

A two-dimensional bin-packing problem with conflict penalties

Kunpeng Li^{a,*}, Hailan Liu^a, Yong Wu^b, Xianhao Xu^a

^a School of Management, Huazhong University of Science & Technology, Wuhan, Hubei, P.R. China. 430074.

^b Department of International Business & Asian Studies, Griffith University, Gold Coast Campus, QLD 4222, Australia

*Corresponding author.

Tel.: +86-13554436029.

Fax: +86-27-87556437.

E-mail address: likp@mail.hust.edu.cn.

Postal address: School of Management, Huazhong University of Science

& Technology, 1037 Luoyu Road, Wuhan, Hubei, P. R. China. 430074.

A two-dimensional bin-packing problem with conflict penalties

Abstract

In this paper, we address the two-dimensional bin-packing (2BP) problem with variable conflict penalties, which incur if conflicting items are loaded into the same bin. Such a problem is observed in applications such as supermarket chains and automobile components transportation. The problem not only focuses on minimization of number of bins used, but also deals with the conflict penalties at the same time. We propose a heuristic method based on the IMA algorithm initially proposed by El Hayek et al. (2008) and adapt it to solve this problem. A local search procedure is also designed to further improve the solutions. For instances derived from benchmark test data, the computational results indicate that the adapted IMA heuristic algorithm with local search effectively balances the number of bins used and the conflict penalties. The algorithm outperforms several adapted approaches that are well known for the 2BP problems.

Keywords: bin packing with conflict penalties; heuristics; tabu search;

1. Introduction

The well known two-dimensional bin-packing (2BP) problem is to load a set of rectangular items into larger identical rectangles (bins), with the objective to minimize the number of bins used without overlapping loaded items. This problem can be observed in many industries, such as the stock cutting in wood industry, the packing and container loading in transportation operations, and the paging of articles in newspapers. Each problem in real practice has its own features (El Hayek et al., 2008). For example, in the case when some items are incompatibly loaded into the same bin, we have the two-dimensional bin-packing problem with conflicts.

In this paper, we address the 2BP problem with conflict penalties. In practice, there might be some items in a given set that could not be loaded into the same bin (strict conflict constraint). However, there are also cases where items can be loaded

into the same bin by incurring penalties, such as safety distance (which takes more space), or extra protection (which increases cost and takes space at the same time) between the items. In this case, the conflict penalty value for each pair of items should not always be either 0 or 1, as often assumed in the literature (Khanafer et al., 2012a). This problem can be observed in the replenishment of supermarket chains with central warehouses. When a pallet is transferred from the warehouse to the store to be unpacked, it may be time-consuming if the racks for the commodities on the pallet are far away from each other in the store. Therefore, commodities can be said to have conflict penalties to reflect the time and labor consumption for pallet unpacking, when they are loaded into the same bin but belong to different storage areas. The same problem can be observed in other industries, such as the automobile manufacturing supply chains in which thousands of components are to be transferred from suppliers to manufacturers. For example, a famous automobile manufacturer located in central China acquires most of its parts from suppliers scattered around Shanghai and Guangzhou where one supply hub is set up in each area, respectively. In a milk-run system, the pallets with component packages are collected from suppliers, transferred to the supply hub, and subsequently repacked to pallets. Then, they are transported to the manufacturer. In order to control the component quality during transportation, each supplier requires that its products should not be mix-loaded with other suppliers' products. If this requirement cannot be fulfilled, suppliers will only allow stacking of their own products and take no responsibility for quality problems if this condition is violated. Hence, it is desirable to directly transfer the pallets from each supplier to the manufacturer but obviously the transportation cost would be unreasonably high. Therefore, it is necessary not only to minimize the number of pallets used but also to minimize the unpacking operations in the supply hub for the incoming pallets. In other words, the component packages from the same supplier are better packed in the same pallet in the supply hub. The conflict penalties between different packages from different suppliers indicate the undesirable operations of unpacking and combining incoming pallets.

The 2BP problem with conflict penalties is a typical k -partite graph $G = (V, E)$

(Diestel, 2000, p. 14), where the items can be represented by the vertices (V) and the penalties represented by the edges (E). Any combination of vertices in one partition can be loaded into the same bin (while space allows) without incurring penalty, while any combination of vertices from more than one partition will incur penalties.

Many industrial applications can be modeled as a typical 2BP problem, such as newspaper paging (Lodi et al., 2002a), time tabling (Laporte and Desroches, 1984), scheduling, and database storage (Jansen, 1999), etc. Variations of the 2BP problem have been studied in the research community in different forms. Reviews of the 2BP literature, for example, are provided by Lodi et al. (2002a), Lodi et al. (2002b) and Riff et al. (2009). It is obvious that the 2BP with conflict penalties is NP-hard since the simpler one-dimensional bin-packing problem is proven to be strongly NP-hard (Garey and Johnson, 1979). Various exact and approximate algorithms have been proposed for the 2BP with conflicts in the literature. The exact methods, for example, are developed by Muritiba et al. (2010) and Elhedhli et al. (2011). Lower bounds are also derived by Gendreau et al. (2004), Muritiba et al. (2010), and Khanafer et al. (2010). Heuristic algorithms, as an effective means to provide solutions to such problems, also received much attention (e.g., Gendreau et al., 2004; Epstein et al., 2008; Khanafer et al., 2012b).

Different variants of the 2BP with conflicts are also studied in the literature. Epstein et al. (2011) address the online variable-sized bin packing problem with conflicts. Hamdi-Dhaoui et al. (2012) investigate the 2BP with partial conflicts, where two conflicting items can be packed into the same bin as long as they maintain a safety distance between each other. To the best of our knowledge, the proposed problem resembles the published work in Khanafer et al. (2012a) most but differs in the objective function. The work in Khanafer et al. (2012a) develops heuristic algorithms to address the min-conflict packing problem, in which the number of violated conflicts is minimized while the number of bins used is fixed. A bi-objective version of the problem is also considered in order to find the trade-off between the number of bins used and the violation of the conflict constraints. Our work combines the number of bins used and conflict minimization into one objective function by

assigning different weights to them while Khanafer et al. (2012a) take the Pareto front of these two. Moreover, unlike the common practice of assigning either 0 or 1 to conflicts in existing literature, we utilize different levels of penalty to reflect the fact that not all conflicting pairs of items (edges in the graph G) should incur the same level of penalty in practice. In the supermarket chain example, it is natural to assume that the cost (and thus penalty) should be high if two items' storage areas are far away from each other. Therefore, we assign various conflict penalty values for the conflicts among the items in this paper.

The remainder of this paper is organized as follows: In Section 2 we give the notation and the definitions used throughout the paper, as well as the mathematical formulation of this problem. Efficient and easy-to-implement algorithms are subsequently proposed in Sections 3 and 4, respectively. Section 5 presents the computational performance of the proposed algorithms on test problems derived from the literature and Section 6 concludes the paper.

2. Problem Formulation

Suppose there is a set A of n items $a_i = (w_i, h_i)$, $i = 1, \dots, n$, where w_i and h_i indicate the item's width and height, respectively. We denote W and H as the width and height of the identical bins $B = (W, H)$, in which the items will be packed. Without loss of generality, we assume that the sizes of the bin and items are positive integers satisfying $w_i \leq W$ and $h_i \leq H$, for every item $a_i \in A$. In practice, items a_i and a_j packed in the same bin might need to be stored on different racks after unpacking. Therefore, there exists transportation cost or conflict penalty in the order of distance from a_i 's rack to a_j 's rack, and vice versa. If these racks are close to each other, the conflict penalty should obviously be small. We use c_{ij} to denote the conflict penalty between a_i and a_j if they are packed in the same bin. Therefore, the problem in this paper involves two objectives, i.e., both the bin number and the conflict penalties (e.g., follow up transportation cost) should be minimized, while most of the work in the literature only consider bin number minimization. We transform this problem into a single objective function by assigning a weight to the conflict penalty in the objective

function, which is denoted by ∂ . Obviously, the value of the weight ∂ should be decided according to practical circumstances. In our example, if the transportation distance from the warehouse to the store is far longer than the transportation distance in the store, then the weight for ∂ in the objective function should be small.

The specific 2BP with conflict penalties addressed in this paper has the following characteristics:

- There may have identical items in the item set A .
- The number of items is given, while the number of bins is unlimited.
- The conflict penalty c_{ij} is between 0 and 1.
- Items can be rotated by 90° , as long as they are within the boundary of the bin and keep their sides parallel to bin sides. This expands the search space considerably compared with the fixed orientation cases.

The parameters and variables that are used to describe the mathematical formulation are as follows:

Parameters:

- M An arbitrarily large number;
- (W, H) Width and height of bin;
- (w_i, h_i) Width and height of item a_i ;
- c_{ij} Penalty cost between items a_i and a_j ;
- n The number of items;

Variables:

- b_{ij} A binary variable which equals 1 if item a_i is packed in bin j , otherwise 0;
- z_i A binary variable which equals 1 if bin i is used, otherwise 0;
- r_i A binary variable which equals 1 if item a_i is rotated, otherwise 0;
- (x_i, y_i) Coordinates of the left-bottom corner of item a_i ;
- l_{ij}, f_{ij} Binary variables used to indicate the relative placement of items. l_{ij} or f_{ij} equals 1 if item a_i is on the left side of, or in front of item a_j , respectively; otherwise 0;
- s_{ij} A binary variable which equals 1 if items a_i and a_j are packed in the

same bin, otherwise 0.

The objective function is:

$$\min \sum_{i=1}^n z_i + \partial \sum_{i=1}^{n-1} \sum_{j=1}^{i+1} c_{ij} s_{ij}$$

Subject to:

$$x_i + (1-r_i)w_i + r_i h_i \leq W \quad \text{for all } i, j \quad (1)$$

$$y_i + (1-r_i)h_i + r_i w_i \leq H \quad \text{for all } i, j \quad (2)$$

$$x_i + (1-r_i)w_i + r_i h_i \leq x_j + (1-l_{ij})M \quad \text{for all } i, j, i \neq j \quad (3)$$

$$y_i + (1-r_i)h_i + r_i w_i \leq y_j + (1-f_{ij})M \quad \text{for all } i, j, i \neq j \quad (4)$$

$$l_{ij} + l_{ji} + f_{ij} + f_{ji} \geq s_{ij} \quad \text{for all } i, j, i < j \quad (5)$$

$$\sum_{j=1}^n b_{ij} = 1 \quad \text{for all } i \quad (6)$$

$$\sum_{i=1}^n b_{ij} \leq z_j M \quad \text{for all } j \quad (7)$$

$$s_{ij} \geq b_{ik} + b_{jk} - 1 \quad \text{for all } i, j, k, i < j \quad (8)$$

The objective function, as discussed before, is to minimize the sum of the numbered bins and the weight adjusted total penalty costs. Constraints (1) and (2) ensure that items are always packed within the bin boundaries. Constraints (3)–(5) guarantee that items will not overlap when they are packed into the same bin. Constraints (6) and (7) dictate that any item can only be packed into one bin and only when this bin is used. Constraints (8) enforce that the binary variable s_{ij} only counts when items a_i and a_j are packed into the same bin k .

Due to the high complexity of the problem, exact approaches become intractable for solving large scale instances that often arise in practice. In most of the cases, researchers employ heuristic or meta-heuristic methods that are able to produce near optimal solutions within reasonable computational time. In this paper, we propose two algorithms to tackle the problem. The first one adapts the IMA algorithm initially proposed by El Hayek et al. (2008) and applies a local search algorithm to improve

the solutions generated; the second one is a tabu search based algorithm which on one hand provides another approach to solve the problem and on the other hand enables numerical comparisons with the adapted IMA algorithm.

3. The Heuristic Algorithms

Parreño et al. (2008) and El Hayek et al. (2008) present a maximal space algorithm to solve the 3D and 2D container loading problem, where a GRASP algorithm and a heuristic called IMA are proposed, respectively. A variable neighborhood search algorithm is further developed for the same problem in Parreño et al. (2010). These algorithms perform well on benchmark problems. Obviously, the basic idea behind these algorithms can be extended to further develop more efficient problem-specific approaches. In this section, we propose an algorithm based on the maximal-space algorithm which has been discussed and presented in Parreño et al. (2008) and El Hayek et al. (2008). For the sake of completeness, the main ideas and steps of the maximal space algorithm and the IMA are briefly discussed in Sections 3.1 and 3.2 below.

3.1 The maximal space algorithm

The maximal space algorithm utilizes the concept of “maximal space”, which is illustrated in Figure 1. When packing a rectangle into an empty bin, the first item is packed at the bottom left corner of the bin. Two maximal spaces, not necessarily to be disjoint though, are generated according to the definition given in Parreño et al. (2008). They are drawn separately on the right hand side of Figure 1.1, where we can see that spaces 1 and 2 have a shared space. Figure 1.2 shows the maximal spaces after another rectangle is packed into the bin at the top right corner of maximal space 1, which results in three maximal spaces 3, 4, and 5.

[Insert Figure 1 at about here]

Algorithm 1: The maximal space algorithm in Parreño et al. (2008).

Data: a set of boxes to be packed, a set of empty maximal spaces S

```
1 while not all boxes have been packed do
2   Choose the box to be packed;
3   Choose the maximal space in  $S$ ;
4   Pack the chosen box into the chosen maximal space;
5   Update the maximal space list  $S$ ;
6 Report the packing results;
```

Algorithm 1 briefly describes the main steps of the maximal space algorithm in Parreño et al. (2008). Needless to say, the most important issue in the maximal space algorithm is the criteria to select the box and the maximal space in Steps 2 and 3, respectively. It is clear that appropriate criteria will result in good performance of the algorithm for problems with special features. In Parreño et al. (2008), the maximal space with the minimum distance to a bin corner is selected. Two criteria are used for selecting the box to be packed: the first is the Best-Volume criterion, which selects the box producing the largest increase in volume utilization; the second is the Best-Fit criterion which selects the box that best fits into the maximal space.

3.2 The IMA algorithm

El Hayek et al. (2008) develop a more complex procedure for the selections of the box and the maximal space, where a pair of box and maximal space is selected when it is producing the largest value calculated by the following equation:

$$\theta(a_i, ma) = q_1(w_i h_i) / (w_{ma} h_{ma}) + q_2(dx_i / w_{ma}) + q_3(dy_i / h_{ma}) + q_4(dx_i^2 + dy_i^2) / (w_{ma}^2 + h_{ma}^2)$$

Where a_i is the i^{th} item and ma denotes the current maximal space; w_i , h_i and w_{ma} , h_{ma} are the width and height of a_i and ma , respectively; dx_i and dy_i are the projections of the edges of the packed item a_i on x - and y -axis; q_1 to q_4 are the four weights, each between 0 and 1 and all four sum to 1. Different combinations of q_i values affect the evaluation value significantly and the best combination can be searched in an iterative way. By increasing the value of each q_i from 0 to 1 with a step value (e.g., 0.05), the best combination can be identified once all possible combinations of the four q_i are

examined.

3.3 The adapted IMA algorithm

The objective function in this paper not only considers minimizing the number of bins used, but also incorporates the weighted conflict penalties. We adapt the maximal space algorithm and the preliminary experiments indicate that the criteria in El Hayek et al. (2008) perform much better than that of Parreño et al. (2008) for this particular problem. The pair selection criterion in El Hayek et al. (2008) is further modified in this paper to better address the research problem and to reflect the specific problem characteristics. One additional term is introduced in the following equation, which is used to calculate the value for a pair of box and maximal space. Clearly, the one with maximum value is selected.

$$\theta(a_i, ma) = q_1(w_i h_i) / (w_{ma} h_{ma}) + q_2(dx_i / w_{ma}) + q_3(dy_i / h_{ma}) + q_4(dx_i^2 + dy_i^2) / (w_{ma}^2 + h_{ma}^2) + q_5(m - \sum_j c_{ij}) / m$$

The additional term, which measures the total conflict penalties, is introduced with a corresponding weight q_5 . The sum from q_1 to q_5 is 1 and each is between 0 and 1. m indicates the number of items packed in the bin. Note that j is the index for all the items that are already loaded in the bin. The configuration of weights is identical to that of the previous one. In order to reduce computational time, the step value is adjusted to be 0.1 when evaluating all the combinations of the five q_i . Experimental tests indicate that there is little difference for the search of the best combination of q_i when the step value increases from 0.05 to 0.1.

Algorithm 2 shows how the adapted IMA works. The procedure basically tries to iteratively find the maximum θ while keeping an eye on the conflict penalty. The resulting penalty is assessed by placing an item into a particular bin. Another coefficient α is also introduced to control whether a new bin should be used when the conflict penalty reaches a certain level. This is achieved by combining with the coefficient for conflict penalty ∂ to reflect its contribution to the final objective function value. The values for the parameters are determined by preliminary experiments. Thus, the step size to control the increment of α is set to 0.2 while its

value changes between the range $[0.1, 0.9]$.

Algorithm 2: The adapted IMA algorithm

Data: the set of boxes to be packed A and the list L_A which manages them; step sizes s_q and s_α for adjusting weights q_i and α ; list L_S to manage available maximal spaces, N_B which is the number of bins used

```

1 for  $q_1 = 0.0; q_1 \leq 1.0; q_1 = q_1 + s_q$  do
2   for  $q_2 = 0.0; q_2 \leq 1.0 - q_1; q_2 = q_2 + s_q$  do
3     for  $q_3 = 0.0; q_3 \leq 1.0 - q_1 - q_2; q_3 = q_3 + s_q$  do
4       for  $q_4 = 0.0; q_4 \leq 1.0 - q_1 - q_2 - q_3; q_4 = q_4 + s_q$  do
5         for  $\alpha = 0.1; \alpha \leq 0.9; \alpha = \alpha + s_\alpha$  do
6            $q_5 = 1.0 - q_1 - q_2 - q_3 - q_4;$ 
7           Initialize list  $L_S$ ;  $L_A = A$ ; number of bins used  $N_B = 0$ ;
8           while the list  $L_A$  is not  $\emptyset$  do
9              $N_B = N_B + 1;$ 
10            while the list  $L_S$  is not  $\emptyset$  do
11               $\theta_{\max} = 0.0;$ 
12              for each maximal area  $ma_j$  in  $L_S$  do
13                for each box  $a_i$  in  $L_A$  do
14                  if  $a_i$  can be packed in  $ma_j$  without overlapping
15                    and  $\theta(a_i, ma_j) > \theta_{\max}$  then
16                       $\theta_{\max} = \theta(a_i, ma_j); a^* = a_i; ma^* = ma_j;$ 
17                  if  $\theta_{\max} > 0.0$  and the additional penalty of loading  $a^*$ 
18                     $< \alpha/\partial$  then
19                      Pack  $a^*$  at  $ma^*$ ; update  $L_S$  and  $L_A$ ;
20                  else
21                    Close the current bin; break;
22            Calculate the current objective function value;
23            if a lower current objective function value is achieved then
24              Record the objective function value and the number of bins
25              used;

```

3.4 A local search algorithm

In order to reduce total conflict penalties of a loading scheme generated by the adapted IMA, a local search improvement algorithm is proposed. The algorithm takes

the solution produced by the adapted IMA algorithm as the initial solution and subsequently exchanges items among the bins in the hope to reduce the total conflict penalties. It is rather clear that the number of the bins used will remain unchanged since only item exchanges will happen. The detailed steps of the algorithm are illustrated in Algorithm 3, in which X denotes the index for bins and K denotes the index for items in a bin.

Algorithm 3: Improving the adapted IMA solutions by local search.

Data: solutions generated by the adapted IMA algorithm

```

1 Calculate the total penalties for each bin;
2 Arrange the bins in decreasing order of the total penalties;
3 Set the bin index  $X = 1$ ;
4 while  $X$  not exceeding the number of bins used do
5   for each item in current bin  $X$  do
6     Calculate the reduced penalty if this item is removed;
7   Arrange the items in  $X$  in the order of non-increasing reduced penalty;
8    $K = 1$ ;
9   while  $K$  not exceeding the number of items in current bin  $X$  do
10    Remove the item  $K$  from current bin  $X$ ;
11    Identify those bins that will result in a reduction in total penalty if  $K$  is
        inserted to them;
12    Arrange the identified bins in non-increasing order of tentative total penalties;
13     $Y = 1$ , Switch = False;
14    while  $Y$  not exceeding the number of identified bins and Switch is False do
15      Insert  $K$  into the bin and use the adapted IMA algorithm to pack it;
16      if the packing result is feasible then
17        Complete the taking away of  $K$  from  $X$  and insertion into  $Y$ ;
18        Switch = True; goto Step 1;
19       $Y = Y + 1$ ;
20     $K = K + 1$ ;
21   $X = X + 1$ ;
22 Report the packing results;
```

The local search iteratively re-arranges items in pairs in the hope that the overall conflict penalty will be reduced. The local search starts with the item that incurs the largest penalty within its current bin. The item is then re-arranged into other bins that have the potential to reduce the overall penalty (potential bins are indexed by Y in

Algorithm 3). Among those bins, the best feasible option is chosen for the re-arrangement. All the other items are treated in a similar way until no further reduction can be made by switching items among all bins.

4. A Tabu Search Algorithm

In order to evaluate the performance of the proposed heuristic algorithm in Section 3, we propose a meta-heuristic algorithm based on tabu search to solve the problem. Tabu search is a local search meta-heuristic proposed independently by Glover (1986) and Hansen and Jaumard (1990). According to recent literature, tabu search is widely applied for solving container loading problems (Lodi et al. 2004; Bortfeldt et al. 2003; Harwig et al. 2006; Alvarez-Valdes et al. 2005, Khanafer et al, 2012a, etc.).

A tabu search procedure starts from an initial feasible solution. During the search process, a set of neighbors is generated and the best feasible one is chosen. A tabu list will also be maintained to guide the search and prevent the search from becoming stuck at local optimal points. A neighbor can be either tabu or non-tabu; furthermore, the tabu status may be overridden by an aspiration criterion. Intensification and diversification schemes are also applied during the search.

In the initial solution generating phase for tabu search, the solution is generated in a similar way to the adapted IMA algorithm listed in Algorithm 2. However, the selection criterion for matching a pair of maximal space and an item is the maximum space utilization ratio, not θ . This has some computational advantages since the iterative search for the best combination of q_i is eliminated. Once an initial solution is generated, the tabu search follows the steps of TSpack in Lodi et al. (2004).

5. Computational Experiments

The presented algorithms described in this paper were tested on the instances derived from the well-known two-dimensional finite bin-packing instances of Berkey and Wang (1987). We assume that in the warehouse, the items are loaded into a set of bins which are subsequently transported to the supermarket. The set of items in the bins will be assigned to six storage areas in the supermarket. Within a given bin, if

two items are assigned to the same storage area, there is no conflict penalty incurred; otherwise, the penalty value will depend on the distance between the two storage areas the two items belong to. The penalty value is increased by 0.2 each time if the difference of the indexes between the two storage areas is increased by 1. Also, the weight of the conflict penalties in the objective function is tested with five values: 0.0, 0.01, 0.05, 0.1 and 0.5, respectively. Allowing the weight to be zero, the problem will deteriorate to a typical 2BP. In Berkey and Wang (1987), there are five sets of data that are generated based on $n = 20, 40, 60, 80$, and 100, respectively. For each data set, there are six classes according to different bin sizes. We generated 10 instances for each problem configuration and the average of these 10 instances is reported. The algorithms were coded in C++. The computational experiments were conducted on a personal computer with Intel Pentium dual-core 2.8 GHz CPU and 2 GB RAM.

We compare the results on the test instances given by four algorithms: 1) the original IMA algorithm proposed in El Hayek et al. (2008) (indicated as IMA); 2) the adapted IMA in this paper (IMA_A); 3) the adapted tabu search algorithm in this paper (TS); and 4) the proposed adapted IMA algorithm with the local search (IMA_A_LS). As mentioned in Section 3.3, preliminary experiments indicated that the IMA algorithm in El Hayek et al. (2008) outperforms the maximal space algorithm of Parreño et al. (2008) for this particular problem. Thus, the results of the original maximal space algorithm were not reported here.

Figure 2 illustrates the impact of the weight ∂ on the ultimate packing results since it directly contributes to the final objective function value. A 20-item case is selected and solved by IMA_A and the solutions with ∂ at 0.01, 0.05, 0.1 and 0.5 are presented. It can be observed that when the weight is not very big (in this example, when $\partial = 0.01, 0.05$ and 0.1) and therefore contributing less to the objective function value, the packing solutions are the same excepting the objective function value changes as the weight increases. This means that the additional conflict penalties combined with the weight is not big enough to outweigh the usage of a new bin. When the weight increases to 0.5 and contributes a lot to the objective function value, it can be observed that two additional bins are used and the packing results (grouping

items into bins) also change accordingly.

[Insert Figure 2 at about here]

Figure 3 graphically presents the packing results provided by different approaches for a 20-item instance when the weight δ is set at 0.01. Although the number of bins used by each algorithm is identical, the objective function value, and thus the conflict penalties vary between different approaches. The IMA approach, which does not consider the conflict penalties when packing, results in the largest objective function value, while IMA_A_LS has the smallest. The packing results are considerably different when the conflict penalties (even when $\delta = 0.01$) are contributing to the final objective function value.

[Insert Figure 3 at about here]

The computational results of the four algorithms for each of the aforementioned weights δ are shown in Tables 1 to 5, respectively. Apart from Table 1 where $\delta=0.0$, we also list IMA's results in Tables 2 to 5 for reference even though it does not consider the conflict penalties during packing. The first two columns in each table include the class of the test instances (with bin dimensions indicated) and the number of items to be packed. From a computational time perspective, except a few cases where the CPU time exceeds 1 second, the majority of the test instances are solved within a very short time which is much less than 1 second. Therefore, the computational times used by each algorithm are not reported here.

According to the average results of algorithms IMA and TS for solving instances of *Class_01*, the proportion of conflict penalty in the objective function increases from 0 to 0.40 when the weight δ increases from 0 to 0.5. For the results of algorithm IMA_A and IMA_A_LS, the proportion of conflict penalty is always less than 0.1 when the weight δ changes. It would be reasonable to conclude that better results should contain smaller proportion of the conflict penalty in the objective value

for all of the test instances.

[Insert Table 1 here]

Table 1 presents the computational results for the test problems with $\partial=0.0$, which makes the problem a typical 2BP. The results reported are the average objective function value (also the number of bins used) over the 10 test instances for each problem configuration. The column IMA_A reports for both IMA_A and IMA_A_LS since the local search will not reduce the number of bins used. Overall, the results are not so different among the four algorithms. However, there is still a margin between IMA and the TS approaches for most of the cases. The IMA_A and IMA_A_LS achieves equivalent to the IMA except on a few occasions it is outperformed by the IMA. This is rather interesting to note, since the only difference between the IMA and the IMA_A is the pair selection criteria. The IMA_A and IMA_A_LS consider conflict penalties when packing. However, this does not contribute to the objective function value, since the weight ∂ is at 0.0.

[Insert Tables 2-5 at about here]

Tables 2 to 5 display the computational results for the test problems with different weights. The results are organized in two major parts: one is the objective function value and the other is the number of bins used for each problem configuration. Note that the column IMA_A, which is the last column of the ‘Number of bins used’ section of Table 2 to 5, reports for both IMA_A and IMA_A_LS, since the local search will not change the number of bins used. Same as in Table 1, the results are the average of the 10 test instances within each problem configuration. In terms of objective function, it can be observed that once the conflict penalties are considered, the IMA_A_LS approach always achieves the best results. On some occasions, IMA_A gets equally good ones. This indicates that the IMA_A_LS provides effective packing results among the four approaches. Compared with IMA_A, the results also suggest that the local search effectively helps to improve the

performance even when the number of bins used does not change. As mentioned before, the performance of IMA is only for reference, and we can see that as the weight of conflict penalties increases, the performance gap between IMA and IMA_A_LS amplifies. This clearly demonstrates the effectiveness of IMA_A_LS's ability to handle conflict penalties. The same can be observed for the TS approach. The results are somewhat surprising since the objective function value is sometimes even bigger than the IMA approach. Of course, the results are linked with the number of bins used and it seems that the TS approach still focuses more on minimizing the number of bins.

In the above experiments, we evenly and randomly distribute the items within the six storage areas for each test instance. However, it is worthwhile to evaluate the performance of the four approaches when the clustering of items changes for each test instance. Since the size of the resulting test instances will be extremely large, we just select one instance from the previous experiments (in this case, Class_03). The items are subsequently re-grouped to create different group structures. We randomly select 20%, 40%, 60% and 80% of the items and allocate them into the third storage area. Then, we randomly and evenly distribute the rest items in the other five storage areas. Therefore, we have four group structures. The average performance of the four algorithms under different weight settings ($\partial = 0.01, 0.05, 0.1$ and 0.5 , respectively) is summarized and reported in Figure 4. Each subfigure shows one weight setting. The percentage of items allocated to the third storage area is indicated as the x axis. The y axis displays the objective function value (note that the scale varies between different subfigures), while the horizontal line within each bar indicates the number of bins used. The order of algorithms presented in the figure is: TS, IMA, IMA_A and IMA_A_LS, as marked in Figure 4a.

In general, we can observe that as the percentage of items belong to the third storage area increases, the objective function values of all four approaches decrease accordingly, since the potential penalty is reduced when more items belong to one storage area. The performance of the four algorithms follows what we have described in Tables 2 to 5, i.e., TS incurs the highest objective function value for most of the

times, while IMA_A_LS performs the best. The number of bins used by the IMA_A and IMA_A_LS changes when the weight of the penalty and the percentage of items in the same storage area change. However, for IMA and TS, as discussed before, seem to be rather insensitive to these changes. Compared with TS and IMA, IMA_A and IMA_A_LS always balance the number of bins used and the weighted penalty while minimizing the objective function value.

[Insert Figure 4 at about here]

From a pure bin number minimization perspective, we can observe that the IMA_A_LS approach (including IMA_A) can achieve the best performance for some test instances when the weight ∂ is small. As the weight increases, the number of best performing times (in terms of bin number minimization) quickly diminishes which is an indication that the IMA_A_LS treats the conflict seriously and balances the number of bins used and the overall objective function value. The adapted tabu search approach, on the other hand, maintains the number of best performing cases on the results of bin numbers and sometimes even with more cases as the weight ∂ increases. It again suggests that the adapted tabu search approach concentrates on bin number minimization.

In summary, we found throughout our investigation that the adapted IMA combined with the local search algorithm performed strongly in comparison to the adapted IMA and the adapted tabu search algorithms. With the portion of the conflict penalties increasing in the objective value, the corresponding gaps increase. This indicates that the proposed approach can effectively deal with the conflict penalties in the objective function.

6. Conclusion

Traditional two-dimensional bin-packing problem takes the bin number minimization as the only objective. In practice, however, there might be cases where pairs of items have variable degrees of penalty between them, such as additional labor or time consumption for unpacking bins in supermarkets or warehouses. Therefore,

we introduce a new 2BP problem with variable degree of conflict penalties and deal with bin number minimization and penalty reduction in an integrative way. We propose a heuristic method based on the IMA algorithm (El Hayek et al., 2008) and adapt it to tackle this problem. A local search algorithm is further introduced to improve the results provided by the adapted IMA algorithm. An adapted tabu search algorithm is also proposed, partly for the purpose of validating the performance of the adapted IMA algorithm. The computational results reveal that the adapted IMA with local search provides the best solution compared to other approaches introduced in this paper.

Future work related to this interesting problem might include developing lower bounds to gauge the effectiveness of the proposed heuristic algorithms. Developing exact algorithms should also be considered, especially for medium sized problems. Furthermore, it is also desirable to combine this loading stage with the subsequent routing stage and provide loading and routing decisions in an integrated way.

Acknowledgements

The authors are indebted to the anonymous reviewers for their valuable comments, suggestions, and remarks. This research is supported by National Nature Science Foundation of China (No. 70972019, No.71131004 and No.71372133), and Program for New Century Excellent Talents in University (NCET-10-0410).

References

- Alvarez-Valdes, R., F. and Parreno J.M., 2005. Tamarit, A tabu search algorithm for the pallet loading problem. *OR Spectrum*, 27(1): p. 43-61.
- Berkey, J.O. and Wang, P.Y., 1987. Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5), 423--429.
- Bortfeldt, A., Gehring, H. and Mack, D., 2003 A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5), 641-662.
- Diestel, R. 2000. *Graph Theory* (2nd edn). Springer, New York.
- El Hayek, J., Moukrim, A. and Negre, S. 2008. New resolution algorithm and pretreatments for the two-dimensional bin-packing problem. *Computers & Operations Research*, 35(10), 3184-3201.
- Elhedhli, S., Li, L., Gzara, M., Naoum-Sawaya, J., 2011. A Branch-and-Price Algorithm for the Bin Packing Problem with Conflicts, *INFORMS Journal on computing*, 23(3), 404-415.
- Epstein, L., Favrholt, L.M., Levin, A., 2011. Online variable-sized bin packing with conflicts, *Discrete Optimization*, 8(2), 333-343.
- Epstein, L., Levin, A., Van Stee, R., 2008. Two-dimensional packing with conflicts. *Acta Informatica*, 45(3), 155--75.
- Garey, M.R. and Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York.
- Gendreau, M., Laporte, G., Semet, F., 2004. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31, 347--58.
- Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533--549.
- Hamdi-Dhaoui, K., Labadie, N., Yalaoui, A. 2012. Algorithms for the two dimensional bin packing problem with partial conflicts, *Rairo-Operations Research*, 46(1), 41-62.
- Hansen, P. and B. Jaumard, 1990. Algorithms for the maximum satisfiability problem. *Computing*, 44(4), 279--303.
- Harwig, J.M., Barnes, J.W. and Moore, J.T. 2006. An adaptive tabu search approach for 2-dimensional orthogonal packing problems. *Military Operations Research*, 11(2), 5-26.
- Jansen K. An approximation scheme for bin packing with conflicts. 1999. *Journal of Combinatorial Optimization*, 3, 363--77.
- Khanafer, A., Clautiaux, F., Talbi E.G., 2010. New lower bounds for bin packing problems with conflicts. *European Journal of Operational Research*, 206(2), 281--288.
- Khanafer, A., Clautiaux, F., Hanafi, S., Talbi, E., 2012a. The min-conflict packing problem, *Computers & Operations Research*, 39(9), 2122-2132.
- Khanafer, A., Clautiaux, F., Talbi, E., 2012b. Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts, *Computers & Operations Research*, 39(1), 54-63.
- Laporte G, Desroches S., 1984. Examination timetabling by computer. *Computers & Operations Research*, 11(3), 351--60.
- Lodi, A., Martello, S. and Monaci, M., 2002a. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2), 241-252.
- Lodi, A., S. Martello and D. Vigo, 2002b. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1-3), 379-396.

- Lodi, A., Martello, S. and Vigo, D. 2004. TSpack: a unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, 131, 203–213.
- Muritiba, A. E. F., Iori, M., Malaguti, E. and Toth, P., 2010. Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 22(3), 401-415.
- Parreño, F., Alvarez-Valdes, R., Tamarit, J. M. and Oliveira, J. F., 2008. A maximal-space algorithm for the container loading problem. *INFROMS Journal on Computing*, 20(3), 412-422.
- Parreño, F., Alvarez-Valdes, R., Tamarit, J. M. and Oliveira, J. F., 2010. Neighborhood structures for the container loading problem: a VNS implementation. *Journal of Heuristics*, 16(1), 1-22.
- Riff, M.C., Bonnaire. X. and Neveu, B., 2009. A revision of recent approaches for two-dimensional strip-packing problems, *Engineering Applications of Artificial Intelligence*, 22, 823–827.

Table 1 Computational results for test problems with $\partial=0.0$.

Class and bin dimension	n	IMA	IMA_A	TS
Class_01 10x10	20	6.60	6.60	6.60
	40	12.90	12.90	12.90
	60	19.50	19.50	19.60
	80	27.00	27.00	27.10
	100	31.30	31.30	31.90
	<i>Avg.</i>	19.46	19.46	19.62
Class_02 30x30	20	1.00	1.00	1.00
	40	1.90	1.90	2.00
	60	2.50	2.50	2.50
	80	3.10	3.10	3.10
	100	3.90	3.90	3.90
	<i>Avg.</i>	2.48	2.48	2.50
Class_03 40x40	20	4.70	4.70	4.90
	40	9.40	9.40	9.40
	60	13.50	13.50	13.70
	80	18.40	18.60	18.90
	100	22.20	22.30	22.50
	<i>Avg.</i>	13.64	13.70	13.88
Class_04 100x100	20	1.00	1.00	1.00
	40	1.90	1.90	1.90
	60	2.50	2.50	2.50
	80	3.10	3.10	3.20
	100	3.70	3.70	3.80
	<i>Avg.</i>	2.44	2.44	2.48
Class_05 100x100	20	5.90	5.90	5.90
	40	11.40	11.50	11.50
	60	17.40	17.60	17.50
	80	23.90	23.90	24.10
	100	27.90	27.90	28.50
	<i>Avg.</i>	17.30	17.36	17.50
Class_06 300x300	20	1.00	1.00	1.00
	40	1.70	1.70	1.90
	60	2.10	2.10	2.20
	80	3.00	3.00	3.00
	100	3.20	3.40	3.40
	<i>Avg.</i>	2.20	2.24	2.30

Table 2 Computational results for test problems with $\hat{\sigma}=0.01$.

Class and bin dimension	n	Objective function value (avg.)				Number of bins used (avg.)		
		IMA	IMA_A	TS	IMA_A_LS	IMA	TS	IMA_A
Class_01 10x10	20	6.69	6.66	6.69	6.65	6.60	6.60	6.60
	40	13.09	13.01	13.08	12.97	12.90	12.90	12.90
	60	19.77	19.68	19.88	19.63	19.50	19.60	19.50
	80	27.34	27.20	27.44	27.12	27.00	27.10	27.00
	100	31.86	31.57	32.39	31.50	31.40	31.90	31.30
	Avg.	19.75	19.62	19.90	19.57	19.48	19.62	19.46
Class_02 30x30	20	1.72	1.72	1.72	1.72	1.00	1.00	1.00
	40	3.55	2.79	3.67	2.68	1.90	2.10	2.00
	60	5.38	3.85	5.66	3.61	2.50	2.50	2.60
	80	7.00	4.63	7.20	4.35	3.10	3.10	3.40
	100	9.10	5.48	9.63	5.06	3.90	3.90	4.00
	Avg.	5.35	3.69	5.58	3.48	2.48	2.52	2.60
Class_03 40x40	20	4.93	4.82	5.02	4.78	4.80	4.90	4.70
	40	9.68	9.60	9.68	9.50	9.40	9.40	9.40
	60	14.00	13.81	14.12	13.68	13.60	13.70	13.50
	80	19.13	19.01	19.46	18.81	18.60	18.90	18.60
	100	22.97	22.78	23.23	22.53	22.30	22.50	22.30
	Avg.	14.14	14.00	14.30	13.86	13.74	13.88	13.70
Class_04 100x100	20	1.76	1.76	1.76	1.76	1.00	1.00	1.00
	40	3.70	2.78	3.82	2.73	1.90	1.90	2.00
	60	5.37	3.87	5.61	3.64	2.50	2.50	2.80
	80	6.92	4.79	7.21	4.37	3.10	3.20	3.40
	100	8.98	5.54	9.21	5.08	3.80	3.80	4.00
	Avg.	5.35	3.75	5.52	3.52	2.46	2.48	2.64
Class_05 100x100	20	5.99	5.97	6.00	5.95	5.90	5.90	5.90
	40	11.71	11.66	11.71	11.58	11.50	11.50	11.50
	60	17.89	17.82	17.79	17.71	17.60	17.50	17.60
	80	24.27	24.20	24.51	24.05	23.90	24.10	23.90
	100	28.45	28.36	29.06	28.17	27.90	28.50	27.90
	Avg.	17.66	17.60	17.82	17.49	17.36	17.50	17.36
Class_06 300x300	20	1.78	1.78	1.78	1.78	1.00	1.00	1.00
	40	3.87	2.77	4.01	2.69	1.70	2.20	2.00
	60	5.25	3.75	5.51	3.63	2.10	2.20	2.50
	80	7.07	4.43	7.34	4.20	3.00	3.00	3.00
	100	9.11	5.44	9.44	4.97	3.40	3.40	3.80
	Avg.	5.42	3.63	5.62	3.45	2.24	2.36	2.46

Table 3 Computational results for test problems with $\partial=0.05$.

Class and bin dimension	n	Objective function value (avg.)				Number of bins used (avg.)		
		IMA	IMA_A	TS	IMA_A_LS	IMA	TS	IMA_A
Class_01 10x10	20	7.03	6.87	7.03	6.83	6.60	6.60	6.60
	40	13.84	13.40	13.82	13.28	12.90	12.90	12.90
	60	20.87	20.33	20.99	20.17	19.50	19.60	19.60
	80	28.70	27.92	28.79	27.59	27.00	27.10	27.00
	100	33.72	32.59	34.34	32.24	31.40	31.90	31.50
	Avg.	20.83	20.22	21.00	20.02	19.48	19.62	19.52
Class_02 30x30	20	4.58	2.96	4.58	2.82	1.00	1.00	2.00
	40	10.01	4.30	9.94	4.30	2.00	2.20	3.10
	60	16.75	5.57	18.30	5.57	2.60	2.50	4.40
	80	22.47	5.94	23.59	5.94	3.20	3.10	5.70
	100	29.86	6.00	32.55	6.00	4.00	3.90	6.00
	Avg.	16.73	4.95	17.79	4.92	2.56	2.54	4.24
Class_03 40x40	20	5.43	5.15	5.51	5.08	4.80	4.90	4.70
	40	10.81	10.07	10.80	9.85	9.40	9.40	9.50
	60	15.59	14.68	15.78	14.35	13.60	13.70	13.60
	80	21.26	20.13	21.69	19.61	18.60	18.90	18.90
	100	25.66	24.01	26.15	23.30	22.30	22.50	22.50
	Avg.	15.75	14.81	15.99	14.44	13.74	13.88	13.84
Class_04 100x100	20	4.79	3.00	4.79	2.91	1.00	1.00	2.10
	40	10.89	4.31	11.49	4.27	1.90	1.90	3.00
	60	16.87	5.62	17.27	5.62	2.50	4.20	4.60
	80	21.96	5.95	23.23	5.95	3.30	3.20	5.80
	100	29.72	6.12	30.85	6.11	3.80	3.80	6.10
	Avg.	16.85	5.00	17.53	4.97	2.50	2.82	4.32
Class_05 100x100	20	6.37	6.16	6.41	6.11	5.90	5.90	5.90
	40	12.55	12.11	12.56	11.92	11.50	11.50	11.50
	60	19.06	18.53	18.97	18.16	17.60	17.50	17.60
	80	25.76	25.19	26.16	24.63	23.90	24.10	24.20
	100	30.65	29.73	31.29	29.11	27.90	28.50	28.40
	Avg.	18.88	18.35	19.08	17.99	17.36	17.50	17.52
Class_06 300x300	20	4.88	2.89	4.88	2.86	1.00	1.00	2.00
	40	12.12	4.32	9.87	4.32	1.90	2.70	3.00
	60	17.64	5.55	18.22	5.53	2.20	3.00	4.60
	80	23.37	5.93	24.71	5.93	3.00	3.00	5.80
	100	31.97	6.00	33.59	6.00	3.40	3.40	6.00
	Avg.	18.00	4.94	18.25	4.93	2.30	2.62	4.28

Table 4 Computational results for test problems with $\partial=0.1$.

Class and bin dimension	n	Objective function value (avg.)				Number of bins used (avg.)		
		IMA	IMA_A	TS	IMA_A_LS	IMA	TS	IMA_A
Class_01 10x10	20	7.45	7.10	7.46	7.07	6.60	6.60	6.60
	40	14.77	13.63	14.75	13.45	12.90	12.90	13.10
	60	22.24	20.76	22.38	20.48	19.50	19.60	19.90
	80	30.41	28.43	30.49	28.03	27.00	27.10	27.40
	100	36.03	33.36	36.78	32.79	31.40	31.90	31.90
	Avg.	22.18	20.65	22.37	20.36	19.48	19.62	19.78
Class_02 30x30	20	8.16	3.54	8.16	3.54	1.00	1.00	2.30
	40	18.02	5.39	17.59	5.36	2.00	2.40	3.70
	60	30.90	5.98	34.11	5.98	2.60	2.50	5.90
	80	41.67	6.00	44.08	6.00	3.30	3.10	6.00
	100	55.72	6.00	61.20	6.00	4.00	3.90	6.00
	Avg.	30.89	5.38	33.03	5.38	2.58	2.58	4.78
Class_03 40x40	20	6.06	5.67	6.13	5.52	4.80	4.90	4.80
	40	12.21	10.40	12.20	10.21	9.40	9.40	9.70
	60	17.59	15.22	17.82	14.98	13.60	13.80	14.30
	80	23.91	20.91	24.48	20.35	18.60	18.90	19.80
	100	29.03	25.02	29.80	24.20	22.30	22.50	23.40
	Avg.	17.76	15.44	18.08	15.05	13.74	13.90	14.40
Class_04 100x100	20	8.58	3.65	8.58	3.62	1.00	1.00	2.60
	40	19.88	5.47	21.08	5.45	1.90	1.90	3.80
	60	31.25	5.97	30.34	5.97	2.50	4.20	5.90
	80	40.61	6.00	40.71	6.00	3.30	6.80	6.00
	100	55.56	6.17	55.85	6.12	3.90	6.90	6.10
	Avg.	31.18	5.45	31.31	5.43	2.52	4.16	4.88
Class_05 100x100	20	6.83	6.39	6.92	6.31	5.90	5.90	5.90
	40	13.60	12.60	13.62	12.39	11.50	11.50	11.90
	60	20.52	19.15	20.43	18.88	17.60	17.50	18.40
	80	27.62	25.84	28.14	25.30	23.90	24.30	24.60
	100	33.37	30.44	34.06	29.75	28.00	28.60	29.10
	Avg.	20.39	18.88	20.63	18.52	17.38	17.56	17.98
Class_06 300x300	20	8.75	3.61	8.75	3.60	1.00	1.00	2.50
	40	22.34	5.41	16.74	5.34	1.90	3.60	4.40
	60	33.09	6.00	33.44	6.00	2.20	3.00	6.00
	80	43.74	6.00	46.42	6.00	3.00	3.00	6.00
	100	60.54	6.00	58.42	6.00	3.40	9.70	6.00
	Avg.	33.69	5.40	32.76	5.39	2.30	4.06	4.98

Table 5 Computational results for test problems with $\partial=0.5$.

Class and bin dimension	n	Objective function value (avg.)				Number of bins used (avg.)		
		IMA	IMA_A	TS	IMA_A_LS	IMA	TS	IMA_A
Class_01 10x10	20	10.79	8.29	10.92	8.23	6.80	6.60	7.60
	40	21.85	14.96	22.14	14.77	13.10	12.90	13.90
	60	33.09	22.54	33.25	22.14	19.80	20.70	21.40
	80	43.71	30.29	43.55	29.59	27.50	28.60	28.70
	100	53.86	35.63	53.69	35.06	32.20	35.10	34.60
	Avg.	32.66	22.34	32.71	21.96	19.88	20.78	21.24
Class_02 30x30	20	36.80	5.33	36.80	5.25	1.00	1.00	4.70
	40	82.10	6.00	77.68	6.00	2.00	3.10	6.00
	60	144.10	6.00	160.54	6.00	2.60	2.50	6.00
	80	195.15	6.00	208.02	6.00	3.30	3.10	6.00
	100	262.58	6.00	290.42	6.00	4.00	3.90	6.00
	Avg.	144.15	5.87	154.69	5.85	2.58	2.72	5.74
Class_03 40x40	20	11.08	6.93	11.03	6.84	4.90	4.90	5.90
	40	23.28	11.65	21.30	11.46	9.60	11.90	10.90
	60	33.44	16.57	30.92	16.32	13.80	18.30	15.50
	80	45.17	22.61	42.28	22.06	18.70	25.50	21.50
	100	55.62	26.42	52.67	26.02	22.50	34.10	25.60
	Avg.	33.72	16.84	31.64	16.54	13.90	18.94	15.88
Class_04 100x100	20	38.89	5.26	38.89	5.24	1.00	1.00	4.70
	40	91.82	6.00	97.80	6.00	1.90	1.90	6.00
	60	146.24	6.00	134.18	6.00	2.50	5.40	6.00
	80	189.86	6.00	176.08	6.00	3.30	7.40	6.00
	100	262.18	6.20	251.65	6.20	3.90	6.90	6.20
	Avg.	145.80	5.89	139.72	5.89	2.52	4.52	5.78
Class_05 100x100	20	10.51	7.61	10.80	7.58	6.00	6.00	6.60
	40	21.83	14.08	20.97	13.71	11.70	13.00	12.90
	60	31.81	21.01	30.92	20.59	18.10	19.20	19.50
	80	42.22	27.97	41.88	27.10	24.10	26.80	26.20
	100	53.57	32.64	52.68	32.07	28.60	35.10	31.50
	Avg.	31.99	20.66	31.45	20.21	17.70	20.02	19.34
Class_06 300x300	20	39.77	5.39	39.77	5.39	1.00	1.00	4.60
	40	104.08	6.00	67.13	6.00	1.90	5.80	6.00
	60	156.64	6.00	154.49	6.00	2.20	4.00	6.00
	80	206.70	6.00	220.11	6.00	3.00	3.00	6.00
	100	289.10	6.00	253.29	6.00	3.40	9.70	6.00
	Avg.	159.26	5.88	146.96	5.88	2.30	4.70	5.72

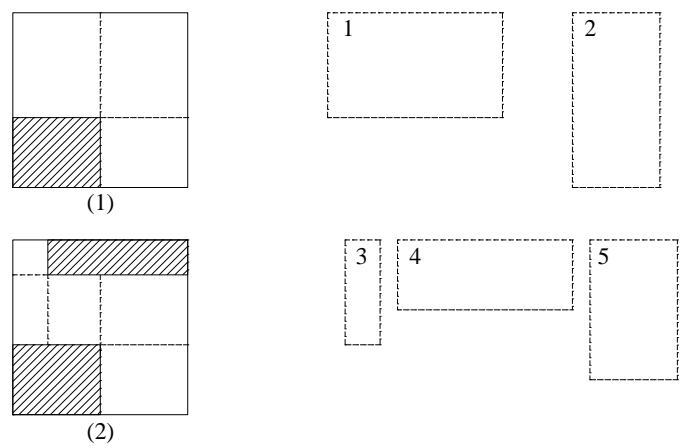
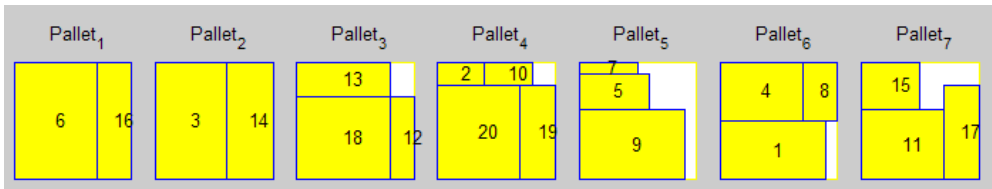
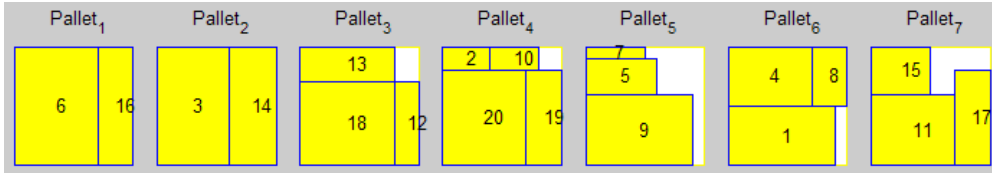


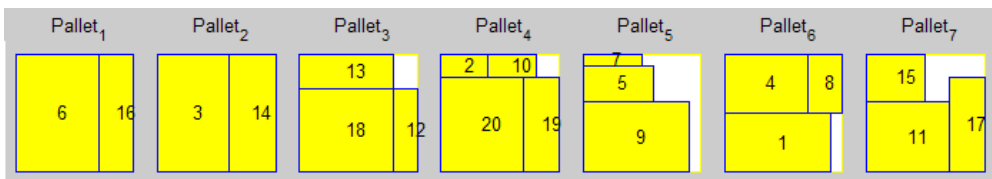
Figure 1: Maximal spaces in two dimensions



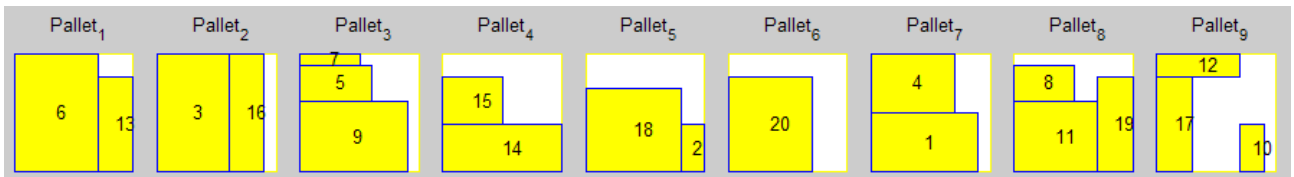
(a) $\delta = 0.01$, objective function value = 7.078.



(b) $\delta = 0.05$, objective function value = 7.39.

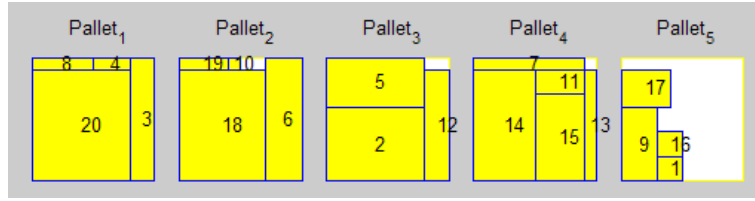


(c) $\delta = 0.1$, objective function value = 7.78.

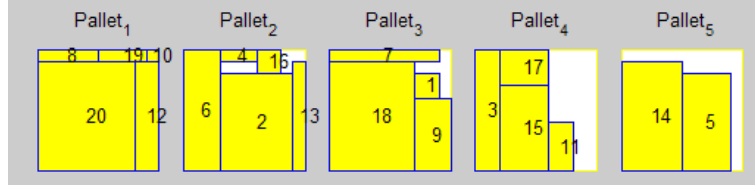


(d) $\delta = 0.5$, objective function value = 9.30.

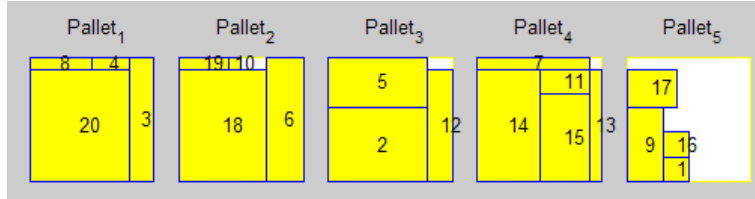
Figure 2: Exemplary packing results achieved by IMA_A subject to different weights of δ



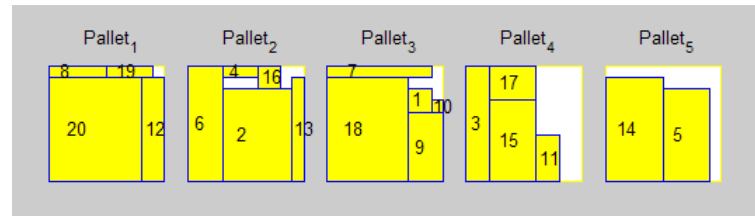
(a) IMA, objective function value = 5.096.



(b) IMA_A, objective function value = 5.064.



(c) TS, objective function value = 5.096



(d) IMA_A_LS, objective function value = 5.05.

Figure 3: Exemplary packing results achieved by the four algorithms with $\partial = 0.01$

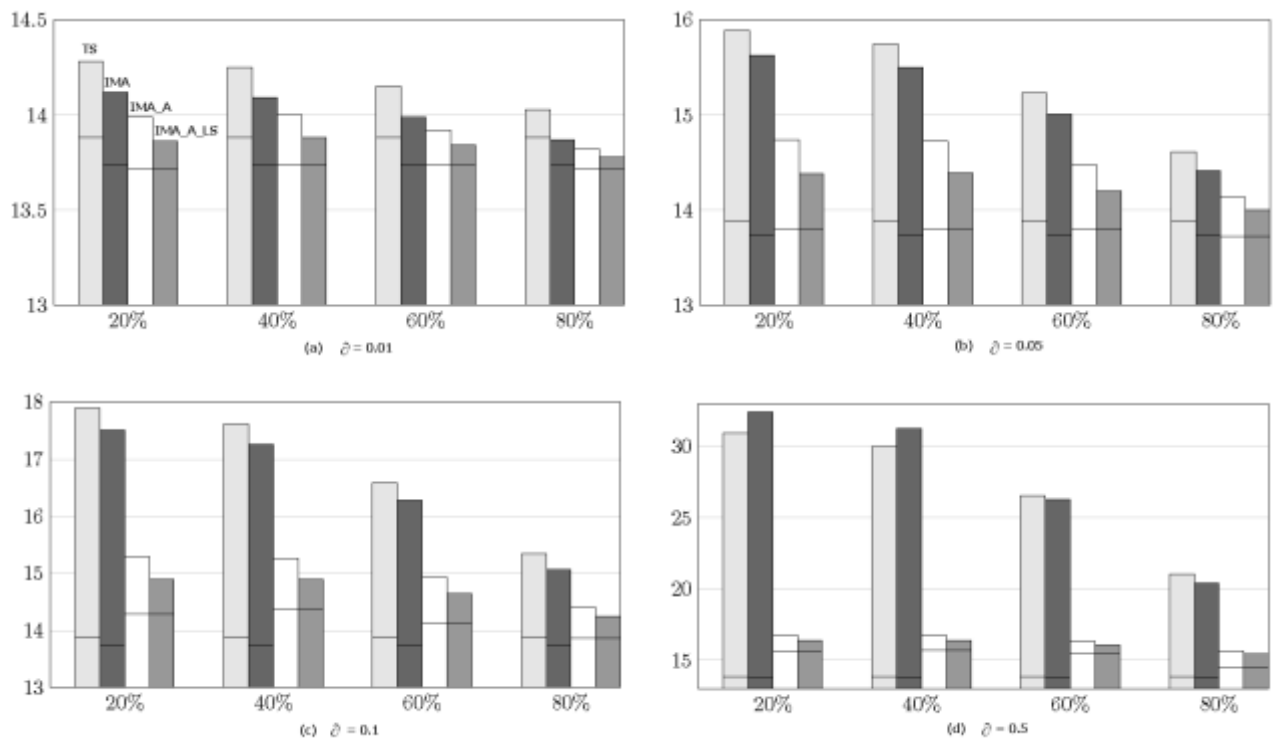


Figure 4: Impacts of grouping structure on the performance of the four algorithms