Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 541931, 14 pages
http://dx.doi.org/10.1155/2015/541931

Research Article

A Hybrid Demon Algorithm for the Two-Dimensional Orthogonal Strip Packing Problem

Bili Chen,[1] Yong Wang,[2] and Shuangyuan Yang[1]

[1]*Software School, Xiamen University, Xiamen 361005, China*
[2]*School of Economics and Business Administration, Chongqing Key Laboratory of Logistics, Chongqing University, Chongqing 400044, China*

Correspondence should be addressed to Yong Wang; wangyongkt@163.com

Received 3 September 2014; Accepted 23 December 2014

Academic Editor: Anders Eriksson

This paper develops a hybrid demon algorithm for a two-dimensional orthogonal strip packing problem. This algorithm combines a placement procedure based on an improved heuristic, local search, and demon algorithm involved in setting one parameter. The hybrid algorithm is tested on a wide set of benchmark instances taken from the literature and compared with other well-known algorithms. The computation results validate the quality of the solutions and the effectiveness of the proposed algorithm.

1. Introduction

Cutting and packing are a very active field of research within operational research, computer science, mathematics, and management science. The two-dimensional cutting and packing problem is widely applied in optimally cutting raw materials such as glass, textile, steel, and paper and transportation and logistics fields. For example, in textile or glass industries, rectangular components have to be cut from large sheets of material. In warehousing, goods have to be placed on shelves. In newspapers paging, articles and advertisements have to be arranged in pages [1]. In order to raise the profitability of the manufacturing or logistics company, the consumption of the raw materials or the cost of transportation should be minimized. Obviously, if an enterprise designs a production scheme using the least waste raw material, it can reduce the manufacturing costs and increase the product's competitiveness in the market.

The two-dimensional orthogonal strip packing problems (2SP) addressed in this paper consist of packing rectangular pieces into a large rectangular sheet of fixed width and unlimited height in order to minimize the used height, where the rectangular pieces are placed orthogonally without overlap and no rotations are allowed. This problem is of significance both from a theoretical and a practical point of view because it arises in various production processes and has many applications in the glass, steel, paper, and textile industries, and they also have indirect applications in other fields [2] such as layout designing, transportation, and logistics. More extensive survey and classification on cutting and packing problems may refer to Lodi et al. [1] and Wäscher et al. [3].

2SP is known to be NP-hard, some exact algorithms are proposed by Martello et al. [4] and Kenmochi et al. [5], but the size of instances that exact algorithms can handle tends to be small. Therefore, many heuristic algorithms have been suggested in the literature. Baker et al. [6] proposed a bottom-left-fill (BLF) algorithm for 2SP and variants of BLF [7]. Different types of construction heuristics have also been proposed recently, for example, the best-fit heuristic [8], a recursive heuristic [9], a bricklaying heuristic [10], a least waste heuristic [11], and a scoring heuristic [12].

In comparison to the literature on construction heuristic algorithms to packing problems, metaheuristic algorithms are paid more and more attention recently. Bortfeldt [13] proposed a genetic algorithm. The above two algorithms explore placements directly and allow infeasible solutions. However, most metaheuristic algorithms have been developed by incorporating a construction heuristic to improve the quality of solutions. Hopper and Turton [14] implemented

Table 1: New scoring function score (i, h_1, h_2, w) for $h_1 \geq h_2$.

	Conditions	f	f_{new}	m
$h_1 \geq h_2$	Case (1): $w = r[i]\cdot$width and $h_1 = r[i]\cdot$length	4	4	−2 or −1
	Case (2): $w = r[i]\cdot$width and $h_1 < r[i]\cdot$length	3	3	0
	Case (3): $w = r[i]\cdot$width and $h_1 > r[i]\cdot$length	2	**1**	0
	Case (4): $w > r[i]\cdot$width and $h_1 = r[i]\cdot$length	1	**2**	0
	Case (5): $w > r[i]\cdot$width	0	0	+1

a simulated annealing, tabu search, and genetic algorithm by incorporating BLF, respectively. Lesh et al. [15] proposed new heuristic and interactive approaches based on BLF for 2SP. Alvarez-Valdes et al. [16] presented a greedy randomized adaptive search procedure (GRASP) that involves learning some instances to determine the desirable parameter settings for 2SP. Wei et al. [11] presented a least waste algorithm by combining a simulated annealing algorithm for rectangle packing problem. Burke et al. [17] implemented a simulated annealing, tabu search, and genetic algorithm by incorporating BF. Leung et al. [18] proposed a simulated annealing algorithm based on a scoring rule heuristic. Some other algorithms based on different types of strategies have also been proposed, for example, SVC [19] and SWL [20]. Zhang et al. [21] developed a binary search heuristic algorithm based on randomized local search for the rectangular strip packing problem. In particular, an efficient algorithm for strip packing problem can be extended to solve other problems such as bin packing problem [22, 23]. In this paper, we present a hybrid demon algorithm for 2SP which combines a demon algorithm with local search and an improved heuristic. This paper mainly has three contributions: firstly, a new scoring rule is presented; secondly, a least waste strategy is proposed; at last, a demon algorithm with fewer parameters than simulated annealing algorithm is applied to solve 2SP.

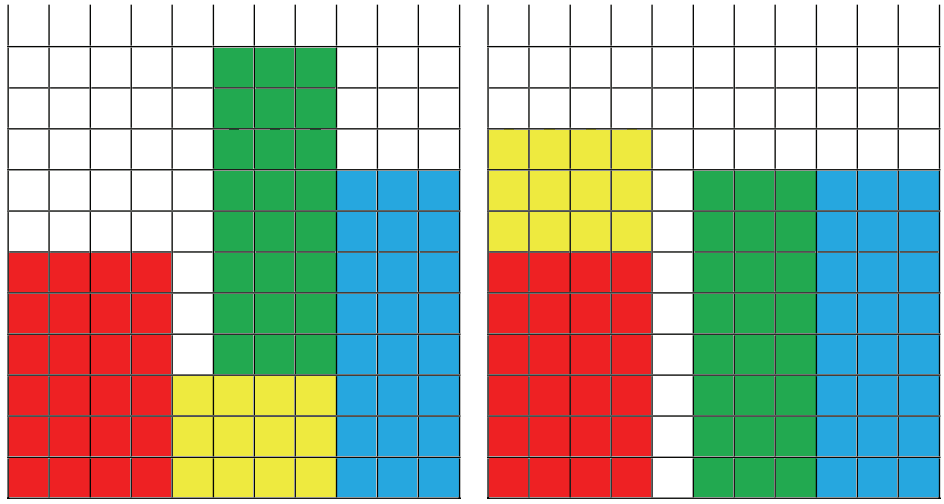
The remainder of the paper is organized as follows. Section 2 describes the hybrid algorithm: an improved scoring rule and demon algorithm. Section 3 investigates the effect of the parameter of demon algorithm and the least waste strategy. Section 4 reports the experimental results. Section 5 summarizes the conclusions and proposes future work.

2. Hybrid Algorithm

2.1. Improved Construction Heuristic Algorithm. It has been reported that construction heuristic algorithm is one of the best heuristics while combining with a simulated annealing algorithm [18]. Construction heuristic algorithm is stated as follows: give a rectangle piece sequence, the algorithm finds an unplaced piece with the maximum score for the lowest and the most left space, and then place it. Repeat the above process until all the pieces are placed. Leung et al. [18] proposed the scoring rules as Table 1 which is very important to select one unplaced piece, where w is the width of the available space and h_1 and h_2 are the height of the left and right, respectively, wall of the available space. $r[i] \cdot$ width and $r[i] \cdot$ length denote width and length, respectively, of rectangular piece i, m denotes the change number of spaces, and f is fitness value as Leung et al. [18]. However, they do not explain why case (3) in Table 1 has higher score than case (4). In fact, the two cases have the same m; namely, the number of available spaces does not increase. Due to the fact that the objective of the problem is to minimize the height of the sheet, the result obtained by the scoring rule in Table 1 may be bad, while case (3) has higher score than case (4). For example, given a piece sequence: red, blue, yellow, and green, the packing result is shown in Figure 1, the yellow piece will be first placed according to the original scoring rule f after the red piece and blue piece are placed, and then the green piece is placed as Figure 1(a), so the height obtained is 11. However, the height obtained is 9 if the new scoring rule f_{new} is used that the green piece should be placed earlier than the yellow piece. The above example shows that the new scoring rule may lead to a better result than the original scoring rule. So the new scoring rule f_{new} for $h_1 \geq h_2$ as shown in Table 1 is used in this paper; for $h_1 < h_2$, the new scoring rule f_{new} can be calculated similarly.

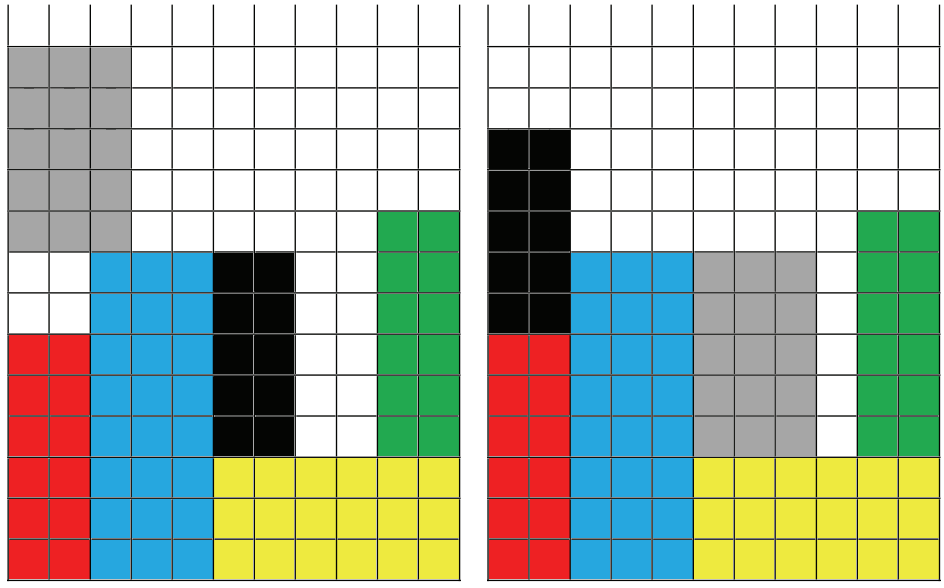
In addition, one piece r is selected by case (4) in Table 1, and we can try to place it into the current available space s; if it leads to the waste of the remainder space, then find another piece q with the maximal width from all the unplaced pieces that meets $h_q = h_r$ and can be packed into s. For example, given a piece sequence: red, yellow, blue, green, black, and grey, the packing result is shown in Figure 2(a), the width of the current available space s is 4 after the red, yellow, blue, and green pieces are placed, the black piece r ($h_r = 5$ and $w_r = 2$) is selected by the scoring rule, and after it is placed, the width of the remainder space is 2. If the width of all the unpacked piece is larger than 2, the remainder space 2×3 will be wasted. The grey piece is placed after the black piece is placed. The height obtained is 13. However, the obtained height is 11 if we use the least waste strategy. Namely, we can find another piece q ($h_q = 5$ and $w_q = 3$) from all the unplaced pieces. Obviously, placing the piece q makes waste least and obtains the smaller height. Similarly, if one piece r is selected by case (5) in Table 1, and placing it leads to the waste of the remainder space, then find another unplaced piece q with the maximal width from all the unplaced pieces that meets $c_q \geq c_r$ and can be packed into s, where c_q and c_r denote the perimeter of the pieces q and r, respectively.

2.2. Hybrid Demon Algorithm. The simulated annealing algorithm (SA) was invented to allow computer simulation of equilibria in statistical physics. It is a powerful randomized search algorithm, and the computational results have shown that ISA [18] based on SA is the best algorithm for 2SP. However, SA has to set the initial temperature, the annealing rate, and the Markov chain length. These parameters have



(a) Case (3) has higher score than case (4) (b) Case (4) has higher score than case (3)

Figure 1: Placement by the different scoring rules.



(a) (b)

Figure 2: The least waste strategy.

significant effect on the performance of SA, but they have no general setting rule. Therefore, some variants with fewer parameters are paid attention by some researchers.

Demon algorithm is a simulated annealing based algorithm that uses computationally simpler acceptance function. This paper applies demon algorithm for 2SP. In order to solve it, the hybrid demon algorithm can be stated as in Algorithm 1, where LS() is similar to that in Leung et al. [18] and is stated as in Algorithm 2, where L is the number of pieces, *besth* saves the best solution during the search process, and *currenth* is the height returned by HeuristicPacking(X'). The process of HeuristicPacking(X') is the same as that of Leung et al. [18] except that a new scoring rule and a least waste strategy are used. Line 14 means keeping the old sequence X. D is a key parameter that controls the acceptance function.

HDA first searches a better solution according to a local search algorithm LS() and then makes use of demon algorithm to improve the solution. Multistart ingredient (lines 16–20) is employed to help HDA to escape from possible local minima.

3. Effect of Parameter D

HDA only involves in setting one parameter D, so it is simpler than simulated annealing algorithm. This section gives some

```
HDA()
(1)  LS();
(2)  give an initial value D and best = besth;
(3)  while the stop criterion is not yet satisfied do
(4)     for i ← 1 to L do
(5)          randomly select two pieces j and k in X;
(6)          swap the order of pieces j and k;
(7)          currenth ← HeuristicPacking(X′);
(8)          Δ = currenth − best;
(9)          if Δ < D then
(10)              best ← currenth;
(11)              if best < besth then
(12)                 besth ← currenth;
(13)                 D = D − Δ;
(14)              else
(15)                 swap the order of pieces j and k;
(16) randomly flip a coin;
(17) if coin comes up heads then
(18)    sort all the pieces by non-increasing ordering of perimeter size to obtain X;
(19) else
(20)    sort all the pieces by non-increasing ordering of width size and obtain X;
(21) return besth;
```

Algorithm 1: Hybrid demon algorithm.

```
LS()
(1)  sort all unpacked pieces by non-increasing ordering of length size to obtain X;
(2)  besth ← HeuristicPacking(X);
(3)  for i ← 1 to n − 1 do
(4)     for j ← i + 1 to n do
(5)          swap the order of pieces i and j in X and obtain a new ordering X′;
(6)          currenth ← HeuristicPacking(X′);
(7)          if currenth < besth then
(8)               besth ← currenth;
(9)               X ← X′;
```

Algorithm 2

experiments to further select the value of D. In order to add the effect of demon algorithm, we only use some small and medium instances to save more time for demon algorithm to improve the solution. Several difficult data sets NT, gcut, Nice1~6, and Path1~6 are selected because they contain zero-waste instances and nonzero-waste instances. Figures 3~5 give the results of demon algorithm on the above instances, where x-axis denotes the different value of D and y-axis denotes the average gap over 10 runs. Although there exists some slight difference for different data sets, we note that Gap will increase as D increases. Therefore, $D = 1$ is selected for the experiments in next section.

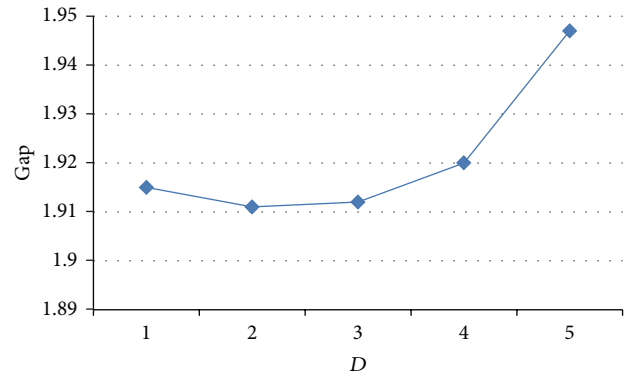


Figure 3: Effect of D on the data set NT.

Figures 6, 7, and 8 report the results of hybrid demon algorithm with and without the least waste strategy, where x-axis denotes the problem instance and y-axis denotes the average gap over 10 runs. From these figures, we can note that the results obtained by the algorithm without the least waste strategy are slightly better than that obtained by the algorithm with the least waste strategy for zero-waste data sets C. The algorithm with the least waste strategy can obtain the better results than that without the least waste strategy for nonzero-waste data sets. Because nonzero-waste data sets

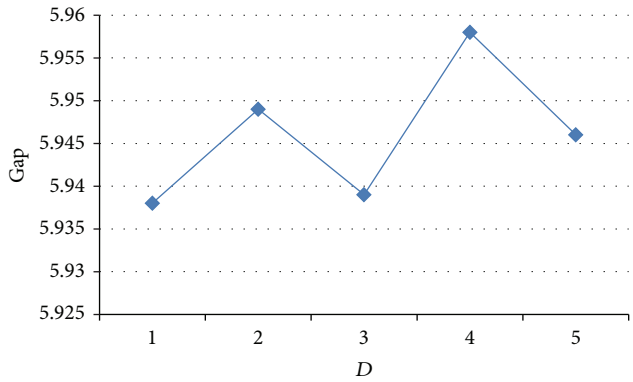


Figure 4: Effect of D on the data set gcut.

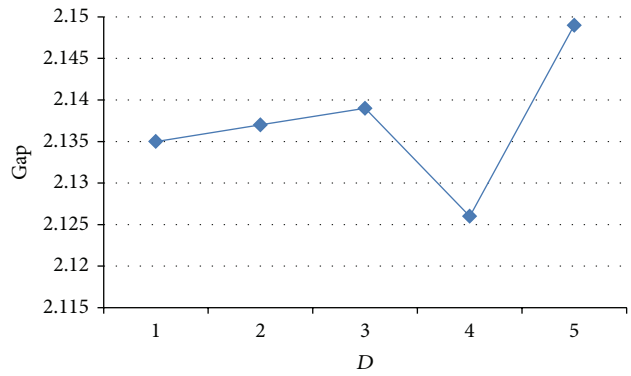


Figure 5: Effect of D on the data set Nice1~6 and Path1~6.

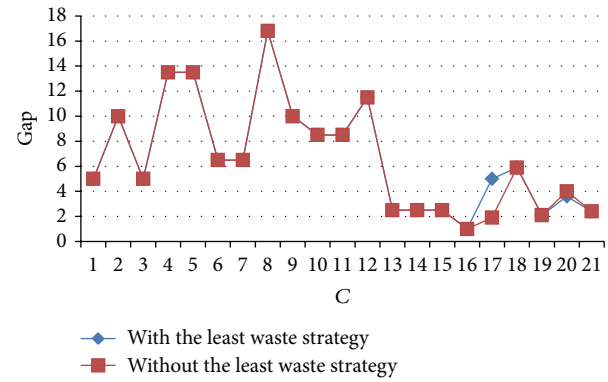


Figure 6: Effect of the least waste strategy on the data set C.

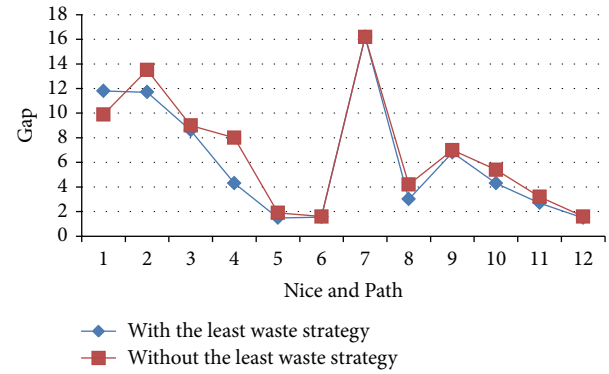


Figure 7: Effect of the least waste strategy on the data set Nice1~6 and Path1~6.

contain more instances, so we use the least waste strategy in the experiments in next section.

4. Experimental Results

In this section, we present the results obtained in a set of experiments we conducted in order to evaluate the performance of the hybrid demon algorithm (HDA) proposed in this paper. This paper uses the same data sets C, N, NT and CX, 2sp, BWMV, Nice, and Path as Leung et al. [18]. All the data sets are publicly available at http://algorithm.xmu.edu.cn:10000/Download.aspx#p4. These data sets include zero-waste instances and nonzero-waste instances from the literature. Zero-waste instances were created from known optimal solutions, and nonzero-waste instances do not necessarily have an optimal solution and their optimal solutions involve some waste regions.

The algorithm was implemented in Visual C++6.0 and the experimental tests were run on a computer with an Intel core 2 CPU 2.13 GHz and 0.99 GB RAM. GRASP [16], SVC [19], and ISA [18] are among the supposedly excellent algorithms in the current literature, so they are selected to compare with HDA. Computational results of GRASP, SVC, and ISA were taken from Leung et al. [18]. They were run on a computer with Intel Xeon CPU E5405 2.00 GHz 1.99 GB RAM and were run 10 times with a time limit of 60 seconds per run for each instance. HDA was allowed 60-second duration too. The computational results are reported in Tables 2~9, where *Instance* denotes problem instance, n is the number of rectangular pieces, W is the width of the rectangular sheet, LB is the optimal height for zero-waste instances and is the lower bound for nonzero-waste instances, and *meanh* denotes the average height obtained by each algorithm, running 10 times for each instance, respectively. Gap is defined as follows: Gap $= 100 \times (meanh - \text{LB})/\text{LB}$. The detailed results on the best and worst cases are available from the authors after this paper is published.

4.1. Zero-Waste Instances. Tables 2~9 report the results of four algorithms on zero-waste problem instances. In each table, the best results obtained by four algorithms are marked by boldface. The first four columns describe the characteristics of each instance. Columns 5~8 correspond to the average height obtained by GRASP, SVC, ISA, and HDA, respectively. Columns 9~12 correspond to the Gap obtained by GRASP, SVC, ISA, and HDA, respectively.

Table 2 shows the results of four algorithms for the data set C [14], which have been used by many authors. From Table 2, GRASP, SVC, and ISA return an average gap of 0.95, 1.03, and 0.76, respectively. HDA with an average gap of 0.71 performs better than GRASP, SVC, and ISA. In addition, we can observe that HDA performs well for large instances.

Table 2: Results obtained by GRASP, SVC, ISA, and HDA on C.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
C11	16	20	20	20	20	20.0	20	0.0	0.0	0.0	0.0
C12	17	20	20	20	21	20.0	20	0.0	5.0	0.0	0.0
C13	16	20	20	20	20	20.0	20	0.0	0.0	0.0	0.0
C21	25	40	15	15	15	15.0	15	0.0	0.0	0.0	0.0
C22	25	40	15	15	15	15.0	15	0.0	0.0	0.0	0.0
C23	25	40	15	15	15	15.0	15	0.0	0.0	0.0	0.0
C31	28	60	30	30	30	30.0	30	0.0	0.0	0.0	0.0
C32	29	60	30	31	31	31.0	30.9	3.3	3.3	3.3	3.0
C33	28	60	30	30	30	30.0	30	0.0	0.0	0.0	0.0
C41	49	60	60	61	61	61.0	61	1.7	1.7	1.7	1.7
C42	49	60	60	61	61	61.0	61	1.7	1.7	1.7	1.7
C43	49	60	60	61	61	60.9	61	1.7	1.7	1.5	1.7
C51	73	60	90	91	91	91.0	91	1.1	1.1	1.1	1.1
C52	73	60	90	91	91	90.8	91	1.1	1.1	0.9	1.1
C53	73	60	90	91	91	91.0	91	1.1	1.1	1.1	1.1
C61	97	80	120	122	121	121.0	121	1.7	0.8	0.8	0.8
C62	97	80	120	121	121	121.0	121	0.8	0.8	0.8	0.8
C63	97	80	120	122	121	121.0	121	1.7	0.8	0.8	0.8
C71	196	160	240	244	242	242.0	241	1.7	0.8	0.8	0.4
C72	197	160	240	243	242	241.0	241	1.3	0.8	0.4	0.4
C73	196	160	240	243	242	242.0	241	1.3	0.8	0.8	0.4
Average				83.19	82.95	82.84	**82.76**	0.95	1.03	0.76	**0.71**

Table 3: Results obtained by GRASP, SVC, ISA, and HDA on N.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
N1	10	40	40	40	40	40.0	40	0.0	0.0	0.0	0.0
N2	20	30	50	50	50	50.0	50	0.0	0.0	0.0	0.0
N3	30	30	50	51	50	50.1	50	2.0	0.0	0.2	0.0
N4	40	80	80	81	81	80.0	80	1.3	1.3	0.0	0.0
N5	50	100	100	102	101	101.0	100	2.0	1.0	1.0	0.0
N6	60	50	100	101	101	100.9	100.7	1.0	1.0	0.9	0.7
N7	70	80	100	101	101	100.0	100	1.0	1.0	0.0	0.0
N8	80	100	80	81	81	81.0	81	1.3	1.3	1.3	1.3
N9	100	50	150	151	151	150.9	151	0.7	0.7	0.6	0.7
N10	200	70	150	151	151	150.8	151	0.7	0.7	0.5	0.7
N11	300	70	150	151	151	150.7	150.8	0.7	0.7	0.5	0.5
N12	500	100	300	304	301	301.0	301	1.3	0.3	0.3	0.3
N13	3152	640	960	965	963	960.0	960	0.5	0.3	0.0	0.0
Average				179.15	178.62	178.18	**178.12**	0.95	0.63	0.41	**0.32**

Table 3 shows the results of four algorithms for the data set N generated by Burke et al. [8]. From Table 3, GRASP, SVC, and ISA return an average gap of 0.95, 0.63, and 0.41, respectively. HDA with an average gap of 0.32 performs better than GRASP, SVC, and ISA. What is more, the optimal solution of N13 is obtained by ISA and HDA.

Table 4 shows the results of four algorithms for the data set NT, which is generated by Hopper and Turton [14]. From Table 4, GRASP, SVC, and ISA return an average gap of 2.32, 2.27, and 2.24, respectively. HDA with an average gap of 1.91 performs better than GRASP, SVC, and ISA. In addition, we can observe that HDA performs the same or better for large instances n7a~e and t7a~e.

Table 5 shows the results of four algorithms for the extra-large data set CX, which is generated by Pinto and Oliveira [24]. From Table 5, GRASP, SVC, and ISA return an average

Table 4: Results obtained by GRASP, SVC, ISA, and HDA on NT.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
n1a	17	200	200	200	202	200.0	200	0.0	1.0	0.0	0.0
n1b	17	200	200	209	200	211.2	200	4.5	0.0	5.6	0.0
n1c	17	200	200	200	200	200.0	200	0.0	0.0	0.0	0.0
n1d	17	200	200	200	200	200.0	200	0.0	0.0	0.0	0.0
n1e	17	200	200	200	200	200.0	200	0.0	0.0	0.0	0.0
n2a	25	200	200	206	205	204.0	201.3	3.0	2.5	2.0	0.7
n2b	25	200	200	206	209	209.4	210	3.0	4.5	4.7	5.0
n2c	25	200	200	208	209	208.5	206.3	4.0	4.5	4.3	3.2
n2d	25	200	200	209	207	207.8	205.9	4.5	3.5	3.9	3.0
n2e	25	200	200	206	205	206.7	206.1	3.0	2.5	3.3	3.1
n3a	29	200	200	209	208	206.1	206.3	4.5	4.0	3.1	3.2
n3b	29	200	200	208	207	209.0	208.9	4.0	3.5	4.5	4.5
n3c	29	200	200	205	207	206.1	205	2.5	3.5	3.1	2.5
n3d	29	200	200	207	208	204.3	204.5	3.5	4.0	2.2	2.3
n3e	29	200	200	207	207	208.0	208.1	3.5	3.5	4.0	4.1
n4a	49	200	200	206	205	206.0	205.9	3.0	2.5	3.0	3.0
n4b	49	200	200	207	205	205.0	204.7	3.5	2.5	2.5	2.3
n4c	49	200	200	205	205	206.0	205.3	2.5	2.5	3.0	2.7
n4d	49	200	200	206	205	204.8	204.9	3.0	2.5	2.4	2.5
n4e	49	200	200	205	205	206.0	206.1	2.5	2.5	3.0	3.1
n5a	73	200	200	205	204	205.1	205.5	2.5	2.0	2.6	2.8
n5b	73	200	200	204	204	203.6	203.1	2.0	2.0	1.8	1.6
n5c	73	200	200	206	204	204.4	204.5	3.0	2.0	2.2	2.3
n5d	73	200	200	204	205	205.0	204.7	2.0	2.5	2.5	2.3
n5e	73	200	200	206	205	204.7	204.9	3.0	2.5	2.3	2.5
n6a	97	200	200	204	203	202.8	202.7	2.0	1.5	1.4	1.3
n6b	97	200	200	204	204	203.0	202.9	2.0	2.0	1.5	1.5
n6c	97	200	200	204	204	203.6	203.2	2.0	2.0	1.8	1.6
n6d	97	200	200	204.1	202	203.8	203	2.1	1.0	1.9	1.5
n6e	97	200	200	204	203	203.5	203	2.0	1.5	1.8	1.5
n7a	199	200	200	202	202	201.0	201	1.0	1.0	0.5	0.5
n7b	199	200	200	203	202	202.0	201	1.5	1.0	1.0	0.5
n7c	199	200	200	203	202	201.9	201	1.5	1.0	1.0	0.5
n7d	199	200	200	203	202	201.9	201	1.5	1.0	1.0	0.5
n7e	199	200	200	203	202	201.9	201	1.5	1.0	1.0	0.5
t1a	17	200	200	200	200	200.0	200	0.0	0.0	0.0	0.0
t1b	17	200	200	200	211	200.0	200	0.0	5.5	0.0	0.0
t1c	17	200	200	200	210	200.0	200	0.0	5.0	0.0	0.0
t1d	17	200	200	200	200	211.8	200	0.0	0.0	5.9	0.0
t1e	17	200	200	200	209	200.0	200	0.0	4.5	0.0	0.0
t2a	25	200	200	204	207	207.0	206.3	2.0	3.5	3.5	3.2
t2b	25	200	200	208	205	207.0	205.8	4.0	2.5	3.5	2.9
t2c	25	200	200	208	206	206.0	207.4	4.0	3.0	3.0	3.7
t2d	25	200	200	206	207	209.3	204.4	3.0	3.5	4.7	2.2
t2e	25	200	200	206	207	207.4	205.9	3.0	3.5	3.7	3.0
t3a	29	200	200	207	208	209.0	209	3.5	4.0	4.5	4.5
t3b	29	200	200	209	207	208.1	207.9	4.5	3.5	4.1	4.0
t3c	29	200	200	206	207	206.6	206.3	3.0	3.5	3.3	3.2
t3d	29	200	200	207	208	206.4	206.4	3.5	4.0	3.2	3.2
t3e	29	200	200	208	206	205.0	205	4.0	3.0	2.5	2.5

Table 4: Continued.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
t4a	49	200	200	205	205	205.0	204.7	2.5	2.5	2.5	2.3
t4b	49	200	200	205	205	206.1	205.9	2.5	2.5	3.1	3.0
t4c	49	200	200	206	205	204.9	204.7	3.0	2.5	2.5	2.3
t4d	49	200	200	206	205	205.7	205.4	3.0	2.5	2.8	2.7
t4e	49	200	200	207	205	205.2	205.5	3.5	2.5	2.6	2.8
t5a	73	200	200	206	204	204.4	204.5	3.0	2.0	2.2	2.3
t5b	73	200	200	204	204	204.0	204.4	2.0	2.0	2.0	2.2
t5c	73	200	200	205	204	205.0	205.5	2.5	2.0	2.5	2.8
t5d	73	200	200	204	205	204.9	204.7	2.0	2.5	2.5	2.3
t5e	73	200	200	204	204	204.0	204.7	2.0	2.0	2.0	2.3
t6a	97	200	200	204	204	203.2	203.7	2.0	2.0	1.6	1.8
t6b	97	200	200	204	202	203.4	203.2	2.0	1.0	1.7	1.6
t6c	97	200	200	204	204	203.0	202.7	2.0	2.0	1.5	1.3
t6d	97	200	200	204	204	203.5	203.7	2.0	2.0	1.8	1.8
t6e	97	200	200	205	204	203.5	203.5	2.5	2.0	1.8	1.8
t7a	199	200	200	203	201	201.2	201	1.5	0.5	0.6	0.5
t7b	199	200	200	203	202	201.0	201	1.5	1.0	0.5	0.5
t7c	199	200	200	204	202	201.0	201	2.0	1.0	0.5	0.5
t7d	199	200	200	202	202	202.0	201	1.0	1.0	1.0	0.5
t7e	199	200	200	203	202	201.7	201	1.5	1.0	0.8	0.5
Average				204.64	204.54	204.48	**203.83**	2.32	2.27	2.24	**1.91**

Table 5: Results obtained by GRASP, SVC, ISA, and HDA on CX.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
50cx	50	400	600	613	603	620.2	607.3	2.2	0.5	3.4	1.2
100cx	100	400	600	617	616	615.8	617.3	2.8	2.7	2.6	2.9
500cx	500	400	600	605	604	601.0	601	0.8	0.7	0.2	0.2
1000cx	1000	400	600	602	601	600.0	600	0.3	0.2	0.0	0.0
5000cx	5000	400	600	600	600	600.0	600	0.0	0.0	0.0	0.0
10000cx	10000	400	600	600	600	600.0	600	0.0	0.0	0.0	0.0
15000cx	15000	400	600	600	600	600.0	600	0.0	0.0	0.0	0.0
Average				605.29	**603.43**	605.29	603.66	0.88	**0.57**	0.88	0.61

gap of 0.88, 0.57, and 0.88, respectively. HDA with an average gap of 0.61 performs better than GRASP and ISA and slightly worse than SVC.

4.2. Nonzero-Waste Instances. Section 4.1 has discussed the results on the zero-waste instances with known optimal solutions. In some practical applications, the optimal solutions often include some wasted regions. So nonzero-waste instances are more general and are widely used in the literature. We can compute the lower bound of these instances, and the optimal solutions of some instances are known because they are confirmed by other algorithms. In this section, we investigate if HDA can still be successfully employed to solve such instances. Tables 6~9 summarize the computational results on nonzero-waste instances.

Table 6 shows the results of nonzero-waste data sets 2sp which include ngcut [25], gcut [26], cgcut [27], and Beng [28]. The problem size of 2sp is very small; namely, n is less than 200, so it is used by many authors to test the performance of the algorithms. From Table 6, GRASP, SVC, and ISA return an average gap of 2.68, 2.8, and 3.02, respectively. HDA with an average gap of 3.04 performs slightly worse than GRASP, SVC, and ISA. However, HDA can obtain smaller average *meanh* 1538.16 than GRASP, SVC, and ISA, so HDA is still efficient for 2sp.

Table 7 shows the results of nonzero-waste data sets BWMV which include C01~C06 [29], C07~C10 [30]. The problem size of BWMV is very small ($n \leq 100$), so it is used by many authors to test the performance of the algorithms. From Table 7, GRASP, SVC, and ISA return an average gap of 1.77, 1.80, and 1.66, respectively. HDA with an average gap of 1.63 performs better than GRASP, SVC, and ISA. So HDA is superior to GRASP, SVC, and ISA.

Table 8 shows the results of data sets Nice and Path which are floating-point data sets and are generated by Valenzuela and Wang [31], where Nice1~Nice5t are the sets of similarly

Table 6: Results obtained by GRASP, SVC, ISA, and HDA on 2sp.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
cgcut1	16	10	23	23	23	23.0	24	0.0	0.0	0.0	4.3
cgcut2	23	70	63	65	65	65.0	65	3.2	3.2	3.2	3.2
cgcut3	62	70	636	661	661	660.2	662.2	3.9	3.9	3.8	4.1
gcut1	10	250	1016	1016	1016	1016.0	1016	0.0	0.0	0.0	0.0
gcut2	20	250	1133	1191	1187	1187.0	1187	5.1	4.8	4.8	4.8
gcut3	30	250	1803	1803	1803	1803.0	1803	0.0	0.0	0.0	0.0
gcut4	50	250	2934	3002	3017	3010.5	3002	2.3	2.8	2.6	2.3
gcut5	10	500	1172	1273	1273	1273.0	1273	8.6	8.6	8.6	8.6
gcut6	20	500	2514	2627	2632	2632.0	2629.5	4.5	4.7	4.7	4.6
gcut7	30	500	4641	4693	4693	4693.0	4693.8	1.1	1.1	1.1	1.1
gcut8	50	500	5703	5912	5876	5890.4	5884.2	3.7	3.0	3.3	3.2
gcut9	10	1000	2022	2317	2317	2317.0	2317	14.6	14.6	14.6	14.6
gcut10	20	1000	5356	5964	5973	5964.8	5964.9	11.4	11.5	11.4	11.4
gcut11	30	1000	6537	6899	6891	6884.4	6883.6	5.5	5.4	5.3	5.3
gcut12	50	1000	12522	14690	14690	14690.0	14690	17.3	17.3	17.3	17.3
gcut13	32	3000	4772	4994	4977	4965.9	4963.2	4.7	4.3	4.1	4.0
ngcut1	10	10	23	23	23	23.0	23	0.0	0.0	0.0	0.0
ngcut2	17	10	30	30	30	31.0	31	0.0	0.0	3.3	3.3
ngcut3	21	10	28	28	28	28.0	28	0.0	0.0	0.0	0.0
ngcut4	7	10	20	20	20	20.0	20	0.0	0.0	0.0	0.0
ngcut5	14	10	36	36	36	36.0	36	0.0	0.0	0.0	0.0
ngcut6	15	10	29	31	31	31.0	31	6.9	6.9	6.9	6.9
ngcut7	8	20	20	20	20	20.0	20	0.0	0.0	0.0	0.0
ngcut8	13	20	32	33	34	34.0	34	3.1	6.3	6.3	6.3
ngcut9	18	20	49	50	51	52.0	51	2.0	4.1	6.1	4.1
ngcut10	13	30	80	80	80	80.0	80	0.0	0.0	0.0	0.0
ngcut11	15	30	50	52	52	52.0	52	4.0	4.0	4.0	4.0
ngcut12	22	30	87	87	87	87.0	87	0.0	0.0	0.0	0.0
beng1	20	25	30	30	30	31.0	30.7	0.0	0.0	3.3	2.3
beng2	40	25	57	57	57	57.0	57	0.0	0.0	0.0	0.0
beng3	60	25	84	84	84	84.0	84	0.0	0.0	0.0	0.0
beng4	80	25	107	107	107	107.0	107	0.0	0.0	0.0	0.0
beng5	100	25	134	134	134	134.0	134	0.0	0.0	0.0	0.0
beng6	40	40	36	36	36	36.0	36	0.0	0.0	0.0	0.0
beng7	80	40	67	67	67	67.0	67	0.0	0.0	0.0	0.0
beng8	120	40	101	101	101	101.0	101	0.0	0.0	0.0	0.0
beng9	160	40	126	126	126	126.0	126	0.0	0.0	0.0	0.0
beng10	200	40	156	156	156	156.0	156	0.0	0.0	0.0	0.0
Average				1539.95	1539.05	1538.64	**1538.16**	**2.68**	2.8	3.02	3.04

dimensioned pieces and Path1~Path5t are of vastly differing dimensions. Nice and Path are regarded as nonzero-waste instances because the integer data are obtained by multiplying the original data by 10 and rounding to the nearest integer [16]. Nice and Path are more difficult because the optimal solutions of many instances are not known. From Table 8, GRASP, SVC, and ISA return an average gap of 1.65, 0.96, and 0.72, respectively. HDA with an average gap of 0.63 performs better than GRASP, SVC, and ISA. HDA performs the same or better for 65 out of 72 instances, so its advantage is very obvious.

Table 9 reports results of large data set ZDF which is generated by combining zero-waste and nonzero-waste data [12]. ZDF includes several extra-large instances ZDF14~16 ($n > 15000$). Generally, it is very difficult to obtain the optimal solutions of ZDF14~16 within a reasonable time.

Table 7: Results obtained by GRASP, SVC, ISA, and HDA on BWMV.

	Instances			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
C01	20	10	60.3	61.4	61.4	61.3	61.3	1.8	1.8	1.7	1.7
	40	10	121.6	121.9	122	121.8	121.8	0.2	0.3	0.2	0.2
	60	10	187.4	188.6	188.6	188.6	188.6	0.6	0.6	0.6	0.6
	80	10	262.2	262.6	262.6	262.6	262.6	0.2	0.2	0.2	0.2
	100	10	304.4	305	304.9	304.9	304.9	0.2	0.2	0.2	0.2
C02	20	30	19.7	19.8	19.8	19.9	19.8	0.5	0.5	1.0	0.5
	40	30	39.1	39.1	39.1	39.1	39.1	0.0	0.0	0.0	0.0
	60	30	60.1	60.3	60.1	60.1	60.1	0.3	0.0	0.0	0.0
	80	30	83.2	83.3	83.2	83.2	83.2	0.1	0.0	0.0	0.0
	100	30	100.5	100.6	100.5	100.5	100.5	0.1	0.0	0.0	0.0
C03	20	40	157.4	163.5	164.6	164.0	163.7	3.9	4.6	4.2	4.0
	40	40	328.8	334.2	333.9	333.8	333.8	1.6	1.6	1.5	1.5
	60	40	500	506.6	506.9	505.8	505.9	1.3	1.4	1.2	1.2
	80	40	701.7	709.7	710.1	709.2	709.5	1.1	1.2	1.1	1.1
	100	40	832.7	840.2	839.9	837.8	838.4	0.9	0.9	0.6	0.7
C04	20	100	61.4	63.3	63.8	63.9	63.4	3.1	3.9	4.1	3.3
	40	100	123.9	126.2	126.2	126.1	125.8	1.9	1.9	1.8	1.5
	60	100	193	196.6	195.6	195.5	195.5	1.9	1.3	1.3	1.3
	80	100	267.2	272	270.5	269.8	270.1	1.8	1.2	1.0	1.1
	100	100	322	327.3	325.3	324.6	324.6	1.6	1.0	0.8	0.8
C05	20	100	512.2	533.9	537.9	534.6	534.1	4.2	5.0	4.4	4.3
	40	100	1053.8	1074.4	1076.4	1073.6	1073.5	2.0	2.1	1.9	1.9
	60	100	1614	1645.5	1647.6	1643.4	1644.0	2.0	2.1	1.8	1.9
	80	100	2268.4	2290.5	2288.9	2289.0	2289.3	1.0	0.9	0.9	0.9
	100	100	2617.4	2651.1	2653.5	2644.4	2646.4	1.3	1.4	1.0	1.1
C06	20	10	159.9	167.2	169.6	169.6	168.7	4.6	6.1	6.1	5.5
	40	10	323.5	333.4	332.6	334.0	333.2	3.1	2.8	3.2	3.0
	60	10	505.1	519.9	517.2	519.0	518.5	2.9	2.4	2.8	2.6
	80	10	699.7	718.4	714.7	715.5	715.4	2.7	2.1	2.3	2.2
	100	10	843.8	865.1	860.6	861.1	861.2	2.5	2.0	2.1	2.1
C07	20	30	490.4	501.9	501.9	501.9	501.9	2.3	2.3	2.3	2.3
	40	30	1049.7	1059	1059.9	1059.0	1059.4	0.9	1.0	0.9	0.9
	60	30	1515.9	1529.6	1530	1529.6	1529.6	0.9	0.9	0.9	0.9
	80	30	2206.1	2222.2	2222.1	2222.1	2222.1	0.7	0.7	0.7	0.7
	100	30	2627	2644	2644	2645.4	2644.1	0.6	0.6	0.7	0.7
C08	20	40	434.6	458.3	461.2	458.6	458.0	5.5	6.1	5.5	5.4
	40	40	922	954.3	956.5	951.9	951.8	3.5	3.7	3.2	3.2
	60	40	1360.9	1405	1403.5	1399.4	1403.1	3.2	3.1	2.8	3.1
	80	40	1909.3	1971.5	1965	1954.7	1960.9	3.3	2.9	2.4	2.7
	100	40	2362.8	2436.8	2425	2410.8	2418.3	3.1	2.6	2.0	2.4
C09	20	100	1106.8	1106.8	1106.8	1106.8	1106.8	0.0	0.0	0.0	0.0
	40	100	2189.2	2190.6	2190.6	2190.6	2191.1	0.1	0.1	0.1	0.1
	60	100	3410.4	3410.4	3410.4	3410.4	3410.4	0.0	0.0	0.0	0.0
	80	100	4578.6	4588.1	4588.1	4588.1	4588.1	0.2	0.2	0.2	0.2
	100	100	5430.5	5434.9	5434.9	5434.9	5434.9	0.1	0.1	0.1	0.1
C10	20	100	337.8	350.5	351.5	350.4	350.1	3.8	4.1	3.7	3.6
	40	100	642.8	664.4	667	664.0	663.7	3.4	3.8	3.3	3.3
	60	100	911.1	934.7	936.6	933.1	933.2	2.6	2.8	2.4	2.4
	80	100	1177.6	1209.9	1212.4	1204.1	1205.2	2.7	3.0	2.3	2.3
	100	100	1476.5	1512.3	1514	1504.2	1506.5	2.4	2.5	1.9	2.0
Average				1043.34	1043.19	**1041.53**	1041.92	1.77	1.80	1.66	**1.63**

Table 8: Results obtained by GRASP, SVC, ISA, and HDA on Nice and Path.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
Nice1	25	1000	1000	1034	1037	1040.7	1034.8	3.4	3.7	4.1	3.5
Nice2	50	1000	1001	1047	1038	1047.2	1037.7	4.6	3.7	4.6	3.7
Nice3	100	1000	1001	1041	1035	1036.5	1030.9	4.0	3.4	3.5	3.0
Nice4	200	1000	1001	1037	1026	1030.9	1023	3.6	2.5	2.9	2.2
Nice5	500	1000	1000	1024	1017	1015.0	1008	2.4	1.7	1.5	0.8
Nice6	1000	1000	999	1020	1014	1011.0	1004	2.1	1.5	1.2	0.5
Nice1t	1000	1000	1001	1026	1015	1011.0	1006	2.5	1.4	1.0	0.5
	1000	1000	1001	1022	1016	1010.0	1005	2.1	1.5	0.9	0.4
	1000	1000	1000	1020	1013	1011.0	1005	2.0	1.3	1.1	0.5
	1000	1000	1000	1019	1013	1010.0	1005	1.9	1.3	1.0	0.5
	1000	1000	1000	1022	1014	1010.0	1005	2.2	1.4	1.0	0.5
	1000	1000	1001	1020	1014	1010.0	1005	1.9	1.3	0.9	0.4
	1000	1000	1000	1022	1014	1010.0	1006	2.2	1.4	1.0	0.6
	1000	1000	1001	1021	1016	1012.0	1007	2.0	1.5	1.1	0.6
	1000	1000	1000	1022	1017	1012.0	1005	2.2	1.7	1.2	0.5
	1000	1000	1001	1027	1016	1012.0	1007	2.6	1.5	1.1	0.6
Nice2t	2000	1000	1001	1016	1008	1006.0	1004	1.5	0.7	0.5	0.3
	2000	1000	1001	1015	1011	1005.0	1005	1.4	1.0	0.4	0.4
	2000	1000	1000	1016	1008	1007.0	1005	1.6	0.8	0.7	0.5
	2000	1000	1000	1014	1007	1006.0	1003	1.4	0.7	0.6	0.3
	2000	1000	1000	1015	1008	1006.0	1003	1.5	0.8	0.6	0.3
	2000	1000	1000	1016	1002	1005.0	1004	1.6	0.2	0.5	0.4
	2000	1000	1001	1016	1007	1007.0	1004	1.5	0.6	0.6	0.3
	2000	1000	1001	1014	1006	1006.0	1003	1.3	0.5	0.5	0.2
	2000	1000	1001	1016	1008	1007.0	1006	1.5	0.7	0.6	0.5
	2000	1000	1001	1016	1009	1007.0	1005	1.5	0.8	0.6	0.4
Nice5t	5000	1000	1000	1010	1003	1003.0	1003	1.0	0.3	0.3	0.3
	5000	1000	1001	1011	1005	1003.0	1003	1.0	0.4	0.2	0.2
	5000	1000	1001	1010	1002	1003.0	1003	0.9	0.1	0.2	0.2
	5000	1000	1000	1009	1005	1002.0	1003	0.9	0.5	0.2	0.3
	5000	1000	1001	1011	1006	1003.0	1004	1.0	0.5	0.2	0.3
	5000	1000	1000	1009	1001	1002.0	1001	0.9	0.1	0.2	0.1
	5000	1000	1001	1011	1004	1003.0	1003	1.0	0.3	0.2	0.2
	5000	1000	1000	1011	1004	1002.0	1003	1.1	0.4	0.2	0.3
	5000	1000	1001	1010	1004	1003.0	1003	0.9	0.3	0.2	0.2
	5000	1000	1000	1010	1005	1003.0	1003	1.0	0.5	0.3	0.3
Path1	25	1000	1001	1042	1042	1042.0	1041.8	4.1	4.1	4.1	4.1
Path2	50	1000	1000	1019	1014	1014.7	1011.9	1.9	1.4	1.5	1.2
Path3	100	1000	1000	1027	1022	1022.6	1025.1	2.7	2.2	2.3	2.5
Path4	200	1000	1002	1023	1018	1017.7	1013	2.1	1.6	1.6	1.1
Path5	500	1000	1000	1034	1022	1020.0	1016	3.4	2.2	2.0	1.6
Path6	1000	1000	1002	1026	1018	1011.0	1010	2.4	1.6	0.9	0.8
Path1t	1000	1000	999	1019	1011	1007.0	1003	2.0	1.2	0.8	0.4
	1000	1000	1001	1018	1010	1006.0	1005	1.7	0.9	0.5	0.4
	1000	1000	1001	1018	1013	1008.0	1006	1.7	1.2	0.7	0.5
	1000	1000	1000	1016	1009	1006.0	1003	1.6	0.9	0.6	0.3
	1000	1000	1003	1024	1017	1010.0	1010	2.1	1.4	0.7	0.7
	1000	1000	1002	1018	1013	1010.0	1005	1.6	1.1	0.8	0.3
	1000	1000	999	1019	1012	1008.0	1004	2.0	1.3	0.9	0.5
	1000	1000	1000	1020	1012	1008.0	1006	2.0	1.2	0.8	0.6
	1000	1000	999	1019	1012	1006.0	1003	2.0	1.3	0.7	0.4
	1000	1000	1002	1018	1011	1008.0	1006	1.6	0.9	0.6	0.4

Table 8: Continued.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
Path2t	2000	1000	1000	1015	1009	1006.0	1002	1.5	0.9	0.6	0.2
	2000	1000	1002	1016	1010	1007.0	1004	1.4	0.8	0.5	0.2
	2000	1000	1000	1015	1011	1006.0	1003	1.5	1.1	0.6	0.3
	2000	1000	999	1014	1007	1003.0	1000	1.5	0.8	0.4	0.1
	2000	1000	1002	1018	1012	1008.0	1006	1.6	1.0	0.6	0.4
	2000	1000	1002	1016	1011	1007.0	1003	1.4	0.9	0.5	0.1
	2000	1000	998	1011	1007	1004.0	1001	1.3	0.9	0.6	0.3
	2000	1000	998	1014	1010	1004.0	1003	1.6	1.2	0.6	0.5
	2000	1000	1001	1017	1010	1008.0	1003	1.6	0.9	0.7	0.2
	2000	1000	1003	1018	1009	1009.0	1005	1.5	0.6	0.6	0.2
Path5t	5000	1000	1000	1010	1002	1003.0	1001	1.0	0.2	0.3	0.1
	5000	1000	998	1009	1003	1001.0	1001	1.1	0.5	0.3	0.3
	5000	1000	1000	1011	1002	1003.0	1002	1.1	0.2	0.3	0.2
	5000	1000	995	1006	998	997.0	997	1.1	0.3	0.2	0.2
	5000	1000	1004	1016	1005	1006.0	1006	1.2	0.1	0.2	0.2
	5000	1000	1000	1009	1003	1002.0	1001	0.9	0.3	0.2	0.1
	5000	1000	998	1009	1001	1001.0	1000	1.1	0.3	0.3	0.2
	5000	1000	996	1007	998	999.0	998	1.1	0.2	0.3	0.2
	5000	1000	997	1007	999	999.0	998	1.0	0.2	0.2	0.1
	5000	1000	1002	1013	1004	1004.0	1005	1.1	0.2	0.2	0.3
Average				1016.63	1009.75	1007.36	**1006.57**	1.65	0.96	0.72	**0.63**

Table 9: Results obtained by GRASP, SVC, ISA, and HDA on ZDF.

	Instance			meanh				Gap			
	n	W	LB	GRASP	SVC	ISA	HDA	GRASP	SVC	ISA	HDA
zdf1	580	100	330	333	331	330.0	330	0.9	0.3	0.0	0
zdf2	660	100	357	360	358	357.0	357	0.8	0.3	0.0	0
zdf3	740	100	384	387	385	384.0	384	0.8	0.3	0.0	0
zdf4	820	100	407	410	408	407.0	407	0.7	0.2	0.0	0
zdf5	900	100	434	437	434	434.0	434	0.7	0.0	0.0	0
zdf6	1532	3000	4872	5251	5085	5081.8	5066	7.8	4.4	4.3	4
zdf7	2432	3000	4852	5163	5083	5084.7	5017	6.4	4.8	4.8	3.4
zdf8	2532	3000	5172	5544	5386	5549.0	5397	7.2	4.1	7.3	4.4
zdf9	5032	3000	5172	5476	5468	5404.0	5408	5.9	5.7	4.5	4.6
zdf10	5064	6000	5172	5570	5462	5419.0	5433	7.7	5.6	4.8	5
zdf11	7564	6000	5172	5562	5516	5419.0	5439	7.5	6.7	4.8	5.2
zdf12	10064	6000	5172	—	5651	5454.0	5403	—	9.3	5.5	4.5
zdf13	15096	9000	5172	—	5600	5415.0	5415	—	8.3	4.7	4.7
zdf14	25032	3000	5172	—	5468	5286.0	5353	—	5.7	2.2	3.5
zdf15	50032	3000	5172	—	5960	5172.0	5273	—	15.2	0.0	2
zdf16	75032	3000	5172	—	5931	5172.0	5203	—	14.7	0.0	0.6
Average					3907.88	3773.03	**3769.94**	5.34		2.67	**2.62**

The symbol "—" denotes that the GRASP executable program cannot return the results, so its average gap is not given. From Table 9, we can observe that SVC and ISA return an average gap of 5.34 and 2.67, respectively. HDA with an average gap of 2.62 performs better than SVC and ISA. ISA and HDA perform well for large instances.

5. Conclusions

A hybrid demon algorithm for 2SP is presented in this paper. This algorithm improves the scoring rule presented by Leung et al. [18] and a demon algorithm with one parameter is used to improve the quality of the solutions. Computational results

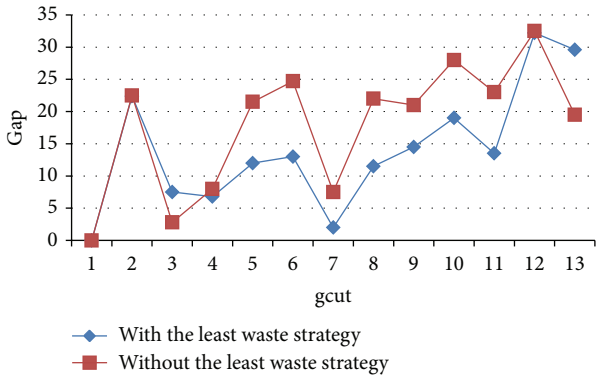


Figure 8: Effect of the least waste strategy on the data set gcut.

have shown that HDA outperforms well-known GRASP, SVC, and ISA. The experiments in Section 3 show that HDA is more stable and efficient for different D value. HDA performs well for large instances; future work is to further improve the quality of the solutions by combining exact algorithms or other metaheuristic algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (Grant no. 71272085).

References

[1] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: a survey," *European Journal of Operational Research*, vol. 141, no. 2, pp. 241–252, 2002.

[2] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2D strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257–300, 2001.

[3] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.

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

[5] M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi, "Exact algorithms for the two-dimensional strip packing problem with and without rotations," *European Journal of Operational Research*, vol. 198, no. 1, pp. 73–83, 2009.

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

[7] S. Jakobs, "On genetic algorithms for the packing of polygons," *European Journal of Operational Research*, vol. 88, no. 1, pp. 165–181, 1996.

[8] E. K. Burke, G. Kendall, and G. Whitwell, "A new placement heuristic for the orthogonal stock-cutting problem," *Operations Research*, vol. 52, no. 4, pp. 655–672, 2004.

[9] D. Zhang, Y. Kang, and A. Deng, "A new heuristic recursive algorithm for the strip rectangular packing problem," *Computers and Operations Research*, vol. 33, no. 8, pp. 2209–2217, 2006.

[10] D. Zhang, S. Han, and W. Ye, "A bricklaying heuristic algorithm for the orthogonal rectangular packing problem," *Chinese Journal of Computers*, vol. 23, no. 3, pp. 509–515, 2008.

[11] L. Wei, D. Zhang, and Q. Chen, "A least wasted first heuristic algorithm for the rectangular packing problem," *Computers and Operations Research*, vol. 36, no. 5, pp. 1608–1614, 2009.

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

[13] A. Bortfeldt, "A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces," *European Journal of Operational Research*, vol. 172, no. 3, pp. 814–837, 2006.

[14] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem," *European Journal of Operational Research*, vol. 128, no. 1, pp. 34–57, 2001.

[15] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, "New heuristic and interactive approaches to 2D rectangular strip packing," *Journal of Experimental Algorithmics*, vol. 10, article 1.2, 2005.

[16] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, "Reactive GRASP for the strip-packing problem," *Computers and Operations Research*, vol. 35, no. 4, pp. 1065–1083, 2008.

[17] E. K. Burke, G. Kendall, and G. Whitwell, "A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem," *INFORMS Journal on Computing*, vol. 21, no. 3, pp. 505–516, 2009.

[18] S. C. H. Leung, D. Zhang, and K. M. Sim, "A two-stage intelligent search algorithm for the two-dimensional strip packing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 57–69, 2011.

[19] G. Belov, G. Scheithauer, and E. A. Mukhacheva, "One-dimensional heuristics adapted for two-dimensional rectangular strip packing," *Journal of the Operational Research Society*, vol. 59, no. 6, pp. 823–832, 2008.

[20] E. K. Burke, M. R. Hyde, and G. Kendall, "A squeaky wheel optimisation methodology for two-dimensional strip packing," *Computers & Operations Research*, vol. 38, no. 7, pp. 1035–1044, 2011.

[21] D. Zhang, L. Wei, S. C. H. Leung, and Q. Chen, "A binary search heuristic algorithm based on randomized local search for the rectangular strip-packing problem," *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 332–345, 2013.

[22] F. G. Ortmann and J. H. van Vuuren, "Modified strip packing heuristics for the rectangular variable-sized bin packing problem," *ORiON*, vol. 26, no. 1, pp. 21–44, 2010.

[23] S. Hong, D. Zhang, H. C. Lau, X. Zeng, and Y.-W. Si, "A hybrid heuristic algorithm for the 2D variable-sized bin packing problem," *European Journal of Operational Research*, vol. 238, no. 1, pp. 95–103, 2014.

[24] E. Pinto and J. F. Oliveira, "Algorithm based on graphs for the non-guillotinable two-dimensional packing problem," in *Proceedings of the 2nd ESICUP Meeting*, Southampton, UK, April 2005.

[25] J. E. Beasley, "An exact two-dimensional nonguillotine cutting tree search procedure," *Operations Research*, vol. 33, no. 1, pp. 49–64, 1985.

[26] J. E. Beasley, "Algorithms for unconstrained two-dimensional guillotine cutting," *Journal of the Operational Research Society*, vol. 36, no. 4, pp. 297–306, 1985.

[27] N. Christofides and C. Whitlock, "An algorithm for two-dimensional cutting problems," *Operations Research*, vol. 25, pp. 30–44, 1977.

[28] B. E. Bengtsson, "Packing rectangular pieces—a heuristic approach," *The Computer Journal*, vol. 25, no. 3, pp. 353–357, 1982.

[29] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin packing algorithms," *Journal of the Operational Research Society*, vol. 38, no. 5, pp. 423–429, 1987.

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

[31] C. L. Valenzuela and P. Y. Wang, "Heuristics for large strip packing problems with guillotine patterns: an empirical study," in *Proceedings of the 4th Metaheuristics International Conference*, pp. 417–421, University of Porto, Porto, Portugal, 2001.