ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/232950526

A dynamic programming-based heuristic for the variable sized two-dimensional bin packing problem

Article *in* International Journal of Production Research · July 2011

DOI: 10.1080/00207543.2010.501549

CITATIONS
9

READS
369

3 authors, including:

Kanliang Wang
Renmin University of China
79 PUBLICATIONS **1,769** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Planning and Scheduling in Operating Theatre View project

scheduling algorithms View project

All content following this page was uploaded by Kanliang Wang on 08 October 2014.

The user has requested enhancement of the downloaded file.

This article was downloaded by: *[2007-2008 Nanyang Technological University]*
On: *26 September 2010*
Access details: *Access Details: [subscription number 910162907]*
Publisher *Taylor & Francis*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713696255

A dynamic programming-based heuristic for the variable sized two-dimensional bin packing problem

Ya Liu[ab]; Chengbin Chu[a]; Kanliang Wang[b]
[a] Ecole Centrale Paris - LGI, 92295 Chatenay-Malabry Cedex, France [b] Xi'an Jiaotong University, 710049 Xi'an, China

First published on: 17 September 2010

To cite this Article Liu, Ya , Chu, Chengbin and Wang, Kanliang(2010) 'A dynamic programming-based heuristic for the variable sized two-dimensional bin packing problem', International Journal of Production Research,, First published on: 17 September 2010 (iFirst)
To link to this Article: DOI: 10.1080/00207543.2010.501549
URL: http://dx.doi.org/10.1080/00207543.2010.501549

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

International Journal of Production Research
2010, 1–17, iFirst

Taylor & Francis
Taylor & Francis Group

A dynamic programming-based heuristic for the variable sized two-dimensional bin packing problem

Ya Liu[ab]*, Chengbin Chu[a] and Kanliang Wang[b]

[a]*Ecole Centrale Paris – LGI, 92295 Chatenay-Malabry Cedex, France;*
[b]*Xi'an Jiaotong University, 28 XianNing Road, 710049 Xi'an, China*

(*Received 9 November 2009; final version received 9 June 2010*)

This paper addresses a variable sized two-dimensional bin packing problem. We propose two heuristics, H1 and H2, stemming from the dynamic programming idea by aggregating states to avoid the explosion in the number of states. These algorithms are elaborated for different purposes: H1 builds a general packing plan for items, while H2 can provide solutions by considering a variety of customer demands, such as guillotine cutting style and rotation of items. The performance of both algorithms is evaluated based on randomly generated instances reported in the literature by comparing them with the lower bounds and optimal solutions for identical bins. Computational results show that the average gaps are 8.97% and 13.41%, respectively, for H1 and H2 compared with lower bounds, and 5.26% and 6.26% compared with optimal solutions for identical bins. We also found that we can save 6.67% of space, on average, by considering variable sized bins instead of a bin packing problem with identical bins.

Keywords: operational research; optimisation

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

1. Introduction

Cutting and packing problems have been studied since the 1950s. Since then, the number of papers dealing with this topic has been growing. The subject is now mainstream in operations research and has been extensively studied in the literature. Cheng and Feiring (1994) and Lodi *et al.* (2002) reviewed the literature on one-dimensional and two-dimensional bin packing problems, respectively. Cheng and Feiring (1994) categorised the research related to the cutting and packing problem in more than 400 books and articles.

Most research assumes that all bins are of the same size. In the few studies of the variable sized problem, most focused on one dimension. Friesen and Langston (1986) presented three efficient approximation algorithms, with the worst-case performance being 2, 3/2 and 4/3, respectively. Chu and La (1994) proposed four polynomial time algorithms and evaluated their worst-case performances. Murgolo (1987) reported a fully polynomial time asymptotic approximation scheme. Kinnersley and Langston (1988), Csirik (1989) and Seiden (2002) considered online variable sized bin packing.

For variable sized two-dimensional bin packing, Valerio de Carvalho (2002) defined an LP formulation. Kang and Park (2003) proposed two greedy algorithms, IFFD and IBFD. They guaranteed a worst-case performance bound of 3/2. Pisinger and Sigurd (2005) used

*Corresponding author. Email: ya.liu@utt.fr

ISSN 0020–7543 print/ISSN 1366–588X online
© 2010 Taylor & Francis
DOI: 10.1080/00207543.2010.501549
http://www.informaworld.com

a branch-and-price algorithm to solve the problem exactly. To solve the pricing problem which consists of two-dimensional knapsack problems, they decomposed the problem into one-dimensional knapsack subproblems. If the solution of the one-dimensional knapsack problem is infeasible for the two-dimensional problem, a valid constraint is added. The procedure is repeated until a feasible solution is found. The computational results show that the lower bounds obtained by column generation are quite tight.

In contrast to two-dimensional bin packing problems with identical bin size and one-dimensional cutting problems with variable bins which have been studied intensively in the literature, the most recent survey papers by Valerio de Carvalho (2002) and Kang and Park (2003) did not report any computational results for variable sized two-dimensional bin packing. Pisinger and Sigurd (2005) proposed an exact algorithm. For instances with 100 items, the computation time even reaches more than one hour and some are not solved to optimality.

In real life, variable sized bin packing problems often arise in industries such as wood, steel, paper and cloth. In glass cutting and wood cutting enterprises, the raw materials are always of different size, while in packing companies the containers are not always of the same size. Compared with the identical bin packing problem, which is considered to be NP-hard, for the variable sized problem it is much more difficult to find an optimal solution in a limited amount of time. In fact, the exact methods proposed by Pisinger and Sigurd (2005) are very demanding both with respect to computation time and memory. That is why we are attempting to develop a heuristic algorithm to obtain satisfactory solutions in an acceptable amount of time. On the other hand, in practical industrial applications, customers may have additional demands, such as the orientation of items and cutting styles. These additional demands make the problem even more difficult to solve, particularly using exact methods. In this paper, besides introducing a heuristic algorithm for a general variable sized problem, we also propose a particular method that can consider a variety of customer demands. To differentiate between these two methods, we denote them by H1 and H2, respectively. The general procedure of these two algorithms is the same, only the subproblems are solved differently.

The paper is organised as follows. Section 2 describes the problem as a set partitioning model. Section 3 presents the general procedure of the two heuristic algorithms. Section 4 states the procedure of H1 and H2 for solving sub-problems. Section 5 reports computational results to evaluate the performance of the algorithms. Section 6 concludes the paper.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

2. Problem description

In this paper, the variable sized bin packing problem has the following characteristics.

(i) All items are rectangular.
(ii) The objective is to minimise the total space of bins used.
(iii) The number of bin types is limited. Bins of different types are of different sizes. The number of bins of each type available is also limited.

The problem consists of packing n items defined by a width w_i and a height h_i into N types of bins. Each bin type $j(j = 1, 2, \ldots, N)$ has its own size with width W_j and height H_j. M_j bins are available for type j. The objective is to minimise the total space of bins used.

This problem can be formulated mathematically as follows:

$$\min \sum_{j=1}^{N} \sum_{k \in P_j} e_j x_{jk}, \tag{1}$$

subject to

$$\sum_{j=1}^{N} \sum_{k \in P_j} a_{ijk} x_{jk} = 1, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{k \in P_j} x_{jk} \leq M_j, \quad j = 1, 2, \ldots, N, \tag{3}$$

$$x_{jk} \in \{0, 1\}, \tag{4}$$

where P_j is the set of all feasible packing patterns for a bin of type j and e_j is the space of bin j, in other words $e_j = W_j \times H_j$. a_{ijk} equals 1 if item i is packed in the kth pattern of type j, and 0 otherwise. The decision variable is x_{jk}. If the kth pattern of bin type j is included in the solution, it equals 1, and 0 otherwise. The first constraint demonstrates that each item should be packed exactly once. The second constraint states that the number of bins used for each type cannot exceed the number available.

3. The general procedure

In this problem, our objective is to select bins loading n items to minimise the total space of bins used. A set of items can be represented by an n-vector B. Similarly, a set of bins can be represented by an N-vector C. Each entry b_i in set B equals 1 if item i belongs to the set, and 0 otherwise. Entry c_j in set C denotes the number of bins of type j contained in the set. At the beginning, let the item set and bin set be B_0 and C_0, respectively. By definition, each entry in B_0 equals 1 and the jth ($j = 1, 2, \ldots, N$) entry in C_0 equals M_j.

The problem can be formulated as a set partitioning problem. For this, B_0 corresponds to the activity set and C_0 corresponds to the resource set. The problem is to partition the set B_0 into subsets, and allocate each subset a bin selected from the set C_0. A packing pattern is a feasible placement of a subset of items into a bin. Each packing pattern incurs a cost (in our problem, it denotes the bin's space). The optimal solution is reached by finding a set of patterns with a minimum total cost such that each item is packed exactly once and the number of bins used for each type does not exceed the number available. Therefore, the problem is to decide how to partition the set B_0, and for each obtained subset, which bin type should be allocated to realise the global objective function.

The problem can be solved optimally using a dynamic programming approach. For any B and C, if items presented by B and bins denoted by C can form packing patterns, then there is a way to form these patterns with the least cost. There exist a B' and a C' such that, in this optimal way, B' and C' form some packing patterns and $B - B'$ and $C - C'$ form a single pattern. The following recursive equation then holds:

$$\gamma(B, C) = \min_{B' \subsetneq B | C - C' | = 1} \{\gamma(B', C') + h(B - B', C - C')\}, \tag{5}$$

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

where $\gamma(B, C)$ is the minimum space used when packing all the items described by set B into bins described by C, $\gamma(0, 0) = 0$ gives the boundary for the recursive formulation, and $h(B - B', C - C')$ is the bin space for packing all the items in set $B - B'$ into one single bin represented by $C - C'$. As $B - B'$ and $C - C'$ form a single packing pattern, there is only one bin that can be allocated to pack the items presented by $B - B'$ in function h. Using this formulation, we can decompose each packing pattern step by step by searching the transition state. The recursive procedure stops when all the items are packed. There may be many final states where all the items are packed. We are only interested in the one leading to the minimum cost, i.e. $\min_{C \le C_0} \gamma(B_0, C)$. The corresponding optimal solution can be found in a backward manner.

In the previous formulation, one must consider all possible states (B, C). The number of possible states (B, C) is $2^n \times \prod_{j=1}^{N}(M_j + 1)$. In other words, the number of states increases exponentially with n and N, and the problem becomes intractable. To reduce the computational complexity, we amalgamate all the possible states (B, C) with the same numbers of items and bins, which are equal to α and β, respectively, into one single state expressed by (α, β). Of course, in this way some information will be lost and hence the solution is no longer optimal. The previous formulation is reformulated as

$$\gamma'(\alpha, \beta) = \min_{\alpha' < \alpha}\{\gamma'(\alpha', \beta - 1) + h'(\alpha, \beta, \alpha')\}. \tag{6}$$

In this formulation, each state (B, C) is replaced by (α, β). α and β respectively denote the number of items and the number of bins. Therefore, as in the previous formulation, $\gamma'(\alpha, \beta)$ is the minimum space used when packing α items into β bins. Similar to $(B - B', C - C')$, (α, β, α') aggregates all the single packing patterns with the same number of items: $\alpha - \alpha'$. Therefore, $h'(\alpha, \beta, \alpha')$ is the minimum space for packing $\alpha - \alpha'$ items into one bin knowing that α' items are already packed into $\beta - 1$ bins with the 'least' cost.

If $h'(\alpha, \beta, \alpha')$ is known, we can use the recursive formulation to obtain a near-optimal solution. From the state $(\alpha', \beta - 1)$ to state (α, β), function h' is the minimum space for packing $\alpha - \alpha'$ items into one bin. The problem is to decide which items and which type of bin to select to realise the minimum space. As $\gamma'(\alpha', \beta - 1)$ already corresponds to the items and bins selected for this state, to reach state (α, β), repeated items and bins should be avoided for h'.

Let $U(\alpha', \beta - 1)$ and $V(\alpha', \beta - 1)$ respectively be the item set and bin set that represent the items and bins selected in state $(\alpha', \beta - 1)$, and $\bar{U}(\alpha', \beta - 1)$ and $\bar{V}(\alpha', \beta - 1)$ are their complementary sets, respectively:

$$\bar{U}(\alpha', \beta - 1) + U(\alpha', \beta - 1) = B_0, \tag{7}$$

$$\bar{V}(\alpha', \beta - 1) + V(\alpha', \beta - 1) = C_0. \tag{8}$$

To obtain $h'(\alpha, \beta, \alpha')$, the problem is to select $\alpha - \alpha'$ items from the set $\bar{U}(\alpha', \beta - 1)$ and one bin from the bin set $\bar{V}(\alpha', \beta - 1)$ to minimise the bin space used. Let α^* be the value of α' giving the minimum value in formula (6). $R(\alpha, \beta, \alpha^*)$ and $S(\alpha, \beta, \alpha^*)$ are the item set and bin set, respectively, consisting of all the items packed and the bin used corresponding to $h'(\alpha, \beta, \alpha^*)$. For each iteration, we have

$$U(\alpha, \beta) = U(\alpha^*, \beta - 1) + R(\alpha, \beta, \alpha^*), \tag{9}$$

$$V(\alpha, \beta) = V(\alpha^*, \beta - 1) + S(\alpha, \beta, \alpha^*). \tag{10}$$

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

To avoid becoming trapped in a local optimum, it is useful to select a secondary objective function. It is used to guide the local search to promising regions where better solutions are likely to be found. It always appears in approximation methods, such as tabu search, genetic algorithms and simulated annealing, to improve the primary objective function indirectly. The need to select a secondary objective function, and which function to choose, highly depend on the characteristics of each problem. In this variable sized two-dimensional bin packing problem, in several early iterations, the smallest bins are easier to select with few items of small size, then in the last iterations, large sized items should be packed into more bins. Logically, to save space, large sized items should be combined with small sized items in a packing pattern. Therefore, we choose a secondary objective function. It is conceived that good packing patterns should fully exploit empty bin space. However, in practice, each packing pattern has some empty space. Therefore, the occupation rate (OR) is an important property and is calculated as

$$OR = \frac{S_{\text{item}}}{S_{\text{bin}}}, \tag{11}$$

where S_{item} is the total area of packed items and S_{bin} is the total bin space used. In an ideal situation, OR is equal to 1, and, in this case, the total space of bins used is minimum. Therefore, the objective of maximising OR leads us to minimise the total bin space, which is expected by the primary objective function. Therefore, the following mathematical formulation holds:

$$\phi(\alpha,\beta) = \max_{\alpha'<\alpha}\left\{\frac{\phi(\alpha',\beta-1) * V(\alpha',\beta-1)^T * Vol + \mu(\alpha,\beta,\alpha') * S(\alpha,\beta,\alpha')^T * Vol}{V(\alpha',\beta-1)^T * Vol + S(\alpha,\beta,\alpha')^T * Vol}\right\}, \tag{12}$$

where $\phi(\alpha,\beta)$ is the maximum occupation rate when packing α items into β bins. We replace $h'(\alpha,\beta,\alpha')$ by $\mu(\alpha,\beta,\alpha')$ in the formula. Here, $U(\alpha',\beta-1)$ and $V(\alpha',\beta-1)$ are the sets of items packed and bins used corresponding to the state $\phi(\alpha',\beta-1)$, and $R(\alpha,\beta,\alpha')$ and $S(\alpha,\beta,\alpha')$ are those corresponding to $\mu(\alpha,\beta,\alpha')$. Vol is the vector denoting the volume of the bins. It is an N-vector with the ith element equaling the area of the bin of type i. T is the transposition operation. Function $\mu(\alpha,\beta,\alpha')$ is defined as the maximum occupation rate for a single pattern, where $\alpha-\alpha'$ items from the set $\bar{U}(\alpha',\beta-1)$ and one bin from the set $\bar{V}(\alpha',\beta-1)$ are selected. Details for solving subproblem $\mu(\alpha,\beta,\alpha')$ will be given in the next section.

4. Solving the subproblem

4.1 *Priority item*

Before solving the subproblem, we will first describe the notion of a priority item.

Definition 4.1: (*priority item*) An item is called a priority item if it can fit into only one type of bin because of its geometrical properties.

For example, we have three bin types 3×6, 5×4 and 6×6 to choose from. Items of size 5×6 can only be packed into a bin with size 6×6. This item is considered to be a priority item for bin 6×6.

As our objective is to pack all the items into bins, in the subproblem the priority item should be packed first. Otherwise, it will be much more difficult to be packed as the

```
Procedure Identify priority item
For each item i in set Ū(α′,β − 1) DO
    If entry i in set Ū(α′,β − 1) equals 1 Then
        count:=0.
        For each bin type j in set V̄(α′,β − 1).
            IF entry j in the above set is positive, w_i ≤ W_j, h_i ≤ H_j
                count:=count+1, k := j;
            End If
        End For
        If count=1 Then
            i is the priority item for bin k.
        End If
    End If
End For
End
```

Figure 1. The process of identifying a priority item.

number of bins available is limited. In the end, it is possible that no feasible solution can be found. The process of identifying priority items is detailed in Figure 1.

We enumerate each item i in set $U(\alpha', \beta - 1)$ with entries equalling 1. Then i is tested by packing it into each bin whose entry is positive in set $\bar{V}(\alpha', \beta - 1)$. 'Count' is used to record the number of different types of bins fitting the item size. 'k' is used to record the bin type. If the size of item i only fits a bin of type k, then item i is defined as the priority item for bin k. We denote by λ_k the number of priority items for bin k.

4.2 *Subproblem for H1*

For $\mu(\alpha, \beta, \alpha')$, the problem is to select $\alpha - \alpha'$ items and one bin to maximise the occupation rate. Three scenarios are possible in this case and are presented in Appendix A.

4.3 *Subproblem for H2*

Sometimes, cutting and packing enterprises need to consider a variety of demands proposed by different customers, such as guillotine cutting, non-guillotine cutting, fixed orientation of items and rotation of items. In algorithm H1, we only consider the non-guillotine cutting and fixed orientation case. In this subsection, we describe another method (H2) that considers additional distinct demands such as guillotine cutting and the possibility of rotation. For enterprises, instead of several methods that are specially designed to deal with a few additional demands, with this method a variety of customer demands can be considered at the same time. The general procedure of H2 is the same as for H1. Only the subproblem is solved differently. We will also consider three cases in the subproblem. These cases are presented in Appendix B.

5. Evaluation

We have implemented two heuristic algorithms, H1 and H2. To evaluate the performance of these two algorithms, we test them on instances reported in the literature. All tests were carried out using a personal computer with an Intel Pentium 933 processor and 1 GB of memory.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

5.1 *Comparison with lower bound*

To our knowledge, experimental results for the variable sized bin packing problem have seldom been reported, with the only result we know of being proposed by Pisinger and Sigurd (2005). We test our algorithms on instances provided by Pisinger and Sigurd (2005). These instances are based on 10 classes of the identical sized bin packing problem summarised by Berkey and Wang (1987) and Martello and Vigo (1998). Let w_j and h_j be the width and height of the items, and, for each class, the distribution of items is as follows:

Class 1: w_j and h_j uniformly random in $[1, 10]$, $W = H = 10$;
Class 2: w_j and h_j uniformly random in $[1, 10]$, $W = H = 30$;
Class 3: w_j and h_j uniformly random in $[1, 35]$, $W = H = 40$;
Class 4: w_j and h_j uniformly random in $[1, 35]$, $W = H = 100$;
Class 5: w_j and h_j uniformly random in $[1, 100]$, $W = H = 100$;
Class 6: w_j and h_j uniformly random in $[1, 100]$, $W = H = 300$;

For Classes 7, 8, 9 and 10, a more realistic situation is considered. The items are classified into four types, where $W = H = 100$:

Type 1: w_j uniformly random in $[\frac{2}{3}W, W]$, h_j uniformly random in $[1, \frac{1}{2}H]$;
Type 2: w_j uniformly random in $[1, \frac{1}{2}W]$, h_j uniformly random in $[\frac{2}{3}H, H]$;
Type 3: w_j uniformly random in $[\frac{1}{2}W, W]$, h_j uniformly random in $[\frac{1}{2}H, H]$;
Type 4: w_j uniformly random in $[1, \frac{1}{2}W]$, h_j uniformly random in $[1, \frac{1}{2}H]$.

Class 7: type 1 with probability 70%, types 2, 3 and 4 with probability 10% each;
Class 8: type 2 with probability 70%, types 1, 3 and 4 with probability 10% each;
Class 9: type 3 with probability 70%, types 1, 2 and 4 with probability 10% each;
Class 10: type 4 with probability 70%, types 1, 2 and 3 with probability 10% each.

In the 10 class instances for the identical sized problem, bin width W and height H can be different. In the literature, for variable sized instances, the number of bin types is limited to five and the sizes are drawn at random with a uniform distribution from $[W/2, W] \times [H/2, H]$. To ensure feasibility of the problem, each item must be packed into at least one bin generated, and we force the fifth bin type to have width W and height H. As the number of bins available in each type is unlimited in the literature, without loss of generality we assume in our dynamic programming formulation that the number of bins available in each type is equal to the quantity of items.

Since the solutions obtained by Pisinger and Sigurd (2005) are not reported, we cannot compare our results with theirs. Therefore, we generate a lower bound using the column generation procedure and compare our results with the lower bound. Details of the column generation procedure are not reported here. To solve the pricing problem, we generate columns heuristically. If the heuristic method fails to find a column with negative reduced cost, an exact method is applied. We refer to the branch-and-bound method of Fekete and Schepers (1997) to solve the pricing problem exactly. CPLEX 11.1 was used to solve the linear programming problem resulting from the column generation.

Tables 1 and 2 show the results for our two heuristic algorithms. Each row in the table is an average over 10 instances. Columns 3 and 4 show the average CPU time and the number of columns generated to obtain the lower bound. Column 5 gives the average gap between H1 and the lower bound. Column 6 gives the average gap between H2 and the lower bound. For classes 2, 4, 6 and 10, as the items are too small compared with the size

Table 1. Average gap comparison with lower bounds for Berkey and Wang's instances.

Class	n	Lower bound		H1	H2
		Time (s)	Avg. cols	Gap (%)	Gap (%)
1	20	80.33	286	4.57	14.43
	40	39.56	525	5.93	22.32
	60	20.92	645	6.27	17.92
	80	76.69	925	5.65	22.11
	100	294.96	994	4.52	17.74
2	20	371.50	560	29.88	29.88
	40	859.13	1295	10.76	10.76
	60				
	80				
	100				
3	20	26.46	339	11.00	14.19
	40	12.55	546	9.86	12.11
	60	101.00	881	7.87	12.31
	80	204.93	1470	11.23	15.35
	100	654.51	1881	10.65	17.69
4	20	257.58	621	17.42	17.42
	40	4215.73	1098	18.83	18.83
	60				
	80				
	100				
5	20	1.33	282	7.22	18.91
	40	54.79	598	6.97	18.65
	60	182.87	936	7.83	17.64
	80	254.75	1135	7.31	14.63
	100	1387.21	1535	7.96	17.01
6	20	0.16	5	0.00	0.00
	40	0.19	7	7.90	0.00
	60				
	80				
	100				

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

of the bins, the pricing algorithm takes a lot of time to consider auxiliary problems. Some instances in these classes fail to obtain a lower bound in 3600 seconds.

Table 3 summarises the results grouped by instance classes. The second column lists the number of instances solved to optimality by Pisinger and Sigurd (2005) in 3600 seconds. With our two heuristic algorithms, all instances are solved in a few seconds compared with their 3600 seconds. It should be noted in this comparison that we consider only the general bin packing problem, and no additional customer demands are involved for both H1 and H2. In most cases, H1 performs better than H2 (see columns Gap1 and Gap2 in Table 3). As the subproblem in H1 is solved by listing items in decreasing order of their area and heuristically packing them into the bin one by one, the subproblem of H1 will naturally lead to a better solution than the subproblem of H2.

5.2 *Comparison for identical bin size*

Due to the absence of comparable algorithms, we consider a special case in which bins are identical and test our algorithms on the instances of Berkey and Wang (1987) and Martello

Table 2. Average gap comparison with lower bounds for Martello and Vigo's instances.

Class	n	Lower bound		H1	H2
		Time (s)	Avg. cols	Gap (%)	Gap (%)
7	20	1.78	250	7.07	25.69
	40	72.22	525	12.95	26.50
	60	334.47	768	14.20	20.79
	80	503.23	1011	6.84	7.08
	100	821.51	1141	7.44	5.38
8	20	2.71	240	6.36	16.85
	40	64.17	442	7.05	19.01
	60	409.39	639	8.92	11.91
	80	625.99	924	9.75	16.50
	100	731.48	1064	8.84	11.58
9	20	0.27	139	0.02	1.33
	40	1.12	306	1.07	2.80
	60	3.39	401	0.78	3.86
	80	5.33	401	1.51	3.55
	100	6.59	745	0.60	2.88
10	20	41.49	461	7.55	12.77
	40	100.53	420	3.42	6.71
	60	1747.92	1428	8.89	16.52
	80	1260.95	802	2.57	3.50
	100				

Table 3. Summary results for each class comparison with lower bounds.

Class	Pisinger, Sigurd		H1			H2		
	No. solved	Time (s)	Gap1 (%)	Time (s)	No. solved	Gap2 (%)	Time (s)	No. solved
1	24	2083.8	5.39	11.25	50	18.90	9.28	50
2	4	3324.4	20.32	29.39	50	20.32	30.39	50
3	22	2244.2	10.12	12.53	50	14.33	11.46	50
4	3	3394.6	18.13	29.46	50	18.13	29.50	50
5	24	1996.2	7.46	9.00	50	17.37	7.46	50
6	3	3385.0	3.95	169.76	50	0.00	4.85	50
7	10	2957.0	9.70	1.93	50	17.09	2.50	50
8	15	2784.0	8.18	11.64	50	15.17	11.51	50
9	50	125.4	0.80	2.02	50	2.88	1.93	50
10	9	2969.4	5.61	23.59	50	9.88	23.46	50
Av.	16	2526.4	8.97	30.06	50	13.41	13.23	50

and Vigo (1998) for the identical sized bin packing problem. We compare solutions provided by H1 and H2 with the optimal solution from the literature. The optimal solution can be found at http://www.or.deis.unibo.it/research_pages/ORinstances/soluz_2dbp.htm.

The test instances consist of 10 classes and, for each class, five scenarios are considered when the number of items equals 20, 40, 60, 80 and 100. The gap between the results

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

Table 4. Comparison with the optimal solution for identical sized bin packing.

Class	20		40		60		80		100		Av.	
	H1 (%)	H2 (%)	H1 (%)	H2 (%)	H1 (%)	H2 (%)	H1 (%)	H2 (%)	H1 (%)	H2 (%)	H1 (%)	H2 (%)
1	2.78	3.33	1.82	3.20	1.14	1.62	0.36	1.08	2.00	2.31	1.62	2.31
2	20.00	20.00	10.00	10.00	10.00	20.00	6.67	10.00	3.33	8.33	10.00	13.67
3	7.83	10.33	4.94	6.84	3.97	3.73	2.68	4.71	3.72	4.99	4.63	6.12
4	10.00	10.00	10.00	10.00	15.00	20.00	10.00	6.67	9.17	11.87	10.83	11.71
5	4.00	5.43	4.25	5.25	3.03	3.03	0.84	1.24	2.27	3.36	2.88	3.66
6	0.00	0.00	20.00	20.00	20.00	20.00	0.00	3.33	10.00	10.00	10.00	10.67
7	8.67	8.67	4.62	4.62	2.61	3.19	4.17	4.17	1.45	1.82	4.30	4.49
8	4.00	6.50	3.62	5.37	2.68	3.83	1.80	2.26	0.66	1.81	2.55	3.95
9	0.71	0.71	1.09	1.09	0.66	0.66	0.18	0.18			0.66	0.66
10	7.83	4.50	3.10	4.52	3.99	6.24	5.23	5.23	5.33	6.21	5.10	5.34

obtained by H1, H2 and the optimal solution is reported in Table 4 for each scenario. For class 9, with 100 items, the optimal solution is not reported in the literature, therefore we do not report a gap. For H1, the average gap for each class is from 0.66 to 10.83%, while for H2 the average gap for each class is from 0.66 to 13.67%.

5.3 *Considering additional customer demands*

In the previous subsections we have compared the solution provided by our algorithms with lower bounds, and we have also compared our results with optimal solutions for identical bins. The results show the good performance of these algorithms, the average gaps being 8.97% and 13.4%, respectively, for H1 and H2 compared with lower bounds, and 5.26% and 6.26% compared with optimal solutions for identical bins. Moreover, H2 is flexible, as it can consider a variety of additional customer demands. We add the guillotine packing style and the possibility of rotation as two additional demands proposed by customers. H2 is tested on instances proposed by Pisinger and Sigurd (2005) considering these two additional demands. In Table 5, for each scenario, the left column is the gap between the results obtained by H2 considering the guillotine packing style and the results obtained by H1 without this demand. The right column for each scenario reports the gap between the results obtained by H1 with fixed orientation of items and the results obtained by H2 with the possibility of rotation. The first row in this table presents the number of items considered in each instance.

5.4 *Influence of number of bin types*

To evaluate the influence of the number of bin types, we compare the difference when the number of bin types is one and five. H1 is tested on instances proposed by Berkey and Wang (1987) and Martello and Vigo (1998) when the number of bin types is one, and it is also tested on instances proposed by Pisinger and Sigurd (2005). Table 6 reports the gap between those two results for each scenario. We find that 6.67% of space is saved, on

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

Table 5. Gaps between guillotine demand/non-guillotine demand, and fixed orientation demand/non-fixed orientation demand.

Class	20		40		60		80		100	
	G (%)	R (%)	G (%)	R (%)	G (%)	R (%)	G (%)	R (%)	G (%)	R (%)
1	10.38	8.48	17.34	6.55	12.01	6.31	16.81	6.34	12.08	6.83
2	0.73	10.36	0.15	11.67	2.03	9.02	3.05	7.22	1.10	11.21
3	2.72	5.87	4.73	6.06	7.40	1.72	6.94	3.21	9.03	1.11
4	1.03	4.89	1.30	6.98	2.17	7.88	0.77	8.87	0.43	8.65
5	12.30	7.64	11.48	5.97	9.89	3.48	7.79	3.76	9.34	3.88
6	0.00	35.84	0.73	34.41	3.31	18.50	2.61	14.75	0.56	15.37
7	26.51	1.57	12.58	7.83	4.55	11.85	0.50	8.26	1.42	9.35
8	11.70	4.04	11.83	7.26	3.58	8.31	7.12	8.52	2.58	9.58
9	1.43	1.56	1.72	1.58	3.11	0.41	1.19	1.55	2.43	0.00
10	5.85	3.26	6.42	4.09	7.11	6.44	2.50	5.57	6.06	5.78
Av.	7.27	8.35	6.83	9.24	5.52	7.39	4.93	6.81	4.50	7.18

Table 6. Gaps between results with variable sized bins and identical sized bins.

Class	20 (%)	40 (%)	60 (%)	80 (%)	100 (%)	Av. (%)
1	11.54	7.12	3.56	4.00	2.04	5.65
2	22.12	25.32	15.74	6.29	7.59	15.41
3	7.59	1.93	1.39	1.77	1.61	2.86
4	8.69	21.73	15.07	7.52	6.92	11.99
5	10.98	5.77	2.53	1.78	1.63	4.54
6	0.00	0.79	3.07	3.70	1.69	1.85
7	12.24	2.17	1.89	2.48	1.04	3.96
8	8.34	3.13	1.10	3.13	3.56	3.85
9	13.31	13.30	13.57	12.05	12.63	12.97
10	11.02	3.29	1.25	1.39	0.88	3.56

average, when the number of bin types is five compared with the bin packing problem with identical bins.

6. Conclusion

The variable sized two-dimensional bin packing problem has a wide application in the wood, paper and steel industries where raw materials to be cut or containers to be placed are of different size. This paper proposes two heuristics, H1 and H2, to solve this problem. Computational results show that the average gaps are 8.97% and 13.4%, respectively, for H1 and H2 compared with the lower bounds, and 5.26% and 6.26% compared with the optimal solutions for identical bins. Both heuristics obtain good quality solutions in a few seconds compared with the existing algorithms, which consume a lot of time and memory. Moreover, customers can propose a variety of additional demands in the production process such as a guillotine or non-guillotine cutting style, fixed orientation of items or possible rotation of items. The H2 algorithm can easily incorporate some of these

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

demands. The principle of these two algorithms can easily be employed to solve other combinatorial problems such as set partitioning problems (Liu 2009).

References

Ben-Messaoud, S., 2004. *Caractérisation, modélisation et algorithmes pour des problèmes de découpe guillotine*. Thesis (PhD). University of Technology of Troyes.

Berkey, J.O. and Wang, P.Y., 1987. Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38 (5), 423–429.

Cheng, C.H. and Feiring, B.R., 1994. The cutting stock problem – A survey. *International Journal of Production Economics*, 36 (3), 291–305.

Chu, C. and La, R., 1994. Variable-sized bin packing: Tight absolute worst-case performance ratios for four approximation algorithms. *SIAM Journal of Computing*, 30 (6), 2069–2083.

Csirik, J., 1989. An on-line algorithm for variable-sized bin packing. *Acta Informatica*, 26 (8), 697–708.

Fekete, S.P. and Schepers, J., 1997. *On more-dimensional packing 3: Exact algorithms*. Technical paper.

Friesen, D.K. and Langston, M.A., 1986. Variale sized bin packing. *SIAM Journal on Computing*, 15 (1), 222–230.

Kang, J. and Park, S., 2003. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147 (2), 365–372.

Kinnersley, N.G. and Langston, M.A., 1988. Online variable sized bin packing. *Discrete Applied Mathematics*, 22 (2), 143–148.

Liu, Y., 2009. *Methods to solve set partitioning problems and applications in packing and operating room scheduling*. Thesis (PhD). University of Technology of Troyes.

Lodi, A., Martello, S., and Monaci, M., 2002. Two-dimensional packing problem: A survey. *European Journal of Operation Research*, 141 (2), 241–252.

Martello, S. and Vigo, D., 1998. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44 (3), 388–399.

Murgolo, F.D., 1987. An efficient approximation scheme for variable sized bin packing. *SIAM Journal on Computing*, 16 (1), 149–161.

Pisinger, D. and Sigurd, M., 2005. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2 (2), 154–167.

Seiden, S.S., 2002. On the online bin packing problem. *Journal of the ACM*, 49 (5), 640–671.

Valerio de Carvalho, J.M., 2002. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141 (2), 253–273.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

Appendix A. Subproblem for H1

In this appendix, we show how the function $\mu(\alpha, \beta, \alpha')$ is computed for algorithm H1. Let λ_k be the number of priority items for bin k. Three cases are possible.

Case 1: $0 < \lambda_k < \alpha - \alpha'$. In this case, there are priority items. As there exist λ_k priority items for bin k, a bin of type k is selected for $\mu(\alpha, \beta, \alpha')$. $\alpha - \alpha'$ is greater than λ_k, therefore we first pack λ_k priority items into bin k. Let $\bar{U}_1(\alpha', \beta - 1)$ be the set of all priority items. For $\mu(\alpha, \beta, \alpha')$, the remaining $\alpha - \alpha' - \lambda_k$ items must be chosen from set $\bar{U}(\alpha', \beta - 1) \backslash \mathrm{U}_1(\alpha', \beta - 1)$ and packed into a bin of type k to maximise the occupation rate.

In this two-stage packing procedure, we will use two notions introduced by Ben-Messaoud (2004): 'possible point' and 'possible rectangle'. A 'possible point' denotes a possible packing position for the next item, and corresponds to the top-left or bottom-right corner of a placed item.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

It is expressed as an x coordinate and a y coordinate. A 'possible rectangle' is an empty rectangle with its bottom-left corner corresponding to a possible point. Its width and height are the maximum width and maximum height, respectively, for the next item corresponding to the possible point. It is denoted by the possible point and its width and height. In the packing procedure, we construct two lists: an item list and a possible rectangle list. Possible rectangles are listed in increasing order of the y coordinates of their corresponding possible points. Initially, the bottom-left corner of the bin is the only packing position for the items, therefore in the rectangle list, there is only one possible rectangle. The packing procedure repeats as follows.

(i) From the head of the rectangle list, try to pack the current item into the first possible rectangle in the list that is suitable to receive it.
(ii) Remove this possible rectangle and the item from the corresponding lists.
(iii) Generate two new possible rectangles. The procedure of generating two new possible rectangles is shown in Figure A1. The newly generated possible rectangles are R_1 and R_2. Possible points correspond to the top-left and bottom-right corner of the item placed. In Figure A1, i is the item packed in rectangle $R(x, y, \bar{w}, \bar{h})$. After packing item i, two new possible points are $(x, y+h_i)$ and $(x+w_i, y)$. For the next item packed at $(x, y+h_i)$, its width cannot be more than $\bar{w}$ and the height limit is $\bar{h}-h_i$. Therefore, the possible rectangle R_1 is expressed by $(x, y+h_i, \bar{w}, \bar{h}-h_i)$. The other possible rectangle R_2 is $(x+w_i, y, \bar{w}-w_i, \bar{h})$.
(iv) Insert the newly created possible rectangles into the list at an appropriate position.
(v) Update the rectangle list.

Initially, we initialise the single possible rectangle as $R=(0, 0, W_k, H_k)$, which corresponds to an empty bin of type k with its possible point $(0, 0)$. We first pack the priority items one by one into bin k following the packing procedure described above. Each time, we try to place the current item into the first suitable possible rectangle in the list. When we decide to pack the item into a rectangle, this rectangle is replaced by two new possible rectangles where the next items can be packed. These two rectangles are inserted in the rectangle list. Finally, certain possible rectangles in the list need to be updated. This procedure is repeated until all the priority items are packed. If in the packing procedure for one priority item there is no suitable possible rectangle to receive the item, we consider that it is not possible to pack λ_k priority items into bin k, or to pack $\alpha-\alpha'$ items. We have $\mu(\alpha, \beta, \alpha')=-\infty$. When all the priority items are packed into bin k, there exists a list of possible rectangles. The second stage selects $\alpha-\alpha'-\lambda_k$ items from set $\bar{U}(\alpha', \beta-1)\backslash\bar{U}_1(\alpha', \beta-1)$. All the items

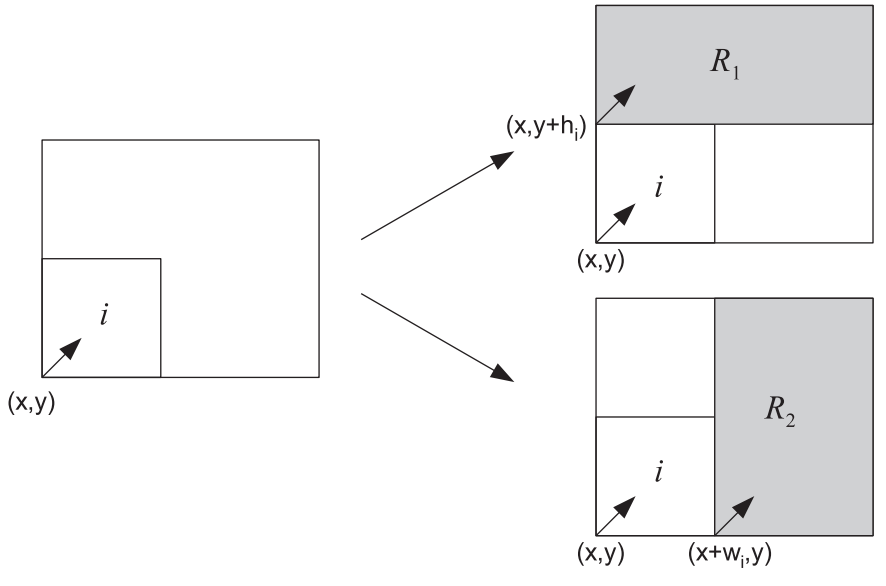

Figure A1. Generating new possible rectangles.

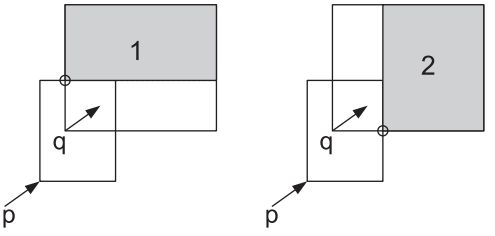

Figure A2. Updating strategy 1.

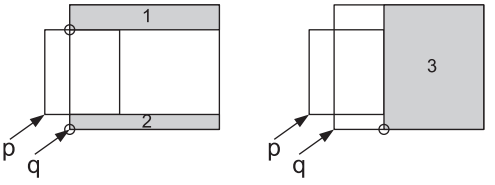

Figure A3. Updating strategy 2.

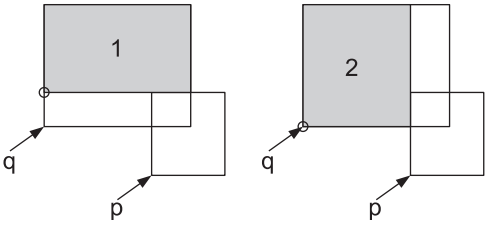

Figure A4. Updating strategy 3.

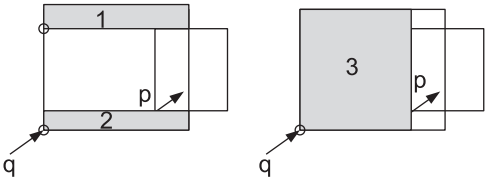

Figure A5. Updating strategy 4.

in set $\bar{U}(\alpha', \beta-1)\backslash\bar{U}_1(\alpha', \beta-1)$ are listed in decreasing order of their area. The second procedure is to pack these items one by one until $\alpha-\alpha'-\lambda$ items are selected or no additional item can be packed into the bin. It begins with the rectangle list that remains after the first-stage procedure. The details of the packing procedure also follow the procedure described above.

The fifth step of the packing procedure is the rectangle list updating strategy. Let p be the item packed at (x_p, y_p) with width w_p and height h_p, and q be a possible rectangle overlapping with p, at position (x_q, y_q) with width $\bar{w}_q$ and height $\bar{h}_q$. The updating rule for the possible rectangle is considered in four different cases and shown in Figures A2–A5.

(i) $x_q \geq x_p$ and $y_q > y_p$. As shown in Figure A2, the possible rectangle q is replaced by two possible rectangles 1 and 2, and the two intersection points between p and q become the possible packing positions of the new possible rectangles 1 and 2, respectively.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

(ii) $x_q \geq x_p$ and $y_q \leq y_p$. The rectangle q is replaced by three new possible rectangles 1, 2 and 3. The intersection point is a possible packing position for rectangles 1 (see Figure A3(left)). For rectangle 3, the possible point is $(x_p + w_p, y_q)$.

(iii) $x_q < x_p$ and $y_q \geq y_p$. Rectangle q is deleted and two new possible rectangles 1 and 2 are created (see Figure A4). They have possible points $(x_q, y_p + h_p)$ and (x_q, y_q), respectively.

(iv) $x_q < x_p$ and $y_q < y_p$. Rectangle q is replaced by three possible rectangles 1, 2 and 3 (see Figure A5). Their possible points are $(x_q, y_p + h_p)$, (x_q, y_q) and (x_q, y_q), respectively. Rectangles 2 and 3 have the same possible points, but they are different in width and height.

Case 2: $\lambda_k \geq \alpha - \alpha'$. In this case, a bin of type k is selected for $\mu(\alpha, \beta, \alpha')$, and we need to select $\alpha - \alpha'$ items from λ_k priority items to maximise the occupation rate. We list all the priority items in decreasing order of their area and pack them one after another in the list until $\alpha - \alpha'$ items are packed. If no more items can be packed while the number of items packed is less than $\alpha - \alpha'$, we set $\mu(\alpha, \beta, \alpha') = -\infty$ without loss of generality. The one-by-one packing process is the same as in Case 1.

Case 3: $\lambda_k = 0$. As no priority item exists, we cannot decide which type of bin to select. Therefore, we enumerate each bin type with a positive entry in set $\bar{V}(\alpha', \beta - 1)$. Let $\omega(f, \alpha, \beta, \alpha')$ be the maximum occupation rate by selecting $\alpha - \alpha'$ items in set $\bar{U}(\alpha', \beta - 1)$ and packing them into a bin of type f. We have $\mu(\alpha, \beta, \alpha') = \max_f \omega(f, \alpha, \beta, \alpha')$. Let ε be the number of items contained in set $\bar{U}(\alpha', \beta - 1)$. Our objective is to select $\alpha - \alpha'$ items out of ε and pack them into a bin of type f to realise the maximum occupation rate.

- $\alpha - \alpha' > \varepsilon$. It is impossible to select $\alpha - \alpha'$ items from ε items. $\omega(f, \alpha, \beta, \alpha') = -\infty$.
- $\alpha - \alpha' \leq \varepsilon$. We list all the items to be packed in set $\bar{U}(\alpha', \beta - 1)$ in increasing order of their area. We select the first $\alpha - \alpha'$ items in the list and decide whether these items can be packed into the bin or not. If these initial items cannot be packed into the bin, we assume that if a set of the smallest items cannot be packed, neither can the bigger ones, so $\omega(f, \alpha, \beta, \alpha') = -\infty$. Otherwise, we obtain an initial occupation rate and then an exchange process is launched to improve this value by modifying the selected items. The details of this process are demonstrated in Figure A6. In Figure A7, ε items indexed from 1 to ε are listed. Initially, i points to the head of the list with index 1 and j points to the end of the list with index ε. The initially selected items are indexed from 1 to $\alpha - \alpha'$ in the list. 'i and j can change' means that we try to replace the item pointed by i with the item pointed by j in the list to update the selected items. If these items can be packed into the bin and the occupation rate is improved, the exchange becomes definitive, and i moves to its next

```
Procedure Exchange
While (i ≤ α−α′ and j > α−α′ ) DO
    If (i and j can change) Then
        Update selected items
        i := i + 1;
        j := j − 1;
    Else
        j := j − 1;
End
```

Figure A6. Algorithm of the exchange procedure.

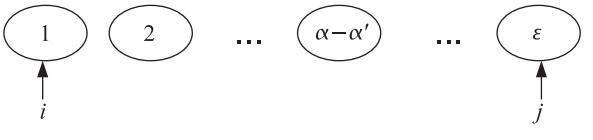

Figure A7. Exchange procedure.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

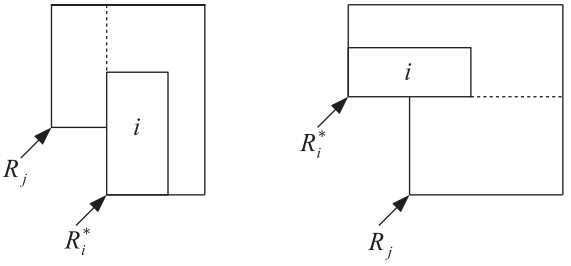

Figure A8. Updating strategy 5.

position with index $i+1$, and j moves to its previous position in the list with index $j-1$. If the exchange fails, j moves to its previous position in the list and we continue to update the selected items. The exchange procedure stops when $i > \alpha - \alpha'$ or $j < \alpha - \alpha'$.

In the exchange process, the core problem is to decide whether a given set of items can be packed into the bin or not. We list the given set of items in decreasing order of their height and pack them into the bin one by one following the packing procedure described above. If items fail to be packed into one bin with this heuristic, we assume that no feasible pattern exists to pack all the given items into the bin. Ben-Messaoud (2004) proposed an updating rule for the fifth step of the packing procedure. This updating rule is based on the items being listed in order of decreasing height. It is expressed as follows. Let the possible rectangle overlapping with item i be $R_j(x_j, y_j, \bar{w}_j, \bar{h}_j)$, and the first suitable rectangle to receive item i is expressed by $R_i^*(x_i^*, y_i^*, \bar{w}_i^*, \bar{h}_i^*)$.

- If $x_j \le x_i^*$: $R_j = (x_j, y_j, x_i^* - x_j, \bar{h}_j)$. Figure A8(left) shows this case. For possible rectangle R_j, as it overlaps with item i, (x_j, y_j) is still the possible packing position for the next items, but the maximum width of the next items should be changed. The next item with width greater than $x_i^* - x_j$ cannot be packed at this position. Therefore, the width of this possible rectangle is reduced to $x_i^* - x_j$.
- If $x_j > x_i^*$: $R_j = (x_j, y_j, \bar{w}_j, y_i^* - y_j)$. In Figure A8(right), the next item with height greater than $y_i^* - y_j$ cannot be packed at this position. The height is reduced to $y_i^* - y_j$. We follow the same updating rule, and repeat the five steps of the packing procedure. If there is no possible rectangle to pack any item in the list, $\mu(\alpha, \beta, \alpha') = -\infty$.

Appendix B. Subproblem for H2

In this appendix, we show how the function $\mu(\alpha, \beta, \alpha')$ is computed for algorithm H2. Three possible cases are considered.

Case 1: $0 < \lambda_k < \alpha - \alpha'$. In this case, as there are λ_k priority items for bin k, a bin of type k is selected for $\mu(\alpha, \beta, \alpha')$. Then $\mu(\alpha, \beta, \alpha')$ is solved by selecting and packing $\alpha - \alpha' - \lambda_k$ non-priority items with the λ_k priority items to realise the maximum occupation rate. We list all non-priority items in increasing order of their area. We select the first $\alpha - \alpha' - \lambda_k$ items in the list, and these items with λ_k priority items form an initial solution. If these items can be packed into bin k, we obtain an initial occupation rate. Otherwise, we assume that since the $\alpha - \alpha' - \lambda_k$ smallest non-priority items cannot be packed into bin k, neither can the larger ones. Therefore, we have $\mu(\alpha, \beta, \alpha') = -\infty$. If a finite initial occupation rate is obtained, we can use the exchange procedure to improve this occupation rate by permuting the initially selected $\alpha - \alpha' - \lambda_k$ non-priority items. The details of this changing procedure are the same as in Appendix A. The core of this exchange procedure is judging whether or not a given set of items can be packed into the bin. Here, it includes both non-priority items and priority items. We try to pack them heuristically following the packing procedure. In this packing procedure, items are listed in decreasing order of their height. This packing procedure may vary according to the distinct customer demands involved in the problem. For rotation or fixed orientation of items, we need to change 'the first suitable rectangle' condition in Step 1. For guillotine cutting or free cutting, the updating procedure is different in Step 5. For rotation

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

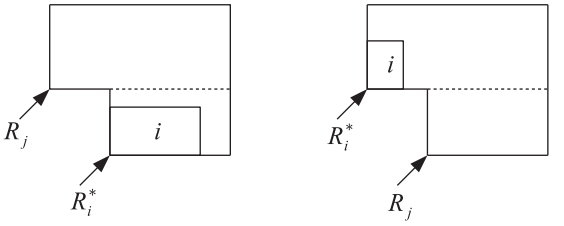

Figure B1. Updating strategy 6.

Downloaded By: [2007-2008 Nanyang Technological University] At: 16:59 26 September 2010

demands, items can be packed in a fixed orientation or rotated by 90°. Beginning with the first possible rectangle in the rectangle list, we examine 'the first suitable rectangle'. A rectangle is considered to be suitable if its width and height are both at least as large as those of the item as it is or after rotation. In the same way for the fixed orientation constraint, if the item's width and height are respectively less than or equal to the rectangle's width and height, we consider this rectangle as 'the first suitable rectangle'. For the non-guillotine constraint, each possible rectangle in the rectangle list overlapping with the current packed item needs to be updated. This updating consists of changing the width or height of the possible rectangles. We examine each possible rectangle in the rectangle list. Let $R_j(x_j, y_j, \bar{w}_j, \bar{h}_j)$ denote the possible rectangle in the list overlapping with the item packed, and the item packed is at position (x_i^*, y_i^*). This updating rule is the same as described before. We have:

- if $x_j \leq x_i^*$: $R_j = (x_j, y_j, x_i^* - x_j, \bar{h}_j)$;
- if $x_j > x_i^*$: $R_j = (x_j, y_j, \bar{w}_j, y_i^* - y_j)$.

For the guillotine constraint, besides updating the possible rectangles in the list overlapping with the last packed item i, the possible rectangle $R_j(x_j, y_j, \bar{w}_j, \bar{h}_j)$ overlapping with 'the first suitable rectangle' $R_i^*(x_i^*, y_i^*, \bar{w}_i^*, \bar{h}_i^*)$ but not overlapping with the item i should also be updated. We have:

- if $x_j \leq x_i^*$: $R_i^* = (x_i^*, y_i^*, \bar{w}_i^*, y_j - y_i^*)$. In Figure B1(left), if the height of the first suitable rectangle is greater than $y_j - y_i^*$, after packing item i, the new possible rectangle created from this first suitable rectangle with the position at the top-left corner of item i may violate the guillotine constraint for the future packing process.
- If $x_j > x_i^*$: $R_j = (x_j, y_j, \bar{w}_j, y_i^* - y_j)$. As shown in Figure B1(right), if the possible rectangle's height is greater than $y_i^* - y_j$, the next item packed at position (x_j, y_j) may violate the guillotine constraint.

Case 2: $\lambda_k \geq \alpha - \alpha'$. This case is very similar to Case 1. A bin of type k is selected for $\mu(\alpha, \beta, \alpha')$. Then $\mu(\alpha, \beta, \alpha')$ is solved by selecting $\alpha - \alpha'$ items from λ_k priority items and packing them into bin k to realise the maximum occupation rate. We list λ_k priority items in increasing order of their area and select the first $\alpha - \alpha'$ items in the list. If these items can be packed into bin k, we obtain an initial occupation rate. Then we apply the exchange procedure to improve this initial occupation rate. The exchange procedure is the same as in Case 1. If $\alpha - \alpha'$ initial priority items are not feasible for bin k, $\mu(\alpha, \beta, \alpha') = -\infty$.

Case 3: $\lambda_k = 0$. This is identical to Case 3 for H1.

View publication stats