A Hyper-Heuristic Approach to Strip Packing Problems

Edmund K. Burke, Qiang Guo, and Graham Kendall

School of Computer Science, University of Nottingham,
Nottingham, NG8 1BB, United Kingdom
{ekb,qxg,gxk}@cs.nott.ac.uk
http://www.asap.cs.nott.ac.uk/

Abstract. In this paper we propose a genetic algorithm based hyper-heuristic for producing good quality solutions to strip packing problems. Instead of using just a single decoding heuristic, we employ a set of heuristics. This enables us to search a larger solution space without loss of efficiency. Empirical studies are presented on two-dimensional orthogonal strip packing problems which demonstrate that the algorithm operates well across a wide range of problem instances.

Keywords: Hyper-heuristic, Strip Packing.

1 Introduction

Cutting and packing problems are a large family of problems arising in many industrial settings, from stock-cutting in the paper, metal, glass and wood industries to container, pallet loading, multi-processor scheduling and other resource allocation problems. Many heuristics have been devised for the problems, and their performance have been intensively studied [1,2]. For the offline version of the problems, where all pieces are known beforehand, results can usually be improved by a meta-heuristic search [3,4,5].

Coffman et al. [1] pointed out that the performance of a single heuristic, in terms of both worst-case and average-case, may vary depending on given instances. Their proofs presume a uniform distribution of item sizes and are applicable to one dimensional problems. The performance for other distributions and higher dimensional instances is not so well understood. The lack of insight into instance properties and heuristic behaviour causes difficulty in practical situations when we need to select an appropriate heuristic for the problem at hand. In addition, many heuristics are designed to guarantee feasible packings, especially in high dimensional cases, but may not be able to construct certain patterns, effectively stopping us being able to find the optimal solution [6]. This limitation is also inherited by any meta-heuristic which employs only one of these heuristics as the mapping function from representation space to solution space.

Hyper-heuristics are currently receiving some attention in the literature [7,8,9]. The approach is motivated by the goal of raising the level of generality of search

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 465–474, 2010.

methodologies [7] and they have been successfully applied to cutting and packing problems [10,11]. In this paper, we propose a novel hyper-heuristic approach which helps intelligently choose a suitable heuristic each time we need to place an item. The main differences from previous, standard meta-heuristic approaches is that a set of heuristics will be utilised. The heuristic set will map the representation space to the solution space so that the algorithm can avoid the shortcomings of only using one heuristic. To demonstrate the effectiveness of the approach, we propose a hyper-heuristic based on a genetic algorithm. The chromosome contains not only which item to pack, but also which heuristic(s) are available to pack that item. Therefore, we enhance the standard genetic algorithm (GA) encoding, where a chromosome is a permutation of items, by adding a set of heuristics together with probabilistic information. Such information will facilitate choice decisions to select heuristics rather than rely on a user's arbitrary judgement. Compared to the hyper-heuristic by Ross [12,10], the learning mechanism updates the probabilities of applying heuristics according to their historical performance, rather than through a learning classifier system.

2 Related Work

In classical two-dimensional orthogonal strip packing problems [6], we are given a container $\mathbf{C}$, with width $\mathbf{W}$ and infinite height. We are required to pack into $\mathbf{C}$ a set of small rectangles $\mathbf{R} = \{r_1, r_2, ..., r_n\}$ with (w_i, h_i) denoting the width and height for each $r_i \in \mathbf{R}$. The objective is to minimise the total height of the packed rectangles. Typical assumptions, as summarised by Fekete and Schepers [13], are:

1. Each edge of the rectangles have to be parallel to one edge of the container (orthogonal);
2. We do not require guillotine cutting (free-form);
3. All rectangles must be within the container (closeness);
4. Rectangles must not overlap with each other (disjoint);
5. Rectangles cannot be rotated (fixed orientation).

The problem can be classified as two-dimensional regular open dimensional packing (2D-R-ODP) according to the typology proposed by Wäscher et al. [14].

Some heuristics for one-dimensional cases can be modified for the strip packing problem. Baker et al. [6] presented a bottom-up left-justified (BL) heuristic, which finds the lowest feasible space, similar to one dimensional First Fit (FF), and packs left justified. Another heuristic has been described by Liu and Teng [15]. Each rectangle is dropped from the top right corner of the container and moved down and then left until it settles at a stable position. The heuristic overlooks any holes formed by preceding rectangles in the partial packing. Therefore, it can be regarded as Next Fit (NF) [1] which never utilises empty spaces produced at an earlier stage. The Best Fit (BF) policy [1] fits a piece into the smallest feasible space. Hayek et al. [16] proposed a way to search for the smallest feasible space. Burke et al. [17] designed a Best Fit Decreasing Width

(BFDW) heuristic, which was later enhanced using a meta-heuristic as a post process operation [20].

Meta-heuristic approaches have also attracted much attention as they have been shown to produce good quality solutions for cutting and packing problems [18,19,20,21]. The most common way of implementing these methodologies for packing problems is to use a hybrid strategy which combines an iterative search component together with a placement heuristic. For example in a typical GA method, the GA searches for the best permutation of shapes that enables a decoder to return a good packing solution. Given that the different decoders and parameter sets perform differently, it normally relies on a user's decision (or even intuition) to make appropriate choices.

Being aware of the difficulties faced by heuristics and meta-heuristics, a natural question to ask is if we can develop an automated system which requires less human interaction and can deal with a wide range of problems? Ross et al. [12,10] presented two approaches for one-dimensional bin packing problems. They associate a set of packing heuristics with different packing statuses. A learning classifier system [12] and a genetic algorithm [10], acting as higher level managers, search for an appropriate packing heuristic to be employed at each step of the packing. In these approaches, a set of predefined problem statuses is used to describe the status of partially filled bins and the remaining pieces. In [12], a classifier system determines the problem status and decides which low level heuristic to call. This approach requires a good understanding between problems and low level heuristics. For many situations, such as in high dimensional cases, the task of gaining such understanding, and enumerating all possible situations, can be non-trivial. In [10], a GA is employed to detect the problem status and suitable heuristics to employ. As we demonstrate in the next section, some heuristics such as left-justify or right-justify, may not be relevant to the problem status, but are still crucial in some cases.

3 The GA-Based Hyper-Heuristic Approach

3.1 Overview

Our hyper-heuristic approach is based on a genetic algorithm for the over-riding search strategy (Fig. 1). To facilitate the choice of heuristics, the standard chromosomes are enhanced by combining the sequence of rectangles with heuristic-probability pairs. Section 3.2 provides more details on the chromosomes. Compared to the hyper-heuristic approach using a static learning classifier system [12,10], our approach adopts a roulette-wheel selection mechanism to choose a heuristic from the candidate set (step 2.3 in Fig. 1), along with an adaptive learning mechanism to intelligently recognise a suitable heuristic within a set (steps 2.5 to 2.7 in Fig. 1).

1.1 **for each** individual c_{ind} random shuffle $\{r_1, r_2, \ldots, r_n\}$;
1.2 **for each** rectangle r_i initialize a set of heuristic-probability pairs (h_i^j, p_i^j);

2 **while** $Iteration \leq Iteration_{max}$
2.1 select parents c_x, c_y;
2.2 generate new child $c_{ind'}$ by crossover and mutation;

2.3 choose a heuristic $h_i^{j^*}$ according to its probability $p_i^{j^*}$;
2.4 pack r_i with $h_i^{j^*}$;

2.5 $\Delta = (Height_{c_{ind}} - Height_{c_{ind'}})/Height_{c_{ind}}$;
2.6 $p_i^{j^*} = max\{0, p_i^{j^*} + \Delta\}$;
2.7 **for each** $j \in \mathbf{J} \setminus j^*$, update p_i^j;

Fig. 1. Pseudo-code of the GA-based hyper-heuristic framework

We also compare two alternative versions of the hyper-heuristic to decide the types of heuristic decoders:

Non-competing heuristic sets (NC-HH). The type of heuristics are fixed. They are all available to pack each shape even if the probability value approaches zero (see step 2.7 in Fig. 2). In this version, although the heuristics h_i^j are arbitrarily chosen and remain static, the probabilities p_i^j are updated adaptively and the search procedure is still a dynamic probability selection mechanism. Fig. 2 shows details of the refined procedures of steps 1.2 and 2.7 for this version of the hyper-heuristic.

1.2 **for each** r_i initialize a set of $|\mathbf{J}|$ heuristics, and set each $p_i^j = \frac{1}{|\mathbf{J}|}$;
2.7 **for each** $j \in \mathbf{J} \setminus j^*$, $p_i^j = max\{0, p_i^j - \frac{\Delta}{|\mathbf{J}|-1}\}$;

Fig. 2. Refined step 1.2 and 2.7 for NC-HH

Competing heuristic sets (C-HH). The hyper-heuristic chooses initial heuristic sets, and it allows badly performing heuristics to be replaced (Fig. 3). When initialising, the hyper-heuristic randomly selects a subset of heuristics from all those available. During the updating process, if the probability of a heuristic drops below a threshold level, it will be replaced by another randomly chosen heuristic. Whenever replacement happens, the probabilities of the heuristics will be reset to allow the newly introduced heuristic a fair chance of competing with the surviving members that are already in the set. In effect, all heuristics are competing against each other in order to stay in the candidate set.

1.2 **for each** r_i random select a set of $|\mathbf{J'}| < |\mathbf{J}|$ heuristics, and set each $p_i^j = \frac{1}{|\mathbf{J'}|}$;
2.7 **if** the incumbent heuristic $h_i^{j^*}$ has $p_i^{j^*} < Prob_{th}$ replace $h_i^{j^*}$ with a new heuristic, set $p_i^j = \frac{1}{|\mathbf{J'}|}$, reset other heuristics' probabilities;
otherwise for each $j \in \mathbf{J'} \setminus j^*$, $p_i^j = max\{0, p_i^j - \frac{\Delta}{|\mathbf{J'}|-1}\}$;

Fig. 3. Refined step 1.2 and 2.7 for C-HH

3.2 Chromosomes

We enhance the standard genetic algorithms' chromosome by including with each item (allele) some probabilistic information for heuristic selection. Each allele is denoted as a set of pairs of heuristic h_i^j and probability p_i^j, $i = 1, 2, ..., n$,where n is the number of items and j is a parameter defining the number of candidate decoding heuristics available to each rectangle. Fig. 4 shows a chromosome for the proposed hyper-heuristic methodology.

r_1	r_2	...	r_n
(h_1^1, p_1^1)	(h_2^1, p_2^1)	...	(h_n^1, p_n^1)
(h_1^2, p_1^2)	(h_2^2, p_2^2)		(h_n^2, p_n^2)
...	...	...	...

Fig. 4. A hyper-heuristic GA chromosome

The values of the probabilities (initially set equal) will be updated through a learning mechanism. The choice of a heuristic for each piece will be rewarded or punished according to the results of the final packing height, i.e. the probabilities of incumbent heuristics will be increased if we obtain a better packing, and decreased otherwise. Therefore, the system learns from its interaction with the search problem. For example, a system may find it tends to apply rules finding lower positions for large pieces, while for small pieces there is less difference in heuristic probabilities. The hyper-heuristic uses this adaptive policy to learn how to utilise the heuristics.

3.3 Decoding Heuristics

Decoding heuristics for higher dimensional problems are concerned with two decisions: which space to select for the placement and where in the chosen space to place the item. For the first decision, we will use three categories of heuristics: First Fit, Next Fit and Best Fit. To implement these heuristics, we maintain a list of feasible spaces, initially containing one element of the size of the strip. We recalculate the list after placing each shape, similar to [16].

First Fit (FF) select the feasible space at the lowest level, break ties by choosing the left most space (equivalent to the bottom-up heuristic [6]);

Best Fit (BF) select the feasible space with the smallest area;
Next Fit (NF) spaces not exposed from the top of the partial packing will be removed from the list, then select the lowest feasible space (equivalent to bottom-left move with downward priority [15]).

For the second decision our hyper-heuristic will consider all four corners of a chosen space. Therefore, in our experiments, we have twelve different placement options for each item. The type and quantity of heuristics will affect the performance of the hyper-heuristic, possibly due to the larger the size of the pool, the potentially larger search space (see section 4.3). Therefore, we limit the candidate sets to a more manageable size of four rather than using all twelve heuristics.

3.4 Selection and Replacement Strategy

Parent selection is carried out by truncated selection. By experimentation we found that it is more effective to select parents from the top third, rather than using roulette-wheel selection from the entire population. A child chromosome replaces the worst member in the population, that is not a replica of an existing chromosome.

3.5 Recombination

The GA recombination operators are standard, involving a random two-point order-based crossover (2OX) [5] and mutation. In particular, when exchanging orders of items in a sequence the associated set of heuristics of each item will be exchanged as well. We have implemented two other operators, partial matching crossover (PMX) and single point crossover (1OX), which also guarantee feasibility. Compared with 2OX, PMX makes little difference and 1OX performs slightly worse. The detailed settings of parameters will be shown in Section 4.

4 Experimental Results

To examine the effectiveness of the proposed hyper-heuristics, we have created a set of instances to demonstrate that hyper-heuristics can explore a wider solution space (Section 4.1). We also compare the average performance with standard GAs (Section 4.2). It is also interesting to investigate the impact of the size of the heuristic sets, which is an important parameter affecting the size of the search space (Section 4.3).

The benchmark instances are taken from Burke et al. [17] and the OR-library `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/`. C1 to C7 are seven categories with three instances in each and N1a to N7e are 35 non-guillotine instances. N1 to N12 have a number of items ranging from 10 to 500, and the other two sets of instances have 16 to 197 items.

The algorithm was implemented in C++ and ran on a grid computer with 2.2GHz CPUs, 2GB memory and GCC compiler. To obtain statistics every experiment was run 100 times.

4.1 Feasibility and Optimality

The first set of experiments is designed to evaluate the effects of multiple decoders. We created some instances where gaps have to exist in the middle of patterns in the optimal solutions (as per Baker et al. [6]). Using only one heuristic will fail to achieve the optimal pattern. An example of such an instance is as follows. Nine Items: 60x60, 60x60, 50x50, 50x50, 40x40, 40x40, 10x10, 10x10, 31x30 are to be packed into a strip of width of 151. (Note if the last item was 30x30 and the strip has a width of 150, the shapes would fit perfectly.) The best results achieved by a meta-heuristic with a single placement heuristic (in our experiments GA+NFBL, GA+FFBL, GA+BFBL) and hyper-heuristics (both C-HH and NC-HH versions) are 120 and 110 respectively Fig. 5). It is simple to verify that 110 is the optimal. Assuming the optimal is less than 110, say 109, the whole area of strip needed (including any utilised and wasted areas) is 16,459 (151x109), which is less than the total area of all items 16,530, therefore it is impossible.

Other instances in our dataset are created by choosing a number of pieces and cutting at random points. The hyper-heuristics demonstrates stronger

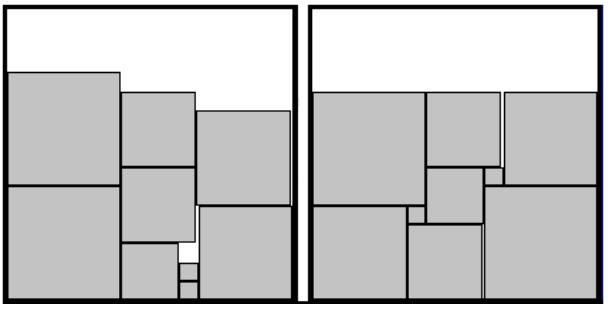

Fig. 5. Best result achieved by meta-heuristic is 120 and optimal achieved by hyper-heuristic is 110

Table 1. Average and best results of new instances

	instance 1		instance 2		instance 3		instance 4		instance 5		instance 6		instance 7		instance 8	
	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg
Next Fit HH	**110**	**112.3**	**110**	**119.3**	110	**113.04**	**110**	120.8	111	**115.26**	120	121.0	112	**114.30**	**116**	**119.67**
GA	120	120.0	120	120.0	110	113.70	120	**120.7**	111	115.45	120	**120.5**	112	115.06	117	120.37
First Fit HH	**110**	**110.1**	**110**	**118.1**	110	**111.85**	**110**	**120.0**	111	**113.10**	120	**120.0**	**110**	**113.26**	116	**117.62**
GA	120	120.0	120	120.0	110	112.43	120	120.1	111	114.31	120	120.2	111	114.20	116	118.40
Best Fit HH	**110**	**110.0**	**110**	**116.3**	110	**111.77**	110	**119.5**	111	**112.43**	120	120.2	**110**	**112.61**	116	**118.50**
GA	120	120.0	120	120.0	110	111.91	110	119.6	111	113.75	**110**	**120.1**	111	113.42	**114**	118.82

performance (Table 1). Comparing best and average results to algorithms applying only one heuristic, hyper-heuristics are better on almost all cases. This experiment provides evidence that hyper-heuristics can avoid the drawbacks of applying only a single heuristic, and find more feasible solutions and, possibly, optimal solutions.

4.2 Performance

In this experiment we further compare our hyper-heuristics to standard meta-heuristics on well known benchmark instances for each category of decoders (FF, NF and BF). The hyper-heuristic (NC-HH) utilises four positioning heuristics while the standard GA uses only one. Table 2 shows that the hyper-heuristics produces superior solutions in more cases on the First Fit and Best Fit and equal solutions on Next Fit. The extra calculations to update the probabilities only causes a minor increase to the CPU time even for larger sized instances, as shown in Table 3 (0.5% on average).

Table 2. Average of all instances

dataset	number of instances	First Fit			Next Fit			Best Fit		
		HH Wins	GA Wins	equal	HH Wins	GA Wins	equal	HH Wins	GA Wins	equal
n1-n12	12	5	5	2	2	9	1	9	2	1
c1-c7	21	12	7	2	8	12	1	11	10	0
n1a-n7e	35	16	19	0	21	14	0	18	17	0
new	8	8	0	0	6	2	0	7	1	0
total	76	41	31	4	37	37	2	45	30	1

Table 3. Average CPU time for 5000 evaluations (milliseconds)

Set size	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12
GA	174	508	924	1546	2224	2216	2820	4076	4786	9902	18358	62022
HH	176	518	906	1568	2230	2208	2824	4074	4856	10008	18542	61950

4.3 Effects of Number of Heuristics in a Set

In the next set of experiments we attempt to find a suitable trade-off between the size of the set (and thus computational time) and solution quality. In Table 4, we present a comparison between four runs of a hyper-heuristic (C-HH version) where heuristics are all randomly chosen and the set size varies between 4 and 8. It can be seen that many of the best results (highlighted) are produced with just four heuristics.

Table 4. Heuristic set size affects results

Set size	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12
size of 4	40.00	51.51	**52.59**	**84.08**	**106.19**	104.78	**110.04**	**85.79**	**156.48**	**154.36**	155.88	317.26
size of 5	40.00	51.32	52.61	84.29	106.42	104.63	110.37	86.26	156.61	154.66	155.97	317.28
size of 6	40.00	**51.08**	52.65	84.39	106.50	104.70	110.20	86.02	156.70	154.80	155.80	317.20
size of 7	40.00	51.22	52.66	84.40	106.42	**104.55**	110.26	86.35	156.63	154.64	**155.75**	**317.13**

5 Conclusion and Future Work

In this paper we have proposed a hyper-heuristic approach to tackle cutting and packing problems. The idea is to combine a set of heuristic decoders with a high level search operator. Empirical studies have demonstrated that the hyper-heuristic approach is superior to standard meta-heuristics which use only one decoder. The potential benefits can be summarised as follows:

- Compared to standard approaches the hyper-heuristic is able to explore a larger solution space. Therefore, it has the potential to find the global optima or deliver better results than other meta-heuristic approaches.
- Its built-in learning mechanism is highly automated requiring less user judgement, as the hyper-heuristic itself will intelligently choose a suitable heuristic to pack a given item. It is also flexible for further expansion by having the option to add new heuristics into the candidate set.

In this paper, the hyper-heuristic utilises a GA as the search methodology and a number of well known heuristics as decoders. The hyper-heuristic is flexible to adopt other search engines, such as Tabu Search or Simulated Annealing. It is also possible to employ other more sophisticated low-level heuristics, such as those considering shared edges. There is scope for further improvement by integrating techniques such as a local search into the hyper-heuristic framework. Like most meta-heuristics, hyper-heuristics are usually computational intensive algorithms. Therefore, it is interesting to investigate if parallelization (e.g. an island model) could solve even lager instances. Further work is also required to understand the dynamics among different level operators and the evolution and interaction between the heuristic search space and solution space.

Acknowledgments. We would like to acknowledge the support of EPSRC (Engineering and Physical Sciences Research Council) for supporting this work (Ref: EP/D061571/1).

References

1. Coffman, E., Garey, M., Johnson, D.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-hard Problems, pp. 46–93. PWS Publishing, Boston (1996)
2. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: a survey. Eur. J. Oper. Res. 141, 241–252 (2002)
3. Aarts, E., Korst, J., Michiels, W.: Simulated annealing. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 7, pp. 187–210. Springer, Heidelberg (2005)
4. Gendreau, M., Potvin, J.: Tabu search. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 6, pp. 165–186. Springer, Heidelberg (2005)

5. Sastry, K., Goldberg, D., Kendall, G.: Genetic algorithms. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 4, pp. 97–126. Springer, Heidelberg (2005)
6. Baker, B., Coffman, E., Rivest, R.: Orthogonal packings in 2 dimensions. SIAM Journal on Computing 9, 846–855 (1980)
7. Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9, 451–470 (2003)
8. Burke, E., Hart, E., Kendall, G., Newall, P., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern research technology. In: Handbook of Metaheuristics, ch. 16, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)
9. Ross, P.: Hyper-heuristics. In: Burke, E., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch. 17, pp. 529–556. Springer Science, Heidelberg (2005)
10. Ross, P., Marín-Blázquez, J.G., Schulenburg, S., Hart, E.: Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003, Part II. LNCS, vol. 2724, pp. 1295–1306. Springer, Heidelberg (2003)
11. Dowsland, K., Gilbert, M., Kendall, G.: A local search approach to a circle cutting problem arising in the motor cycle industry. J. Oper. Res. Soc. 58, 429–438 (2007)
12. Ross, P., Schulenburg, S., Marín-Blázquez, J.G., Hart, E.: Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of Genetic and Evolutionary Computation - GECCO 2002, vol. 6, pp. 942–948 (2002)
13. Fekete, S., Schepers, J.: A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research 29, 353–368 (2004)
14. Wäscher, G., Hauáner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183, 1109–1130 (2007)
15. Liu, D., Teng, H.: An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles. Eur. J. Oper. Res. 112, 413–420 (1999)
16. El Hayek, J., Moukrim, A., Negre, S.: New resolution algorithm and pretreatments for the two-dimensional bin-packing problem. Computers & Operations Research 35, 3184–3201 (2008)
17. Burke, E., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52, 655–671 (2004)
18. Hopper, E., Turton, B.: A review of the application of meta-heuristic algorithms to 2D strip packing problems. Artificial Intelligence Review 16, 257–300 (2001)
19. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive grasp for the strip-packing problem. Computers & Operations Research 35, 1065–1083 (2008)
20. Burke, E., Kendall, G., Whitwell, G.: A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock Cutting Problem. INFORMS Journal on Computing 21(3), 505–516 (2009)
21. Dowsland, K., Herbert, E., Kendall, G., Burke, E.: Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. Eur. J. Oper. Res. 168, 390–402 (2006)