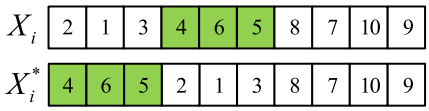


FIGURE 7. Example of searching behavior.

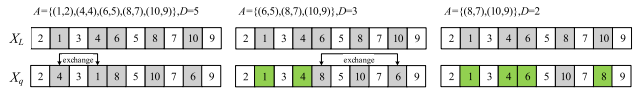
Attacking is another main hunting behavior of grey wolves. To imitate this behavior, the search agent changes the position and moves towards the alpha until the distance between them is no more than D_a, $D_a = n^*\theta_2$, θ_2 is the last parameter of the algorithm. It is required to increase the number of equal x_{pj} & x_{qj} pairs between a search agent and alpha. A series of strategies are proposed to achieve this goal and to guarantee the effectiveness of the coding.

The first step is to calculate the distance between the search agent and alpha because updating the position is not needed if the distance is less than D_a. The second step is to record different numbers between the alpha and search agent, and also record the site of numbers in the sequence. In equation (2), a_k and b_k represent the number and its site in the sequence, respectively, and X_L is the leader wolf, i.e., the alpha, X_p represents the search agent. Set A records all of the different numbers and their sites from 1 to n.

$$A\,(a_k, b_k) = \begin{cases} a_k = x_{Lj}, b_k = j & (x_{Lj} \neq x_{pj}, j = j + 1) \\ a_k = null, & b_k = null(x_{Lj} = x_{pj}, j = j + 1) \end{cases} \qquad (2)$$

The third step is that the search agents change their positions in light of the information provided by set A. Assuming that x_{ib} needs to be replaced by a, first, finding x_{ij} equals to a in the sequence $X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ij}, \ldots, x_{in})$, if x_{ib} is not equal to a, then exchange x_{ij} and x_{ib}; nothing needs to be done when the value of x_{ib} is the same as a.

There is an example (shown in Figure 8) to describe the entire process of attacking behavior. Hypothesizing that θ_2 is 0.2, $D_a = n^*\theta_2 = 10^*0.2 = 2$. After the first two steps, the distance D between the search agent X_q and the alpha X_L is five, and set A records the information, as shown in Figure 8(a). Then the third step is executed. When randomly selecting the first agent in set A, x_{q2} is not equal to 1, while x_{q4} is 1. Therefore, the number of x_{q2} and x_{q4} needs to be exchanged, and the result is shown in Figure 8(b). These steps repeat until the distance D is just less than D_a.



(a) Exchange x_{q2} and x_{q4} (b) Exchange x_{q5} and x_{q9} (c) Attacking is over

FIGURE 8. Example of attacking behavior.

The proposed methods and strategies can not only guarantee the effectiveness of the coding, but also enable the algorithm to have the ability of exploitation and exploration. Note that a new position X^* is generated after searching the behavior (or attacking behavior) of the search agent. The BestFitHeuristic is called on X^*, if X^* is not worse than X, X is replaced by X^*; otherwise, X^* is abandoned.

The pseudocode of the DGWO algorithm is presented in Algorithm 2. The DGWO is slightly different from the GWO in the framework. Generally, the DGWO algorithm mainly includes seven steps. First, the parameter of the algorithm should be set, and then the algorithm randomly creates the position of each search agent. The following step computes the material utilization of every search agent and picks the alpha wolf. The abovementioned four steps are the first stage, and the second stage has three major steps. The first step is the searching behavior of the search agent, and the second step is attacking behavior. Finally, the leader wolf is selected among the search agents according to material utilization. Stage two repeats until the time is over. Importantly, if one of the search agents is better than the leader wolf during the searching step, the searching behavior will terminate and then perform attacking behavior.

Algorithm 2 The DGWO Procedure

DGWO algorithm (Seq, W)
1 Set the parameters M, θ_1, θ_2
2 Randomly produce the position of each search agent
3 **for** $i \leftarrow 1$ to M
4 Calculating the material utilization U_i and the height H_i of the used strip
5 H_i, $U_i \leftarrow$ BestFitHeuristic(X_i, W).
6 Selecting the best search agent as leader wolf $X_\alpha =$ the best search agent
7 **while** it doesn't exceed the limited time
8 **for** $i \leftarrow 1$ to $M-1$
9 Search agent scouting, generate the position X_i^* H_i^*, $U_i^* \leftarrow$ BestFitHeuristic(X_i^*, W)
10 **if** ($U_i^* > U_i$)
11 $X_i \leftarrow X_i^*$; $H_i \leftarrow H_i^*$; $U_i \leftarrow U_i^*$
12 **if** ($U_i^* > U_\alpha$)
13 End this loop, $X_\alpha \leftarrow X_i^*$; $H_\alpha \leftarrow H_i^*$; $U_\alpha \leftarrow U_i^*$
14 **for** $i \leftarrow 1$ to $M-1$
15 Search agent attacking, generate the position X_i^* H_i^*, $U_i^* \leftarrow$ BestFitHeuristic(X_i^*, W)
16 **if** ($U_i^* > U_i$)
17 $X_i \leftarrow X_i^*$; $H_i \leftarrow H_i^*$; $U_i \leftarrow U_i^*$
18 Selecting the best search agent as leader wolf $X_\alpha =$ the best search agent
19 **return the** H_α and U_α

The time complexity analysis of the DGWO algorithm is conducted as follows. In line 5, sorting the rectangular sequences takes $O(nlog(n))$ times, which is similar to the heuristic algorithm of Wei *et al.* [24]. In line 6, the time complexity is $O(1)$, Line 8 to line 13 is a 'for' loop, takes $O((M-1)n\log(n))$ times. Line 14 to line 18 also is a 'for' loop, take $O((M-1)n\log(n))$ times. The total time complexity of the DGWO algorithm is the sum of individual steps, therefore, approximately $O(2(M-1)n\log(n))$.

IV. EXPERIMENTATION AND RESULTS

In this section, the DGWO is tested on the benchmark problems published in the literature. To verify the effectiveness of DGWO, it is compared to the meta-heuristic algorithm and other excellent algorithms. First, a series of computational experiments are presented to determine the appropriate parameters for the algorithm.

A. PROBLEM INSTANCES

The proposed DGWO algorithm was implemented as a sequential algorithm using the DEV C++, and all experiments were conducted on a Windows computer with Intel(R) Core(TM) CPU i5-3740 3.20 GHz and 3.88 GB RAM.

The algorithm was tested on many well-known benchmark instances from the previous literature. All of the instances were classified into two cases: zero-waste and non-zero-waste instances. The former was generated by cutting a large sheet into small pieces to form the rectangles, and a perfectly optimal solution with 100% area utilization was known, it contained the data set C, NT(n), NT(t), N, CX. While the second test set was not produced by cutting the sheet into pieces, and the perfect optimal solution was not known. It was made up of 8 data sets, named ngcut, gcut, cgcut, beng, bwmv, Nice, Path, and ZDF.

The features of the used data sets are summarized in Table 1. The *Inst.* indicates the number of instances in the data set, n is the number of rectangles, W represents the width of the strip, and H^* is the known optimal strip height. The details of all the instances can be obtained at www.computational logistic-s.org/orlib/2dsp/index.html.

B. THE STUDY OF PARAMETERS ON THE ALGORITHM

The DGWO algorithm only has three parameters that needed to be set: the number of wolves M, the searching factor θ_1, and the attacking factor θ_2. Some experiments were conducted to determine a suitable parameter for the 2DSP problem.

The running time of the algorithm was set to 60 s. Running the algorithm 15 times, the average of the obtained height was regarded as the result. The instance used was one of the data sets CX because the size of the problem with 100 rectangles was appropriate, and various rectangles are ranging from small to large. The experiment was carried out many times for each parameter.

TABLE 1. Data set features.

Data set	Inst.	n	W	H^*	Source
C	21	16–197	20–160	15–240	Hopper et al. (2001)
NT(n)	35	17–199	200	200	Hopper (2000)
NT(t)	35	17–199	200	200	Hopper(2000)
CX	7	50-15000	400	600	Pinto et al (2005)
N	13	10-3152	30-640	40-960	Burke et al. (2004)
beng	10	20–200	25–40	Unknow	Bengtsson (1982)
cgcut	3	16–60	10–70	Unknow	Christofides (1977)
gcut	13	10–50	250-3000	Unknow	Beasley JE (1985b)
ngcut	12	7–22	10–30	Unknow	Beasley JE (1985a)
bwmv	300	20-100	10-100	Unknow	Berkey et al. (1987)
bwmv	200	20-100	10-100	Unknow	Martello S (1998)
ZDF	16	580–75032	33–5172	Unknow	Leung et al. (2011)
Nice	36	25-1000	998-1000	Unknow	Wang et al (2001)
Path	36	25-1000	998-1000	Unknow	Wang et al (2001)

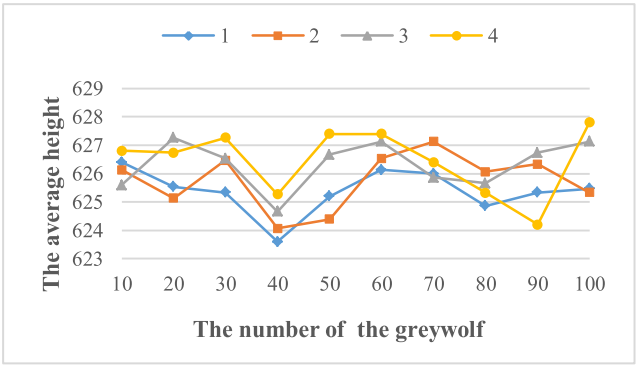
The first experiment studied the influence of the number of wolves on the solution, and the number was divided into ten cases from 10 to 100. The result is shown in Figure 9(a). The overall trend of the height was falling, then rising and falling again, and the shape of the broken line was similar to W. The reason is that the diversity of the solution, and the probability of finding the optimal solution increased with increasing wolves size. Additionally, the number of iterations affected the quality of the solution. With the increase in the number of wolves, the quality of the solution decreased under limited running time. It obtained a balance between the quality of the solution and the number of iterations in a limited time when the number of wolves was approximately 40.

The experiments on the parameters θ_1 and θ_2 were run four times and six times, respectively. Both the values of θ_1 and θ_2 ranged from 0 to 1 and were divided into nine cases. Figure 9(b) shows that the result was unstable when θ_1 was more than 0.2. The algorithm obtained the optimal height if the θ_1 was 0.2, and the result was steady. In Figure 9(c), the algorithm obtained better results than other cases when the value of θ_2 was approximately 0.5. The reason is that the diversity of the solution was slightly spoiled if the distance between the alpha and search agents was too close. In other words, it slightly reduced the exploration ability.

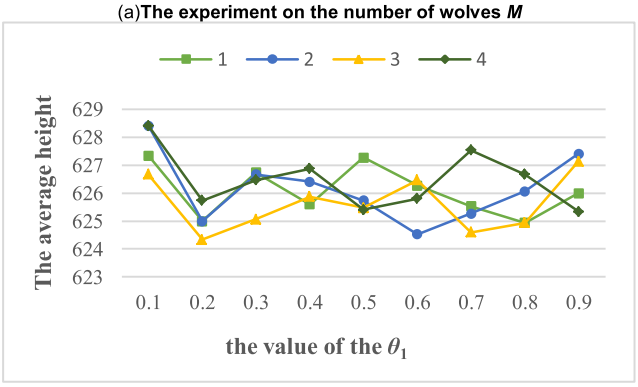
The above study shows that the algorithm performs well for the 2DSP problem as the number of grey wolves M and θ_1 and θ_2 were 40, 0.2, and 0.5, respectively.

C. THE COMPUTATIONAL RESULTS ON THE PERFORMANCE OF THE ALGORITHM

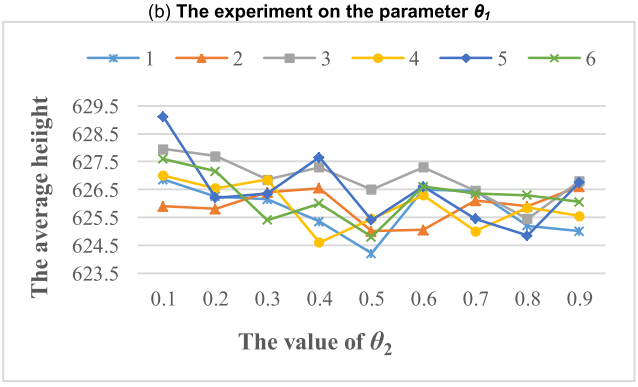
According to Table 1, 737 widely used benchmark instances were used to test the performance of the DGWO algorithm. The DGWO algorithm was run 10 times on each instance and the time limit for each run was also set to 60 s. The procedure



(a)The experiment on the number of wolves M



(b) The experiment on the parameter θ_1



(c) The experiment on the parameter θ_2

FIGURE 9. The experiment on the three parameters.

was terminated when the optimal solution was found, or a time limit of 60 s elapsed. For each instance, assuming that the calculated height is H, and the lower bound is LB, the gap is defined as $gap = 100 \times (H - LB)/LB$, the *best gap* is the best value of gap over 10 runs of each instance.

1) COMPARED WITH THE META-HEURISTIC ALGORITHM

The meta-heuristic algorithms applied to solve the 2DSP were SA+BF [20], GA+BLF [11], SA+BLF [11], ISA [21] and ACO [17]. All of these algorithms obtained a good solution for the problem, but there was room for further improvement.

Table 2 and Table 3 give the comparison on the *best gap* of these algorithms on the data sets C and N respectively, where the columns show the *best gap* for the two data sets, and the best results are highlighted in bold. The results of these algorithms are taken from their original articles.

TABLE 2. Computational results on data set C.

Instance				*best gap(100%)*				
	n	*w*	*LB*	GA+BLF	SA+BLF	SA+BF	ISA	DGWO
C11	16	20	20	**0**	**0**	**0**	**0**	**0**
C12	17	20	20	5	5	**0**	**0**	**0**
C13	16	20	20	**0**	**0**	**0**	**0**	**0**
C21	25	40	15	6.7	6.7	6.7	**0**	**0**
C22	25	40	15	6.7	6.7	6.7	**0**	**0**
C23	25	40	15	6.7	6.7	6.7	**0**	**0**
C31	28	60	30	6.7	6.7	3.3	**0**	**0**
C32	29	60	30	6.7	6.7	3.3	3.3	**0**
C33	28	60	30	6.7	6.7	3.3	**0**	**0**
C41	49	60	60	6.7	6.7	1.7	1.7	1.7
C42	49	60	60	5	6.7	1.7	1.7	1.7
C43	49	60	60	3.3	5	1.7	**0**	**0**
C51	73	60	90	5.6	4.4	**1.1**	**1.1**	**1.1**
C52	73	60	90	5.6	5.6	1.1	**0**	**0**
C53	73	60	90	5.6	5.6	2.2	**1.1**	**1.1**
C61	97	80	120	5.8	5.8	1.7	**0.8**	**0.8**
C62	97	80	120	5	5	0.8	**0.8**	**0.8**
C63	97	80	120	5	5	1.7	**0.8**	**0.8**
C71	196	160	240	6.3	6.3	1.7	**0.8**	**0.8**
C72	197	160	240	4.6	5.4	1.7	**0.4**	0.8
C73	196	160	240	5.8	6.3	2.1	**0.8**	**0.8**
Ave				5.21	5.38	2.34	0.63	**0.50**

TABLE 3. Computational results on data set N.

Instance				*best gap(100%)*					
	n	*w*	*LB*	GA+BLF	SA+BLF	SA+BF	ACO	ISA	DGWO
N1	10	40	40	**0**	**0**	**0**	**0**	**0**	**0**
N2	20	30	50	2	4	**0**	6.4	**0**	**0**
N3	30	30	50	4	4	2	2.3	**0**	**0**
N4	40	80	80	3.8	3.8	2.5	6.5	**0**	**0**
N5	50	100	100	6	6	3	6	1	**0**
N6	60	50	100	3	3	3	4.4	**0**	**0**
N7	70	80	100	6	6	4	7.9	**0**	**0**
N8	80	100	80	6.3	6.3	2.5	7	1.3	1.3
N9	100	50	150	3.3	3.3	1.3	3	**0**	**0**
N10	200	70	150	2.7	2.7	1.3	6.3	**0**	**0**
N11	300	70	150	3.3	3.3	2	5.1	**0**	**0**
N12	500	100	300	4.3	4	2	6	**0.3**	**0.3**
N13	3152	640	960	—	—	0.4	0.1	**0**	0.1
Ave				3.725	3.87	1.85	4.69	0.20	**0.13**

Data sets C and N are widely used to test the effectiveness of the algorithm for 2DSP.

The results are given in Tables 2 and 3, and the proposed algorithm obtained better results on these instances compared to most of the published meta-heuristic algorithms. Although the hardware environment among the algorithms was slightly different, the proposed algorithm had a time limit of 60 s, while the others obtained the best result without a time limit except for the ISA. Note that many of the previously published meta-heuristic algorithms could not obtain the optimal solution with 100% material utilization in instances C32 and N5, but the proposed algorithm easily obtained the optimal solution.

TABLE 4. Computational environments.

algorithm	#lan	CPU	RAM	times	Time limit(s)
GRASP	C++	Pentium 4 Mobile at 2GHz	—	10	60
SRA	C	Intel(R) core(TM) 2 CPU 2.13 GHz	0.99GB	10	60
IDBS	C++	Intel Xeon E5430 2.66 GHz	8GB	10	60
IA	C++	Intel R Xeon R CPU E5405 2.00 GHz	1.99GB	10	60
ISH	C++	Intel(R) Xeon® CPU E5645 2.40 GHz	3.49GB	10	60

TABLE 5. Comparison of the average *best gap* of GRASP, ISH, SRA, IDBS, IA, and DGWO.

Algo.	C	N	CX	NT(n)	NT(t)	2sp	bwmv	ZDF	Nice	Path
GRASP	0.96	0.96	0.93	2.40	2.24	2.68	1.77	4.22	1.84	1.68
ISH	0.64	0.20	0.89	1.99	1.94	2.99	1.69	2.77	0.95	0.75
SRA	0.62	0.13	0.44	1.19	1.31	3.04	1.49	2.56	**0.51**	**0.38**
IDBS	**0.04**	**0**	0.39	**0.91**	1.1	2.99	1.76	—	—	—
IA	0.22	0.13	—	1.09	**1.04**	3.05	**1.37**	—	0.52	0.40
DGWO	0.50	0.13	**0.29**	1.44	1.31	**2.63**	1.66	**1.92**	0.71	0.45

2) COMPARED WITH OTHER ALGORITHMS

In addition to adopting the meta-heuristic to solve the 2DSP problem, there are also some excellent algorithms based on BF. The current excellent algorithms are GRASP [36], SRA [22], IDBS [37], IA [23] and ISH [24]. There is a slight difference between these algorithms in the computational environment. The details about the hardware environment are shown in Table 4. The stopping criterion of all the algorithms was defined as a time limit of 60 s.

Table 5 shows the computational results of DGWO on these instances and the comparison with these algorithms, where the columns show the average *best gap* for the data sets and the best results are highlighted in bold. The used instances are classified into ten sets because the data sets cgcut, gcut, ngcut, and beng comprise a new data set named 2sp. Overall, none of the algorithms can obtain the best results on all data sets in the average of the best gap. Table 5 shows that the IDBS is the best one for data sets C, N, and NT(n), but the IA outperform the other algorithms on data sets NT(t) and bwmv. In addition, Table 5 also suggests that DGWO is superior to the rest of the algorithms on data sets CX, 2sp, and ZDF. Moreover, DGWO also obtained good results on the remaining data sets, and it performed better than GRASP, ISA, and SRA on more than five data sets.

Appendix gives the detailed results of the DGWO on each test set. The best results are highlighted in bold. Tables 9 and 10 give the results of six algorithms for data sets NT(n) and NT(t). We notice that the DGWO is slightly worse

TABLE 6. Computational results on data set C.

Instance				*best gap(100%)*						
	n	*w*	*LB*	GRASP	IA	SRA	ISA	IDBS	ISH	DGWO
C11	16	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C12	17	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C13	16	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C21	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C22	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C23	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C31	28	60	30	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C32	29	60	30	3.3	**0**	3.3	3.3	**0**	**0**	**0**
C33	28	60	30	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C41	49	60	60	1.7	1.7	1.7	1.7	**0**	**0**	1.7
C42	49	60	60	1.7	1.7	1.7	1.7	**0**	1.7	1.7
C43	49	60	60	1.7	**0**	1.7	**0**	**0**	**0**	**0**
C51	73	60	90	1.1	**0**	**0**	1.1	**0**	**0**	1.1
C52	73	60	90	1.1	**0**	**0**	**0**	**0**	**0**	**0**
C53	73	60	90	1.1	**0**	1.1	1.1	**0**	**0**	1.1
C61	97	80	120	1.7	**0**	0.8	0.8	**0**	0.8	0.8
C62	97	80	120	0.8	**0**	0.8	0.8	**0**	**0**	0.8
C63	97	80	120	1.7	**0**	0.8	0.8	**0**	0.8	0.8
C71	196	160	240	1.7	**0.4**	**0.4**	0.8	**0.4**	**0.4**	0.8
C72	197	160	240	1.3	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	0.8
C73	196	160	240	1.3	0.4	0.4	0.4	**0**	0.4	0.8
Ave				0.96	0.22	0.62	0.64	**0.04**	0.21	0.50

TABLE 7. Computational results on data set N.

Instance				*best gap(100%)*						
	n	*w*	*LB*	GRASP	IA	ISA	SRA	IDBS	ISH	DGWO
N1	10	40	40	**0**	**0**	**0**	**0**	**0**	**0**	**0**
N2	20	30	50	**0**	**0**	**0**	**0**	**0**	**0**	**0**
N3	30	30	50	2	**0**	**0**	**0**	**0**	**0**	**0**
N4	40	80	80	1.3	**0**	**0**	**0**	**0**	**0**	**0**
N5	50	100	100	2	**0**	1	**0**	**0**	**0**	**0**
N6	60	50	100	1	**0**	**0**	**0**	**0**	**0**	**0**
N7	70	80	100	1	**0**	**0**	**0**	**0**	**0**	**0**
N8	80	100	80	1.3	1.3	1.3	1.3	**0**	1.3	1.3
N9	100	50	150	0.7	**0**	**0**	**0**	**0**	**0**	**0**
N10	200	70	150	0.7	**0**	**0**	**0**	**0**	**0**	**0**
N11	300	70	150	0.7	**0**	**0**	**0**	**0**	**0**	**0**
N12	500	100	300	1.3	0.3	0.33	0.3	**0**	0.3	0.3
N13	3152	640	960	0.5	0.1	**0**	0.1	**0**	0.1	0.1
Ave				0.96	0.13	0.20	0.13	**0**	0.13	0.13

TABLE 8. Computational results on data set CX.

Instance				*best gap(100%)*					
	n	*w*	*LB*	GRASP	SRA	IDBS	ISA	ISH	DGWO
50cx	50	400	600	2.2	0.3	**0**	3.4	**0**	**0**
100cx	100	400	600	2.8	2.8	2.7	2.6	2.2	**2**
500cx	500	400	600	0.8	**0**	**0**	0.2	**0**	**0**
1000cx	1000	400	600	0.3	**0**	**0**	**0**	**0**	**0**
5000cx	5000	400	600	**0**	**0**	**0**	**0**	**0**	**0**
10,000cx	10000	400	600	**0**	**0**	**0**	**0**	**0**	**0**
15,000cx	15000	400	600	**0**	**0**	**0**	**0**	**0**	**0**
Ave				0.93	0.44	0.39	0.89	0.31	**0.29**

than the others when n is larger than 29 and less than 97. A similar result can be found in Table 6. Table 11 shows that the proposed algorithm obtained excellent results on data sets ngcut and cgnut, for which the number of rectangles is small. Table 8 indicates that the DGWO also obtained excellent results if the scale of the problem is large. DGWO is an efficient method suitable for large-scale problems but is poor for medium-size problems.

TABLE 9. Computational results on data set NT(n).

Instance				*best gap(100%)*					
	n	*w*	*LB*	GRASP	ISA	SRA	IDBS	IA	DGWO
n1a	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
n1b	17	200	200	4.5	5	**0**	**0**	**0**	**0**
n1c	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
n1d	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
n1e	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
n2a	25	200	200	3	2	**0**	**0**	**0**	**0**
n2b	25	200	200	3	4.5	**0**	3	2.5	**0**
n2c	25	200	200	4	3.5	**0**	**0**	2	**0**
n2d	25	200	200	4.5	3	2.5	**0**	**0**	**0**
n2e	25	200	200	3	3	**0**	**0**	2.5	**0**
n3a	29	200	200	4.5	3	**0**	**0**	**0**	**0**
n3b	29	200	200	4	4.5	3.5	**0**	3.5	3.5
n3c	29	200	200	2.5	2.5	**0**	**0**	2.5	2
n3d	29	200	200	3.5	2	2	**0**	**0**	3
n3e	29	200	200	3.5	4	3.5	**0**	2.5	2
n4a	49	200	200	3	3	2	2.5	**1.5**	2.5
n4b	49	200	200	3.5	**2.5**	2	3.5	**2.5**	3
n4c	49	200	200	2.5	2.5	2.5	2.5	**1.5**	2
n4d	49	200	200	3	2	2	2.5	**1.5**	2.5
n4e	49	200	200	2.5	3	3	3	**1.5**	3
n5a	73	200	200	2.5	2.5	2	2	**1.5**	2.5
n5b	73	200	200	2	1	1.5	1.5	**1.5**	2
n5c	73	200	200	3	2	2	2	**1.5**	2.5
n5d	73	200	200	2	2.5	2	2	**1.5**	3
n5e	73	200	200	3	**1.5**	2	2	**1.5**	2
n6a	97	200	200	2	**1**	1.5	**1**	**1**	2
n6b	97	200	200	2	1.5	1.5	1	**0.5**	1.5
n6c	97	200	200	2	1.5	1.5	**0.5**	**0.5**	1.5
n6d	97	200	200	2	1.5	**1**	1.5	**1**	2
n6e	97	200	200	2	1.5	**1**	**1**	**1**	2
n7a	199	200	200	1	0.5	0.5	**0**	0.5	1.5
n7b	199	200	200	1.5	1	0.5	**0**	0.5	1
n7c	199	200	200	1.5	0.5	0.5	**0**	0.5	1
n7d	199	200	200	1.5	0.5	0.5	**0**	0.5	1.5
n7e	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1
Ave				2.40	1.99	1.19	**0.91**	1.09	1.44

D. DISCUSSION

The comparative study results report that DGWO can provide very competitive results. The DGWO algorithm outperforms most of the published meta-heuristic. We analyze the results from the mechanism and structure of the DGWO. From the mechanism perspective, the DGWO shows a good balance between exploration and exploitation. This superior capability is due to the attacking behavior and searching behavior. The exploitation ability in the algorithm is strong, while exploration is not sufficient. The searching behavior with randomness can strengthen the exploration and allow the DGWO algorithm to search globally. This mechanism assists DGWO to provide very good exploration and exploitation simultaneously.

From the structure perspective, the DGWO modified the process and structure of the algorithm. First, during the searching step of the original GWO, the stopping criterion of this step is determined by all the wolves finishing the search process or the searched solution of one wolf being superior to

TABLE 10. Computational results on data set NT(t).

Instance				*best gap(100%)*					
	n	*w*	*LB*	GRASP	ISA	SRA	IDBS	IA	DGWO
t1a	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
t1b	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
t1c	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
t1d	17	200	200	**0**	5	**0**	**0**	**0**	**0**
t1e	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**
t2a	25	200	200	2	3.5	3	**0**	**0**	**0**
t2b	25	200	200	4	3	**0**	**0**	**0**	**0**
t2c	25	200	200	4	3	3	**0**	**0**	**0**
t2d	25	200	200	3	1.5	**0**	**0**	2.5	**0**
t2e	25	200	200	3	3	2.5	**0**	2.5	**0**
t3a	29	200	200	3.5	4.5	**0**	**0**	**0**	3
t3b	29	200	200	4.5	4	3	4	3.5	**0**
t3c	29	200	200	3	3	2.5	**0**	**0**	**0**
t3d	29	200	200	3.5	3	**0**	2	2	2.5
t3e	29	200	200	4	2.5	2.5	**0**	3.5	3.5
t4a	49	200	200	2.5	2.5	2.5	2.5	**2**	**2**
t4b	49	200	200	2.5	3	3	2.5	**1.5**	2.5
t4c	49	200	200	3	2	1.5	2.5	**1.5**	2
t4d	49	200	200	3	2.5	**1**	2.5	1.5	3
t4e	49	200	200	3.5	**2**	**2**	3.5	2.5	2.5
t5a	73	200	200	3	2	2	2.5	**1.5**	2.5
t5b	73	200	200	2	2	1.5	2.5	**1**	2.5
t5c	73	200	200	2.5	2.5	2	2.5	**1.5**	2.5
t5d	73	200	200	2	2	2	2	**1.5**	2.5
t5e	73	200	200	2	2	2	2	**1.5**	2
t6a	97	200	200	2	**1**	1.5	**1**	**1**	1.5
t6b	97	200	200	2	1	1.5	1	**0.5**	1.5
t6c	97	200	200	2	1.5	1.5	**1**	**1**	2
t6d	97	200	200	2	1.5	1.5	1	**0.5**	1.5
t6e	97	200	200	2.5	1.5	1.5	**1**	**1**	2
t7a	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
t7b	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1
t7c	199	200	200	2	**0.5**	**0.5**	**0.5**	**0.5**	1
t7d	199	200	200	1	1	**0.5**	**0.5**	**0.5**	1
t7e	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1
Ave				2.24	1.94	1.31	1.1	**1.04**	1.31

TABLE 11. Computational results on data set 2sp.

Instance				*best gap(100%)*					
	n	*w*	*LB*	GRASP	ISA	SRA	IDBS	IA	DGWO
cgcut1	16	10	23	**0**	**0**	4.4	**0**	4.3	**0**
cgcut2	23	70	63	3.2	3.2	3.2	**0**	3.2	3.2
cgcut3	62	70	636	3.9	3.5	3.9	**0**	3.9	3.6
gcut1	10	250	1016	**0**	**0**	**0**	**0**	**0**	**0**
gcut2	20	250	1133	5.1	4.8	4.8	**0**	4.8	4.8
gcut3	30	250	1803	**0**	**0**	**0**	6.9	**0**	**0**
gcut4	50	250	2934	2.3	2.4	2.4	**0**	2.2	2.8
gcut5	10	500	1172	8.6	8.6	8.6	**6.3**	8.6	8.6
gcut6	20	500	2514	4.5	4.7	4.3	**2**	4.3	4.3
gcut7	30	500	4641	1.1	1.1	1.1	**0**	1.1	1.1
gcut8	50	500	5703	3.7	3.2	**3**	4	2.8	3.6
gcut9	10	1000	2022	14.6	14.6	14.6	**0**	14.6	10.8
gcut10	20	1000	5392	11.4	11.4	11.4	**0**	11.4	19.4
gcut11	30	1000	7374	5.5	5.1	**5**	5.1	**5**	5.5
gcut12	50	1000	12522	17.3	17.3	17.3	**0**	17.3	5.2
gcut13	32	3000	4772	4.7	**4**	4.1	2.9	4.2	4.6
ngcut1	10	10	23	**0**	**0**	**0**	8.6	**0**	**0**
ngcut2	17	10	30	**0**	3.3	3.3	4.3	3.3	**0**
ngcut3	21	10	28	**0**	**0**	**0**	1.2	**0**	**0**
ngcut4	7	10	20	**0**	**0**	**0**	4.8	**0**	**0**
ngcut5	14	10	36	**0**	**0**	**0**	11.4	**0**	**0**
ngcut6	15	10	29	**6.9**	**6.9**	**6.9**	19.5	**6.9**	**6.9**
ngcut7	8	20	20	**0**	**0**	**0**	18.3	**0**	**0**
ngcut8	13	20	32	3.1	6.3	6.3	5.8	6.3	9.4
ngcut9	18	20	49	**2**	6.1	4.1	5.3	4.1	**2**
ngcut10	13	30	80	**0**	**0**	**0**	**0**	**0**	**0**
ngcut11	15	30	50	4	4	4	**3.2**	4	4
ngcut12	22	30	87	**0**	**0**	**0**	4.1	**0**	**0**
beng1	20	25	30	**0**	3.3	3.3	**0**	3.3	**0**
beng2	40	25	57	**0**	**0**	**0**	**0**	**0**	**0**
beng3	60	25	84	**0**	**0**	**0**	**0**	**0**	**0**
beng4	80	25	107	**0**	**0**	**0**	**0**	**0**	**0**
beng5	100	25	134	**0**	**0**	**0**	**0**	**0**	**0**
beng6	40	40	36	**0**	**0**	**0**	**0**	**0**	**0**
beng7	80	40	67	**0**	**0**	**0**	**0**	**0**	**0**
beng8	120	40	101	**0**	**0**	**0**	**0**	**0**	**0**
beng9	160	40	126	**0**	**0**	**0**	**0**	**0**	**0**
beng10	200	40	156	**0**	**0**	**0**	**0**	**0**	**0**
Ave				2.68	2.99	3.04	2.99	3.05	**2.63**

the α wolf. When the stopping criterion is satisfied, the α wolf calls the other wolves to surround the prey and the hunting behavior enters the encircling step. In the earlier stage of the algorithm, we find that the searching result is not good. The search agents more easily find a better solution than the current solution so that the wolves break off the searching step to enter the encircling step frequently. Thus, some wolves do not conduct the searching step but directly enter the encircling step. The wolves more easily trap in local minima. However, in the DGWO, the stopping criterion of the searching step is that all wolves finish the searching step, which can increase the probability of avoiding local minima. Second, the DGWO simplifies the hunting behavior (tracking, encircling, attacking) and redesigns the searching and attacking operators. This modification reduces unnecessary intelligent behavior but does not decrease the quality of the solution. The DGWO can search for solution space more times within a certain time.

In addition, an improved best-fit strategy is developed to solve this packing problem. The improved best-fit strategy based on scoring rules enhances the search ability of the DGWO. Moreover, the embedded evaluation strategy enabled the algorithm to identify the best item to pack during the packing process, allowing it to generate dense packing layouts.

The proposed algorithm can successfully address the 2DSP problem, it exists some limitations: (1) because the searching process of GWO is guided by α, β, and δ, the GWO tends to fall into the local optimal solution. Although DGWO modifies the process and structure of the algorithm, it maintains the central themes of GWO. Thus, DGWO also tends towards premature convergence, (2) the time complexity of the DGWO algorithm is $O(2(M\text{-}1)nlog(n))$, its time complexity is larger than that of the state-of-the-art algorithm. The average runtime of DGWO is 26.46 s on data set CX. Nonetheless, DGWO hybridized with a heuristic placement algorithm BF can obtain very competitive results in an acceptable time.

V. CONCLUSION

In this article, an effective discrete grey wolf optimizer (DGWO) is proposed for the first time to specifically solve the 2DSP problem. The proposed algorithm is easy to implement and only has three parameters that must be set. The performance of DGWO is compared to some excellent

TABLE 12. Computational results on data set bwmv.

Instance				*best gap(100%)*					
	n	*w*	*LB*	GRASP	ISA	SRA	IDBS	IA	DGWO
C01	20	10	60.3	1.8	1.7	1.7	**1.5**	1.7	**1.5**
	40	10	121.6	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
	60	10	187.4	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**
	80	10	262.2	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
	100	10	304.4	0.2	0.2	0.2	0.2	**0.1**	**0.1**
C02	20	30	19.7	0.5	1	0.5	**0**	**0**	0.5
	40	30	39.1	**0**	**0**	**0**	**0**	**0**	**0**
	60	30	60.1	0.3	**0**	**0**	**0**	**0**	**0**
	80	30	83.2	0.1	**0**	**0**	**0**	**0**	**0**
	100	30	100.5	0.1	**0**	**0**	**0**	**0**	**0**
C03	20	40	157.4	**3.9**	4.2	**3.9**	4.2	**3.9**	4.2
	40	40	328.8	1.6	1.5	1.3	1.5	**1.2**	**1.2**
	60	40	500	1.3	1.2	**1**	1.3	**1**	1.1
	80	40	701.7	1.1	1.1	**1**	1.2	**1**	1.2
	100	40	832.7	0.9	0.6	**0.5**	0.7	**0.5**	0.9
C04	20	100	61.4	3.1	4.1	3.4	3.1	**2.9**	3.1
	40	100	123.9	1.9	1.8	1.3	**1**	**1**	1.1
	60	100	193	1.9	1.3	0.9	0.9	**0.8**	**0.8**
	80	100	267.2	1.8	1	0.7	0.9	**0.5**	1.1
	100	100	322	1.6	0.8	0.5	0.6	**0.3**	0.9
C05	20	100	512.2	**4.2**	4.4	**4.2**	4.4	4.3	4.3
	40	100	1053.8	2	**1.9**	**1.9**	2.1	**1.9**	2.1
	60	100	1614	2	1.8	**1.7**	2.2	**1.7**	2.1
	80	100	2268.4	1	0.9	0.9	1.2	**0.8**	1
	100	100	2617.4	1.3	**1**	1.1	1.7	**1**	2
C06	20	10	159.9	**4.6**	6.1	4.9	4.7	4.9	**4.6**
	40	10	323.5	3.1	3.2	2.6	3.4	**2.3**	2.8
	60	10	505.1	2.9	2.8	2.2	3.1	**2**	2.9
	80	10	699.7	2.7	2.3	2	2.8	**1.6**	2.7
	100	10	843.8	2.5	2.1	1.7	2.6	**1.4**	2.5
C07	20	30	490.4	**2.3**	**2.3**	**2.3**	**2.3**	**2.3**	**2.3**
	40	30	1049.7	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
	60	30	1515.9	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
	80	30	2206.1	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**
	100	30	2627	**0.6**	0.7	0.7	0.7	**0.6**	**0.6**
C08	20	40	434.6	5.5	5.5	5.2	5.4	5	**4.9**
	40	40	922	3.5	3.2	2.9	3.6	**2.4**	3
	60	40	1360.9	3.2	2.8	2.5	3.2	**2.1**	2.9
	80	40	1909.3	3.3	2.4	2.4	3.2	**2**	3.2
	100	40	2362.8	3.1	2	2	3.3	**1.7**	3
C09	20	100	1106.8	**0**	**0**	**0**	**0**	**0**	**0**
	40	100	2189.2	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
	60	100	3410.4	**0**	**0**	**0**	**0**	**0**	**0**
	80	100	4578.6	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
	100	100	5430.5	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
C10	20	100	337.8	3.8	3.7	**3.5**	3.9	**3.5**	3.6
	40	100	642.8	3.4	3.3	**3**	3.6	2.8	3.1
	60	100	911.1	2.6	2.4	2.3	3.2	**2**	2.5
	80	100	1177.6	2.7	2.3	2.1	3.3	**1.8**	2.8
	100	100	1476.5	2.4	1.9	1.8	3.3	**1.5**	2.6
Ave				1.77	1.69	1.49	1.76	**1.37**	1.66

TABLE 13. Computational results on data set ZDF.

Instance				*best gap(100%)*			
	n	*w*	*LB*	GRASP	ISA	SRA	DGWO
zdf1	580	100	330	0.9	**0**	**0**	**0**
zdf2	660	100	357	0.8	**0**	**0**	**0**
zdf3	740	100	384	0.8	**0**	**0**	**0**
zdf4	820	100	407	0.7	**0**	**0**	**0**
zdf5	900	100	434	0.7	**0**	**0**	**0**
zdf6	1532	3000	4872	7.8	4.3	**3.1**	5.2
zdf7	2432	3000	4852	6.4	4.8	**3.8**	6.5
zdf8	2532	3000	5172	7.2	7.3	5.7	**1.3**
zdf9	5032	3000	5172	5.9	4.5	7	**1.7**
zdf10	5064	6000	5172	7.7	4.8	**3.9**	4.7
zdf11	7564	6000	5172	7.5	4.8	4.7	**1.7**
Ave				4.22	2.77	2.56	**1.92**

TABLE 14. Computational results on data set Nice.

Instance				*best gap(100%)*				
	n	*w*	*LB*	GRASP	ISA	SRA	IA	DGWO
Nice1	25	1000	1000	3.4	3.5	3.5	**2.8**	3.5
Nice2	50	1000	1001	4.6	4.2	3	2.6	3.9
Nice3	100	1000	1001	4	2.8	3.1	2.3	3.7
Nice4	200	1000	1001	3.6	2.9	2.2	**1.7**	3.2
Nice5	500	1000	1000	2.4	1.5	0.8	**0.6**	1.5
Nice6	1000	1000	999	2.1	1.2	**0.3**	0.5	0.7
Nice1t	1000	1000	1001	2.5	1	**0.4**	0.9	0.7
	1000	1000	1001	2.1	0.9	**0.3**	**0.3**	0.5
	1000	1000	1000	2	1.1	**0.3**	**0.3**	0.7
	1000	1000	1000	1.9	1	**0.4**	0.6	0.7
	1000	1000	1000	2.2	1	0.4	**0.3**	0.6
	1000	1000	1001	1.9	0.9	**0.1**	0.4	0.4
	1000	1000	1000	2.2	1	0.3	**0.2**	0.7
	1000	1000	1001	2	1.1	**0.3**	0.5	0.7
	1000	1000	1000	2.2	1.2	**0.3**	**0.3**	0.6
	1000	1000	1001	2.6	1.1	**0.3**	**0.3**	0.6
Nice2t	2000	1000	1001	1.5	0.5	0.2	0.3	0.2
	2000	1000	1001	1.4	0.4	**0.1**	0.2	0.2
	2000	1000	1000	1.6	0.7	0.2	0.4	0.3
	2000	1000	1000	1.4	0.6	0.2	0.3	0.2
	2000	1000	1000	1.5	0.6	**0.1**	0.2	0.2
	2000	1000	1000	1.6	0.5	**0.1**	0.2	**0.1**
	2000	1000	1001	1.5	0.6	**0.1**	0.4	0.3
	2000	1000	1001	1.3	0.5	**0.1**	0.2	**0.1**
	2000	1000	1001	1.5	0.6	0.2	**0.1**	0.2
	2000	1000	1001	1.5	0.6	**0.2**	0.3	0.3
Nice5t	5000	1000	1000	1	0.3	**0.1**	0.2	**0.1**
	5000	1000	1001	1	0.2	**0**	0.2	**0**
	5000	1000	1001	0.9	0.2	0.1	0.1	**0**
	5000	1000	1000	0.9	0.2	0.1	0.2	**0**
	5000	1000	1001	1	0.2	**0.1**	0.1	**0.1**
	5000	1000	1000	0.9	0.2	**0.1**	0.2	**0.1**
	5000	1000	1001	1	0.2	**0**	0.1	0.1
	5000	1000	1000	1.1	0.2	**0.1**	**0.1**	**0.1**
	5000	1000	1001	0.9	0.2	**0**	0.2	**0**
	5000	1000	1000	1	0.3	**0.1**	**0.1**	0.2
Ave				1.84	0.95	**0.51**	0.52	0.71

meta-heuristic algorithms on many well-known benchmark instances. The computational results indicate that DGWO outperforms SA, GA, ISA, and ACO in most instances. Moreover, in contrast to other state-of-the-art algorithms, such as GRASP, SRA, IDBS, IA, and ISH, the proposed algorithm can obtain a better or comparable solution over the majority of instances. The experimental results demonstrate that the DGWO algorithm is effective for the 2DSP problem. The proposed algorithm can successfully address the 2DSP problem, however, the time complexity of DGWO is larger than that of the state-of-the-art algorithm. Thus, improving the data structure and the framework of the algorithm is the direction of future research.

APPENDIX

THE DETAILED RESULTS OF DGWO ON EACH TEST SET

Tables 6–15 give the detailed results of the DGWO on each test set. The best results are highlighted in bold.

TABLE 15. Computational results on data set Path.

Instance				*best gap(100%)*				
	n	*w*	*LB*	GRASP	ISA	SRA	IA	DGWO
Path1	25	1000	1001	**4.1**	**4.1**	4.2	**4.1**	**4.1**
Path2	50	1000	1000	1.9	1.2	**0.1**	**0.1**	**0.1**
Path3	100	1000	1000	2.7	2	**2.2**	1.8	2.9
Path4	200	1000	1002	2.1	1.5	1.4	**1.1**	1.8
Path5	500	1000	1000	3.4	2	**1.4**	**1.4**	1.8
Path6	1000	1000	1002	2.4	0.9	0.6	0.5	0.6
Path1t	1000	1000	999	2	0.8	**0.2**	**0.2**	0.4
	1000	1000	1001	1.7	0.5	**0.2**	0.3	0.3
	1000	1000	1001	1.7	0.7	**0.2**	**0.2**	0.4
	1000	1000	1000	1.6	0.6	**0.1**	0.3	0.3
	1000	1000	1003	2.1	0.7	0.3	**0.2**	0.4
	1000	1000	1002	1.6	0.8	**0.2**	0.5	0.4
	1000	1000	999	2	0.9	0.3	**0.2**	0.7
	1000	1000	1000	2	0.8	**0.2**	0.4	0.6
	1000	1000	999	2	0.7	**0.3**	0.4	0.4
	1000	1000	1002	1.6	0.6	**0.2**	0.3	**0.2**
Path2t	2000	1000	1000	1.5	0.6	0.1	**0**	0.1
	2000	1000	1002	1.4	0.5	**0.1**	0.2	**0.1**
	2000	1000	1000	1.5	0.6	**0.1**	**0.1**	**0.1**
	2000	1000	999	1.5	0.4	**0**	0.2	**0**
	2000	1000	1002	1.6	0.6	**0.1**	**0.1**	**0.1**
	2000	1000	1002	1.4	0.5	**0**	0.3	0.1
	2000	1000	998	1.3	0.6	**0.1**	**0.1**	**0.1**
	2000	1000	998	1.6	0.6	0.2	0.2	0.2
	2000	1000	1001	1.6	0.7	**0**	0.3	**0.1**
	2000	1000	1003	1.5	0.6	**0**	0.1	**0**
Path5t	5000	1000	1000	1	0.3	0.1	0.1	**0**
	5000	1000	998	1.1	0.3	0.1	**0**	**0**
	5000	1000	1000	1.1	0.3	0.1	0.1	**0**
	5000	1000	995	1.1	0.2	0.1	0.2	**0**
	5000	1000	1004	1.2	0.2	**0**	**0**	**0**
	5000	1000	1000	0.9	0.2	**0**	**0**	**0**
	5000	1000	998	1.1	0.3	0.1	**0**	**0**
	5000	1000	996	1.1	0.3	0.1	0.1	**0**
	5000	1000	997	1	0.2	**0**	0.2	**0**
	5000	1000	1002	1.1	0.2	0.1	**0**	**0**
Ave				1.68	0.75	**0.38**	0.40	0.45

REFERENCES

[1] M. Abdul Quader, S. Ahmed, S. Z. Dawal, and Y. Nukman, "Present needs, recent progress and future trends of energy-efficient ultra-low carbon dioxide (CO_2) steelmaking (ULCOS) program," *Renew. Sustain. Energy Rev.*, vol. 55, pp. 537–549, Mar. 2016.

[2] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, Dec. 2007.

[3] J. E. Beasley, "An exact two-dimensional non-guillotine cutting tree search procedure," *Oper. Res.*, vol. 33, no. 1, pp. 49–64, Feb. 1985.

[4] S. Martello, M. Monaci, and D. Vigo, "An exact approach to the strip-packing problem," *Informs J. Comput.*, vol. 15, no. 3, pp. 310–319, Aug. 2003.

[5] M. Hifi and R. M'Hallah, "An exact algorithm for constrained two-dimensional two-staged cutting problems," *Oper. Res.*, vol. 53, no. 1, pp. 140–150, Feb. 2005.

[6] Y. Cui, Y. Yang, X. Cheng, and P. Song, "A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1281–1291, Apr. 2008.

[7] M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi, "Exact algorithms for the two-dimensional strip packing problem with and without rotations," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 73–83, Oct. 2009.

[8] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM J. Comput.*, vol. 9, no. 4, pp. 846–855, Nov. 1980.

[9] B. Chazelle, "The bottomn-left bin-packing heuristic: An efficient implementation," *IEEE Trans. Comput.*, vol. C-32, no. 8, pp. 697–707, Aug. 1983.

[10] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem," *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 34–57, Jan. 2001.

[11] E. K. Burke, G. Kendall, and G. Whitwell, "A new placement heuristic for the orthogonal stock-cutting problem," *Oper. Res.*, vol. 52, no. 4, pp. 655–671, Aug. 2004.

[12] Ö. B. Aşık and E. Özcan, "Bidirectional best-fit heuristic for orthogonal rectangular strip packing," *Ann. Oper. Res.*, vol. 172, no. 1, pp. 405–427, Nov. 2009.

[13] S. C. H. Leung and D. Zhang, "A fast layer-based heuristic for non-guillotine strip packing," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13032–13042, Sep. 2011.

[14] J. Verstichel, P. De Causmaecker, and G. V. Berghe, "An improved best-fit heuristic for the orthogonal strip packing problem," *Int. Trans. Oper. Res.*, vol. 20, no. 5, pp. 711–730, Sep. 2013.

[15] E. Hopper and B. Turton, "A genetic algorithm for a 2D industrial packing problem," *Comput. Ind. Eng.*, vol. 37, nos. 1–2, pp. 375–378, Oct. 1999.

[16] A. Bortfeldt, "A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces," *Eur. J. Oper. Res.*, vol. 172, no. 3, pp. 814–837, Aug. 2006.

[17] C. Yuan and X. Liu, "Solution to 2D rectangular strip packing problems based on ACOs," in *Proc. Int. Workshop Intell. Syst. Appl.*, May 2009, pp. 1–4.

[18] S. C. H. Leung, D. Zhang, C. Zhou, and T. Wu, "A hybrid simulated annealing Metaheuristic algorithm for the two-dimensional knapsack packing problem," *Comput. Oper. Res.*, vol. 39, no. 1, pp. 64–73, Jan. 2012.

[19] M. K. Omar and K. Ramakrishnan, "Solving non-oriented two dimensional bin packing problem using evolutionary particle swarm optimisation," *Int. J. Prod. Res.*, vol. 51, no. 20, pp. 6002–6016, Oct. 2013.

[20] E. K. Burke, G. Kendall, and G. Whitwell, "A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem," *Informs J. Comput.*, vol. 21, no. 3, pp. 505–516, Aug. 2009.

[21] S. C. H. Leung, D. Zhang, and K. M. Sim, "A two-stage intelligent search algorithm for the two-dimensional strip packing problem," *Eur. J. Oper. Res.*, vol. 215, no. 1, pp. 57–69, Nov. 2011.

[22] S. Yang, S. Han, and W. Ye, "A simple randomized algorithm for two-dimensional strip packing," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 1–8, Jan. 2013.

[23] L. Wei, H. Qin, B. Cheang, and X. Xu, "An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem," *Int. Trans. Oper. Res.*, vol. 23, nos. 1–2, pp. 65–92, Jan. 2016.

[24] L. Wei, Q. Hu, S. C. H. Leung, and N. Zhang, "An improved skyline based heuristic for the 2D strip packing problem and its efficient implementation," *Comput. Oper. Res.*, vol. 80, pp. 113–127, Apr. 2017.

[25] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[26] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, Jan. 2016.

[27] M. Radmanesh and M. Kumar, "Grey wolf optimization based sense and avoid algorithm for UAV path planning in uncertain environment using a Bayesian framework," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 68–76.

[28] G. M. Komaki and V. Kayvanfar, "Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *J. Comput. Sci.*, vol. 8, pp. 109–120, May 2015.

[29] A. Chaman-Motlagh, "Superdefect photonic crystal filter optimization using grey wolf optimizer," *IEEE Photon. Technol. Lett.*, vol. 27, no. 22, pp. 2355–2358, Nov. 15, 2015.

[30] U. Sultana, A. B. Khairuddin, A. S. Mokhtar, N. Zareen, and B. Sultana, "Grey wolf optimizer based placement and sizing of multiple distributed generation in the distribution system," *Energy*, vol. 111, pp. 525–536, Sep. 2016.

[31] M. H. Sulaiman, Z. Mustaffa, M. R. Mohamed, and O. Aliman, "Using the gray wolf optimizer for solving optimal reactive power dispatch problem," *Appl. Soft Comput.*, vol. 32, pp. 286–292, Jul. 2015.

[32] A. E. Ezugwu, O. J. Adeleke, A. A. Akinyelu, and S. Viriri, "A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 6207–6251, May 2020.

[33] V. Kumar and D. Kumar, "An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems," *Adv. Eng. Softw.*, vol. 112, pp. 231–254, Oct. 2017.

[34] M. Chen, M. Zhou, X. Guo, X. S. Lu, J. Ji, and Z. Zhao, "Grey wolf optimizer adapted for disassembly sequencing problems," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2019, pp. 46–51.
[35] H. Qin, P. Fan, H. Tang, P. Huang, B. Fang, and S. Pan, "An effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint," *Comput. Ind. Eng.*, vol. 128, pp. 458–476, Feb. 2019.
[36] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, "Reactive GRASP for the strip-packing problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1065–1083, Apr. 2008.
[37] L. Wei, A. Lim, and W. Zhu, "A skyline-based heuristic for the 2D rectangular strip packing problem," in *Proc. Int. Conf. Ind., Eng. Appl. Appl. Intell. Syst.* Berlin, Germany: Springer, 2011, pp. 286–295.

PENG WANG received the master's degree in mechanical engineering from Southwest Petroleum University, Chengdu, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. His current research interests include intelligent manufacturing, computational intelligence, and combinatorial optimization.

YUNQING RAO was born in Chibi, Hubei, China, in 1968. He received the B.S. and M.S. degrees in mechanical manufacturing and the Ph.D. degree in mechanical manufacturing and automation from the Huazhong University of Science and Technology, Wuhan, China, in 1992 and 1999, respectively.

In 2001, he was with the Management of Industrial Production (MIP) Training, Aachen University of Technology, supported by the Chiang's Fund of Hong Kong. From 2004 to 2005, he visited the University of Oxford. He is currently a Professor with the Huazhong University of Science and Technology. He has authored over 200 journal articles. His research interests include operation optimization of manufacturing systems, manufacturing execution systems (MESs), and intelligent manufacturing.

QIANG LUO received the B.S. degree in mechanical engineering from the Taiyuan University of Technology, Taiyuan, China, in 2016. He is currently pursuing the master's degree in mechanical engineering with the Huazhong University of Science and Technology, Wuhan, China. His current research interests include intelligent manufacturing, computational intelligence, and combinatorial optimization.