Solving the Two-Dimensional Bin-Packing Problem with Variable Bin Sizes by Greedy Randomized Adaptive Search Procedures and Variable Neighborhood Search

Andreas M. Chwatal[1] and Sandro Pirkwieser[1,2]

[1] Destion – IT Consulting OG, Vienna, Austria
[2] Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{chwatal|pirkwieser}@destion.at

Abstract. In this work we present new metaheuristic algorithms to a special variant of the two-dimensional bin-packing, or cutting-stock problem, where a given set of rectangular items (demand) must be produced out of heterogeneous stock items (bins). The items can optionally be rotated, guillotine-cuttable and free layouts are considered. The proposed algorithms use various packing-heuristics which are embedded in a greedy randomized adaptive search procedure (GRASP) and variable neighborhood search (VNS) framework. Our results for existing benchmark-instances show the superior performance of our algorithms, in particular the VNS, with respect to previous approaches.

1 Introduction

In this work we consider a special variant of a two-dimensional bin packing problem where a finite number of bins of different dimensions is given (stock objects), and a set of different two-dimensional rectangular items must be packed into (a subset of) these bins. This problem obviously has many practical applications, e.g. in wood, glass and metal production, where a given demand must be produced from a heterogeneous set of stock items. We consider situations where the obtained layouts must be guillotine-cuttable, which means that it must be possible to cut the items from a stock sheet (bin) by only straight slices, as well as free packing layouts. The items are allowed to be rotated by 90 degrees.

More formally, we are given a set of two-dimensional objects (items) $I = \{1, \ldots, i_{\max}\}$ with dimensions $w_i \times h_i$, for all $i \in I$ and a set of stock objects or bins $B = \{1, \ldots, b_{\max}\}$ with dimensions $w_i \times h_i$, for all $i \in B$. For each bin $b \in B$ we are further given costs $c_b \in \mathbb{N}$. We assume the instances to be feasible, i.e. a feasible packing exists for the given set of items and bins. The optimization goal is to find a feasible packing with minimum costs of the used bins. For a comparison to previous work we also use the utilization percentage, i.e. the total area of all items compared to the area of all used bins, as optimization criterion.

2 Related Work

The two-dimensional cutting/packing problem defined in Sec. 1 is a natural extension of the well known (one-dimensional) bin packing (BP), and cutting-stock (CS) problem. Both problems are closely related to each other, the only difference is that the assortment of particular items (of a specific size) is usually assumed to be small in the context of BP problems, whereas it is usually assumed to be large with regard to CS problems. According to the classification of Dyckhoff [1], we consider a problem of type 2/V/D/M in this work, meaning that we consider 2-dimensional items and stock objects, the items should be assigned to a selection of the stock objects, we are given a heterogeneous stock, and many items of many different shapes are given. According to the extended typology of Wäscher et al. [2] our problem variant can be classified as type 2/V/Ss/S, i.e. we have a strongly heterogeneous assortment according to stock objects (bins) and items. A further recent classification scheme of cutting and packing problems can be found in [3].

The problem considered in this work is $\mathcal{NP}$-hard, as it is a generalization of the BP or CS problem, which are both well known to be $\mathcal{NP}$-hard [4]. Surveys of related work are given in [5, 6].

In [7, 8] the application of metaheuristics to a problem variant similar to the one considered in this work, however with uniform bin costs, has been investigated. In [9] the authors propose an exact algorithm based on column generation for the problem variant with variable costs and bin sizes and an unbounded number of each bin type. Related work can also be found within the cutting-stock literature, e.g. [10] or [11].

3 Construction Algorithms

For the free packing layout we implemented the algorithm *Bottom Left Fill* (BLF) used in [7, 8] which is based on the *Bottom-Left* algorithm introduced in [12]. The algorithm iteratively adds the items to the current bin, trying to place each item bottom-left aligned at the first possible insertion position. For this purpose the algorithm maintains a list of possible insertion regions, which is updated after each iteration. These possible insertion regions then are given by the upper-left and lower-right coordinates of the last added item according to its position and are ranging to the upper-right corner of the bin. The algorithm is also capable of filling holes in the layout obtained by previous insertion steps. This may on the one hand yield denser packings in contrast to the bottom-left algorithm, but on the other hand the possible insertion regions have to be checked for feasibility and possibly updated when a new item is inserted. When a feasible insertion region is found, further alignment operations (shifts) are performed. Details of the algorithm can be found in [7, 8], an illustrative example is given in Fig. 1.

For the guillotine-cuttable layouts we implemented a new level-packing algorithm, which we call *advanced level packing* (ALP). In contrast to many other

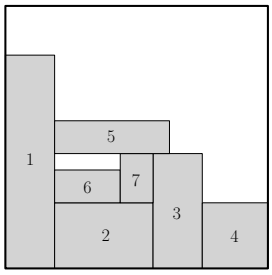

Fig. 1. BLF-packing, with a sequence sorted w.r.t. decreasing area. The hole after the insertion of item 5 is then filled with items 6 and 7.

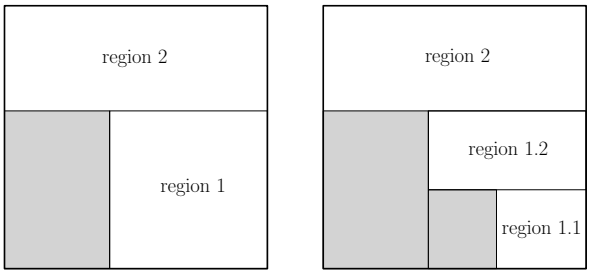

Fig. 2. This figure shows how possible insertion regions are split recursively by ALP after the addition of further items to obtain a better utilization of the space to the right of particular items.

level-packing approaches, we do, however, split the remaining space of one level into two possible insertion regions, each one acting as a new "level" themselves. This leads to a better utilization of the space to the right of the items which initialized a level, i.e. which have previously been packed at the left-most position of the level. The drawback of this approach is a longer running time, as usually more possible insertion regions have to be checked for each item.

Both construction algorithms can be used with various sortings of the input sequence, i.e. decreasing height, decreasing area, decreasing perimeter etc. Results for various settings are shown in Sec. 5. Further both algorithms (try to) rotate each item by 90 degrees with a probability of 0.5 in order to introduce more diversity, which generally is of advantage when improving the solutions later on.

4 Metaheuristics

In this section we present our solution approaches based on *greedy randomized adaptive search procedures* (GRASP) [13] and *variable neighborhood search* (VNS) [14]. Although minimizing the total bin costs $\sum_{b \in B_u} c_b$, where B_u is the set of used bins, is our primary objective, we use an additional measure denoted as *packing score* to also distinguish solutions w.r.t. to their packing, similar to [7, 8]: $\left(\sum_{b \in B_u} \left(A_b^{items}/A_b^{bin}\right)^2\right)/|B_u|$, where A_b^{items} is the items' area in bin b, and A_b^{bin} is the area of bin b itself. Whenever two solutions have equal bin costs we prefer those yielding a higher packing score, hence favoring solutions with a denser packing.

4.1 A GRASP-like algorithm

In the first step of GRASP a randomized solution construction is performed, with the goal to produce good starting-solutions for a subsequent local search.

These steps are then performed iteratively. The randomized construction is usually performed by utilizing a *restricted candidate list* (RCL), with the purpose of limiting the number of meaningful extension candidates in each step to a small number. In our case, however, there is no need to explicitly use an RCL, as the construction algorithms consistently find solutions of reasonable quality, even with random ordering of the items and bins. The placement of the item is then performed in a deterministic way, into the first feasible, or best available insertion region. A set of solutions constructed by these randomized construction heuristics is sufficiently diverse, and furthermore their quality is good enough to act as reasonable starting point for the subsequent local search.

As local search method we use a *variable neighborhood descent* (VND) algorithm. VND systematically changes the neighborhoods – usually using a predefined order – since a local optima for a given neighborhood is not necessarily one for another neighborhood. Whenever an improvement is found the process starts over beginning with the first neighborhood again, otherwise the next neighborhood is selected. Following three neighborhood structures are applied in the given order:

1. **Use cheaper bin:** Consecutively all used bins are considered and an empty bin having less costs is sought for which offers potentially enough space to occupy the items. If such a pair of bins is found it is tried to actually re-pack the items into the newly selected bin.
2. **Clear one bin:** The used bins are considered in the order of increasing utilization (i.e. starting with the least-filled bins), one bin is "emptied" via unassigning the packed items and the latter are re-inserted in the partial solution.
3. **Clear two bins:** Similar to the previous neighborhood structure but emptying two bins, whereas the second bin is taken from the bin order currently in use (one of the pre-defined sorting criteria).

The first neighborhood structure directly aims at replacing improperly chosen bins, while the successive emptying and re-inserting is expected to yield denser packed bins and hence occasionally allows using one bin less.

The only algorithmic parameter for the GRASP-like algorithm is the number of iterations it_{G}, i.e. independent solution construction and improvement phases.

4.2 Variable Neighborhood Search

VNS applies random steps in neighborhoods of increasing size for diversification in order to escape local optima, referred to as *shaking*, and uses an embedded local search component for intensification. Over time it has been successfully applied to a wide range of combinatorial optimization problems. We apply a general VNS, i.e. we utilize the proposed VND for performing the local improvement. We use two different neighborhood structures for shaking:

– **Swap items:** For two randomly selected used bins it is tried to swap a given number of items. We proceed by selecting the item sets s.t. an exchange move

Table 1. Detailed shaking neighborhood order used in the VNS.

k	$\mathcal{N}_k$
1	exchange one item (or clear one bin as fallback)
2	clear one bin
3	exchange two items (or clear two bin as fallback)
4	clear two bins
5	exchange three items (or clear three bins as fallback)
6	clear three bins

is potentially possible (w.r.t. the area offered by the bins). If such sets are found several re-packing trials are performed (consistent limit of 20).

– **Clear bins:** Similar to the variants used in the VND, except that the bins to be emptied are selected at random.

The actually applied shaking neighborhoods based on these neighborhood structures are given in Tab. 1. Since swapping of items might not yield a feasible packing (either because it is not possible at all and/or the heuristic is unable to find it) as a fallback strategy clearing of bins is applied otherwise.

5 Experimental Results

The algorithms have been implemented in C++, compiled with gcc-4.4 and executed on a single core of an Intel i7 860 @ 2.80 GHz computer with 8 GB of RAM. The test instances are from [7, 8] and made available via the OR-Library[3]. These instances basically feature bins having uniform costs since the area of the bins directly corresponds to its cost, in this case minimizing the costs equals maximizing the utilization. To also investigate the interesting case of dealing with non-uniform costs we modified the bins' data to reflect this (denoted by the subscript NUC). Thereby the modified costs are not chosen completely random but lie on average within 20% of the initial costs. For each setting of bin sizes (unregarded the costs) there are five different item sets which will, however, in the following be treated as one problem setting. The number of items as well as bins are the following: 100 and 16 for setting 1, 100 and 18 for setting 2, and 150 and 20 for setting 3.

In order to also compare our CPU runtimes to those of [7, 8] we used a factor of 1/100 according to `http://www.cpubenchmark.net` for their Pentium II 350 MHz processor. For the construction heuristics we performed 1000 runs, for all other methods 100 runs. The results on the instances with uniform cost bins are shown in Tab. 2, those for non-uniform cost bins in Tab. 3. We state the maximal and average bin utilization as well as the minimal and average bin costs for the corresponding setting. Also the average runtime is given in milliseconds. In the upper half of the tables results for free packing layouts are shown, those for the guillotine-cuttable variant in the lower half. For BLF and ALP

[3] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpacktwoinfo.html`

Table 2. Average results on instances with bins having uniform costs.

	M1			M2			M3		
	max	avg	avg time [ms]	max	avg	avg time [ms]	max	avg	avg time [ms]
BLF-R,R [7, 8]	92.7	–	–	88.0	–	–	90.7	–	–
BLF-DA,R [7, 8]	97.8	–	39	93.3	–	44	94.6	–	153
Sim. Ann. [7, 8]	–	97.9	8560	–	94.8	9360	–	95.0	18640
BLF-R,R	93.5	85.2	2	90.9	82.4	3	92.3	85.8	5
BLF-DA,R	98.4	92.3	2	95.7	88.9	2	96.3	90.8	5
BLF-DA,DA	91.6	91.6	2	93.1	92.4	2	92.6	92.1	5
VND-1	98.4	97.6	4	95.4	94.8	5	95.7	95.3	8
VND-2	98.4	97.6	16	95.9	95.0	20	96.3	95.4	46
VND-3	98.4	97.7	102	95.9	95.1	130	96.3	95.5	480
GRASP-1	98.4	97.8	46	95.4	95.4	64	95.7	95.7	69
GRASP-2	98.4	97.8	186	95.9	95.6	235	96.3	95.8	409
GRASP-3	98.4	97.8	1270	96.1	95.6	1448	96.3	95.8	4779
VNS-1	98.4	97.6	346	95.9	95.0	270	96.0	95.4	532
VNS-2	98.4	97.7	510	96.5	95.1	405	96.8	95.6	862
VNS-3	98.4	97.7	2563	96.5	95.5	2264	96.8	96.1	8433
ALP-R,R	92.2	83.0	2	89.9	80.6	2	90.3	83.6	5
ALP-DA,R	97.6	91.2	2	95.4	87.7	2	96.0	90.1	5
ALP-DA,DA	91.6	91.6	2	93.1	90.9	2	92.6	91.6	4
VND-1	97.6	97.3	3	95.2	93.9	6	95.7	94.6	7
VND-2	97.6	97.4	15	95.6	94.1	23	95.7	94.8	45
VND-3	98.4	97.4	85	95.6	94.2	136	96.0	95.0	446
GRASP-1	98.4	97.7	45	95.4	95.0	86	95.7	95.5	81
GRASP-2	98.4	97.8	202	96.1	95.1	304	96.3	95.6	518
GRASP-3	98.4	97.7	1182	96.1	95.1	1802	96.6	95.6	4516
VNS-1	97.6	97.3	309	95.4	94.0	293	96.0	94.7	471
VNS-2	98.4	97.5	427	95.6	94.3	395	96.0	95.0	795
VNS-3	98.4	97.6	2085	95.9	94.7	2123	97.1	95.8	7230

we explicitly state the applied sorting criteria for the items as well as the bins: R=random (shuffle), DA=decreasing area, IC=increasing costs, IRC=increasing relative costs (i.e. c_b/A_b^{bin}), where the latter two are only applicable for bins and are used in case of non-uniform bin costs. VND-x, GRASP-x, as well as VNS-x are the corresponding variants only using the VND neighborhoods up to x as stated in Section 4.1. For GRASP we set $it_G = 10$. Preliminary results suggested to always use the first feasible region for insertion.

In Tab. 2 we also contrast our results of the free packing layouts to those of [7, 8] where the simulated annealing performed best, also stating their BLF results for completeness. Unfortunately they did not report all relevant data for all methods. However, it is clear that our solution approaches outperform them in all aspects but the average utilization for setting M1 where a small gap remains. Anyway, probably most important from a potential users perspective is that the maximal utilization (corresponding to minimal costs) is improved in all cases. Looking at the average performance of BLF and ALP we decided to use the sorting combination (DA, DA) for uniform costs and (DA, IRC) for non-uniform

Table 3. Average results on instances with bins having non-uniform costs.

	$M1_{NUC}$			$M2_{NUC}$			$M3_{NUC}$		
	min	avg	avg time [ms]	min	avg	avg time [ms]	min	avg	avg time [ms]
BLF-DA,DA	2972.0	2972.3	2	2880.0	2905.2	2	4872.0	4893.2	5
BLF-DA,IC	2700.0	2775.1	2	2974.0	3048.2	3	5086.0	5259.8	5
BLF-DA,IRC	2720.0	2737.0	2	2880.0	2884.3	3	4302.0	4305.1	5
VND-1	2544.0	2598.3	5	2880.0	2880.8	3	4262.0	4269.6	6
VND-2	2544.0	2586.6	14	2790.0	2860.1	13	4262.0	4269.5	51
VND-3	2544.0	2570.9	159	2754.0	2823.3	240	4250.0	4267.4	775
GRASP-1	2516.0	2545.6	51	2862.0	2879.1	36	4262.0	4262.0	64
GRASP-2	2544.0	2548.0	153	2760.0	2809.5	139	4262.0	4262.0	516
GRASP-3	2544.0	2544.0	1839	2754.0	2773.4	2834	4246.0	4258.4	7721
VNS-1	2524.0	2574.9	185	2734.0	2821.5	187	4212.0	4270.4	362
VNS-2	2524.0	2547.3	347	2734.0	2782.2	408	4142.0	4254.8	921
VNS-3	2506.0	2525.8	2519	2688.0	2749.3	3509	4094.0	4172.2	10876
ALP-DA,DA	2972.0	2972.0	2	2880.0	2955.0	2	4872.0	4916.7	4
ALP-DA,IC	2700.0	2885.6	2	2974.0	3079.0	3	5086.0	5347.6	5
ALP-DA,IRC	2720.0	2755.7	2	2880.0	2925.5	2	4302.0	4363.1	5
VND-1	2544.0	2643.3	6	2880.0	2889.0	5	4262.0	4321.1	7
VND-2	2544.0	2629.5	15	2790.0	2853.1	15	4262.0	4310.8	52
VND-3	2544.0	2594.8	149	2772.0	2819.6	215	4246.0	4286.1	635
GRASP-1	2544.0	2557.4	41	2880.0	2880.0	30	4262.0	4264.8	68
GRASP-2	2544.0	2549.4	132	2772.0	2818.0	124	4262.0	4264.7	555
GRASP-3	2544.0	2545.9	1431	2736.0	2772.5	2139	4236.0	4258.3	6968
VNS-1	2524.0	2606.4	189	2744.0	2832.7	187	4246.0	4317.8	366
VNS-2	2524.0	2577.5	377	2732.0	2797.8	385	4186.0	4281.1	846
VNS-3	2524.0	2529.3	2487	2704.0	2759.2	3255	4110.0	4204.2	9498

costs within the metaheuristics. It is observable that generally it pays off to use one of the metaheuristics instead of only the construction heuristics, and the runtimes are still acceptable. VNS with full VND yields the best results in general for both cost types, followed by the GRASP-like approach which performs similar in case of uniform bin costs. Hence it seems that shaking is beneficial when dealing with non-uniform cost bins. Also notable is the overall marginal difference for the respective objective between free layout and guillotine-cuttable packing, although the latter imposes a considerably constraint in principle. Finally, also a statistical comparison of the methods is given in Tab. 4 applying a Wilcoxon rank sum test with an error level of 5%; basically confirming what was reported before.

6 Conclusions

In this work we presented new metaheuristic algorithms for a special variant of the two-dimensional bin-packing problem. Several configurations including different packing-heuristics, neighborhoods and parameters of VNS and GRASP have been experimentally tested on existing benchmark-instances. On these datasets improved results regarding average solution quality and running-times have been obtained. Hence, these algorithms, in particular the VNS, are found to be very suitable for the considered packing problem, and can likely also successfully be applied to many other variations.

Table 4. Results of pairwise statistical significance tests on all 30 instances (6×5) with free layout packing stating how often method 1 (row) is significantly better/worse than method 2 (column).

	VND-1	VND-2	VND-3	GRASP-1	GRASP-2	GRASP-3	VNS-1	VNS-2	VNS-3
BLF	0/24	0/29	0/30	0/26	0/29	0/30	0/30	0/30	0/30
VND-1	–	0/7	0/12	0/18	0/24	0/25	0/10	0/21	0/26
VND-2		–	1/9	4/16	0/22	0/24	0/8	0/18	0/25
VND-3			–	5/15	1/20	0/24	7/4	1/17	0/24
GRASP-1				–	1/8	0/13	16/6	12/12	4/18
GRASP-2					–	0/9	21/2	13/8	4/17
GRASP-3						–	23/1	18/4	4/16
VNS-1							–	0/18	0/24
VNS-2									0/25

References

1. Dyckhoff, H.: A typology of cutting and packing problems. European Journal of Operational Research **44**(2) (1990) 145 – 159
2. Wascher, G., Hausner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**(3) (2007) 1109–1130
3. Ntene, N.: An Algorithmic Approach to the 2D Oriented Strip Packing Problem. PhD thesis, University of Stellenbosch, South Africa (2007)
4. Garey, M.R., Johnson, D.S.: "Strong" NP-completeness results: Motivation, examples, and implications. Journal of the ACM **25** (1978) 499–508
5. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research **141**(2) (2002) 241–252
6. Lodi, A., Martello, S., Vigo, D.: Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics **123**(1-3) (2002) 379–396
7. Hopper, E., Turton, B.C.H.: An empirical study of meta-heuristics applied to 2d rectangular bin packing - part i. Studia Informatica Universalis **2**(1) (2002) 77–92
8. Hopper, E., Turton, B.C.H.: An empirical study of meta-heuristics applied to 2d rectangular bin packing - part ii. Studia Informatica Universalis **2**(1) (2002) 93–106
9. Pisinger, D., Sigurd, M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optimization **2**(2) (2005) 154–167
10. Alvarez-valdes, R., Parajon, A., Tamarit, J.M.: A computational study of heuristic algorithms for two-dimensional cutting stock problems. In: MIC'2001 Metaheuristics International Conference. (2001)
11. Cintra, G., Miyazawa, F., Wakabayashi, Y., Xavier, E.: Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. European Journal of Operational Research **191**(1) (2008) 61–85
12. Chazelle, B.: The bottom-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers **32**(8) (1983) 697–707
13. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**(2) (1999) 109–133
14. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11) (1997) 1097 – 1100