Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 9, No. 3, pp. 391–412, July 2013

SOLVING TWO-DIMENSIONAL BIN PACKING PROBLEMS WITH TWO-STAGE GUILLOTINE CUTTING BY COMBINED LOCAL SEARCH HEURISTICS

T. M. Chan, Filipe Alvelos, Elsa Silva and J. M.Valério de Carvalho

Abstract: In this paper, a new efficient algorithm named combined local search heuristics which comprise two local search heuristics, Variable Neighborhood Descent (VND) and Random Neighbor Selection (RNS), is designed and proposed to solve two-dimensional guillotine bin packing problems. The objective of these problems is to pack smaller pieces of rectangular items into large rectangular bins without overlapping such that the total number of used bins is minimized. A constructive heuristic (CH) is conceived to construct a solution by packing items into bins with the use of a defined item packing sequence. VND and RNS, which consist of three deterministic neighborhood structures and three random neighbor selection operators, respectively, are used for improving a solution given by the CH. Benchmark instances were adopted to verify the effectiveness of the designed algorithm via computational experiments. Computational results show that, in terms of the quality of solutions, the proposed approach is better than other heuristics and meta-heuristics. In terms of computational times, the proposed algorithm cannot be compared to other algorithms and the computational experiments cannot offer enough evidence of showing any good running-time behavior of the proposed algorithm because of different models of computers used. However, from a practical point of view, easy implementation and reasonable and affordable computational times confirm the usefulness of the proposed algorithm.

Key words: *two-dimensional guillotine bin packing problems, local search heuristics, variable neighborhood descent (VND), random neighbor selection (RNS), optimization*

Mathematics Subject Classification: *90C27, 90C59*

1 Introduction

Bin packing problems are real-world combinatorial optimization problems which are frequently tackled by different industries. To cite some examples, paper and metal industries aim at cutting all required smaller rectangular sheets of material by using the fewest larger rectangular sheets. Also, in newspaper and magazine offices, rectangular advertisements and articles need to be properly placed within fixed size rectangular pages.

The objective of these problems is to pack a given set of rectangular items in an unlimited number of identical rectangular bins such that the total number of used bins is minimized obeying three constraints: (i) all items must be packed in bins, (ii) items cannot overlap, and (iii) the edges of items are parallel to those of the bins. Items should be packed into bins based on the required cutting method which can be a free cutting or guillotine cutting. A free cutting means that the cutting does not have any restriction and an example of this cutting is depicted in Figure 1. The sheet can be arbitrarily cut to separate the twelve

ISSN 1348-9151

individual rectangles without waste attached. However, a guillotine cutting signifies the one from an edge of the rectangle to the opposite edge. A guillotine cutting which is applied n times is referred to as an n-stage guillotine cutting. For a two-stage guillotine cutting problem (i.e. $n = 2$), horizontal cuts are applied in the first stage, and then vertical cuts are applied in the second stage. Regarding a three-stage guillotine cutting problem (i.e. $n = 3$), the same operations in the two stages aforementioned are performed and then horizontal cuts are adopted in the third stage. Figures2 illustrates examples of two-stage and three-stage guillotine cutting problems. In the former problem, three strips are produced via horizontal cuts in the first stage, and all items are segregated for each strip via vertical cuts in the second stage. In the latter problem, two strips are generated by means of horizontal cuts in the first stage, and stacks are separated for each strip through vertical cuts in the second stage. Finally, all items and waste are partitioned via horizontal cuts in the last stage.

In order to separate items from waste, trimming (i.e. further horizontal cuts) may be used after the last stage. Accordingly, both two-stage and three-stage problems can be categorized to two different cases, (i) non-trimming (i.e. exact case) and (ii) trimming (i.e. non-exact case), as portrayed in Figure 3. Suppose that there are 13 and 23 items packed in bins for the two-stage and three-stage problems, respectively. Since trimming is not allowed for the two-stage exact case shown in Figure 3(a), the height of each item in the same shelf must be identical. However, such restriction is not applied in the non-exact case shown in Figure 3(b), and consequently the height of each item in the same shelf can be different. Similarly, as trimming is not permitted for the three-stage exact case shown in Figure 3(c), the width of each item in the same stack must be equivalent. Nevertheless, this rule is not used in the non-exact case shown in Figure 3(d) and therefore the width of each item in the same stack can be distinct. Note that the leftmost item in each shelf is also a stack which determines the height of the shelf.

The possible search methods to solve bin packing problems as a type of combinatorial optimization problems are exact methods, heuristics methods and meta-heuristics methods [16, 17, 4, 22]. Although exact methods guarantee that an optimal solution can be found, the difficulty of obtaining an optimal solution increases drastically if the problem size increases. Thus, for large instances, using heuristics or meta-heuristics methods may be seen as a more appropriate alternative when the goal is to attain a good-quality solution in a reasonable amount of time.

In this study, the Single Bin Size Bin Packing Problem (SBSBPP) [23] with two-stage guillotine cutting, trimming (i.e. non-exact case) and fixed orientation items is considered. Also, a new efficient algorithm called combined local search heuristics which comprise two local search heuristics, Variable Neighborhood Descent (VND) and Random Neighbor Selection (RNS), is designed and proposed to improve a solution given by a constructive heuristic (CH). CH contains a number of simple procedures for constructing a solution by packing items into bins with the use of a defined item packing sequence. Different combinations of criteria for packing items into existing shelves and bins must be set before CH is implemented. Then, CH is executed M times to obtain M solutions where M is the total number of combinations of criteria. The best of M solutions is chosen as the initial (current) solution, and only the corresponding combination of criteria will be employed when the VND and RNS are subsequently implemented. VND and RNS embrace three deterministic neighborhood structures called swap adjacent item types, swap adjacent item subsequences, and reverse item subsequences and three random neighbor selection operators called cut-and-paste, split, and swap blocks, respectively. These neighborhood structures and neighbor selection operators are all adopted to try to enhance the quality of the current solution by modifying the current item packing sequence. After a new sequence is obtained, the CH is

executed again to generate a new solution. If the new solution outperforms the current solution, the current solution is replaced by the new. Otherwise, the current solution remains unchanged.

This paper is organized as follows. Section 2 offers a literature review of different heuristics used in solving various bin packing problems. Sections 3 and 4 provide the details of the CH, and VND and RNS respectively. Sections 5 and 6 present the computational results and discussions, and conclusions respectively.

2 Literature Review

Some related work about application of heuristics to solve bin packing problems and about local search heuristics to solve other problems will be presented as follows.

The following studies are the ones using various heuristic methods to tackle various bin packing problems. Four heuristic algorithms, Finite next-fit (FNF), Finite first-fit (FFF), Finite best-strip (FBS), and Finite bottom-left (FBL), were proposed by Berkey and Wang [1] to attack the two-dimensional guillotine bin packing problem without allowing item rotations: (i) FNF is a level-oriented next-fit heuristic that packs directly into finite bins, (ii) FFF is a level-oriented first-fit heuristic that packs directly into finite bins, (iii) FBS is a best-fit heuristic that is similar to hybrid first-fit, which is a two-step algorithm in which the pieces are packed into an infinite height-strip and the strip is divided into blocks, which are then packed into finite bins, and (iv) FBL is a bottom-left approach which packs directly into finite bins.

A new hybrid heuristic algorithm called hybrid first fit (HFF) was suggested by Chung et al. [6] to cope with two dimensional bin packing problems. Creating a strip packing for the set of rectangular items by utilizing the first-fit decreasing height heuristic algorithm is the first step of HFF. Then, a collection of blocks of non-increasing heights, each of which comprises a subset of rectangular items, is obtained. These blocks can be viewed as a new collection of rectangles and the first-fit decreasing heuristic algorithm can be adopted to pack those blocks into the bins.

A new heuristic based on a best-fit algorithm was proposed by Hayek et al. [12] to deal with the two-dimensional bin packing problem with free cutting and item rotations. In an iteration of the heuristic, a list of unpacked items and a list containing all the maximal areas available in the opened bin are considered at each stage of packing an item. Also, the decisions of determining an item to be packed and maximal area are made at each step of item packing.

The floor-ceiling algorithm was conceived by Lodi et al. [14] to attack the two-dimensional guillotine bin packing problem with item rotations. This algorithm used a special packing strategy that items are packed not only from left to right while their bottom edges touch the floor level, but also from right to left while their top edges touch the ceiling level, that is, the horizontal line drawn on the top of the tallest item packed on the floor.

Heuristic algorithms and a meta-heuristic approach, tabu search, were proposed by Lodi et al. [15] to solve each of four variants of two-dimensional bin packing problems. These four variants are: (i) the items cannot be rotated and guillotine cutting is required, (ii) the items may be rotated and guillotine cutting is required, (iii) the items cannot be rotated and cutting is free, and (iv) the items may be rotated and cutting is free. The first two heuristic algorithms for the first two variants rely on the idea that shelf packings are determined by solving a series of 0-1 knapsack problems. The heuristic algorithm for the third variant is based on the idea of exploiting the feasibility of non-guillotine patterns by packing the items into the bins in alternate directions, i.e., from left to right, then from right to left in the

lowest possible position, and so on. The last one relies on the idea that the first item is always packed in the bottom-left corner of a bin and each subsequent item is packed in a so-called normal position, i.e., with its bottom edge touching either the bottom of the bin or the top edge of another item, and with its left edge touching either the left edge of the bin or the right edge of another item. Moreover, the feature of the proposed tabu search is the use of a unified parametric neighborhood, whose size and structure are dynamically varied during the search.

A set-covering-based heuristic was suggested by Monaci and Toth [20] to tackle the two-dimensional bin packing problem with free cutting and without allowing item rotations. Greedy procedures and fast constructive heuristic algorithms were used to generate a small subset of columns rather than the entire set in the first phase of the proposed approach. Then, a Lagrangian-based heuristic algorithm was used to solve the associated set-covering instance in the second phase.

A newly proposed heuristic method named HBP was proposed by Boschetti and Mingozzi [3] to deal with the two-dimensional bin packing problems with three various cases, (i) some items can be rotated by 90° and cutting is free, (ii) the items are oriented and cutting is free, and (iii) the items may be rotated by 90° and cutting is free. The idea of the heuristic approach is that the items are ordered by using a given criterion, and then allocated by considering a bin at the time. When no more items can be allocated to the current bin, this bin is closed and a new one is opened. The process stops when all items are allocated. This procedure is then repeated with changing the order of the items.

The work related to VND will be reviewed as follows. Pan et al. [21] proposed a discrete particle swarm optimization algorithm with VND to solve the no-wait flowshop scheduling problem with both makespan and total flowtime criteria. Two different neighborhood structures named “Swap” and “Insert” were adopted in the VND. The former one means two jobs are swapped while the latter one means a job is removed from one dimension and inserted into another dimension.

Gao et al. [8] tackled the flexible job shop scheduling problem by using a hybrid genetic algorithm. Three objectives were considered: (a) minimize the makespan of the jobs, (b) minimize the maximal machine workload, and (c) minimize the total workload. Local search of moving one operation and that of moving two operations were the two neighborhood structures used in the VND. Moving an operation was achieved by deleting the operation, searching an assignable time interval, and allocating it to that interval.

Gendron et al. [9] suggested a parallel hybrid heuristic comprising slope scaling and VND to deal with the multicommodity capacitated location problem with balancing requirements. The objective is to minimize the costs incurred by moving flows of commodities through the network to satisfy the supplies at origins and the demands at destinations. The first neighborhood structure used in the VND is the move of closing a depot and the second one is the move of simultaneously closing a depot and opening another one.

Borgulya [2] solved the 3-boolean satisfiability problem by using three evolutionary algorithms with VND. The objective is to determine the assignment of variables to satisfy all clauses. Three neighborhood structures were proposed for the VND. The first one is bit-flip which means a single bit (variable) is randomly flipped in the descendant. The second one is bit-flip-flop which means that, if two variables corresponding to two randomly selected bits have different values in the descendant, these two bits will be flipped. The last one is bit-repair which sequentially chooses all clauses which are not satisfied yet and flips a single uniformly selected variable in the chosen clause.

Degila and Sanso [7] dealt with the topological design problem of a yottabit-per-second multidimensional lattice network by employing tabu search with VND. The objective is

to find the arrangement of the edge nodes in the lattice structure to minimize the traffic weighted mean hop distance between the origin and the destination edge nodes. There are three neighborhood structures used in VND, (a) d-move which swaps nodes within an agile core (AC) of the d^{th} dimension, (b) $d_1 - d_2$ move which exchanges nodes belonging to an AC of the ${d_1}^{th}$ dimension with those belonging to an AC of the ${d_2}^{th}$ dimension, (c) ac-move which swaps two ACs.

Liang and Wu [13] proposed a VND algorithm to solve the redundancy allocation problem. The objective is to choose the optimal combination of components and redundancy levels to maximize the system reliability while satisfying the system level constraints, i.e. cost and weight constraints. Four neighborhood structures were adopted in the VND. The first one is to simultaneously replace one type of component with a different type within the same subsystem. The second one is to change the number of a particular component type by either adding or subtracting one. The third one is to simultaneously delete one component in a subsystem and add one component to another subsystem. The last one is to simultaneously exchange components in any two subsystems.

3 Constructive Heuristic

In this section, a CH for solving the two-dimensional guillotine bin packing problems is presented. The design of this heuristic is simple but effective and can satisfy the guillotine-cutting constraint. The idea of the heuristic is based on sorting item types for defining an initial packing sequence and on iteratively trying to pack items into existing shelves or bins according to the following criteria. For the former one, the three criteria considered to sort the item types in the descending order are: (i) by width, (ii) by height, and (iii) by area. For the latter one, it is possible that more than one existing shelf or bin can accommodate each item. Therefore, criteria should be established to determine which existing shelf and bin should be selected for packing the items. The three criteria used to achieve this purpose are: (i) minimize the residual width after packing the item, (ii) minimize the residual height after packing the item, and (iii) minimize the residual area after packing the item. The residual width, residual height and residual area are defined as $w_r - w_i$, $h_r - h_i$, and $a_r - a_i$ respectively where w_r, h_r, a_r are the width, height and area of a rectangular free space respectively and w_i, h_i, a_i are the width, height and area of an item to be packed in the rectangular free space respectively. Note that different criteria can be adopted for shelve and bin selection.

The pseudo code of the CH is given in Figure 4. The first step of the heuristic is to define a packing sequence for items by sorting the item types based on a specified criterion. Then, iteratively, each item is packed into an existing shelf which minimizes a specified criterion. If this is not possible, we try to pack it into a used bin which minimizes a specified criterion. If this is not possible again, it is placed into a new bin. Finally, a solution is obtained by running the heuristic with several sets of criteria and then selecting the best one among the obtained solutions. The time complexity of the CH is $O(N \log N + M \times F)$ where N is the total number of item types, M is the total number of items, and F is the total number of free rectangles in which items will be placed. All combinations of the three aforementioned criteria were used in the experiments.

Figure 5 illustrates an example of the CH. Assume that the items are sorted by height and four items have already been loaded into the bin. The item 5 currently being considered can be loaded into an existing shelf (free space C or D), on the top of shelves of the bin (free space E) or a new empty bin. The place where the item is packed depends on the pre-selected criterion and the type of the problem tackled.

Suppose that a two-stage problem with trimming is now being solved. In this case, only three possible choices of empty spaces, C, D and E, are available for packing item 5. If the criterion (i) is set, it will be packed into C to fulfil the criterion; if the criterion (ii) is selected, it will be packed into D; if the criterion (iii) is chosen, it will be placed into D.

3.1 Solution Representation

It is crucial to realize the solution representation and evaluation function for executing the CH and devising neighborhood structures for VND and random neighbor selection operators for RNS. A solution is represented by a sequence of items satisfying the demand of each item. Let us consider an example as follows. Assume that items 1, 2, 3 have demands 3, 5, 2 respectively. A feasible solution is represented by 2, 2, 2, 2, 2, 1, 1, 1, 3, 3. The first five items of type 2 will be firstly packed. Next, three items of type 1 will be loaded, and finally the last two items of type 3. Note that the solution representation before implementing any neighborhood structure and random neighbor selection operator is the same as it is afterwards. Moreover, in order to examine the quality of a solution, an evaluation function f is required for evaluating the solution, which is given by

$$f = n_b \times a_b + o_b - m \tag{3.1}$$

where n_b is the number of used bins, a_b is the area of a bin, o_b is the occupied area of the bin in solution with the smallest occupied area, and m is the number of items packed in the same bin. While the first two terms are measured in area, the last one does not represent an area. The reason is that the evaluation function is used to distinguish between solutions with the same area, but it is easier to empty a bin in one solution (the solution where there are more number of items packed in the less occupied bin) than in another one. Also, the function can work appropriately for instances such that the widths and heights of the bins and items are small (e.g. smaller than 0.01). In this case, the last term should be multiplied by a constant which is less than 1 in order to ensure that a solution with a smaller number of used bins is always better than a solution with a greater number of used bins. The rationale behind this function is that a solution with fewer used bins always outperforms another solution with more used bins and that if two solutions have the same number of used bins, the solution where it is easier to empty one used bin is better than another one.

4 Combined Local Search Heuristics

Combined local search heuristics are proposed to improve a solution given by the CH, which consist of two local search heuristics, VND and RNS. VND is a meta-heuristic proposed by Hansen and Mladenovic [11, 10, 19]. The concept of VND is to systematically use different neighborhood structures. The brief description of VND is given as follows. First, a set of neighborhood structures which will be adopted in the descent is chosen and an initial solution is found. Now, an iteration loop starts. The first neighborhood structure is used to search the best neighbor of the current solution. If the best neighborhood solution outperforms the current solution, the latter is replaced by the former. Otherwise, the next (second, third, and so on) neighborhood structure is considered. Finally, the iteration loop repeats until all neighborhood structures are used. VND used in this study embraces three neighborhood structures named “swap adjacent item types”, “swap adjacent item subsequences”, and “reverse item subsequences”. It is a sequential VND which is different from the common one described above, because the three neighborhood structures, arranged in the fixed order, are implemented one by one.

RNS is adopted in a way similar to VND. The differences between RNS and VND are that RNS (i) impose the restriction of using all random neighbor selection operators rather than deterministic or mixed ones, and (ii) use the fixed number of iterations to explore better neighbor solutions for each of all neighbor selection operators. In each RNS operator, instead of finding the best neighbor of the current solution by complete enumeration, only one neighbor is randomly generated in each time and then compared with the current solution. The advantage of this modification is that a large computational load is not required to search all neighbors of the current solution, especially when the problem size is huge. RNS adopted in this study comprises three neighbor selection operators titled "Cut-and-Paste", "Split-and-Redistribute", and "Swap Block". Like those in VND, these three neighbor selection operators arranged in the fixed order are implemented one by one.

Figure 6 depicts the pseudo code of the combined local search method. The three neighborhood structures of VND are firstly executed, followed by the three RNS operators. In the following, the details of the three neighborhood structures in VND and three neighbor selection operators in RNS will be given.

4.1 Swap Adjacent Item Types

"Swap adjacent item types" is aimed at swapping all items in two adjacent item types. Figure 7 illustrates an instance of this neighborhood structure. Assume that two highlighted adjacent item types, 2 and 4, are chosen. One item in the first type and three items in the second type are exchanged to produce the new solution.

4.2 Swap Adjacent Item Subsequences

The mission of "swap adjacent item subsequences" is to swap two adjacent item subsequences, both of which have the same size. A size parameter is used to define the size of the neighborhood. Figure 8 portrays an instance of this neighborhood structure for the size of the neighborhood, 2. Two shaded adjacent item subsequences (1, 3) and (3, 2) are selected in the current solution and then swapped to produce the new solution.

4.3 Reverse Item Subsequences

The objective of "reverse item subsequences" is to reverse the order of an item subsequence with a given size. A size parameter is used to define the size of the item subsequence. Figure 9 shows an example of this neighborhood structure for the size of the item subsequence, 3. The item subsequence (3, 2, 4) is selected, and then their packing orders are reversed to generate a new solution.

4.4 Cut-and-Paste

"Cut-and-Paste" was initially proposed as a genetic operation which was applied in the Jumping-Gene paradigm to solve multi-objective optimization problems [5]. In this implementation, the "jumping" element is cut from an original position and pasted into a new position of a chromosome. However, since this operation can also be used as a random neighbor selection operator, in this study, "Cut-and-Paste" is applied to the solution in a way that there is only one "jumping" segment in the solution, and the length, original position and new position of the "jumping" segment are randomly chosen. Figure 10 depicts an example of this neighbor selection operator. Given that the randomly generated length is 4, the original position is 6 and the new position is 2. In other words, the highlighted

segment (4, 2, 1, 4) is randomly selected. The segment is cut from the original position and then pasted into the new position to complete the operation.

4.5 Split-and-Redistribute

The objective of " Split-and-Redistribute " is to split various blocks with a given length from the solution and redistribute them to the solution. The total number, length, original positions, and new positions of the blocks are randomly selected and all blocks have the same length. Figure 11 illustrates an instance of this neighbor selection operator. Suppose that the randomly selected total number is 3, the length is 2, the original positions are 1, 5 and 10, and the new positions are 10, 1 and 5 for the three blocks respectively. That is, the three highlighted blocks (1, 3), (3, 4), and (4, 4) are randomly chosen. These three blocks are split and then redistributed to their new positions to acquire the new solution.

4.6 Swap Block

"Swap Block" is aimed at exchanging two different blocks with a given length in the solution. The length and the positions of the two blocks are randomly selected and the lengths of these two blocks are equivalent. Figure 12 portrays an example of this neighbor selection operator. Assume that the randomly selected length is 3 and the randomly chosen positions are 4 and 8. That means that the two highlighted blocks (2, 3, 4) and (1, 4, 4) are randomly selected. Then, these two blocks are swapped to produce the new solution.

5 Computational Results

In order to show the effectiveness of the proposed method (i.e. CH + combined local search heuristics), it is compared with other approaches, FFF, FBS, Knapsack Packing (KP), TS with FBS, TS with KP, CH + pure VND, and CH + pure RNS. The computational results of the first five algorithms are obtained from the study [15] and the computer program for these algorithms was written in FORTRAN 77 and run on a Silicon Graphics INDY R10000sc 195MHz. The computer program for the last two algorithms and the suggested method was written in C++ and run on a PC with 2.00 GHz Intel CoreTM 2 CPU with 1 GB memory. Six classes of instances named B&W from the Berkey and Wang study [1] and four classes of instances named M&V from the Martello and Vigo study [18] were adopted. Each class contains five subclasses, each of which has ten instances and therefore there are a total of 300 instances and 200 instances respectively. Also, five values of total number of items are considered in each class: 20, 40, 60, 80 and 100. Assume that W and H are the bin width and height respectively, w_i and h_i are the width and height of item i respectively, $i = 1, 2, \ldots, n$, where n is the total number of items. The information of these ten classes is given as follows:

B&W - Class I: w_i and h_i are uniformly and randomly generated between [1, 10], $W = H = 10$.

B&W - Class II: w_i and h_i are uniformly and randomly generated between [1, 10], $W = H = 30$.

B&W - Class III: w_i and h_i are uniformly and randomly generated between [1, 35], $W = H = 40$.

B&W - Class IV: w_i and h_i are uniformly and randomly generated between [1, 35], $W = H = 100$.

B&W - Class V: w_i and h_i are uniformly and randomly generated between [1, 100], $W = H = 100$.

B&W - Class VI: w_i and h_i are uniformly and randomly generated between [1, 100], $W = H = 300$.

For the M&V instances, the items are classified into four types:

Type 1: w_i is uniformly and randomly generated between $[2/3\,W,\ W]$, h_i is uniformly and randomly generated between $[1,\ 1/2H]$.

Type 2: w_i is uniformly and randomly generated between $[1,\ 1/2\,W]$, h_i is uniformly and randomly generated between $[2/3H,\ H]$.

Type 3: w_i is uniformly and randomly generated between $[1/2\,W,\ W]$, h_i is uniformly and randomly generated between $[1/2H,\ H]$.

Type 4: w_i is uniformly and randomly generated between $[1,\ 1/2\,W]$, h_i is uniformly and randomly generated between $[1,\ 1/2H]$.

M&V - Class VII: Type 1 with probability 70%, type 2, 3, 4 with probability 10% each, $W = H = 100$.

M&V - Class VIII: Type 2 with probability 70%, type 1, 3, 4 with probability 10% each, $W = H = 100$.

M&V - Class IX: Type 3 with probability 70%, type 1, 2, 4 with probability 10% each, $W = H = 100$.

M&V - Class X: Type 4 with probability 70%, type 1, 2, 3 with probability 10% each, $W = H = 100$.

This study considered the scenario two-stage guillotine cutting, trimming (i.e. non-exact case) and fixed orientation items. The maximum time limit for executing VND is 300 seconds. If three consecutive new solutions generated by any neighborhood structure cannot improve the current solution, VND will terminate and RNS will start next. In addition, in pure RNS and in the proposed method, the order of neighbor selection operators used in the computational tests is the same as that described in the previous Section. The total number of iterations used is 500 for each RNS operator. Since both pure RNS and the proposed approach adopt RNS operators, 30 runs were conducted to obtain the best, average, and worst solution in each problem of each subclass in order to allow a statistical analysis.

Table 1 shows the computational results of various approaches in terms of the ratio of heuristic solution to lower bound for all 50 subclasses. z is a heuristic solution value and LB is a lower bound value. The lowest ratio is bold-faced in each subclass. By comparing the z/LB of the leftmost six methods with the best z/LB of pure RNS and the proposed approach, it can be seen that the proposed approach is the best performer, obtaining lower ratios in a total of 29 subclasses. By comparing the z/LB of TS with FBS to the best z/LB of the proposed method, TS with FBS and the proposed method obtained lower ratios in a total of 15 and 17 subclasses respectively while they acquired the same ratios in a total of 18 subclasses. It shows that the proposed method is slightly better than TS with FBS.

Furthermore, by comparing (i) the best z/LB, (ii) average z/LB and (iii) worst z/LB of the proposed approach to those of CH + pure RNS, the proposed approach and CH + pure RNS got lower ratios in a total of 7 and 4 subclasses respectively while they obtained the same ratios in a total of 49 subclasses for the case (i). For the case (ii), the proposed approach and CH + pure RNS acquired lower ratios in a total of 22 and 2 subclasses respectively while they obtained the same ratios in a total of 36 subclasses. For the case (iii), the proposed

approach and CH + pure RNS got lower ratios in a total of 27 and 0 subclasses respectively while they obtained the same ratios in a total of 33 subclasses. Although the results acquired by the proposed approach is a bit similar to those by CH + pure RNS, the combined local search is necessary and useful because it can produce lower ratios in more subclasses than CH + pure RNS with only small additional computational times.

Overall, the results reveal that the deterministic neighborhood structures and RNS operators of the proposed approach are able to explore many different neighbor solutions that increase the chance of finding promising solutions. The deterministic neighborhood structures in the proposed method can locate a better starting solution for the RNS operators to further enhance the quality of the solution, and thus it has a slight improvement over both CH + pure VND and CH + pure RNS.

Besides, Table 2 shows the computational times of various approaches. Note that the leftmost five methods from the past studies and the remaining methods were run in different computers with different models. Although the exact computational times of FFF, FBS, and KP were not given by the study [15], it indicated that their computational times did not exceed 0.5 seconds for all subclasses and thus ≤ 0.5 are shown for all subclasses for these three algorithms. The computational times of CH + pure RNS and of the proposed approach in each subclass are obtained by summing the total time of 30 runs per instance for all 10 instances and then dividing the sum by 10. As FFF, FBS, KP and CH + pure VND contain simple procedures, their computational times are short. Similarly, CH + Pure RNS also comprises simple procedures. However, the cut-and-paste and split-and-redistribute operations in RNS have slightly larger computational load than other operations in VND in terms of written codes. Therefore, the computational times of CH + pure RNS are longer than those of CH + Pure VND. It can be observed that, as expected, the proposed method, CH + combined local search heuristics, requires longer computational time than CH + pure VND and CH + pure RNS because both VND and RNS operators were used. Although TS with FBS and TS with KP have the largest computational times, they were run on a computer which is different from the one used in this paper. Thus, the proposed algorithm cannot be compared to those two algorithms in terms of computational times and the computational experiments cannot offer enough evidence of showing any good running-time behavior of the proposed algorithm.

However, from a practical point of view, the usefulness of the proposed algorithm is that the constructive heuristic, VND and RNS in the proposed algorithm are easy to be implemented compared with procedures in other algorithms proposed in the past. Although they are not as simple as heuristics like FBS, the quality of solutions obtained by the proposed algorithm is better with little extra computational load. Also, good solutions can be acquired by the proposed algorithm with reasonable and affordable computation times.

Finally, the components inside the proposed approach, CH + combined local search heuristics, are analyzed and discussed in the following. Regarding the constructive heuristic, among all combinations of the three mentioned criteria used in the experiments, the three combinations, (a) the item types are sorted in the descending order of item height, (b) minimize the residual width after packing the item to the shelf, and (c) minimize the residual width, height or area after packing the item to the used bin, are able to generate better solutions than other combinations. These three combinations acquire the lowest number of used bins with the lowest function value in 312 instances out of a total of 500 instances.

Table 3 shows the averages of the total number of loop iterations executed in the VND. Since each subclass contains ten instances, each average value is obtained by averaging the ten values of ten instances and then rounding off. The averages of the total number of loop iterations are ranged from 7 to 15. As these values are not large, the computational load is

not high and thus the proposed method is efficient with short computational times.

Table 4 lists the averages of the total number of improved solutions produced by the three different operations in the VND. Each average value is obtained by averaging the ten values of ten instances and then rounding off. These average values are ranged from 3 to 7. It can be observed that the three operations have similar effectiveness to improve solutions. Even though the averages of the total number of improved solutions are not large for these three operations, they can be used to increase the chance of improving solutions by testing different combinations of solution components which other algorithms and operations cannot search.

Table 5 shows the averages of the total number of improved solutions produced by the three different operations in the RNS. Each average value is obtained by averaging the ten values of ten instances and then rounding off. These average values are ranged from 3 to 8. All of these three operations are able to improve solutions and the operation, Cut-and-Paste, is slightly better than the other two operations, Split-and-Redistribute and Swap Block. The reason may be that Cut-and-Paste can search different solutions with various order-like combinations. Again, although the averages of the total number of improved solutions are not large for these three operations, they can be used to increase the chance of improving solutions by testing different combinations of solution components which other algorithms and operations cannot search.

6 Conclusions

In this paper, a new efficient algorithm named combined local search heuristics comprising VND and RNS has been studied. The objective of employing the proposed approach is to further improve the solution given by the CH. Three deterministic neighborhood structures and three RNS operators were designed in the proposed approach. The quality of solutions found by the proposed approach is quite good, as it obtains better performance ratios in a total of 29 out of 50 subclasses, outperforming three heuristics, two tabu search meta-heuristics and CH + pure VND. By comparing the best, average and worst results of the proposed approach to those of CH + pure RNS, the former is more stable and robust.

References

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

[2] I. Borgulya, An evolutionary framework for 3-SAT problems, in *Proceedings of the 25th International Conference on Information Technology Interfaces*, Cavtat, Croatia, Jun. 16-19, 2003, pp. 471–476.

[3] M.A. Boschetti and A. Mingozzi, Two-dimensional finite bin packing problems. Part II: New lower and upper bounds, *4OR* 1 (2003) 135–147.

[4] M.W. Carter and C.C. Price, *Operations Research: A Practical Introduction*, CRC Press, Boca Raton, 2001.

[5] T.M. Chan, K.F. Man, S. Kwong and K.S. Tang, A jumping gene paradigm for evolutionary multiobjective optimization, *IEEE Transactions on Evolutionary Computation* 12 (2008) 143–159.

[6] F.K.R. Chung, M.R. Garey and D.S. Johnson, On packing two-dimensional bins, *SIAM Journal on Algebraic Discrete Methods* 3 (1982) 66–76.

[7] J. R. Degila and B. Sanso, Topological design optimization of a yottabit-per-second lattice network, *IEEE Journal on Selected Areas in Communications* 22 (2004) 1613–1625.

[8] J. Gao, L. Sun and M. Gen, A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, *Computers & Operations Research* 35 (2008) 2892–2907.

[9] B. Gendron, J.Y. Potvin and P. Soriano, A parallel hybrid heuristic for the multicommodity capacitated location problem with balancing requirements, *Parallel Computing* 29 (2003) 591–606.

[10] P. Hansen and N. Mladenovic, An introduction to variable neighborhood search, in *Meta-heuristics, Advances and Trends in Local Search Paradigms for Optimization*, S. Voss et al. (eds.), Kluwer Academic Publishers, Boston, 1999, pp. 433–458.

[11] P. Hansen and N. Mladenovic, Variable neighborhood search: Principles and applications, *European Journal of Operational Research* 130 (2001) 449–467.

[12] J.E. Hayek, A. Moukrim and S. Negre, New resolution algorithm and pretreatments for the two-dimensional bin-packing problem, *Computers & Operations Research* 35 (2008) 3184–3201.

[13] Y.C. Liang and C.C. Wu, A variable neighbourhood descent algorithm for the redundancy allocation problem, *Industrial Engineering & Management Systems* 4 (2005) 94–101.

[14] A. Lodi, S. Martello and D. Vigo, Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem, in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss et al. (eds.), Kluwer Academic Publishers, Boston, 1998, pp. 125–139.

[15] A. Lodi, S. Martello and D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, *INFORMS Journal on Computing* 11 (1999) 345–357.

[16] A. Lodi, S. Martello and M. Monaci, Two-dimensional packing problems: A survey, *European Journal of Operational Research* 141 (2002) 241–252.

[17] A. Lodi, S. Martello and D. Vigo, Recent advances on two-dimensional bin packing problems, *Discrete Applied Mathematics* 123 (2002) 379–396.

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

[19] N. Mladenovic and P. Hansen, Variable neighborhood search, *Computers & Operations Research* 24 (1997) 1097–1100.

[20] M. Monaci and P. Toth, A set-covering-based heuristic approach for bin-packing problems, *INFORMS Journal on Computing* 18 (2006) 71–85.

[21] Q.K. Pan, M.F. Tasgetiren and Y.C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers & Operations Research* 35 (2008) 2807–2839.

[22] S.M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*, IEEE Computer Society, Los Alamitos, CA, 1999.

[23] G. Wäscher, H. Haubner and H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research* 183 (2007) 1109–1130.

Manuscript received 31 October 2011
revised 28 February 2012, 6 June 2012, 26 June 2012
accepted for publication 27 June, 2012

T. M. Chan
Centro de Investigação Algoritmi, Universidade do Minho
4710-057 Braga, Portugal
E-mail address: tmchantmchan@hotmail.com

Filipe Alvelos
Centro de Investigação Algoritmi, Universidade do Minho
4710-057 Braga, Portugal; and
Departamento de Produção e Sistemas, Universidade do Minho
4710-057 Braga, Portugal
E-mail address: falvelos@dps.uminho.pt

Elsa Silva
Centro de Investigação Algoritmi, Universidade do Minho
4710-057 Braga, Portugal
E-mail address: elsa@dps.uminho.pt

J. M.Valério de Carvalho
Centro de Investigação Algoritmi, Universidade do Minho
4710-057 Braga, Portugal; and
Departamento de Produção e Sistemas, Universidade do Minho
4710-057 Braga, Portugal
E-mail address: vc@dps.uminho.pt