A Hybrid Algorithm for Strip Packing Problem with Rotation Constraint

Huan Chen[1], Furong Ye[1] and Yain-Whar Si[2]

[1]*Department of Computer Science, Xiamen University, Xiamen, China*
[2]*Department of Computer and Information Science, University of Macau, Macau*

Abstract. Strip packing is a well-known NP-hard problem and it was widely applied in engineering fields. This paper considers a two-dimensional orthogonal strip packing problem. Until now some exact algorithm and mainly heuristics were proposed for two-dimensional orthogonal strip packing problem. While this paper proposes a two-stage hybrid algorithm for it. In the first stage, a heuristic algorithm based on layering idea is developed to construct a solution. In the second stage, a great deluge algorithm is used to further search a better solution. Computational results on several classes of benchmark problems have revealed that the hybrid algorithm improves the results of layer-heuristic, and can compete with other heuristics from the literature.

1 Introduction

Strip packing is a well-known NP-hard problem and has many practical applications in the engineering fields. For example, strip packing can be applied for tasks such as placing goods on shelves in the warehouses, arranging articles and advertisements during the type-setting of the newspapers, cutting rectangular pieces from large sheets of material in the wood or glass industries, and laying very-large-scale integration (VLSI) in VLSI floor planning industry. These applications can be formalized as a strip packing problem according to different constraints and objectives [1]. The interested reader is referred to the literature [1]-[3] for more information on packing problems.

This paper considers a two-dimensional orthogonal strip packing problem: Given a rectangular sheet of given width and unlimited height, and a set of rectangles with arbitrary length and width, the orthogonal strip packing problem is to place each rectangle on the sheet such that no two rectangles overlap and the used height h of sheet is minimized. Let W be the width of the rectangular sheet, and n is the number of rectangles, let hi and wi be the length and width of rectangle i (i=1, 2,..., *n*) respectively. Where, we assume that the edges of each rectangle are parallel to the edges of the sheet. In addition, all rectangles except for the rectangular sheet are permitted to be rotated when they are placed. A formal definition for this problem can be found in [4].

The two-dimensional orthogonal strip packing problem mentioned above belongs to a subset of classical cutting and packing problems and has been shown to be NP-hard [5]. Some exact algorithms for orthogonal two-dimension cutting and packing problem were proposed in [6] and [7]. However, they might be impractical for large problems because large amount of computational time is needed to obtain an optimal solution. Therefore, heuristic algorithms, which can produce a good approximation solution within an acceptable time, are preferred to solving this class of problems. Examples of heuristic algorithms include the well-known bottom-left (BL), bottom-left-fill (BLF) and other heuristic methods [8]. In addition, some new heuristic algorithms such as constructive approach [9], new placement heuristics [10], and heuristic recursion algorithm [4] were developed to solve this class of packing problems. These heuristic algorithms can obtain a good solution in a short time. In order to obtain a better solution, heuristic algorithms are often combined with meta-heuristic algorithms such as genetic algorithms [11]-[13], neural network and GA [14], simulated annealing algorithms [15]-[17] and other meta-heuristics [18]-[23]. An empirical investigation of meta-heuristic and heuristic algorithms for the orthogonal packing problem of rectangles is given by Hopper and Turton [24]. Recently, some excellent hybrid algorithms were developed [25]-[27]. In particular, a class of deterministic heuristics is developed by a number of researchers and they are successful in obtaining the better results [20], [28], [29]. Leung and Zhang [30] proposed a fast layer-heuristic algorithm, which have many applications in routing problem with loading constraints [31], [32]. Several efficient heuristic algorithms for the variants of packing problem were also developed [33]-[35] recently. In this paper, a hybrid algorithm for solving the orthogonal strip packing problem with rotation constraint is developed by combining layer-based heuristics with a great deluge algorithm. Computational results on several classes of benchmark problem instances have shown that the hybrid algorithm can compete with other heuristics.

2 Imprroved layer-based heuristic

Layer-based heuristic was proposed by Leung and Zhang [30]. It is a fast heuristic algorithm for strip packing problem. The idea of layer-based heuristic is as follows:

(1) Select a reference item r from unpacked items.

(2) Stack some unpacked items above the item r to determine a reference line.

(3) Pack the available space under the reference line as follows:

(3.1) Determine the lowest available space

(3.2) Select an unpacked item i with maximal fitness value from unpacked items.

(3.3) Pack the item i and update the available space s;

(3.4) If there is an unpacked item can be packed into the available space s, go to (3.1);

(4) if there exists an unpacked item, go to (1).

The detailed process of layer-based heuristic based on the idea of fitness value is given in Leung and Zhang [30]. There are four cases when the fitness value is computed in the process of layer-based heuristic [30]. In fact, there are two special cases that should be given a bigger fitness value when $h1 \geqslant h2$ and $h1 < h2$, respectively. The case for $h1 \geqslant h2$ is shown in Figure 1(a). And for the case given in Figure 1(b), the item R should be given higher fitness value than the item R from Figure 1(a). Also, the item R in Figure 1(d) should be given higher fitness value than the item R from Figure 1(c). Meanwhile, there also are two special cases as $h1 < h2$, and they are shown in Fig. 2.

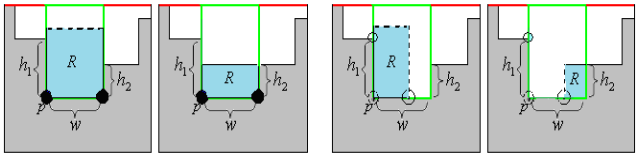

Figure 1. Two special cases of h1⩾h2.

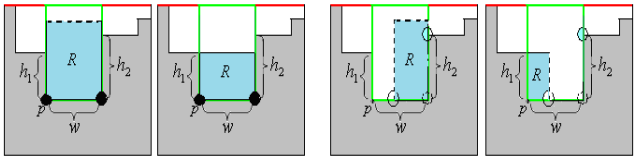

Figure 2. Two special cases of h1<h2.

3 Hybrid algorithm

The great deluge algorithm (GDA) is a general algorithm which is applied to solving optimization problems. It is similar to the hill-climbing and simulated annealing algorithms in many ways. GDA comes from the analogy that in a great deluge a person climbing a hill will try to move in any direction that does not get his/her feet wet in the hope of finding a way up as the water level rises. In a typical implementation of the GDA, the algorithm starts with an initial solution which is a poor approximation S of the optimum solution. A numerical value called "badness" is computed based on S and it measures how undesirable the initial approximation is. The higher the value of badness, the more undesirable the approximate solution is. Another numerical value called "tolerance" is calculated based on a number of factors, often including the initial badness.

For a given solution x, we can construct a neighborhood $N(x)$. A new approximate solution x' which is called a neighbor of x, is selected from $N(x)$. The badness b' of x' is computed and compared with the tolerance. If b' is better than tolerance, then the algorithm is restarted by setting $x:=x'$, and tolerance:=decay(tolerance), where decay is a function that lowers the tolerance (representing a rise in water levels). If b' is worse than tolerance, a different neighbor x^* of x is chosen and the process is repeated. If all the neighbors of x produce approximate solutions beyond the tolerance value, then the algorithm is terminated and x is returned as the best approximate solution obtained.

```
GDA()
1   best=besth,level=3;
2   while(1)
3    for(i=0;i<2*typen;i++)
4        k1=rand()%(typen-1);
5        do k2=rand()%typen;
6        while(k2==k1);
7        swap(x[k1],x[k2]);
8        flag1=Heuristicplacing(x);
9        up=flag1-best;
10       if(up<=level)  best=flag1;
11       if(best<besth)
12           besth=best;
13           level=level-up;
14       else
15           swap(x[k1],x[k2]);
16       end = clock();
17       double runtime=(float)(end -
start) / CLK_TCK;
18       if(runtime>60.0)
19           return besth;
20   rrr=rand()%3;
21   if(rrr==0) sort1(x);
22   else if(rrr==1)sort2(x);
23   else if(rrr==2)sort3(x);
```

The detailed GDA for strip packing problem is as follows:

where x is an array and records the orderings of rectangular pieces. Function Sort1(x) sorts a sequence of rectangle pieces according to the non-increasing ordering of area size. Sort2(x) sorts a sequence of rectangle pieces according to the non-increasing ordering of length size. Sort3(x) sorts a sequence of rectangle pieces according to the non-increasing ordering of perimeter size. The parameter "besth" denotes the current minimal height.

Function Heuristicplacing(x) is the same heuristic algorithm used in Leung and Zhang [30]. Therefore, the proposed algorithm based on GDA (FH+GDA) is as follows:

```
FH+GDA()
1  Fastheuristic();
2  GDA();
```

where Fastheuristic() is the function proposed in Leung and Zhang [30]. Note that the performance of heuristic algorithm significantly depends on the packing ordering X of the rectangles. Some results have shown that the packing orderings affect the performance of the presented algorithm [31]. In this paper, we consider three orderings to avoid the search process from being trapped in a local minimum

4 Computational results

In order to test the performance of the proposed algorithm (FH+GDA), a large amount of benchmark are selected and the results of the proposed algorithm are compared with results of some published heuristics. The benchmark instances (C11~C73) in [24] include 21 problem instances ranging from 16 to 197 items. And the instances 50cx~15000cx are taken from [36], whose sized ranges from 25 to 5000. Considering more constraints in strip packing problem, instances zdf1~zdf16 with fix-orientation and instances beng1~beng10 with small-scale are also selected. All problem classes can be downloaded from
http://algorithm.xmu.edu.cn:10000/Download.aspx#p4.
In the section, FH+GDA is with compared BF+TS, BF+SA, BF+GA, HRP, FH and DHA. BF+TS, BF + SA, BF + GA were tested on a Pentium IV machine with 2.0 GHz. The algorithms are executed for 10 times with a time limit of 60s per run. HRP is implemented in C#.net programming language and it was tested on a 2 GHz Pentium 4 notebook computer with 2048 MB of RAM. FH is a deterministic algorithm and it was tested on a 2 GHz Pentium 4 notebook computer with 2048 MB of RAM with the time limit of 60s per run. DHA was implemented in C programming language and executed as a single process/thread, and all tests were performed on a personal computer with 3.0 GHz CPU and 2.0 GB memory with a time limit of 1000s. FH+GDA was coded in C++ and executed on a TECRA M2V computer with a 1.6GHz CPU and 256 MB of RAM. The proposed algorithm is executed five times and each execution last for maximum of 60 seconds. The comparison of the performance of different algorithms is based on h and $time$, where, h and $time$ denote the height and running time (in second) respectively.

4.1 The computational results on the test problems permitting rotation

In the part, the computational results are from the instances permitting rotation. Meanwhile, BF+GA, BF+SA and BF+TS in [16], HRP [20], FH [30] and DHA [29] are introduced for comparison. The results of BF+GA, BF+SA and BF+TS are from [16], the results of HRP and FH are from [30], and the results of DHA are from [29]. In tables, "op" represents the optimal height, OPTMA# denotes the number of the optimal height obtained by different algorithms. BEST# denotes the number of the best height obtained by different algorithms.

Since the previous paper did not supply the time of experiment. The comparison among BF+TS, BF+SA, BF+GA and FH+GDA is only concentrated on the h, and the results is shown in Table1. Obviously, FH+GDA achieves the best result in every instances among four strategies. The comparison among FH+GDA and other three algorithms on $time$ and h is presented in Table 2. Also, FH+GDA shows efficiency on all instances, FH+GDA gets best results in all instances and do not cost much time. Table 1 and Table 2 clearly indicate that FH+GDA performs better than other algorithms on instances permitting rotation.

Table 1. The results of BF+TS, BF+SA, BF+GA and FH+GDA on instances permitting rotation.

in	op	BF +TS(h)	BF +SA(h)	BF +GA(h)	FH+GDA
C11	20	**20**	**20**	**20**	**20**
C12	20	21	**20**	21	**20**
C13	20	**20**	**20**	**20**	**20**
C21	15	16	16	16	**15**
C22	15	16	16	16	**15**
C23	15	16	16	16	**15**
C31	30	31	31	31	**30**
C32	30	32	31	32	**30**
C33	30	31	31	31	**30**
C41	60	62	61	62	**60**
C42	60	62	61	62	**60**
C43	60	61	61	62	**60**
C51	90	92	91	92	**90**
C52	90	92	91	92	**90**
C53	90	92	92	92	**90**
C61	120	122	122	122	**120**
C62	120	121	121	121	**120**
C63	120	122	122	122	**120**
C71	240	245	244	245	**241**
C72	240	244	244	244	**240.6**
C73	240	245	245	245	**241**
OPTMA#		2	3	2	18
BEST#		2	3	2	21

Table 2. The results of HRP, FH, DHA and FH+GDA on instances permitting rotation.

in	op	HRP		FH		DHA		FH+GDA	
		h	$time$	h	$time$	h	$time$	h	$time$
C11	20	**20**	0.03	**20**	0	**20**	0	**20**	0.01
C12	20	**20**	0.19	**20**	0.01	21	0.3	**20**	0

C13	20	**20**	0.03	21	0	**20**	0.02	**20**	0
C21	15	**15**	0.27	16	0.02	**15**	0.59	**15**	0.07
C22	15	**15**	0.06	**15**	0	**15**	0.23	**15**	0
C23	15	**15**	0.05	**15**	0	**15**	0.06	**15**	0
C31	30	31	0.61	31	0.02	31	3.27	**30**	0.01
C32	30	31	0.61	31	0.02	32	5.14	**30**	0.97
C33	30	31	0.63	32	0.02	**30**	0.25	**30**	0.08
C41	60	61	1.84	61	0.16	61	20.86	**60**	0.20
C42	60	**60**	0.23	61	0.11	61	16.95	**60**	1.20
C43	60	61	2.02	61	0.08	**60**	14.62	**60**	0.05
C51	90	91	4.3	91	0.28	**90**	2.02	**90**	0.18
C52	90	**90**	1.34	**90**	0	**90**	15.62	**90**	0.07
C53	90	91	4.3	91	0.3	**90**	39.18	**90**	1.38
C61	120	121	9.84	121	0.83	121	169.53	**120**	0.04
C62	120	121	8.38	121	0.89	121	139.25	**120**	2.07
C63	120	121	9.94	121	0.76	**120**	2.7	**120**	4.21
C71	240	241	61.48	241	13.91	241	1000	**241**	60.00
C72	240	241	58.91	241	11.64	241	1000	**240.6**	1.73
C73	240	241	62.9	241	15	241	1000	**241**	60.00
OPTMA#		8		5		11		18	
BEST#		10		7		13		21	

In order to extensively test the performance of FH+GDA, A large-scale problem is considered. Table 3 presents the comparison of computational results of four algorithms with instances 50cx~15000cx. According to the table, FH+GDA outperforms other three algorithms. FH+GDA find smaller *h* than FH in 50xc and 100cx, and solutions of other instances obtained by FH+GDA are the same as FH. It reveals that FH+GDA can improve the solutions obtained by FH. We can observe that BEST# and OPTIMA# of FH+GDA is highest, which means FH+GDA shows efficiency with large-scale problems as well. Meanwhile, the *time* of FH+GDA is smaller than the *time* of HRP and DHA for most instances.

Table 3. The results of large-scale instances.

in	op	HRP		FH	
		h	*time*	*h*	*time*
50cx	600	615	10.09	624	0.33
100cx	600	615	47.88	619	3.56
500cx	600	611	87.31	**600**	2.00
1000cx	600	607	86.08	**600**	0.02
5000cx	600	607	9256.37	**600**	0.05
10000cx	600	—	—	**600**	0.05
15000cx	600	—	—	**600**	0.06
OPTMA#		0		5	
BEST#		0		5	

	op	DHA		FH+GDA	
		h	*time*	*h*	*time*
50cx	600	644	26.91	**606.8**	60.00
100cx	600	637	266.64	**609**	60.00
500cx	600	601	1000	**600**	4.62
1000cx	600	**600**	628.48	**600**	1.38
5000cx	600	**600**	3.92	**600**	0.01
10000cx	600	**600**	6.5	**600**	0.02
15000cx	600	**600**	9.88	**600**	0.02
OPTMA#		4		5	
BEST#		4		7	

4.2 The computational results on test problems with fixed orientation

In this paper, similar to [30], we test the performance of FH+GDA with several large-scale instances with fixed orientation. The well-known GRASP, SVC, and FH algorithms are selected for comparison in the Table 4. The computation results of other three algorithms can be accessed from [30] directly. Likewise, FH+GDA gets best results among four algorithm in most instances. Due to the large scale of problems, FH+GDA only achieve better solution in two instances and improve FH slightly, since FH+GDA is executed in limited 60 seconds.

Table 4. The results of large-scale instances with fixed orientation.

in	op	GRASP	SVC	FH	FH+GDA
zdf1	330	333	331	**330**	**330**
zdf2	357	360	358	**357**	**357**
zdf3	384	387	385	**384**	**384**
zdf4	407	410	408	**407**	**407**
zdf5	434	437	**434**	**434**	**434**
zdf6	4872	5300	**5085**	5311	5102
zdf7	4852	5163	**5083**	5311	5102
zdf8	5172	5544	5386	**5360**	**5360**
zdf9	5172	5476	5468	**5402**	**5402**
zdf10	5172	5570	5462	**5178**	**5178**
zdf11	5172	5608	5516	**5174**	**5174**
zdf12	5172	—	5651	5175	**5173**
zdf13	5172	—	5600	**5172**	**5172**
zdf14	5172	—	5468	5257	**5244**
zdf15	5172	—	5960	**5402**	**5402**
zdf16	5172	—	5931	**5402**	**5402**
OPTMA#		0	1	**6**	**6**
BEST#			3	**12**	**14**

4.3 The computational results on problems with non-zero waste

Similar to [29], we apply the benchmark Beng [37] which includes ten small-scale instances ($20 \leqslant n \leqslant 200$) with fixed orientation constraint. The lower bounds of these problems are known but the optimal solutions are still not yet known.

Table 5 reports the results achieved by the exact staircase algorithms [38], DHA and FH+GDA. The

staircase algorithms were executed on a 3.0 GHz Pentium 4 computer with 1.0 GB memory with the time limit of 3600s. W denotes the width of the rectangular board in the table. From Table 5, we could clearly observe that three algorithms obtain the same height as the lower bounds, namely the optimal height for all the instances. However, FH+GDA is much faster than DHA, for all instances, the *time* of FH+GDA do not beyond 0.01s.

Table 5. The results on problems with non-zero waste.

in	n	W	op	Staircase		DHA		FH+GDA	
				h	time	h	time	h	time
beng1	20	25	30	**30**	0.08	**30**	0.78	**30**	0.003
beng2	40	25	57	**57**	0.12	**57**	6.28	**57**	0.001
beng3	60	25	84	**84**	0.1	**84**	17.06	**84**	0.001
beng4	80	25	107	**107**	0.08	**107**	34.74	**107**	0.002
beng5	100	25	134	**134**	0.08	**134**	52.34	**134**	0
beng6	40	40	36	**36**	0.1	**36**	7.08	**36**	0
beng7	80	40	67	**67**	0.11	**67**	36.17	**67**	0.001
beng8	120	40	101	**101**	0.17	**101**	85.78	**101**	0.001
beng9	160	40	126	**126**	0.23	**126**	145.11	**126**	0
beng10	200	40	156	**156**	0.81	**156**	231.27	**156**	0.001
OPTMA#				10		10		10	

5 Conclusions

A hybrid algorithm based on a layer-based heuristic and a great deluge algorithm for the strip packing problem is presented. The proposed algorithm is able to solve the strip packing problem efficiently and further improve the packing results of FH. Extensive computational results show that FH+GDA can compete with meta-heuristics in terms of both solution quality and execution time. FH+GDA is superior especially for large test problems. Therefore, FH+GDA may be of great practical value to the rational layout of the rectangular objects from the engineering fields, such as the wood-, glass- and paper industry, and the textile and leather industry. Future work is to further improve the performance of this algorithm and extend it to solve other packing problems.

Acknowledgment

The work was partially supported by the National Nature Science Foundation of China (61272003).

References

1. A. Lodi, S. Martello, M. Monaci, Eur. J. Oper. Res. **141**, 241–252 (2002)
2. D.Pisinger, Eur. J. Oper. Res. **141**, 382–392 (2002)
3. E. Silva, J.F. Oliveira, G. Wäscher, Eur. J. Oper. Res. **237**, 846-856 (2014)
4. D. Zhang, Y. Kang, A. Deng, Comput. Oper. Res. **33**, 2209–2217 (2006)
5. J.E. Beasley, Oper. Res. **33**, 49–64 (1985)
6. S. Martello, M. Monaci, D. Vigo, INFORMS J. Comput. **15**, 310–319 (2003).
7. Y. Cui, L. Huang, Comput. Optim. Appl. **33**, 287-301 (2006)
8. B. Chazelle, IEEE Trans. Comput, **32**, 697–707 (1983)
9. M. Hifi, R.M Hallah, Int. Trans. Oper. Res. **10**, 1–22 (2013)
10. E.K. Burke, G. Kendall, G. Whitwell, Oper. Res. **52**, 655–671 (2014)
11. S. Jakobs, Eur. J. Oper. Res. **88**, 165–181(1996)
12. D. Liu, H. Teng, Eur. J. Oper. Res. **112**, 413–419 (1999)
13. D. Zhang, S. Chen, Y. Liu, Acat Automatica Sin. **33**, 911-916 (2007)
14. C.H. Dagli, P. Poshyanonda, J. Intell. Manuf. **8**, 177–190 (1997)
15. L. Wei, D. Zhang, Q. Chen, Comput. Oper. Res. **36**, 1608-1614 (2009)
16. E.K. Burke, G. Kendall, G. Whitwell, INFORMS J. Comput. **21**, 505–516 (2009)
17. S.C.H. Leung, D. Zhang, K. M. Sim, Eur. J. Oper. Res. **215**, 57-69 (2011)
18. D. Zhang, A. Deng, Y. Kang, *Lecture Notes in Computer Science 3514*, 783–791 (2005)
19. D. Zhang, Y. Liu, S. Chen, X. Xie, *Lecture Notes in Computer Science 3612*, 1235–1241 (2005)
20. W. Huang, D. Chen, R. Xu, Comput. Oper. Res. **34**, 3270–3280 (2007)
21. D. Zhang, S. Han, W. Ye, Chin. J. Comput, **23**, 509–515 (2008)
22. R. Alvarez-Valdes, F. Parreño, J.M. Tamarit, Comput. Oper. Res. **35**, 1065–1083 (2008)
23. G. Belov, G. Scheithauer, E.A. Mukhacheva, J. Oper. Res. Soc. **59**, 823– 832 (2008)
24. H. Hopper, B.C.H. Turton, Eur. J. Oper. Res. **128**, 34–57 (2001)
25. D. Zhang, L. Wei, S.C.H. Leung, Q. Chen, INFORMS J. Comput. **25**, 332-345 (2013)
26. Y. Wang, L. Chen, Expert Syst. Appl. **42**, 3297-3305 (2015)
27. L. Wei, W.C. Oon, W. Zhu, A. Lim, Eur. J. Oper. Res. **215**, 337-346 (2011)
28. K. He, W. Huang, Y. Jin, Comput. Oper. Res. **39**, 1355-1363 (2012)
29. K. He, Y. Jin, W. Huang, Expert Syst. Appl. **40**, 5542-5550 (2013)
30. S.C.H. Leung, D. Zhang, Expert Syst. Appl. **38**, 13032-13042 (2011)
31. S.C.H. Leung, X. Zhou, D. Zhang, J. Zheng, Comput. Oper. Res. **38**, 205-215 (2010)
32. S.C.H Leung, J. Zheng, D. Zhang, X. Zhou, Flex. Serv. Manuf. J. **22**, 61-82 (2010)
33. S. Hong, D. Zhang, H.C. Lau, X. Zeng, Y. Si, Eur. J. Oper. Res. **238**, 95-103 (2014)
34. L. Wei, T. Tian, W. Zhu, A. Lim, Eur. J. Oper. Res. **239**, 58-69 (2014)
35. K. He, P. Ji, C. Li, Eur. J. Oper. Res. **241**, 674-685 (2015)
36. E. Pinto, J.F. Oliveira, *Second ESICUP Meeting,* **29**, 531– 540 (2015)
37. B.E. Bengtsson, Comput. J. **25**, 353–357 (1982)
38. M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, H. Nagamochi, Eur. J. Oper. Res. **198**, 73–83 (2009)