IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 24, 2020, accepted December 16, 2020, date of publication December 18, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3045905

Hybrid Heuristic Algorithm Based On Improved Rules & Reinforcement Learning for 2D Strip Packing Problem

KAI ZHU, NAIHUA JI, AND XIANG DONG LI
School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266400, China
Corresponding author: Kai Zhu (zhu_kaicom@163.com)

This work was supported by the Shandong Provincial Natural Science Foundation of China under Grant ZR2019PEE013.

ABSTRACT A hybrid heuristic algorithm based on improved rules and reinforcement learning is proposed to solve the 2D strip packing problem (2DSPP). Firstly, the scoring rules based on the skyline algorithm are improved by considering the "two-step" successive items with a set of width relaxation factors. The improved scoring rules can reduce space waste efficiently. Secondly, as a reinforcement learning approach, the Deep Q-Network (DQN) is established to get the initial rectangular items sequence and at the same time as an essential supplement for the placement rules. It can improve space utilization and prevent the algorithm from falling into the local optimum. Combining the new "two-step" placement rules and DQN, the heuristic algorithm based on simple random algorithm (SRA) is proposed and finally called reinforcement learning based simple random algorithm (RSRA). Experiments on eight datasets by five algorithms have been conducted for comparison. Results show the RSRA has achieved the best performance on eight datasets (C, N, CX, NT, 2sp, NP, ZDF, BWMV) and has dropped Ave. Gap% by 45.86%, 45.16%, 30.89% and 20.56% than GRASP, SRA, IA, ISH respectively. It can be concluded that the RSRA algorithm would achieve better performance than the other four algorithms on eight datasets, especially on the larger datasets.

INDEX TERMS 2D strip packing problem (2DSPP), heuristic algorithm, improved rules, deep Q-Network (DQN), reinforcement learning.

I. INTRODUCTION

As a typical combination optimization problem, the Packing Problem has been proven to be an NP-hard problem [1]. Packing problems with different constraints and objectives are widely used in the manufacturing, transportation, and computer industry. This paper focuses on the 2DSPP.

A. 2DSPP MATHEMATICAL MODEL

2DSPP in this paper are described as follows: Given a set of the rectangle of size $w_i \times h_i, i = 1 \cdots n$, and a strip with W width and infinite height. Let the lower-left corner of strip be the origin of the two-dimensional coordinate system, and put rectangular objects into the strip. The goal is to use the minimum height h of the strip, and all rectangles in strip meet the following conditions:

(1): rectangles cannot be overlapped.

(2): the rectangle edges must be parallel to the X-or Y-axis.

(3): each rectangle width cannot exceed the bounds of the strip.

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

(x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) represent the lower-left corner and the upper right corner of the rectangle respectively. The mathematical description [41] of 2DSPP is as follows:

$$min\ H$$

$$\begin{aligned}
s.t.\ &(1)\ H = \max\{y_{i2}, i = 1, 2, \ldots, n\};\\
&(2)\ \mathrm{x}_{i2} - x_{i1} = w_{\mathrm{i}} \text{ and } y_{i2} - y_{i1} = y_i, \quad i = 1, 2, \ldots, n;\\
&(3)\ \max\{x_{i1} - x_{j2}, x_{j1} - x_{i2}, y_{i1} - y_{j2}, y_{j1} - y_{j2}\} \geq 0\\
&\qquad\qquad i, j = 1, 2, \ldots, n,\ and\ i \neq j;\\
&(4)\ 0 \leq x_{i2} \leq (W - w_i)\ and\ y_{i1} \geq 0, \quad i = 1, 2, \ldots, n;
\end{aligned}$$

The first constraint indicates that the total height of the strip used is labeled as H, the second constraint indicates that the rectangles must be placed horizontally, the third constraint demonstrates that there is no overlap between the rectangles. The fourth constraint means that each rectangle must be in the strip.

B. PREVIOUS WORK

Researchers proposed several algorithms for the problem. Early scholars tried a bunch of exact algorithms. Beasley [2] proposed a tree-searching algorithm. Martello textitet al. [3] used the branch-and-bound algorithm. In 2019 Wei *et al.*

[4] tried to eliminate the conflict between the two items by branching first, and then used dynamic programming to solve the problem of two-constraint backpacks on the leaf node. Experiments have shown the algorithm was superior to the existing branch-and-bound method. Bezerra *et al.* [5] proposed the use of two mixed integer linear programming model. The results show that the model can produce more optimal solutions. However, exact algorithms are only more suitable for the problem with fewer samples.

Considering the time performance of the precise algorithm, heuristic and meta-heuristic algorithms are also introduced to 2DSPP solving. Meta-heuristic algorithms include genetic algorithms [6], [7], simulated annealing algorithms [8], [9], particle swarm optimization [10], [11]. Classic heuristic algorithms include the Bottom Left [12] algorithm, the Bottom Left [13] algorithm, the Best Fit [14] algorithm, Alvarez-Valdés [15] *et al.* proposed the Greedy Random Adaptive Search Algorithm (GRASP), which proved to be one of the best algorithms to solve the problem of 2DSPP. The study found that a single heuristic algorithm has the problem of premature convergence and falling into local optimum. On this basis, some scholars have put forward a hybrid heuristic algorithm, which combines the advantages of different heuristic algorithms, and effectively avoids the problems above. For example, Huang *et al.* [16] and others proposed the lowest level of the optimal fitting algorithm with memory and combined with PSO algorithms. The experiment shows that the hybrid heuristic algorithm is better than the single heuristic algorithm. Rakotonirainy [38] *et al.* proposed a hybrid approach in which the method of simulated annealing is combined with a heuristic construction algorithm. They also proposed the second algorithm involves application of the method of simulated annealing directly in the space of completely defined packing layouts, without an encoding of solutions. Experiments for the 2DSPP prove that these two algorithms are better than the existing meta-heuristic algorithms.

In addition, there are some excellent heuristic algorithms; for example. In 2011, Leung *et al.* [17] proposed to use a two-stage intelligent search algorithm(ISA) composed of local search(LS()) and simulated annealing. LS() just swaps two positions of items in a given sequence in turn. Then simulated annealing will be implemented to obtain a better solution in global. Yang *et al.* [18] improved Leung algorithm by replacing the simulated annealing algorithm with a SRA without setting any parameters. In 2016, Wei *et al.* [39] proposed an efficient improved algorithm (IA) in 2016, adding a greedy selection stage based on Leung *et al.* [17] and Yang *et al.* [18], and preferentially selecting a better initial solution to enter the local search phase. In 2017, Wei *et al.* [40] proposed an improved skyline-based heuristic algorithm (ISH). The ISH with a complexity of $O(n \cdot log(n))$ has proven to be superior to most heuristic algorithms. Although the heuristic algorithm can quickly obtain an approximate optimal solution, the algorithm is less popular and must be specially designed for different problems.

As an artificial intelligence approach, reinforcement learning (RL) method can directly extract useful information from the data, so that it can potentially learn better heuristic algorithms. RL has been widely used on combination optimization issues such as travelling salesman problem [19], vehicle routing problem [20], Graph Coloring [21], maximum independent set [22], and packing problems, etc. Bello [23] *et al.* proposed a combination optimization framework based on reinforcement learning, which has yielded good results on traveler issues and backpacking issues. In view of the complexity of the combinatorial optimization problems, the DQN [24] combining the reinforcement learning function and the artificial neural network has demonstrated its strong performance, especially for the Maximum Cut [25], the Maximum Common Subgraph [26] and other combinatorial optimization problems. A deep reinforcement learning framework based on dual DQN has been proposed for an online two-dimensional packing study [27]. Although reinforcement learning has better performance than heuristic algorithms on the problem of combinatorial optimization, the performance is influenced by the amount of training data.

C. OUR WORK

In this paper, a hybrid heuristic algorithm is proposed on DQN and SRA. The main contributions of this paper are as follows:

(1) Previous rules have just considered "single-step" item placement. In this paper, the "two-step" successive items sequence is considered to get better combination by introducing a set of width relaxation factors.

(2) In order to minimize the waste, a set of relaxation factors $(\alpha, \beta, 0 < \alpha < 1, 0 < \beta < 1)$ is designed to give different fitness values in multiple width intervals.

(3) Introduction of another "Scorer" based on reinforcement learning. DQN is used to establish the evaluation function to get the placement sequence score of rectangular items. Compared with other simple sequences sorted by the perimeter, width, etc., DQN can not only generate initial sequence, but also improve overall space utilization. Another advantage is preventing the heuristic algorithm from falling into the local optimum and reducing the number of iterations.

(4) Algorithm fusion. The RSRA combining DQN and SRA is proposed without too many parameters in which the scoring rule is determined by combining the improved scoring rules (*Scorer I*) and DQN (*Scorer II*).

The remainder of this paper is organized as follows. Part II presents the hybrid heuristic algorithm based on improved scoring rules of skyline algorithm and the evaluation function based on DQN combined with SRA. Part III has shown experiments and analysis. The last part includes a conclusion and a prospect.

II. HYBRID HEURISTIC ALGORITHM BASED ON REINFORCEMENT LEARNING

A. SKYLINE ALGORITHM

As shown in Fig.1, the placement space on the lowest and the leftmost candidate line segment s_{lowy} (give priority to

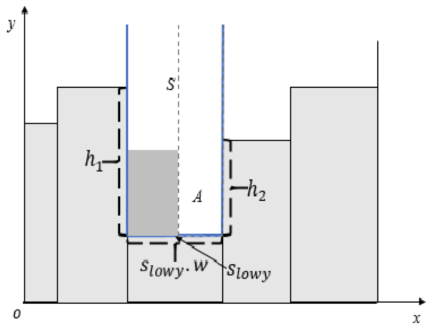


FIGURE 1. Example of rectangular item placement.

"the lowest") is labeled as S (in the blue box in Fig. 1). The height difference of the horizontal segment adjacent to s_{lowy} the larger is h_1 and the smaller is h_2. The width of the s_{lowy} is recorded as $s_{lowy}.w$. The height and width of the candidate rectangles are recorded as $r.h$ and $r.w$. The smallest rectangle in the unplaced item sequence is recorded as $r_{\min}$ and its width is recorded as $r_{\min}.w$. The space remaining after placing rectangular items in S is recorded as A (in the dotted box in the Fig. 1) and the width is labeled $A.w$.

The skyline algorithm [28] is used as the basic framework to determine the arrangement rules of rectangular items. The algorithm steps are as follow: for a given rectangular items sequence, repeat the following four steps until all rectangles are placed: 1) Find the s_{lowy} as the initial position to place items; 2) Calculate the fitness value for each item in turn from the original sequence; 3) Place the rectangular item with the highest fitness value in the S; 4) update the skyline.

B. NEW SKYLINE SCORING RULES

The disadvantages of Yang's rules [18] and Gao's rules [29] are as follows:

(1) There is no restriction or control over the width of the item to be placed in Yang's rules. So items with smaller widths may be put first, which will waste more space than putting items with larger widths. It is not reasonable that putting items with different widths in the same position getting the same fitness values in Yang's rules.

(2) Gao [29] argued that space waste on width is inevitable or even necessary. A relaxation factor α is introduced specially to ensure a wider item to place first to minimize the width waste. However, there might be such a case that the remaining space after putting the wider item can also be put into one more item with minimum width. (See Fig.2(a)–(b)). So the rules should be improved.

(3) For the height controlling, Gao set items with large height to low fitness values by introducing a relaxation factor in the height dimension. So shorter item would be placed prior to the higher one, and it results in small s_{lowy} (See Fig. 3(a)–(b)). However, the actual testing results on public datasets are not ideal. The main reason is if the following item to place has a bigger height, the final s_{lowy} from the two items may be higher, and that leads to the increase of the overall height finally (See Fig. 3(c)–(d)).

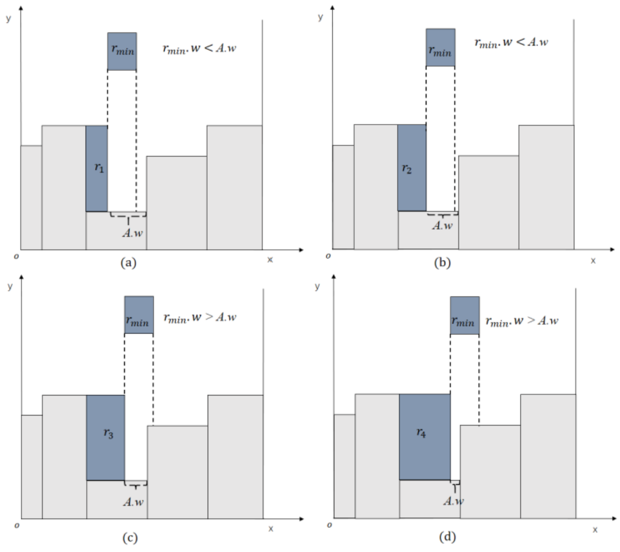


FIGURE 2. "Two-step" situations of considering $r_{\min}$.

Actually, previous rules have just considered "single-step" item placement. In this paper, the two successive items are considered together to get a better combination to place by introducing a set of width relaxation factors. The main improvements are as follows:

(1) $A.w$ represents the width of the remaining space after placing one item. $r_{min}.w$ is the width of the smallest item from the remaining unplaced items. $r_{\min}.w \leq A.w$ means that $r_{\min}$ can also be placed here to further reduce the space waste. As Fig. 2(a)–(b) show, placing r_1 or r_2 will result in the same score. Fig.2(c)-(d) are showing it's evitable to cause a waste on width when $r_{min}.w > A.w$. In this case, item r_4 with a larger width will be placed priority over r_3 to reduce the width waste.

(2) Since that space waste is unavoidable to some extent, so the wider of the item to place, the higher the fitness value should be. In order to minimize the waste, a set of relaxation factors ($\alpha, \beta, 0 < \alpha < 1, 0 < \beta < 1$) is designed to give different fitness values in multiple width intervals. The optimal values $\alpha = 0.2$ and $\beta = 0.5$ are obtained by a number of experiments.

(3) Height relaxation factor is abandoned to avoid excessive height value after combination placement (Fig. 3). Considering two situations of when $r.w = s_{lowy}.w$ as Fig. 4(a)–(b) ($r.h = h_1 vs. r.h = h_2$), the fitness value of the former is set higher so that it can make the space more flat which is more conducive to place the next item. The overlapping rules in literature [29] are also considered in the new rules.

The details are shown in Table 1 and Figure 4(a)-(f).

C. REINFORCEMENT LEARNING METHODS

Reinforcement learning is based on environmental feedback mechanisms. Through constant interaction with the environment, trial and error, reinforcement learning achieves maximizes the benefits of the overall action.

Q-learning algorithm is one of the value-based algorithms in reinforcement learning. Q(s, a) is the reward value that

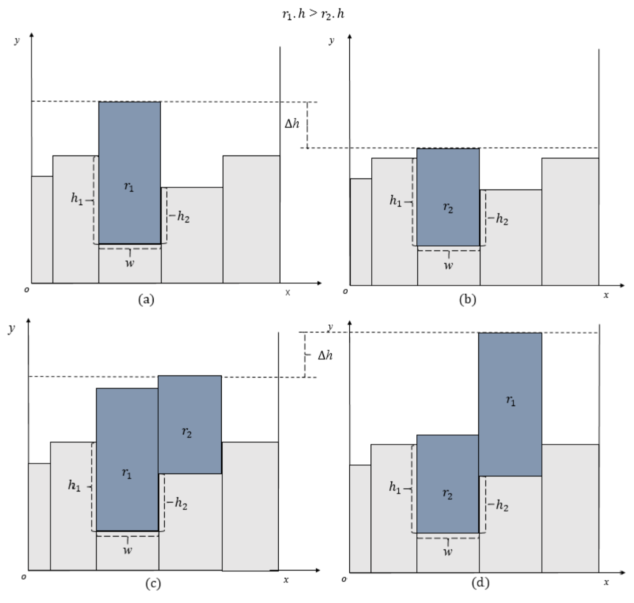


FIGURE 3. Possible bad effects from controlling height in "one-step".

TABLE 1. Placement rules as *Scorer I*.

SITUATION	FITNESS VALUE
$r.w = s_{lowy}.w \text{ and } r.h = h_2$	7
$r.w = s_{lowy} \text{ and } r.h = h_1$	6
$r.w = s_{lowy}.w \text{ and } r.h \neq h_1 \text{ and } r.h \neq h_2$	5
$r.h = h_1 \text{ or } r.h = h_2 \text{ and } A.w > r_{\min}.w$	4
$r.h = h_1 \text{ or } r.h = h_2 \text{ and } A.w \leq 0.2r_{\min}.w$	3
$r.h = h_1 \text{ or } r.h = h_2 \text{ and } 0.2r_{\min}.w < A.w \leq 0.5r_{\min}.w$	2
$r.h = h_1 \text{ or } r.h = h_2 \text{ and } 0.5r_{\min}.w < A.w < r_{\min}.w$	1
$r.w \neq s_{lowy}.w \text{ and } r.h \neq h_1 \text{ and } r.h \neq h_2$	0

represents the execution of action a in the case of state s. Q function selects the maximum action execution value in the case of the state s. As a table function in the Q-learning algorithm, Q-table is not suitable for 2DSPP. So the DQN is used instead of Q-table for Q-learning. The principle of the DQN is to use an artificial the neural network to replace the action-value function. The neural network is much more powerful than artificial feature search because it has strong expression ability and can automatically extract features. The network structure used in this paper has been shown in Fig. 5.

Data pre-processing approaches are shown in Table 2. For the input data, h_{large} is calculated from the height difference between $r.h$ and h_1. h_{small} is calculated from the height difference between $r.h$ and h_2. W is calculated from the difference between the width of the $r.w$ and the width of the space to be placed $s_{lowy}.w$. h_r is the ratio of $r.h$ to $r_{h\max}.h$ which is the maximum height of the remaining items. And w_r is the ratio of $r.w$ to the $r_{w\max}$ which is the maximum width of the remaining items.

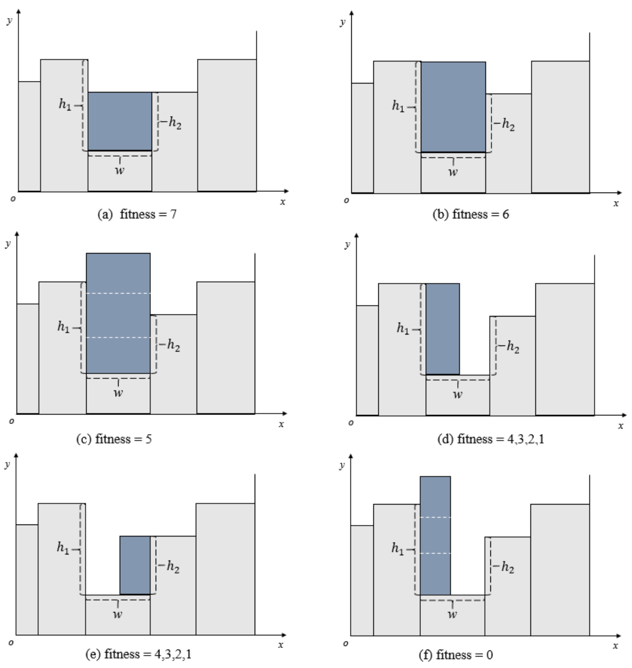


FIGURE 4. Fitness values in proposed new rules (*Scorer I*).

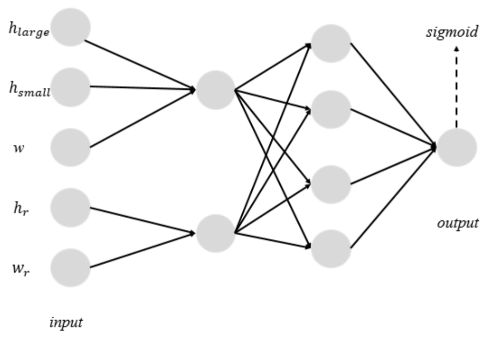


FIGURE 5. Neural network as *Scorer II*.

The output value of network is the matching degree of between the rectangle r and the space S. The three attributes h_{large}, h_{small} and w have an intermediate value through linear transformation and activation function. Similarly, the other two attributes h_r and w_r can also have the corresponding intermediate value by the same operations. Then the two intermediate values first undergo a linear transformation of 2×4 and activation function tanh, and then undergo a linear transformation of 4×1 and activation function sigmoid to obtain the final result. The value range of sigmoid is (0, 1).

D. DQN-BASED HEURISTIC ALGORITHM (RSRA)

DQN has been used to construct the evaluation function. The experimental results show that the performance of this algorithm is better than the sequence of items sorted by width, length, area, or perimeter under most datasets. Algorithm 1 has shown the details: (1) The first line shows items sequence will be sorted for preliminary selection by an algorithm which is similar to the greedy selection stage in IA [39]: the skyline algorithm will be implemented respectively after

TABLE 2. Data preprocessing for DQN.

DATA	APPROACH
h_{large}	$1-tanh(\frac{r.h-h_1}{r.h+h_1}\bullet\frac{h_1}{h_1+h_2})^2$
h_{small}	$1-tanh(\frac{r.h-h_2}{r.h+h_2}\bullet\frac{h_2}{h_1+h_2})^2$
W	$\frac{s_{lowy}.w-r.w}{s_{lowy}.w}$
h_r	$\frac{r.h}{r_{h\,max}.h}$
w_r	$\frac{r.w}{r_{w\,max}.w}$

Algorithm 1 RSRA

```
Input: Rectangular item sequence $R(r_1, r_2, \ldots, r_n)$
Output: besth

1: $R \leftarrow MinH\{skyline(R_P), skyline(R_a), skyline(R_h),$
   $skyline(R_W), skyline(R_{SII})\}$;
2: LS($R$)
3: While No longer than the required program run time do
4:     for i←1 to n do
5:         Randomly select two sequence numbers $j$ and $k$ in
           $R$;
6:         Exchange $j$ and $k$ in Order r to obtain the new
           sequence $R^*$;
7:         currenth←$skyline(R*, Scorer\ I, Scorer\ II)$;
8:       if current $<$ besth then
9:         besth←currenth;
10:          $R \leftarrow R^*$;
11:      else
12:          $p$ ←currenth/(currenth+besth);
13:          if $p$ <random(0,1) then
14:            $R \leftarrow R^*$;
15:          end if
16:      end if
17:   end for
18: end while
19: return besth;
```

generating the origin sequence from FIVE indexes: perimeter, area, width, length (from *Scorer I*) and the *Scorer II*. The solution of the smallest height in the FIVE sequences will be saved for the next stage. (2) The second line shows a local search algorithm (LS()) proposed by Leung *et al.* [17] is used for a further better solution. The LS() will swap the two items in a given sequence in turn and implement the skyline algorithm to get a "local best" solution. (3) For the line 3-line 19, the SRA will be used to improve the solution ultimately. It should be noted that *line 7* combines *Scorer I* and *Scorer II* to make decision: *Scorer II* will be used first. *Scorer II* rule will be used only if the scores from *Scorer I* are equal. And if the scores from Scorer II are also equal, the items will be placed in order.

III. EXPERIMENTAL ANALYSIS

A. NETWORK MODEL TRAINING

For the training of neural network *Scorer II*, C [30], NT [31], CX [32], BWMV [13], [33], N [14], ZDF [34] and NP [35] and 500 groups of data sets are generated based on the algorithm proposed by Bortfeldt and Gehring [37]. Randomly select 80% of the data as *TrainSet* and 20% as *TestSet*. The training process is as follows:

(1) At the beginning of the training, randomly select n sets of data from *TrainSet* to get the *TrainSet*∗ as the dataset used in this cycle, and then sequentially pack each set of data in *TrainSet*∗, which is a sequence R composed of rectangular items.

(2) Randomly select a number m from (0, num) in which num is the number of items in R. Use skyline algorithm under *Scorer II* to pack m items from R. Mark the lowest line here ass_{lowy}. Let k be equal to $num - m$. A very small k value would result in low credibility of the DQN. Set a threshold t for k.$k < t$ would skip to the next loop. Otherwise the k items at this time are processed according to the data pre-processing method of Table 2 and recorded as x_j.

(3) Try to place each x_j ($j = 1 \ldots k$) at s_{lowy} to execute the skyline algorithm and get the final height h_j. Record the maximum height $h_{\max}$ and the minimum height $h_{\min}$.

(4) Use the equation $y_j = \frac{h_{\max}-h_j}{h_{\max}-h_{\min}}$ to calculate y_j from each x_j. Save all x_j and y_j.

(5) Finally, use the saved x_j as input and y_j as the target value to train the *Scorer II*. The training would end when the set time or the optimal effect has been achieved. The detail is described as Algorithm 2.

B. EXPERIMENTAL DATA AND EXPERIMENTAL ENVIRONMENT

Current excellent algorithms including GRASP [15], SRA [18], IA [39] and ISH [40] have been compared with RSRA in order to verify the performance. At the same time, according to the method used by Lenug *et al.* [17], the problem can be subdivided into two: the zero-waste problem and non-zero waste problem. The optimal solution to the zero-waste problem is already known, while the optimal solution to the non-zero waste problem is mostly unknown which will generally contain some waste space. The C, N, CX and NT datasets are used for the zero-waste problem. And the non-zero waste problem will be tried to be solved on the 2sp [2], [32], [36], NP, ZDF, and BWMV datasets.

The computer is with Intel(R) Core(TM) i7-10750H CPU 2.60 GHz, RAM 16GB. The results from GRASP, SRA, IA and ISH are based on the literatures [15], [18], [39] and [40] respectively.

C. EXPERIMENTAL RESULT

All instances of the eight datasets and the best solutions from the RSRA can be found in the website (https://github.com/Gitlixiangdong/BestSolutios). Tables 3-10 in the Appendix

Algorithm 2 Network Model Training

Input: *TrainSet*
Output: *Scorer II*

```
1: Random initialization neural network $Scorer\ II$
2: While No longer than the required program run time or
     the desired results do
3:     Randomly select n sets of data from $TrainSet$ as
         $TrainSet*$
4:     for i←1 to n do
5:         $R \leftarrow TrainSet*$[i]
6:         $num \leftarrow$ number of items in $R$
7:         $m \leftarrow$ randomint(0,num)
8:         $Scorer\ II$ is used as a scoring rule to pack
            $m$ rectangular items in $R$ by skyline algorithm
9:         $k \leftarrow num-m$
10:        if $k < t$ then
11:            Continue
12:        end if
13:        for j← 1 to $k$ do
14:            Put the $j$-$th$ item in the sequence of $k$ items into
                 $s_{lowy}$
15:            Let the input of item j and the current position
                 data be preprocessed as $x_j$
16:            Continue with the skyline algorithm until all
                 rectangular items are packed and get $h_j$
17:        end for
18:        $h_{\min} \leftarrow \min(h_j)$, $h_{\max} \leftarrow \max(h_j)$
19:        for j← 1 to k do
20:            $y_j = \frac{h_{\max}-h_j}{h_{\max}-h_{\min}}$
21:            Save $x_j$ and $y_j$ to $x$, $y$
22:        end for
23:    Train the $Scorer\ II$ based on the obtained $x$ as input
       and $y$ as output
24: end while
25: return $Scorer\ II$
```

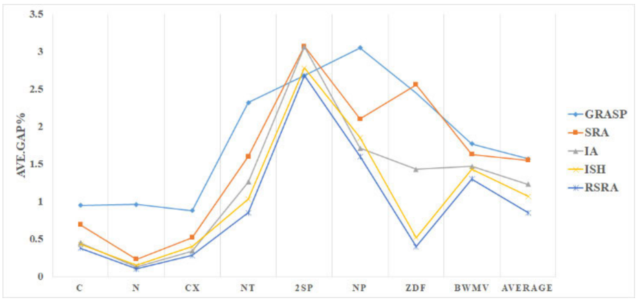


FIGURE 6. Ave. Gap% comparison of five algorithms.

are showing the experimental results from RSRA, GRASP, SRA, IA and ISH on the eight datasets respectively. The meanings of the symbols in the tables are as follow: n represents the number of rectangular items. W represents the width of the rectangular plate. LB represents the lower bound value of the problem instance. Gap% is defined the same as Leung *et al.* [17], namely Gap% $= 100 \times (BestH - LB)/LB$, where

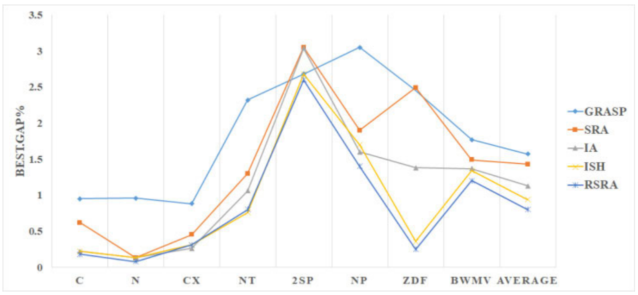


FIGURE 7. Best. Gap% comparison of five algorithms.

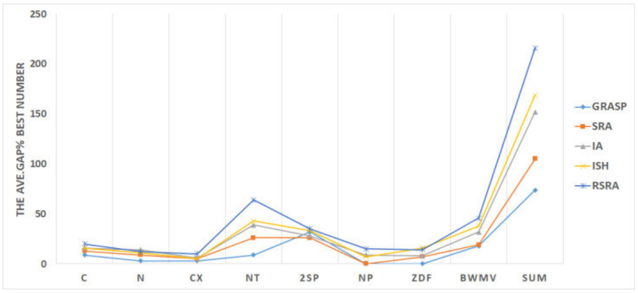


FIGURE 8. Count numbers of optimal solutions from Ave. Gap%.

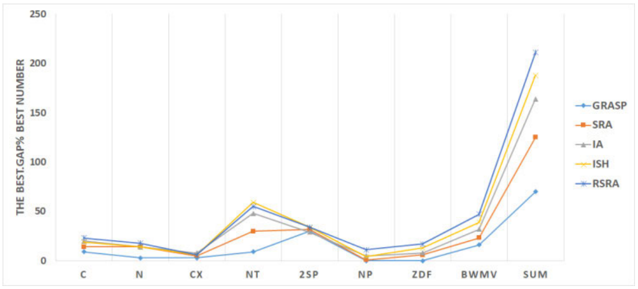


FIGURE 9. Count numbers of optimal solutions from the Best. Gap%.

BestH is the best height found in one run. Ave. Gap% and Best. Gap% denote the average Gap and the best (smallest) Gap during 10 runs for each instance respectively [18].

Figures 6 & 7 have shown the average values of Ave. Gap% and Best. Gap% on eight datasets from five algorithms. For the Ave. Gap% as shown in Fig.6, the RSRA algorithm has achieved the best results on all the 8 datasets comparing to other four algorithms. The average Ave. Gap% value of RSRA algorithm is 45.86%, 45.16%, 30.89% and 20.56% lower than that of GRASP, SRA, IA, and ISH respectively. For the Best. Gap% as shown in Fig.7, the RSRA algorithm has obtained the best (smallest) values on 6 data sets (on dataset of C, N, CX, NT, NP, BWMV) out of 8 ones. And the average Best. Gap% value of RSRA algorithm is 48.86%, 35.21%, 18.76% and 9.57% lower than that of GRASP, SRA, IA, and ISH respectively.

Furthermore, the values showing the optimal solutions between all the five algorithms for each instance are labeled in bold in Tables 4-11. And figures 8 & 9 have shown the count numbers of optimal solutions with the Ave. Gap% and Best. Gap% values obtained on all the data sets of the five

TABLE 3. Dataset C calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
C11	16	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C12	17	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C13	16	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C21	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C22	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C23	25	40	15	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C31	28	60	30	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C32	29	60	30	3.3	3.3	1.3	**0**	1.3	3.3	3.3	**0**	**0**	**0**
C33	28	60	30	**0**	**0**	**0**	2	**0**	**0**	**0**	**0**	**0**	**0**
C41	49	60	30	1.7	1.7	1.7	0.7	**1**	1.7	1.7	1.7	**0**	1
C42	49	60	30	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**	**1.7**
C43	49	60	30	1.7	1.7	**0**	**0**	**0**	1.7	1.7	**0**	**0**	**0**
C51	73	60	90	1.1	0.9	0.8	0.2	**0**	1.1	**0**	**0**	**0**	**0**
C52	73	60	90	1.1	0.4	**0**	**0**	**0**	1.1	**0**	**0**	**0**	**0**
C53	73	60	90	1.1	1.1	**1**	**1**	**1**	1.1	1.1	**0**	**0**	**0**
C61	97	80	120	1.7	0.8	**0.7**	0.8	0.8	1.7	0.8	**0**	0.8	**0**
C62	97	80	120	0.8	0.8	**0.3**	0.6	**0.3**	0.8	0.8	**0**	**0**	**0**
C63	97	80	120	1.7	0.8	0.8	0.8	**0.6**	1.7	0.8	**0**	0.8	**0**
C71	196	160	240	1.7	**0.4**	**0.4**	**0.4**	**0.4**	1.7	**0.4**	**0.4**	**0.4**	**0.4**
C72	197	160	240	1.3	**0.4**	**0.4**	**0.4**	**0.4**	1.3	**0.4**	**0.4**	**0.4**	**0.4**
C73	196	160	240	1.3	**0.4**	**0.4**	**0.4**	**0.4**	1.3	**0.4**	**0.4**	**0.4**	**0.4**
Average				0.95	0.69	0.45	0.43	**0.38**	0.95	0.62	0.22	0.22	**0.2**

TABLE 4. Dataset N calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
N1	10	40	40	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
N2	20	30	50	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
N3	30	20	50	2	**0**	**0**	**0**	**0**	2	**0**	**0**	**0**	**0**
N4	40	80	80	1.3	**0**	**0**	**0**	**0**	1.3	**0**	**0**	**0**	**0**
N5	50	100	100	2	**0**	**0**	**0**	**0**	2	**0**	**0**	**0**	**0**
N6	60		100	1	0.2	**0**	0.1	**0**	1	**0**	**0**	**0**	**0**
N7	70	80	100	1	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**
N8	80	100	80	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**
N9	100	50	150	0.7	0.6	**0**	0.2	**0**	0.7	**0**	**0**	**0**	**0**
N10	200	70	150	0.7	0.4	**0**	**0**	**0**	0.7	**0**	**0**	**0**	**0**
N11	300	70	150	0.7	0.1	**0**	**0**	**0**	0.7	**0**	**0**	**0**	**0**
N12	500	100	300	1.3	0.3	0.3	0.3	**0**	50 1.3	**0.3**	**0.3**	**0.3**	**0.3**
N13	3152	640	960	0.5	**0.1**	**0.1**	**0.1**	**0.1**	0.5	**0.1**	**0.1**	**0.1**	**0.1**
Average				0.96	0.23	**0.13**	0.15	0.11	0.96	**0.13**	**0.13**	**0.13**	**0.13**

TABLE 5. Dataset CX calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
50cx	50	400	600	2.2	0.3	**0**	0.2	**0**	2.2	0.3	**0**	**0**	**0**
100cx	100	400	600	2.8	3.3	2.4	2.6	**2**	2.8	2.8	**1.8**	2.2	2.2
500cx	500	400	600	0.8	**0**	**0**	**0**	**0**	0.8	**0**	**0**	**0**	**0**
1000cx	1000	400	600	0.3	**0**	**0**	**0**	**0**	0.3	**0**	**0**	**0**	**0**
5000cx	5000	400	600	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
10000cx	10000	400	600	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
15000cx	15000	400	600	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Average				0.88	0.52	0.34	0.4	**0.31**	0.88	0.45	**0.26**	0.31	0.31

algorithms. For the Ave. Gap%, the RSRA algorithm has the largest number of optimal solutions on 6 datasets (C, CX, NT, 2sp, NP, BWMV) out of 8 ones, and has the largest number of optimal solutions in SUM. For the Best. Gap%, the RSRA algorithm has the largest number of optimal solutions on 5 datasets (C, N, NP, ZDF, BWMV) out of 8 ones, and has

TABLE 6. Dataset NT calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
n1a	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
n1b	17	200	200	4.5	**0**	**0**	**0**	**0**	4.5	**0**	**0**	**0**	**0**
n1c	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
n1d	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
n1e	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
n2a	25	200	200	3	**0**	1.3	**0**	**0**	**3**	**0**	**0**	**0**	**0**
n2b	25	200	200	3	**0**	2.6	**0**	**0**	3	**0**	2.5	**0**	**0**
n2c	25	200	200	4	1.5	2.2	1.4	1.4	4	**0**	2	**0**	**0**
n2d	25	200	200	4.5	2.9	**0**	0.3	0	4.5	2.5	**0**	**0**	**0**
n2e	25	200	200	3	2.6	2.6	**0**	0	3	**0**	2.5	**0**	1.6
n3a	29	200	200	4.5	0	1.7	2.7	0	4.5	**0**	**0**	**0**	**0**
n3b	29	200	200	4	4.3	3.5	3.4	**3**	4	3.5	3.5	**3**	**3**
n3c	29	200	200	2.5	**0**	2.9	2.7	**0**	2.5	**0**	2.5	2	**0**
n3d	29	200	200	3.5	2	**0**	**0**	**0**	3.5	2	**0**	**0**	2
n3e	29	200	200	3.5	3.6	2.5	1.8	1.8	3.5	3.5	2.5	**0**	1.8
n4a	49	200	200	3	2.6	**2**	2.2	2.2	3	**2**	1.5	**2**	**2**
n4b	49	200	200	3.5	2.2	2.8	2.5	2.2	3.5	**2**	2.5	**2**	**2**
n4c	49	200	200	2.5	2.6	2	2	2	2.5	2.5	**1.5**	**1.5**	2
n4d	49	200	200	3	2.6	**1.5**	1.6	**1.5**	3	2	**1.5**	**1.5**	**1.5**
n4e	49	200	200	2.5	3.3	1.9	**1.5**	**1.5**	2.5	3	**1.5**	**1.5**	**1.5**
n5a	73	200	200	2.5	2.4	1.8	1.6	**1**	2.5	2	1.5	1.5	**1**
n5b	73	200	200	2	1.9	1.6	**1.4**	**1.4**	2	1.5	1.5	**1**	**1**
n5c	73	200	200	3	2.1	1.8	1.9	**1.5**	3	2	**1.5**	**1.5**	**1.5**
n5d	73	200	200	2	2.3	1.9	2	**1.5**	2	2	**1.5**	**1.5**	**1.5**
n5e	73	200	200	3	2.4	1.6	1.7	**1.6**	2	2	1.5	**1**	1.6
n6a	97	200	200	2	1.5	1.4	1.2	1.2	2	1.5	**1**	**1**	1.2
n6b	97	200	200	2	1.5	**1**	1.1	**1**	2	1.5	**0.5**	**1**	1
n6c	97	200	200	2	1.7	1	1	1	2	1.5	**0.5**	1	1
n6d	97	200	200	2.1	1.5	1.2	1.3	1.2	2	1	1	1	1.2
n6e	97	200	200	2	1.4	1.1	1.3	1.1	2	1	1	1	1.1
n7a	199	200	200	1	**0.5**	**0.5**	**0.5**	**0.5**	1	**0.5**	**0.5**	**0.5**	**0.5**
n7b	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
n7c	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
n7d	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
n7e	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	**1.5**	**0.5**	**0.5**	**0.5**	**0.5**
t1a	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t1b	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t1c	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t1d	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t1e	17	200	200	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t2a	25	200	200	2	3.4	**0**	**0**	**0**	2	3	**0**	**0**	**0**
t2b	25	200	200	4	2.1	**0**	**0**	**0**	4	**0**	**0**	**0**	**0**
t2c	25	200	200	4	3.1	**0**	**0**	**0**	4	3	**0**	**0**	**0**
t2d	25	200	200	3	1.6	2.6	**0**	**0**	3	0	2.5	**0**	**0**
t2e	25	200	200	3	3	2.6	**0**	**0**	3	2.5	2.5	**0**	**0**
t3a	29	200	200	3.5	1.7	0	**0**	**0**	3.5	0	0	**0**	**0**
t3b	29	200	200	4.5	3.5	3.5	3.1	3	4.5	2.5	3.5	**0**	**0**
t3c	29	200	200	3	3.1	0.5	**0**	0.	3	2.5	**0**	**0**	**0**
t3d	29	200	200	3.5	**0**	2	**0**	**0**	3.5	3	2	**0**	**0**
t3e	29	200	200	4	2.5	3.5	3.4	2.5	4	1.5	3.5	3	2
t4a	49	200	200	2.5	2.6	2.4	2.2	2.2	2.5	**1**	2	2	2
t4b	49	200	200	2.5	3.2	2	2	2	2.5	2	**1.5**	**1.5**	1.8
t4c	49	200	200	3	2.3	**1.5**	1.6	1.6	3	2	**1.5**	**1.5**	1.6
t4d	49	200	200	3	2.6	**2**	2.2	**2**	3	**1.5**	**1.5**	2	2
t4e	49	200	200	3.5	**2.2**	2.8	2.5	**2.2**	3.5	2	2.5	**2**	**2**
t5a	73	200	200	3	2.2	**1.8**	1.9	**1.8**	3	2	**1.5**	**1.5**	1.6
t5b	73	200	200	2	2.1	**1.4**	1.5	**1**	2	1.5	**1**	**1**	**1**
t5c	73	200	200	2.5	2.3	1.8	**1.6**	**1**	2.5	2	**1.5**	**1.5**	**1.5**

the largest SUM number of optimal solutions in all the five algorithms.

The results can be analyzed further according to the size of the dataset. As shown in Fig.6, even though the results

TABLE 6. (Continued) Dataset NT calculation results.

t5d	73	200	200	2	2.4	**1.9**	2	2	2	2	**1.5**	**1.5**	**1.5**
t5e	73	200	200	2	2.3	1.6	**1.4**	**1.2**	2	2	1.5	**1**	**1**
t6a	97	200	200	2	2	**1.1**	**1.1**	**1.1**	2	1.5	**1**	**1**	**1**
t6b	97	200	200	2	1.5	**1**	1	**1**	2	1.5	**0.5**	1	**0.5**
t6c	97	200	200	2	1.5	**1.1**	1.3	**1.1**	2	1.5	**1**	**1**	**1**
t6d	97	200	200	2	1.8	**1**	1.1	**1**	2	1.5	**0.5**	1	**0.5**
t6e	97	200	200	2.5	1.7	**1**	1.1	**1**	2.5	1.5	**1**	**1**	**1**
t7a	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
t7b	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
t7c	199	200	200	2	**0.5**	**0.5**	**0.5**	**0.5**	2	**0.5**	**0.5**	**0.5**	**0.5**
t7d	199	200	200	1	**0.5**	**0.5**	**0.5**	**0.5**	1	**0.5**	**0.5**	**0.5**	**0.5**
t7e	199	200	200	1.5	**0.5**	**0.5**	**0.5**	**0.5**	1.5	**0.5**	**0.5**	**0.5**	**0.5**
Average				2.32	1.6	1.26	1.03	**0.85**	2.32	1.3	1.06	**0.76**	0.8

TABLE 7. Dataset 2sp calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
cgcut1	16	10	23	**0**	4.4	4.3	**0**	**0**	**0**	4.4	4.3	**0**	**0**
cgcut2	23	70	63	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
cgcut3	62	70	636	3.9	4	3.9	3.7	3.7	3.9	3.9	3.9	3.5	3.3
gcut1	10	250	1016	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
gcut2	20	250	1133	5.1	**4.8**	5	4.9	**4.8**	5.1	**4.8**	**4.8**	**4.8**	**4.8**
gcut3	30	250	1803	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
gcut4	50	250	2934	**2.3**	2.4	**2.3**	2.5	**2.3**	2.3	2.4	2.3	**2.2**	**2.2**
gcut5	10	500	1172	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**	**8.6**
gcut6	20	500	2514	4.5	4.5	**4.3**	**4.3**	**4.3**	4.5	**4.3**	**4.3**	**4.3**	**4.3**
gcut7	30	500	4641	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**
gcut8	50	500	5703	3.7	3.2	**2.9**	**2.9**	**2.9**	3.7	**3**	2.8	2.7	**2.5**
gcut9	10	1000	2022	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**	**14.6**
gcut10	20	1000	5356	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**	**11.4**
gcut11	30	1000	6537	5.5	**5**	5.1	5.1	**5**	5.5	**5**	**5**	5.1	**5**
gcut12	50	1000	12522	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**	**17.3**
gcut13	32	3000	4772	4.7	4.2	**4.1**	**4.1**	**4.1**	4.7	4.1	4.2	**4**	**4**
ngcut1	10	10	23	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut2	17	10	30	**0**	3.3	3.3	**0**	**0**	**0**	3.3	3.3	**0**	**0**
ngcut3	21	10	28	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut4	7	10	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut5	14	10	36	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut6	15	10	29	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9
ngcut7	8	20	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut8	13	20	32	**3.1**	6.3	6.3	6.3	3.1	**3.1**	6.3	6.3	6.3	6.3
ngcut9	18	20	49	**2**	4.1	4.1	**2**	**2**	**2**	4.1	4.1	**2**	**2**
ngcut10	13	30	80	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ngcut11	15	30	50	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
ngcut12	22	30	87	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng1	20	25	30	**0**	3.3	3.3	2.7	2.5	**0**	3.3	3.3	**0**	**0**
beng2	40	25	57	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng3	60	25	84	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng4	80	25	107	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng5	100	25	134	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng6	40	40	36	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng7	80	40	67	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng8	120	40	101	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng9	160	40	126	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
beng10	200	40	156	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Average				**2.68**	3.07	3.06	2.78	**2.68**	2.68	3.05	3.04	2.68	**2.67**

of RSRA are much better than that of the other four algorithms, they are very close to that of GRASP and ISH. This is probably because the dataset of 2sp is so small (most of the instances are with $n \leq 50$). The CX and ZDF datasets have contained of instances with $n \geq 10,000$. And the performance of the RSRA seems much better than the other four algorithms. The count numbers of problem instances in other five datasets (C, N, NT, NP, BWMV) are mostly with $50 \leq n \leq 200$. RSRA algorithm is also effective on datasets with medium problem size. Therefore, it can be concluded

TABLE 8. Dataset Np calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
Nice1	25	1000	1000	3.4	3.7	**3.4**	3.6	**3.4**	3.4	3.5	2.8	3.5	**3**
Nice2	50	1000	1001	4.6	3.7	**2.9**	3.5	**2.9**	4.6	3	**2.6**	3.4	2.9
Nice3	100	1000	1001	4	3.3	**2.5**	3.1	**2.5**	4	3.1	2.3	2.6	**2.2**
Nice4	200	1000	1001	3.6	2.2	1.9	2.3	**1.8**	3.6	2.2	**1.7**	2.2	**1.7**
Nice5	500	1000	1000	2.4	0.8	**0.6**	0.7	0.7	2.4	0.8	**0.6**	0.7	**0.6**
Nice6	1000	1000	999	2.1	0.3	0.5	0.4	**0.2**	2.1	0.3	0.5	0.3	**0.2**
Path1	25	1000	1001	4.1	4.2	4.1	**4**	**4**	4.1	4.2	4.1	3.3	**3.2**
Path2	50	1000	1000	1.9	0.9	**0.1**	**0.1**	**0.1**	1.9	**0.1**	**0.1**	**0.1**	**0.1**
Path3	100	1000	1000	2.7	2.3	1.9	**1.8**	**1.8**	2.7	2.2	1.8	**1.6**	**1.6**
Path4	200	1000	1002	2.1	1.4	**1.2**	**1.2**	**1.2**	2.1	1.4	1.1	**1**	**1**
Path5	500	1000	1000	3.4	1.4	**1**	**1**	**1**	3.4	1.4	1.4	0.9	**0.8**
Path6	1000	1000	1002	2.4	0.6	**0.5**	**0.5**	**0.5**	2.4	0.6	**0.5**	**0.5**	**0.5**
Average				3.05	2.1	1.71	1.85	**1.6**	3.05	1.9	1.6	1.7	**1.5**

TABLE 9. Dataset ZDF calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
zdf1	580	100	330	0.9	**0**	**0**	**0**	**0**	0.9	**0**	**0**	**0**	**0**
zdf2	660	100	357	0.8	**0**	**0**	**0**	**0**	0.8	**0**	**0**	**0**	**0**
zdf3	740	100	384	0.8	**0**	**0**	**0**	**0**	0.8	**0**	**0**	**0**	**0**
zdf4	820	100	407	0.7	**0**	**0**	**0**	**0**	0.7	**0**	**0**	**0**	**0**
zdf5	900	100	434	0.7	**0**	**0**	**0**	**0**	0.7	**0**	**0**	**0**	**0**
zdf6	1532	3000	4872	7.8	3.7	2.9	**2.8**	2	7.8	3.1	2.6	2.2	**2**
zdf7	2432	3000	4852	6.4	4.1	3.1	**2.7**	2.8	6.4	3.8	2.6	2.6	3.8
zdf8	2532	3000	5172	7.2	5.7	4.2	**0.9**	**0.5**	7.2	5.7	4.2	**0.5**	**0.5**
zdf9	5032	3000	5172	5.9	7	2	0.4	**0.1**	5.9	7	2	**0**	**0**
zdf10	5064	6000	5172	7.7	3.9	3.8	**0.1**	**0**	7.7	3.9	3.8	**0**	**0**
zdf11	7564	6000	5172	7.5	4.7	2.4	**0.5**	**0.5**	7.5	4.7	2.4	**0.1**	**0.1**
zdf12	10064	6000	5172	-	4.7	1.2	**0.5**	**0.5**	-	4.7	1.2	**0.1**	**0.1**
zdf13	15096	9000	5172	-	4.5	1.2	**0.5**	**0.5**	-	4.5	1.2	0.3	**0.2**
zdf14	25032	3000	5172	-	1.8	2.1	**0**	**0**	-	1.8	2.1	**0**	**0**
zdf15	50032	3000	5172	-	0.7	**0**	**0**	**0**	-	0.7	**0**	**0**	**0**
zdf16	75032	3000	5172	-	**0**	**0**	**0**	**0**	-	**0**	**0**	**0**	**0**
Average				-	2.56	1.43	0.52	**0.4**	-	2.49	1.38	**0.36**	0.45

that the RSRA algorithm would achieve better performance than the other four algorithms on eight datasets, especially on the relatively large datasets.

The RSRA algorithm has also been compared with the SPSAL and IAm algorithms proposed by Rakotonirainy *et al.* [38] in 2020. All the instances used in experiments are from the benchmark clusters proposed by Van Vuuren and Rakotonirainy [42]. Cluster1 consists of instances with predominantly narrow items of elongated rectangular shape, which are widely varying in size. Cluster2 mainly contains a number of items with large size and are predominantly homogeneous. Cluster3 mainly consists of approximately square items with different sizes. These items are much smaller than the width of the strip. Cluster4 mainly contains approximately square items with the same size.

Table 11 in the Appendix has shown the results from the RSRA compared with the SPSAL, IAm, GRASP, SRA, IA and ISH algorithms. Since Rakotonirainy *et al.* [38] did not provide the algorithm source code, so we apply RSRA algorithm on the data set from literature [38]. Mean performance ratios achieved by the various algorithms together with their ranks at a 5% level of significance have been shown in parentheses in the table. A rank of 1 indicates that the algorithm achieved the smallest mean packing height over the instances in a cluster of benchmark data. The other algorithm experiment results come from literature [38], the result data ranking is shown in parentheses. Table 11 has shown that the RSRA has achieved best results on all the four data clusters. Even on the "complex" Cluster1, RSRA has achieved very good result. And the performance of RSRA has no statistical difference from IAm on the other three clusters at the significance level of 5%.

Compared to IAm, the advantages of RSRA can be concluded as follow: (1) The scoring rules (*Scorer I*) used by the RSRA are more refined and reasonable than IAm. This makes it more effective to generate the best rectangular items sequence for placement. (2) DQN was used in RSRA to obtain the evaluation function as *Scorer II*. So the

TABLE 10. Dataset BWMV calculation results.

Instance				Ave.Gap%					best.Gap%				
	n	W	LB	GRASP	SRA	IA	ISH	RSRA	GRASP	SRA	IA	ISH	RSRA
C01	20	10	60.3	1.8	1.7	1.7	**1.5**	1.7	1.8	1.7	1.7	**1.5**	**1.5**
	40	10	121.6	**0.2**	**0.2**	**0.2**	**0.2**	**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**	**0.6**	**0.6**	**0.6**	**0.6**
	80	10	262.2	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
	100	10	304.4	**0.2**	**0.2**	**0.1**	**0.1**	**0.1**	**0.2**	**0.2**	**0.1**	**0.1**	**0.1**
C02	20	30	19.7	**0.5**	0.9	0.6	**0.5**	**0.5**	0.5	0.5	**0**	0.5	**0**
	40	30	39.1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	60	30	60.1	0.3	**0**	**0**	**0**	**0**	0.3	**0**	**0**	**0**	**0**
	80	30	83.2	0.1	**0**	**0**	**0**	**0**	0.1	**0**	**0**	**0**	**0**
	100	30	100.5	0.1	**0**	**0**	**0**	**0**	0.1	**0**	**0**	**0**	**0**
C03	20	40	157.4	3.9	4.1	4.1	3.4	**3**	3.9	3.9	3.9	3.4	**3**
	40	40	328.8	1.6	1.4	**1.3**	**1.3**	**1.3**	1.6	1.3	**1.2**	**1.2**	**1.2**
	60	40	500	1.3	1.1	**1**	**1**	**1**	1.3	**1**	**1**	**1**	**1**
	80	40	701.7	1.1	1.1	**1**	**1**	**1**	1.1	**1**	**1**	**1**	**1**
	100	40	832.7	0.9	0.6	**0.5**	**0.5**	**0.5**	0.9	0.5	0.5	**0.4**	**0.4**
C04	20	100	61.4	3.1	3.9	3.4	3.2	**3**	3.1	3.4	2.9	3.1	**2.8**
	40	100	123.9	1.9	1.6	**1.2**	**1.2**	**1.2**	1.9	1.3	**1**	**1**	**1**
	60	100	193	1.9	1.1	0.9	**0.8**	**0.8**	1.9	0.9	**0.8**	**0.8**	**0.8**
	80	100	267.2	1.8	0.9	0.7	**0.6**	**0.6**	1.8	0.7	0.5	**0.4**	0.6
	100	100	322	1.6	0.7	0.5	**0.4**	**0.4**	1.6	0.5	**0.3**	**0.3**	**0.3**
C05	20	100	512.2	**4.2**	4.3	4.3	4.3	4	**4.2**	**4.2**	4.3	**4.2**	**2**
	40	100	1053.8	2	1.9	1.9	1.8	**1.6**	2	1.9	1.9	1.8	**1.6**
	60	100	1614	2	**1.8**	**1.8**	**1.8**	**1.8**	2	**1.7**	**1.7**	**1.7**	**1.7**
	80	100	2268.4	1	1	**0.8**	0.9	**0.8**	1	0.9	0.8	0.8	**0.7**
	100	100	2617.4	1.3	1.3	1.1	**1**	**1**	1.3	1.1	**1**	**1**	**1**
C06	20	10	159.9	**4.6**	5.6	5.3	5	4.5	4.6	4.9	4.9	4.5	4.5
	40	10	323.5	3.1	3	**2.5**	2.6	**2.5**	3.1	2.6	2.3	2.2	**2**
	60	10	505.1	2.9	2.6	**2.2**	**2.2**	**2.2**	2.9	2.2	**2**	**2**	**2**
	80	10	699.7	2.7	2.2	**1.8**	1.9	**1.8**	2.7	2	**1.6**	1.7	**1.6**
	100	10	843.8	2.5	1.9	**1.6**	**1.6**	**1.6**	2.5	1.7	**1.4**	**1.4**	**1.4**
C07	20	30	490.4	**2.3**	**2.3**	**2.3**	**2.3**	**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**	**0.9**	**0.9**	**0.9**	**0.9**
	60	30	1515.9	**0.9**	**0.9**	**0.9**	**0.9**	**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**	**0.7**	**0.7**	**0.7**	**0.7**
	100	30	2627	**0.6**	**0.7**	**0.6**	**0.6**	**0.6**	**0.6**	**0.7**	**0.6**	**0.6**	**0.6**
C08	20	40	434.6	5.5	5.4	5.1	**5**	4.5	5.5	5.2	5	4.8	**4.5**
	40	40	922	3.5	3.2	2.7	2.8	**2.6**	3.5	2.9	**2.4**	**2.4**	2.6
	60	40	1360.9	3.2	2.8	2.4	**2.3**	**2.3**	3.2	2.5	2.1	2.1	**2**
	80	40	1909.3	3.3	2.7	2.3	2.1	**0**	3.3	2.4	2	**1.9**	2
	100	40	2362.8	3.1	2.3	1.9	**1.8**	**1.8**	3.1	2	1.7	**1.6**	**1.6**
C09	20	100	1106.8	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	40	100	2189.2	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
	60	100	3410.4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	80	100	4578.6	**0.2**	**0.2**	**0.2**	**0.2**	**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**	**0.1**	**0.1**	**0.1**	**0.1**
C10	20	100	337.8	3.8	3.6	3.6	**3.5**	**3.5**	3.8	3.5	3.5	**3.4**	**3.4**
	40	100	642.8	3.4	3.1	2.9	2.9	**2.8**	3.4	3	2.8	2.8	**2.6**
	60	100	911.1	2.6	2.4	2.1	2.1	**2**	2.6	2.3	2	**1.9**	2
	80	100	1177.6	2.7	2.4	**2**	**2**	**2**	2.7	2.1	1.8	1.8	**1**
	100	100	1476.5	2.4	2	**1.6**	**1.6**	**1.6**	2.4	1.8	1.5	**1.4**	**1**
Average				1.77	1.63	1.47	1.43	**1.3**	1.77	1.49	1.37	1.34	**1.2**

TABLE 11. Dataset cluster calculation results comparing to ref [38].

Mean(Rank)	GRASP	SRA	IA	ISH	SPSAL	IAm	RSRA
Cluster1	1.11(4)	1.04(2)	1.05(3)	1.04(2)	1.24(5)	1.04(2)	1.03(1)
Cluster2	1.11(5)	1.08(3)	1.07(2)	1.08(3)	1.10(4)	1.07(2)	1.06(1)
Cluster3	1.15(5)	1.10(3)	1.10(3)	1.10(3)	1.14(4)	1.09(1)	1.09(1)
Cluster4	1.12(2)	1.06(1)	1.06(1)	1.06(1)	1.16(2)	1.06(1)	1.06(1)

perimeter, area, width, height and the score from *Scorer II* are used to implement the skyline algorithm respectively. And the solution with the smallest height will be saved for the next stage. This ensures that the heuristic algorithm can obtain a better initial solution to improve the overall performance.

IV. CONCLUSION

This paper has proposed a hybrid heuristic algorithm for 2DSPP problem. Firstly, scoring rules based on skyline have been improved to reduce the space waste called *Scorer I*. Secondly, the reinforcement learning method has been used to enhance the local search ability and reduce the number of iterations called *Scorer II*. The *Scorer I and II* have been combined with a SRA to be the RSRA. Experiments show that compared with the other four excellent heuristic algorithms (GRASP, SRA, IA, ISH), the RSRA has achieved the best performance on eight datasets (C, N, CX, NT, 2sp, NP, ZDF, BWMV) and has dropped the Ave. Gap% by 45.86%, 45.16%, 30.89% and 20.56% than GRASP, SRA, IA, ISH respectively. It can be roughly concluded that the RSRA would achieve better performance than the other four algorithms on eight datasets, especially on the relatively large datasets.

Researchers have found reinforcement learning can achieve excellent performance for combinatorial optimization problems [24], even though it also has some limitations. For the running time, RL seems to be more time consuming. Another problem is how to improve the generalization ability. That is, train on a dataset and generalize it to other similar datasets. It's also found that the performance of RL mainly depends on the structure of the neural network. So in the future, we will focus on the neural network structure and training algorithm to enhance the learning ability of RL, and improve the generalization of the algorithm.

APPENDIX

See Tables 3–11.

REFERENCES

[1] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *J. ACM*, vol. 32, no. 1, pp. 130–136, Jan. 1985.

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

[3] 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.

[4] L. Wei, M. Lai, A. Lim, and Q. Hu, "A branch-and-price algorithm for the two-dimensional vector packing problem," *Eur. J. Oper. Res.*, vol. 281, no. 1, pp. 25–35, Feb. 2020.

[5] V. M. R. Bezerra, A. A. S. Leao, J. F. Oliveira, and M. O. Santos, "Models for the two-dimensional level strip packing problem—A review and a computational evaluation," *J. Oper. Res. Soc.*, vol. 71, no. 4, pp. 606–627, 2019, doi: 10.1080/01605682.2019.1578914.

[6] S. Jakobs, "On genetic algorithms for the packing of polygons," *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 165–181, 1996.

[7] J. F. Gonçalves, "A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1212–1229, Dec. 2007.

[8] C. H. Dagli, "Simulated annealing approach for solving stock cutting problem," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. Conf.*, Los Angeles, CA, USA, 1990, pp. 221–223.

[9] 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.

[10] Y. B. Shin and E. Kita, "Solving two-dimensional packing problem using particle swarm optimization," *Comput. Assist. Methods Eng. Sci.*, vol. 19, no. 3, pp. 241–255, 2017.

[11] T.-H.-S. Li, C.-Y. Liu, P.-H. Kuo, N.-C. Fang, C.-H. Li, C.-W. Cheng, C.-Y. Hsieh, L.-F. Wu, J.-J. Liang, and C.-Y. Chen, "A three-dimensional adaptive PSO-based packing algorithm for an IoT-based automated e-fulfillment packaging system," *IEEE Access*, vol. 5, pp. 9188–9205, 2017, doi: 10.1109/ACCESS.2017.2702715.

[12] 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.

[13] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin-packing algorithms," *J. Oper. Res. Soc.*, vol. 38, no. 5, pp. 423–429, May 1987.

[14] 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, 2004.

[15] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit, "Reactive GRASP for the strip-packing problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1065–1083,2008.

[16] L. Huang, Z. Liu, and Z. Liu, "An improved lowest-level best-fit algorithm with memory for the 2D rectangular packing problem," in *Proc. Int. Conf. Inf. Sci., Electron. Electr. Eng.*, Sapporo, 2014, pp. 1279–1282, doi: 10.1109/InfoSEEE.2014.6947877.

[17] 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.

[18] 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.

[19] Z. Xing and S. Tu, "A graph neural network assisted Monte Carlo tree search approach to traveling salesman problem," *IEEE Access*, vol. 8, pp. 108418–108428, 2020, doi: 10.1109/ACCESS.2020.3000236.

[20] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "UAV autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, vol. 7, pp. 117227–117245, 2019, doi: 10.1109/ACCESS.2019.2933002.

[21] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," 2019, *arXiv:1911.04936*. [Online]. Available: http://arxiv.org/abs/1911.04936

[22] K. Abe, Z. Xu, I. Sato, and M. Sugiyama, "Solving np-hard problems on graphs by reinforcement learning without domain knowledge," 2019, *arXiv:1905.11623*. [Online] Available: https://arxiv.org/abs/1905.11623

[23] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*. [Online]. Available: http://arxiv.org/abs/1611.09940

[24] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," 2020, *arXiv:2003.03600*. [Online]. Available: http://arxiv.org/abs/2003.03600

[25] T. D. Barrett, W. R. Clements, J. N. Foerster, and A. I. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," 2019, *arXiv:1909.04063*. [Online]. Available: http://arxiv.org/abs/1909.04063

[26] Y. Bai, D. Xu, A. Wang, K. Gu, X. Wu, A. Marinovic, C. Ro, Y. Sun, and W. Wang, "Fast detection of maximum common subgraph via deep Q-learning," 2020, *arXiv:2002.03129*. [Online]. Available: http://arxiv.org/abs/2002.03129

[27] O. Kundu, S. Dutta, and S. Kumar, "Deep-pack: A vision-based 2D online bin packing algorithm with deep reinforcement learning," in *Proc. 28th IEEE Int. Conf. Robot Hum. Interact. Commun. (RO-MAN)*, New Delhi, India, Oct. 2019, pp. 1–7, doi: 10.1109/RO-MAN46459.2019.8956393.

[28] L. Wei, W.-C. Oon, W. Zhu, and A. Lim, "A skyline heuristic for the 2D rectangular packing and strip packing problems," *Eur. J. Oper. Res.*, vol. 215, no. 2, pp. 337–346, Dec. 2011.

[29] D. C. Gao, "Heuristic algorithms for 2D strip packing problem," M.S. thesis, Dept. Comput. Sci. Technol., Zhejiang Univ., Hangzhou, China, 2019.

[30] 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.

[31] E. Hopper, "Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods," Ph.D. dissertation, School Eng., Univ. Wales, Cardiff, U.K., 2000.

[32] N. C. Christofides Whitlock, "An algorithm for two-dimensional cutting problems," *Oper. Res.*, vol. 25, no. 1, pp. 30–44, Feb. 1977.

[33] S. Martello and D. Vigo, "Exact solution of the two-dimensional finite bin packing problem," *Manage. Sci.*, vol. 44, no. 3, pp. 388–399, Mar. 1998.

[34] 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.

[35] C. L. Mumford-Valenzuela, J. Vick, and P. Y. Wang, "Heuristics for large strip packing problems with guillotine patterns: An empirical study," in *Metaheuristics: Computer Decision-Making*. Norwell, MA, USA: Kluwer, 2003, ch. 24, pp. 501–522.

[36] B. E. Bengtsson, "Packing rectangular pieces—A heuristic approach," *Comput. J.*, vol. 25, pp. 253–300, 1982.

[37] A. Bortfeldt and H. Gehring, "New large benchmark instances for the two-dimensional strip packing problem with rectangular pieces," in *Proc. 39th Annu. Hawaii Int. Conf. IEEE Syst. Sci. (HICSS)*, vol. 2, Jan. 2006, p. 30b.

[38] R. G. Rakotonirainy and J. H. van Vuuren, "Improved metaheuristics for the two-dimensional strip packing problem," *Appl. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106268, doi: 10.1016/j.asoc.2020.106268.

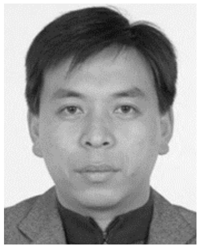
[39] L. J. Wei, Q. Hu, C. Brenda, and X. H. Xu, "An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem," *Int. Trans. Oper. Res.*, vol. 23, nos. 1–2, pp. 65–92, Jan. 2016.

[40] 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.

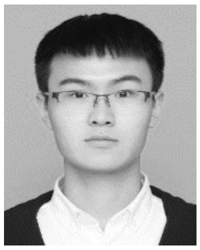
[41] Z. Chen and J. Chen, "An effective corner increment-based algorithm for the two-dimensional strip packing problem," *IEEE Access*, vol. 6, pp. 72906–72924, 2018.

[42] J. H. Van Vuuren and R. G. Rakotonirainy. (2018). *The 2D Rectangular Strip Packing Problem (Clustered Benchmarks)*. [Online]. Available: http://www.vuuren.co.za/main.php

KAI ZHU received the bachelor's degree in information and computing science from the China University of Petroleum (East China), in 2008, and the master's and Ph.D. degrees, in 2011 and 2014, respectively. He is currently a Teacher with the Qingdao University of Technology. His research interests include reinforcement learning, big data analysis, machine learning algorithms, and applications.



NAIHUA JI received the bachelor's degree in computer science and technology and the master's degree in software and theory from Yanshan University, in 1997 and 2004, respectively. He worked as an Intern with Beijing BiXi Radio and Television Technology Company Ltd., for one and a half years. He has been teaching with the Qingdao University of Technology since 2004. His research interests include reinforcement learning, integrated manufacturing, and enterprise informatization.



XIANG DONG LI is currently pursuing the master's degree with the Qingdao University of Technology. His current research interests include machine learning and combinatorial optimization algorithm.