On Solving the Oriented Two-Dimensional Bin Packing Problem under Free Guillotine Cutting: Exploiting the Power of Probabilistic Solution Construction

Article · September 2012

Source: arXiv

Christian Blum
Spanish National Research Council

Verena Schmid
Bayer Business Services, Leverkusen, Germany

On Solving the Oriented Two-Dimensional Bin Packing Problem under Free Guillotine Cutting: Exploiting the Power of Probabilistic Solution Construction

Christian Blum[1] and Verena Schmid[2] and Lukas Baumgartner[2]

[1]ALBCOM Research Group
Universitat Politècnica de Catalunya, Barcelona, Spain
cblum@lsi.upc.edu

[2]Department of Business Administration
Universität Wien, Vienna, Austria
{verena.schmid,lukas.baumgartner}@univie.ac.at

Abstract

Two-dimensional bin packing problems are highly relevant combinatorial optimization problems. They find a large number of applications, for example, in the context of transportation or warehousing, and for the cutting of different materials such as glass, wood or metal. In this work we deal with the oriented two-dimensional bin packing problem under free guillotine cutting. In this specific problem a set of oriented rectangular items is given which must be packed into a minimum number of bins of equal size. The first algorithm proposed in this work is a randomized multi-start version of a constructive one-pass heuristic from the literature. Additionally we propose the use of this randomized one-pass heuristic within an evolutionary algorithm. The results of the two proposed algorithms are compared to the best approaches from the literature. In particular the evolutionary algorithm compares very favorably to current state-of-the-art approaches. The optimal solution for 4 previously unsolved instances could be found.

1 Introduction

Bin packing problems (BPPs) are well studied and highly popular combinatorial optimization problems. The main reason for their popularity is a large number of real-world applications. Moreover, in general they can be easily expressed in mathematical terms. In this work we deal with a specific variant of the two-dimensional bin packing problem (2BP), which consists in packing a set $\mathcal{Q} = \{1, \ldots, n\}$ of n rectangular items into a minimum number of bins of height H and width W such that items do not overlap. Each item $j \in \mathcal{Q}$ is characterized by its height h_j and its width w_j. Real world applications for the 2BP include, for example, cutting glass, wood or metal and packing in the context of transportation or warehousing (see [11, 24]).

According to Lodi et. al [15] there are four different cases of the 2BP as described above. The differences between these four cases are based on two aspects: (1) a rotation of 90° of the items may, or may not, be permitted; (2) guillotine cutting may be required or free. The four resulting problem versions are as follows:

- 2BP|O|G: Items are oriented and guillotine cutting is required.
- 2BP|O|F: Items are oriented and guillotine cuttings is free.
- 2BP|R|G: Items may be rotated by 90° and guillotine cutting is required.
- 2BP|R|F: Items may be rotated by 90° and guillotine cutting is free.

In this paper we exclusively focus on the 2BP|O|F version of the problem. Note that in the remainder of the paper the abbreviation 2BP will refer to this problem version. Concerning the complexity of the 2BP, Garey and Johnson classified the problem as NP-hard (see [10]).

1.1 Existing Work

In general, different versions of the 2BP have been tackled in the literature by means of different integer programing models, heuristics, and exact algorithms. A good overview on the early work regarding the 2BP can be obtained from [16, 17, 13, 7]. In the following we will focus on existing heuristics as well as metaheuristics.

1.2 Heuristics

Concerning heuristics, the literature mainly distinguishes between *one-phase* and *two-phase approaches*. One-phase algorithms pack the items directly into the bins, whereas two-phase algorithms first pack the items into levels of one infinitely high strip with width W and then stack these levels into the bins. Level-packing algorithms place items next to each other in each level. Hereby, the bottom of the first level is the bottom of the bin. For the next level the bottom is a horizontal line coinciding with the highest item of the level below. Note that, items can only be placed besides each other in each level, in contrast to packing items on top of each other.

Well known *level-packing algorithms* are Next-Fit Decreasing Height (NFDH), First-Fit Decreasing Height (FFDH) and Best-Fit Decreasing Height (BFDH) [5]. These strategies were originally developed for the one-dimensional bin packing problem, but have also been adapted to strip packing problems and for the application to the two-dimensional case. All three heuristics require the items to be sorted by non-increasing height, which represents the order in which they are packed. Moreover, they pack the items into one bin of infinite hight.

Next, *two-phase level-packing algorithms* are shortly described. Hybrid Next-Fit (HNF) (see [9]) is based on NFDH, Hybrid First-Fit (HFF) [4] on FFDH and Finite Best-Strip (FBS) [2], which is also sometimes referred to as Hybrid Best-Fit, is based on BFDH. The first phase of all three algorithms consists in the execution of the heuristic on which they are based. This produces in each case a set of levels, which must then be packed into bins of finite hight. This is done by using the same strategy as the one that was used for the packing of the items into levels. Another example for a two-phase level-packing algorithm is Knapsack Packing (KP) [15]. Phase one of KP consists in packing the levels by solving knapsack problems. Hereby, the tallest unpacked item, say j, initializes each new level. The remaining horizontal distance up to the right bin border $(W - w_j)$ is taken as the capacity of the knapsack problem to be solved. Moreover, the width w_i of any unpacked item i is regarded as its weight, while the items' area $w_i \cdot h_i$ is regarded as its value (or profit). This procedure is repeated until all items are packed into levels. In the second phase the remaining one-dimensional bin packing problem is solved by using a heuristic such as Best-Fit Decreasing or an exact algorithm. Finally, Floor Ceiling (FC) [15] can be seen as an improvement over FBS. Again, the first phase is used for packing items into levels, whereas these levels are packed into bins in the second phase.

Among the most important *one-phase non-level-packing algorithms* are Alternate Direction (AD) [15], Bottom-Left Fill (BLF) [1], Improved Lowest Gap Fill (LGFi) [25] and Touching Perimeter (TP) [15]. In the following we describe these techniques shortly. AD sorts the items by non-increasing height and initializes L bins, where L is a lower bound for the necessary number of bins. Afterwards the bottom of the bins are filled from left to right using a best-fit decreasing strategy. Then one bin after another is being filled. In this context items are packed in bands from left to right and from right to left until no more items can be packed into the current bin. BLF initializes bins by placing the first item at the bottom left corner. The top left and bottom right corners of already placed items are positions at which the bottom left corner of new items may potentially be placed. BLF tries to place the items starting from the lowest to the highest available position. When positions with an equal height are encountered, the position closer to the left is tried first. LGFi has a preprocessing and a packing stage. In the preprocessing stage, items are sorted by non-increasing area as a first criterion. Ties are broken by non-increasing absolute difference between height and width of the items. The packing stage starts by initializing a bin with the first unpacked item, which is placed at the bottom left corner. Then items are placed at the bottom leftmost position. If possible, an item is chosen such that either the horizontal gap or the vertical gap is filled completely. If this is not possible, the largest fitting item is placed at this position. This is repeated until all items are packed. TP, the last one-phase non-level-packing algorithm considered here, first sorts the items by non-increasing area and initializes L bins, where L is a lower bound for the number of necessary bins. Furthermore, depending on a specific position in the bin, a score is associated to each item: the percentage of the edges of the item touching either an edge of another item or the border of the bin. Each item is now considered for different positions in the bin and for each of these positions the corresponding score is calculated. Each item is then placed at the position at which its score is highest.

The best heuristic for the 2BP which is currently available (labelled SCH) is based on solving a set-covering formulation of the problem [20] by means of column generation. In the first phase, a rather small subset of all possible columns is generated by using greedy procedures and fast constructive heuristic algorithms from the literature. In the second phase, the resulting set-covering instance is solved by means of a Lagrangian-based heuristic.

In addition, some heuristics developed for three-dimensional packing can sometimes easily be applied to the 2BP. An example is the extreme point based heuristic from [6]. This heuristic uses extreme points to determine all points in the bin where items can be placed. Extreme points can either be corners of the already placed items or points generated by the extended edges of the placed items. These points are updated every time an item is placed into the bin. For placing the items a modified version of BFDH is used.

1.3 Metaheuristics

The earliest metaheuristic developed for the 2BP is *tabu search* (TS) [14, 16]. An initial solution is created using a heuristic such as FBS, KP, or AD. Moreover, neighborhood moves are based on trying empty certain bins by repacking their items into other bins.

A metaheuristic based on *guided local search* (GLS) has been presented in [8]. This metaheuristic has its origins in constraint satisfaction applications. GLS uses memory to guide the search process away from already explored regions of the search space. This is done by adding a penalty term to the objective function that penalizes bad solution features of previously visited solutions.

A rather simple metaheuristic, labeled HBP, based on a greedy heuristic has been proposed in [3]. HBP assigns a score to each item. Then, for the construction of a solution, the items are considered according to non-increasing values of the scores. After the construction of a solution the scores are updated using a certain criterion. This procedure is iterated until a pre-defined stopping criterion is met.

An approach labeled *weight annealing* (WA) for solving the 2BP was proposed in [18]. The WA technique can be seen as an extension of a greedy heuristic. Hereby, weights are assigned to different parts of the solution space. These weights are changed during the execution of the algorithm on the basis of the generated solutions. Moreover, they have an influence on the decisions of the greedy heuristic when constructing a new solution.

Finally, the currently best-performing metaheuristic is a hybrid between a greedy randomized adaptive search procedure (GRASP) and variable neighborhood descent (VND) [21]. The solution construction phase of GRASP is hereby based on a maximal-space heuristic from the field of container loading.

1.4 Contribution of this Work

In this paper we propose two algorithms based on a randomized version of the LGFi heuristic from the literature. First, a multi-start algorithm is developed. Second, our randomized version of LGFi is embedded into several operators of a comparatively simple evolutionary algorithm. Extensive computational experiments on publicly available benchmark instances show that both algorithms compare very favorably with the state of the art. In fact, the proposed multi-start algorithm and the evolutionary algorithm are able to solve 4 previously unsolved problem instances to optimality. Moreover, summing up the number of used bins concerning all 500 problem instances the evolutionary algorithm reaches a value of 7239, which is the best value reached by any algorithm that has been proposed for this problem.

1.5 Organization of the Paper

In Section 2 we first outline an ILP model for the tackled problem. The proposed algorithms are then presented in Section 3. Finally, an experimental evaluation is provided in Section 4, while conclusions and an outlook to the future are given in Section 5.

2 A New ILP Model

Inspired by the models proposed in [22] and [23] we present in the following an alternative ILP model for the 2BP. For this purpose, we denote by $\mathcal{Q} = \{1, \ldots, n\}$ the set of all items and the set of all bins. W and H refer to the bin-width and the bin-height, while w_i and h_i refer to the width and the height of item $i \in \mathcal{Q}$. W, H, w_i and h_i are all integer values.

The binary decision variable α_{ik} evaluates to 1 if item i is packed into bin k, and 0 otherwise. Only variables α_{ik} where $i \geq k$ are created so that only $\frac{n^2+n}{2}$ instead of n^2 have to be initialized. Furthermore items α_{kk} indicate if bins are opened or not. A bin is considered open if the item with the same index as the bin is placed in that bin. For example item 1 cannot be placed in bin 3 but only in bin 1. Item 3 can be placed in bin 3, in bin 2 in case item 2 is placed in bin 2, or in bin 1, which is always open as item 1 can only be placed in bin 1. It is easy to see that, even with this restricted variable set, all combinations of items packed into one bin are still possible. The integer variables x_i and y_i decide the x- and y-coordinates of each item within a bin. For the overlapping constraints, which we will introduce in the next paragraph, we need the binary variables $ul_{ij}, ua_{ij}, ur_{ij}$ and uu_{ij}. Each one of these four variables decides if item i has to be to the left (ul_{ij}), above (ua_{ij}), to the right (ur_{ij}) or underneath (uu_{ij}) item j. Only variables for $i < j$ are created so that only $\frac{n^2-n}{2}$ instead of n^2 have to be initialized for each variable. This can be done because if item i has to be to the left of item j, item j automatically has to be to the right of item i which makes it unnecessary to initialize the corresponding variable of item j.

$$
\begin{aligned}
& Z = \sum_{i=0}^{n} \alpha_{ii} \rightarrow min && && (1)\\
& \sum_{k=0}^{n} \alpha_{ik} = 1 && i, k \in \mathcal{Q}; i \geq k && (2)\\
& \alpha_{ik} \leq \alpha_{kk} && i, k \in \mathcal{Q}; i \geq k && (3)\\
& x_i + w_i \leq W && i \in \mathcal{Q} && (4)\\
& y_i + h_i \leq H && i \in \mathcal{Q} && (5)\\
& ul_{ij} + ua_{ij} + ur_{ij} + uu_{ij} = 1 && i, j \in \mathcal{Q}; i < j && (6)\\
& x_i + w_i \leq x_j + W \cdot (3 - ul_{ij} - \alpha_{ik} - \alpha_{jk}) && i, j, k \in \mathcal{Q}; k \leq i < j && (7)\\
& y_i + H \cdot (3 - ua_{ij} - \alpha_{ik} - \alpha_{jk}) \geq y_j + h_j && i, j, k \in \mathcal{Q}; k \leq i < j && (8)\\
& x_i + W \cdot (3 - ur_{ij} - \alpha_{ik} - \alpha_{jk}) \geq x_j + w_j && i, j, k \in \mathcal{Q}; k \leq i < j && (9)\\
& y_i + h_i \leq y_j + H \cdot (3 - uu_{ij} - \alpha_{ik} - \alpha_{jk}) && i, j, k \in \mathcal{Q}; k \leq i < j && (10)
\end{aligned}
$$

The objective function (1) minimizes the number of bins used. The constraint (2) ensures that each item is assigned to one and only one bin. That an item i can only be assigned to an open/initialized bin is ensured by (3). Constraints (4) and (5) ensure that each item is placed within the bin. Equation (6) states that item i has to be placed either to the left, above, to the right or underneath item j. The last four equations (7)-(10) ensure that two items do not overlap if assigned to the same bin.

3 The Proposed Algorithms

Both algorithms that we present in this paper are strongly based on heuristic LGFi, as developed by Wong and Lee in [25]. LGFi itself is an improved version of the LGF heuristic presented by Lee in [12]. Note that LGFi is a two-stage heuristic. In the *preprocessing stage* items are sorted into a list, while in the *packing stage* these items are packed from the list into bins. More specifically, in the preprocessing stage items are sorted by non-increasing area as a first criterion. Ties are broken by non-increasing absolute difference between height and width of the items. The packing stage is an iterative process in which the following actions are performed at each iteration. First, the bottom leftmost position at which an item may be placed is identified. This position is henceforth called the *current position*. Then, two gaps are calculated with respect to this position. The *horizontal gap* is defined as the distance between the current position and either the right border of the bin or the left edge of the first item between the current position and the right border of the bin. The distance between the current position and the upper border of the bin defines the value of the *vertical gap*. The value of the smaller gap is called *current gap*. The current gap is compared to either the widths of the items from the list of unpacked items, if the horizontal gap is the current gap, or to the heights of the items from the list of unpacked items, if the vertical gap is the current gap. The first item that fills the gap completely is placed with its bottom left corner at the identified position. If no such item exists, the first item which fits without any overlap is placed with its bottom left corner at the current position. If no such item exists either, some of the area must be declared *wastage area*, which works as follows. A wastage area with the width of the horizontal gap is created. The height of the wastage area is chosen as the height of the upper edge of the lowest neighboring item, or, if no neighboring items exists, as the height of the bin. Finally, if no current position can be found, and if unpacked items exist, a new bin is opened.

Example. Figure 1 shows the working of LGFi by means of a simple example. The left-hand side of each graphic shows the bin which is currently packed. The cross marks the current position, while the dotted lines show the horizontal and the vertical gap (indicated by hgap and vgap). The unpacked items are shown sorted from left to right at the right-hand side of each graphic. The body of each item shows its dimensions. In the initial situation (see Figure 1(a)), the current postion corresponds to the bottom left corner of the empty bin. As the current gap evaluates to 6, no item is able to fill the current gap completely. Therefore, the first item from the list is chosen and placed at the current position. After this first step (see Figure 1(b)), the current position is $(3, 0)$, and the current gap (as defined by the horizontal gap) evaluates to 3. The first item from the list which fills this gap completely is the second item (with dimensions 3×2). Therefore, this item is chosen and placed at the current position. The packing stage of LGFi proceeds in the same way until reaching the situation shown in Figure 1(f). The current position at this point is $(2, 3)$, and the current gap (corresponding to the horizontal gap) evaluates to 1. Unfortunately, the only remaining unpacked item does not fit without overlap at this position. Therefore, a wastage area must be declared. The width of this wastage space is equal to the horizontal gap. The height of the wastage space is 2, because after two vertical space units, the upper border of the neighboring item to the left is reached. Finally, as a last step, the last unpacked item is placed at position $(0, 5)$.

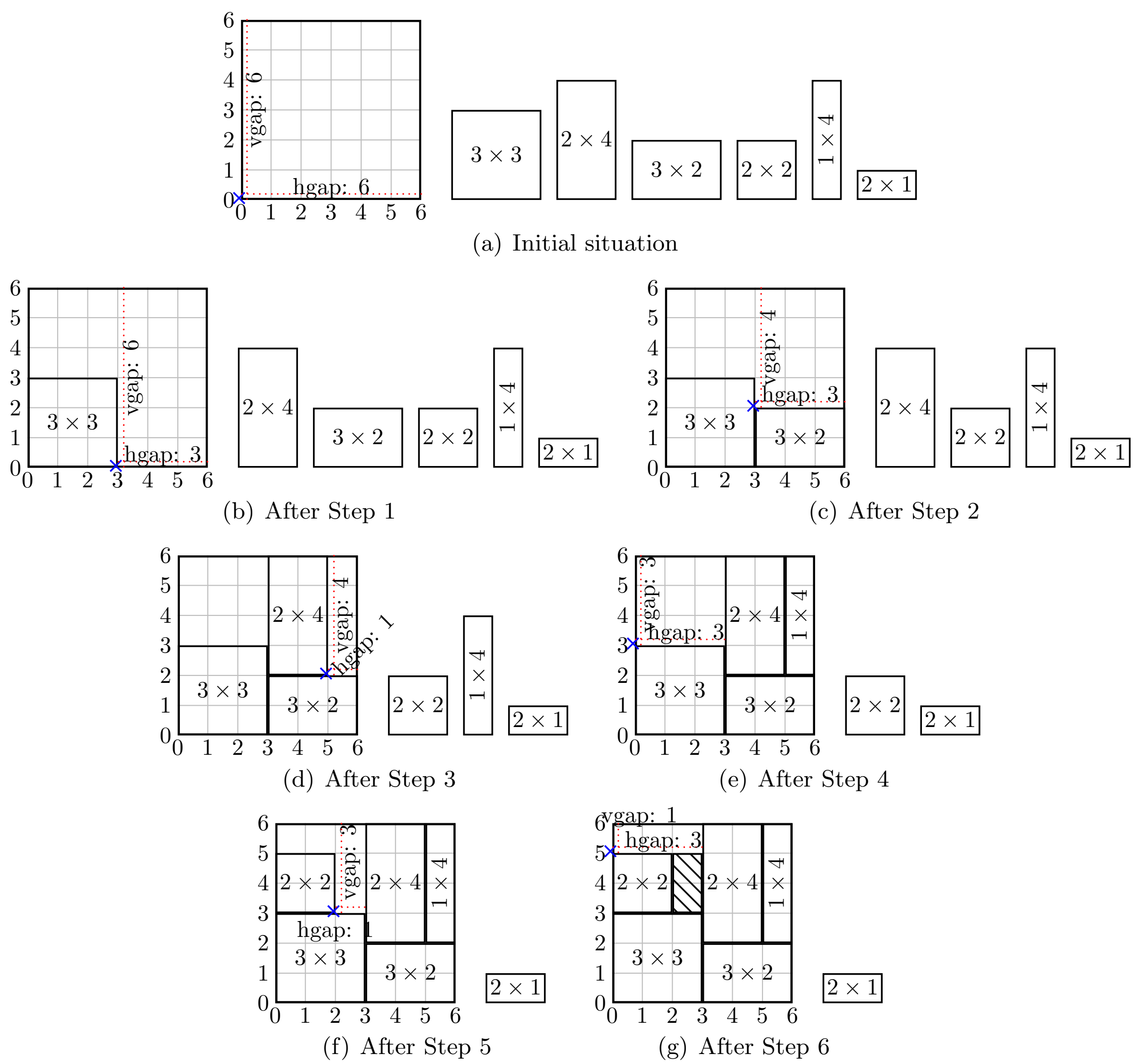

Figure 1: Example of the working of LFGi. (a) shows the initial situtation. Each subsequent subfigure shows the situation after placing one more item. In each subfigure the remaining items are shown to the right of the bin, ordered from left to right. The last step, which is not shown, consists in placing the last remaining item at position $(0, 5)$.

3.1 Multistart LGFi

The main idea of this paper is the use of the LGFi heuristic in a probabilistic way within the preprocessing stage. Our first approach is described in the following. As mentioned before, the preprocessing stage of LGFi generates an input sequence of all items. In this input sequence, items are ordered with respect to non-increasing area. In the following, pos_i refers to the position of an item i in this sequence. Multistart LGFi (MS-LGFi) works as follows. At each iteration, a new input sequence s is probabilistically generated on the basis of the original input sequence. Then, this new input sequence is provided to LGFi for the generation of the packing. At the end of the algorithm, the best found solution is provided as output.

In the following we explain the way in which a new input sequence s is generated based on the original input sequence. Remember that the total number of items is denoted by n. A value v_i is then assigned to each item i in the following way:

$$v_i := (n - pos_i)^{\kappa} \tag{11}$$

where $\kappa \geq 1$ is a parameter. The positions of s are filled from 1 to n in an iterative way. At each step, let $\mathcal{I} \subseteq \mathcal{Q}$ be the set of items that are not yet assigned to s. An item $i \in \mathcal{I}$ is chosen according to probabilities $\mathbf{p}(i|\mathcal{I})$ (for all $i \in \mathcal{I}$) by roulette-wheel-selection. These probabilities $\mathbf{p}(i|\mathcal{I})$ are calculated proportional to v_i:

$$\mathbf{p}(i|\mathcal{I}) = \frac{v_i}{\sum_{i \in \mathcal{I}} v_i} \tag{12}$$

Note that the larger parameter κ, the more similar the newly generated input sequence s will be to the original deterministic sequence.

3.2 Evolutionary Algorithm

The MS-LGFi algorithm, as proposed in the previous subsection, may have the disadvantage that no learning takes place over time. In other words, MS-LGFi may only find good input sequences for LGFi by chance. Moreover, once a good input sequence has been found, the knowledge about this sequence is forgotten at the end of the corresponding iteration. Therefore, we started to investigate if, for example, an evolutionary algorithm would is able learn good input sequences for LGFi. For this purpose the following evolutionary algorithm for the 2BP—henceforth labeled EA-LGFi—was devised.

A solution in the context of EA-LGFi is an input sequence s for LGFi. Note that s is an ordered list of all items that must be packed. The item at position j of this list (where $j = 1, \ldots, n$) is denoted by s_j. The function value $f(s)$ of a solution s is calculated by applying LGFi to s. The pseudo-code of EA-LGFi is shown in Alg. 1. The first step of EA-LGFi consists in generating the initial population of size p_{size} (see function $\mathsf{GenerateInitialPopulation}(p_{\text{size}}, \kappa)$). Then, at each iteration a crossover operator is applied in function $\mathsf{Crossover}(P, c_{\text{rate}}, \delta)$, recreating c_{rate} percent of the population. This provides a population P' with less than p_{size} solutions. The missing $p_{\text{size}} - |P'|$ solutions are generated by function $\mathsf{AddNewSolutions}(P', p_{\text{size}}, \kappa)$. In the following the three functions of algorithm EA-LGFi are outlined in more detail.

Algorithm 1 Evolutionary Algorithm for the 2BP (EA-LGFi)

input: values for parameters p_{size}, c_{rate}, κ and δ
$P :=$ GenerateInitialPopulation$(p_{\text{size}}, \kappa)$
while stopping criterion not met **do**
    $P' :=$ Crossover$(P, c_{\text{rate}}, \delta)$
    $P :=$ AddNewSolutions$(P', p_{\text{size}}, \kappa)$
end while
output: best solution found

GenerateInitialPopulation$(p_{\textbf{size}}, \kappa)$ In this function, p_{size} solutions are probabilistically generated in the same way as in MS-LGFi (see Section 3.1). Parameter κ is used for this purpose.

Crossover$(P, c_{\textbf{rate}}, \delta)$ This operator applies recombination to each of the best $\lfloor c_{\text{rate}} \cdot |P| \rfloor$ solutions of P, where $0 < c_{\text{rate}} \leq 1$ is a parameter of the algorithm. Extensive empirical tests have shown that a crossover rate $c_{\text{rate}} = 0.7$ works best for the instances at hand. For each solution s from the set of best $\lfloor c_{\text{rate}} \cdot |P| \rfloor$ solutions of P, a crossover parter $s^c \in P$ (such that $s^c \neq s$) is chosen from P by means of roulette-wheel-selection. Assume that P is an ordered list in which solutions are sorted according to their objective function values in a non-increasing manner. Ties are broken by the load of the last bin, that is, solutions with a lower load in the last bin are ordered first. Let $pos(s)$ denote the position of a solution s in P. The probability $\mathbf{p}(s^c|s)$ for a solution $s^c \neq s$ to be chosen as a crossover partner for solution $s \in P$ is as follows:

$$\mathbf{p}(s^c|s) := \frac{(p_{\text{size}} - 1 - pos(s^c))^{\delta}}{\sum_{s^o \in P, s^o \neq s} (p_{\text{size}} - 1 - pos(s^o))^{\delta}} \tag{13}$$

where $\delta \geq 1$ is a parameter of the algorithm. Given two crossover partners s and s^c, one offspring solution s^{off} is generated as explained in the following. First, three pointers (k, l and r) are initialized to the first position. Then, the n positions of s^{off} are filled from 1 to n as follows. If $s_k = s^c_l$ then $s^{\text{off}}_r := s_k$. In words, if position k of solution s and position l of solution s^c contain the same item, then this item is placed at position r of the offspring solution s^{off}. Next, position pointer r is incremented, and position pointers k and l are moved to the right until reaching the closest position containing an item which does not yet appear in solution s^{off}. In case $s_k \neq s^c_l$, the item for position r of solution s^{off} is chosen probabilistically among s_k and s^c_l, where a probability of 0.75 is given to the item originating from the better of the two solutions. Afterwards, the position pointer r is incremented. Moreover, the position pointer of the solution from which the item was selected is moved to the right until reaching the closest position containing an item which does not yet appear in solution s^{off}. The resulting solution s^{off} is evaluated by using it as input for LGFi. In case $f(s^{\text{off}}) < f(s)$ or $f(s^{\text{off}}) = f(s)$ and s^{off} has a lower load than s in the last bin, solution s^{off} is added to the new population P', otherwise solution s is added to P'.

AddNewSolutions$(P', p_{\textbf{size}}, \kappa)$ This function probabilistically generates $p_{\text{size}} - |P'|$ solutions in the same way as in MS-LGFi (see Section 3.1). Parameter κ is used for this purpose.

4 Experimental Evaluation

MS-LGFi and EA-LGFi were implemented in ANSI C++ using GCC 4.4 for compiling the software. The experimental results that we outline in the following were obtained on a PC with an AMD64X2 4400 processor and 4 Gigabyte of memory. The proposed algorithms were applied to a benchmark set of 500 problem instances from the literature. After an initial study of the algorithms' behavior, a detailed experimental evaluation is presented.

4.1 Problem Instances

Ten classes of problem instances for the 2BP are provided in the literature. A first instance set, containing six classes (I-VI), was proposed by Berkey and Wang in [2]. For each of these classes, the widths and heights of the items were chosen uniformly at random from the intervals presented in Table 1. Moreover, the classes differ in the width (W) and the height (H) of the bins. Instance sizes, in terms of the number of items, are taken from $\{20, 40, 60, 80, 100\}$. Berkey and Wang provided 10 instances for each combination of a class with an instance size. This results in a total of 300 problem instances.

Table 1: Specification of instance classes I-VI (as provided by [2]).

Class	w_j	h_j	W	H
I	[1,10]	[1,10]	10	10
II	[1,10]	[1,10]	30	30
III	[1,35]	[1,35]	40	40
IV	[1,35]	[1,35]	100	100
V	[1,100]	[1,100]	100	100
VI	[1,100]	[1,100]	300	300

The second instance set, consisting of classes VII-X, was introduced by Martello and Vigo in [19]. In general, they considered four different types of items, as presented in Table 2. The four item types differ in the limits for the width w_i and the height h_i of an item. Then, based on these four item types, Martello and Vigo introduced four classes of instances which differ in the percentage of items they contain from each type. As an example, let us consider an instance of class VII. 70% of the items of such an instance are of type 1, 10% of the items are of type 2, further 10% of the items are of type 3, and the remaining 10% of the items are of type 4. These percentages are given per class in Table 3. As in the case of the first instance set, instance sizes are taken from $\{20, 40, 60, 80, 100\}$. The instance set by Martello and Vigo consists of 10 instances for each combination of a class with an instance size. This results in a total of 200 problem instances.

These 500 instances can be downloaded from `http://www.or.deis.unibo.it/research.html`.

4.2 Algorithm Tuning

In order to study the behavior of MS-LGFi we performed tests with a varying limit for the number of solution evaluations (that is, algorithm iterations) and for various settings of parameter κ. More specifically, we considered limits for the number of solution evaluations from $\{10^3, 5 \cdot 10^3, 2 \cdot 10^4, 5 \cdot 10^4, 10^5, 5 \cdot 10^5, 10^6, 5 \cdot 10^6\}$ and values for κ from $\{1, 5, 10, 15, 20\}$. For

Table 2: Item types for classes VII-X (as introduced in [19]).

Item type	w_j	h_j	W	H
1	$[\frac{2}{3} \cdot W, W]$	$[1, \frac{1}{2} \cdot H]$	100	100
2	$[1, \frac{1}{2} \cdot W]$	$[\frac{2}{3} \cdot H, H]$	100	100
3	$[\frac{1}{2} \cdot W, W]$	$[\frac{1}{2} \cdot H, H]$	100	100
4	$[1, \frac{1}{2} \cdot W]$	$[1, \frac{1}{2} \cdot H]$	100	100

Table 3: Specification of instance classes VII-X (as provided by [19]).

Class	Type 1	Type 2	Type 3	Type 4
VII	70%	10%	10%	10%
VIII	10%	70%	10%	10%
IX	10%	10%	70%	10%
X	10%	10%	10%	70%

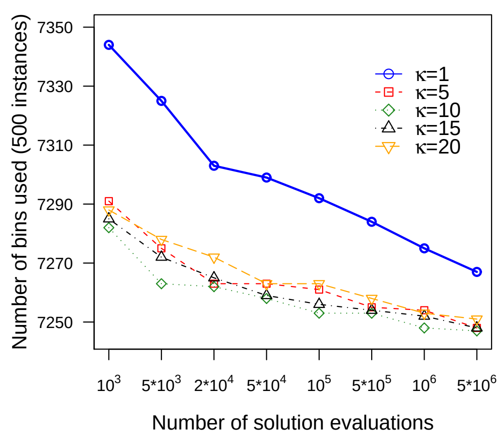

Figure 2: Tuning results for MS-LGFi.

each combination of the two parameters MS-LGFi was applied exactly once to each of the 500 problem instances. The sum of the number of bins used in the best solutions generated for all 500 problem instances is used as a measure. The graphic in Figure 2 provides this information for all parameter value combinations. The best performance is generally achieved (for each solution evaluation limit) with the setting $\kappa = 10$. Moreover, when increasing the number of solution evaluations from 10^6 to $5 \cdot 10^6$, the algorithm performance improves only slightly. Therefore, the final results of MS-LGFi that are presented in the following section are obtained with $\kappa = 10$ and a number of $5 \cdot 10^6$ solutions evaluations.

Concerning EA-LGFi, the same number of solution evaluations as for MS-LGFi was chosen as a stopping criterion (that is, $5 \cdot 10^6$ solution evaluations). Moreover, for parameter κ

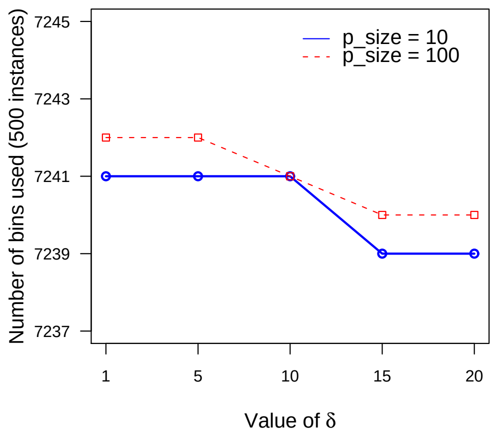

Figure 3: Tuning results for EA-LGFi.

we chose value 10, as in the case of MS-LGFi. However, we tested different population sizes ($p_{\text{size}} \in \{10, 100\}$) and different values for parameter δ ($\delta \in \{1, 5, 10, 15, 20\}$). Remember that the value of δ is used for the calculation of the probabilities for solutions to be selected as crossover partners. In general, the higher the value of δ, the more are good solutions preferred over worse ones, when selecting a crossover partner s^c for s. EA-LGFi was applied for each combination of δ and $p_{\text{size}} \in \{10, 100\}$ exactly once to each of the 500 problem instances. The sum of the number of bins used in the best solutions generated for all 500 instances is shown in Figure 3 for each parameter value combination. Even though differences in algorithm performance are quite small, higher values of δ seem to work better than smaller ones. Moreover, a population size of 10 generally seems to work slightly better than a population size of 100. The final results of EA-LGFi presented in the following section are the ones obtained with $\delta = 20$ and $p_{\text{size}} = 10$.

4.3 Numerical Results

The following six benchmark algorithms were chosen from the literature: The set covering heuristic (SCH) from [20], the hybrid GRASP approach from [21], the HBP approach from [3], the tabu search (TS) algorithm from [14, 16], the guided local search (GLS) approach from [8], and the weighted annealing (WA) metaheuristic from [18]. Among these approaches, SCH and GRASP are currently regarded to be the state-of-the-art techniques for the 2BP.

The results are shown in Tables 4 and 5 in a way which is traditional for the 2BP. For each algorithm the results are shown in two columns. The first column (with heading **value**) provides the sum of the number of bins used in the best solutions generated for the 10 instances of a combination between instance class (I – X) and number of items (20 – 100). For example, the best solutions generated for the 10 instances of Class I (20 items) by algorithm SCH occupy in total 71 bins. In case a value corresponds to the best result obtained by any algorithm, it highlighted in bold. Moreover, in the case of MS-LGFi and EA-LGFi a value is marked by an asterisk if it is better than the best know value as of today. The second column (with heading **time (s)**) shows the average computation time (in seconds) necessary to find the best solutions for the 10 instances of a combination between instance class and number of items. For example, algorithm SCH needed on average 0.06 seconds to find its best solutions for the 10 instances of Class I (20 items). The only exception is GLS for which the second column is missing, as the computation time information was not given in the original paper. Finally, the last line of Table 5 provides a summary of the results over all 500 problem instances. For each algorithm is given the sum of number of bins used, as well as the average computation time, for the 500 instances.

There are several aspects about the results that should be mentioned. First, the number bins used by the best solutions generated by EA-LGFi for the 500 problem instances amounts to 7239, which is the best value ever achieved by any algorithm. The best algorithm so far (GRASP) achieved a value of 7241. Moreover, MS-LGFi achieves a value of 7247, which is the 3rd best value ever obtained by any algorithm. Only GRASP (7241) and SCH (7243) achieve better values. This is remarkable, because MS-LGFi is a simple multi-start algorithm. It is also interesting to note that both MS-LGFi and EA-LGFi are able to solve four problem instances to optimality that have never been solved before. This concerns instances 173 and 174 (both from Class IV, with 60 items), instance 197 (from Class IV, with 100 items), and instance 298 (from Class VI, with 100 items). Detailed results for each single instance are shown in the 10 tables of Appendix A.

Finally, we would like to comment on the computation times. Due to the fact that different processors and different computation time limits have been used for the generation of the results, the computation times are certainly not directly comparable. However, the computation times of all algorithms are, in general, very low. Therefore, no algorithm can be identified to have a particular advantage or disadvantage over the other algorithms for what concerns the computation time requirements. In the following we provide the information about processors and computation time limits for the competitor algorithms: SCH was run on a Digital Alpha 533 MHz with a time limit of 100 seconds per instance. The same machine and time limit was used for HBP, because the results presented in Tables 4 and 5 are the ones from a re-implementation from [20]. GRASP was executed on a Pentium Mobile with 1500 MHz with a stopping criterion of 50000 iterations per application. Furthermore, TS was tested on a Silicon Graphics INDY R10000sc with 195 MHz and a computation time limit of 60 seconds per problem instance. Finally, GLS was executed on a Digital 500au workstation with a 500 MHz 21164 CPU using a computation time limit of 100 seconds per problem instance, while WA was run on a Pentium 4 with 3 GHz.

5 Conclusions and Outlook

In this paper we presented two algorithms for tackling the oriented two-dimensional bin packing problem under free guillotine cutting (2BP). Both algorithms are strongly based on a probabilistic version of an existing one-pass heuristic (LGFi) from the literature. The first algorithm is a simple multistart metaheuristics, whereas the second one is an evolutionary algorithm. The results have shown that both algorithms obtain very good results in comparison to current state-of-the-art approaches. In fact, both algorithms are able to solve four problem instances—which have not been solved yet by any algorithm—to optimality. Moreover, the best solutions generated by the evolutionary algorithm for the 500 instances use, in total, a number 7239 bins. This is the best value ever achieved by any algorithm proposed for the

Table 4: Part A: Numerical results for the 250 instances of the first 5 instance classes (Class I – Class V).

	LB	**SCH**		**GRASP**		**HBP**		**TS**		**GLS**	**WA**		**MS-LGFi**		**EA-LGFi**	
		value	time (s)	value	time (s)	value	time (s)	value	time (s)	value	value	time (s)	value	time (s)	value	time (s)
Class I																
20	71	**71**	0.06	**71**	0.00	**71**	10.09	**71**	24.00	**71**	**71**	0.21	**71**	0.00	**71**	0.00
40	134	**134**	2.42	**134**	0.00	**134**	32.02	135	36.11	**134**	**134**	0.06	**134**	0.00	**134**	0.00
60	197	**200**	7.26	**200**	4.50	201	40.17	201	48.93	201	**200**	0.67	**200**	0.00	**200**	0.01
80	274	**275**	4.63	**275**	1.50	**275**	10.10	282	48.17	**275**	**275**	3.07	**275**	0.00	**275**	0.00
100	317	**317**	5.21	**317**	0.00	319	20.79	326	60.81	321	**317**	9.21	**317**	0.00	**317**	0.00
Class II																
20	10	**10**	0.06	**10**	0.00	**10**	0.06	**10**	0.01	**10**	**10**	0.05	**10**	0.00	**10**	0.00
40	19	**19**	0.67	**19**	0.00	**19**	1.33	20	0.01	**19**	20	0.04	**19**	0.00	**19**	0.00
60	25	**25**	0.07	**25**	0.00	**25**	0.07	27	0.09	**25**	**25**	0.43	**25**	0.00	**25**	0.00
80	31	**31**	0.07	**31**	0.00	**31**	1.35	33	12.00	32	**31**	13.89	**31**	0.00	**31**	0.00
100	39	**39**	0.79	**39**	0.00	**39**	0.26	40	6.00	**39**	**39**	8.70	**39**	0.00	**39**	0.00
Class III																
20	51	**51**	0.07	**51**	0.00	**51**	20.74	55	54.00	**51**	53	0.04	**51**	0.00	**51**	0.02
40	92	**94**	2.66	**94**	3.00	**94**	21.38	97	54.02	95	**94**	2.15	**94**	0.02	**94**	0.01
60	136	**139**	6.21	**139**	4.60	140	40.19	140	45.67	140	**139**	0.16	**139**	0.03	**139**	0.27
80	187	**189**	8.80	**189**	4.10	190	32.72	198	54.31	193	**189**	3.16	191	0.31	**189**	20.68
100	221	**223**	12.80	**223**	4.90	225	41.51	236	60.10	229	224	7.52	225	1.03	224	26.17
Class IV																
20	10	**10**	0.06	**10**	0.00	**10**	0.07	**10**	0.01	**10**	**10**	0.05	**10**	0.00	**10**	0.00
40	19	**19**	0.07	**19**	0.00	**19**	0.08	**19**	0.01	**19**	**19**	0.14	**19**	0.00	**19**	0.00
60	23	25	6.15	25	3.00	25	20.15	26	0.14	25	25	0.05	**23***	11.77	**23***	12.18
80	30	32	10.35	**31**	1.90	32	21.67	33	18.00	33	**31**	12.19	**31**	0.00	**31**	0.00
100	37	38	4.72	38	1.50	38	12.02	38	6.00	39	38	0.36	**37***	0.01	**37***	0.00
Class V																
20	65	**65**	0.06	**65**	0.00	**65**	0.10	66	36.02	**65**	**65**	0.10	**65**	0.00	**65**	0.00
40	119	**119**	1.98	**119**	0.00	**119**	30.78	**119**	27.07	**119**	**119**	0.17	**119**	0.05	**119**	0.03
60	179	**180**	1.93	**180**	1.50	**180**	27.07	182	56.77	181	**180**	1.49	**180**	0.07	**180**	0.14
80	241	**247**	20.66	**247**	9.00	248	62.19	251	56.18	250	**247**	2.66	**247**	0.09	**247**	0.03
100	279	**282**	18.50	**282**	5.20	286	61.07	295	60.34	288	283	3.50	287	1.53	284	27.33

Table 5: Part B: Numerical results for the 250 instances of the last 5 instance classes (Class VI – Class X). The last table row provides a summary of the results for all 10 instances classes.

	LB	**SCH**		**GRASP**		**HBP**		**TS**		**GLS**	**WA**		**MS-LGFi**		**EA-LGFi**	
		value	time (s)	value	time (s)	value	time (s)	value	time (s)	value	value	time (s)	value	time (s)	value	time (s)
Class VI																
20	10	**10**	0.06	**10**	0.00	**10**	0.07	**10**	0.01	**10**	**10**	0.04	**10**	0.00	**10**	0.00
40	15	**17**	6.85	**17**	3.00	**17**	22.69	19	0.03	18	19	0.07	**17**	0.01	**17**	0.03
60	21	**21**	0.66	**21**	0.10	**21**	0.16	22	0.04	22	22	0.05	**21**	0.00	**21**	0.00
80	30	**30**	0.23	**30**	0.00	**30**	0.23	**30**	0.01	**30**	**30**	0.06	**30**	0.00	**30**	0.00
100	32	34	6.29	34	3.00	34	20.42	34	12.00	34	33	21.00	**32***	29.57	**32***	0.58
Class VII																
20	55	**55**	0.13	**55**	0.00	**55**	20.12	**55**	12.02	**55**	**55**	0.05	**55**	0.00	**55**	0.00
40	109	**111**	3.02	**111**	3.00	112	33.56	114	37.01	113	**111**	2.12	**111**	0.01	**111**	0.01
60	156	**158**	8.85	159	4.50	160	43.33	162	36.44	159	159	6.79	159	0.00	159	0.00
80	224	**232**	54.79	**232**	12.00	**232**	80.35	**232**	54.52	**232**	**232**	0.27	**232**	0.00	**232**	0.00
100	269	**271**	25.06	**271**	3.10	273	42.82	277	47.43	275	**271**	1.92	**271**	0.53	**271**	0.01
Class VIII																
20	58	**58**	0.06	**58**	0.00	**58**	0.07	**58**	18.04	**58**	**58**	0.05	**58**	0.01	**58**	0.03
40	112	**113**	0.96	**113**	1.50	**113**	11.36	114	18.72	114	**113**	0.21	**113**	0.01	**113**	0.00
60	159	162	9.05	**161**	4.20	162	30.81	162	20.99	163	162	0.16	**161**	0.00	**161**	0.02
80	223	**224**	11.60	**224**	1.60	225	20.83	226	37.95	225	**224**	0.33	**224**	0.01	**224**	0.00
100	274	279	47.13	278	6.10	279	50.98	284	52.66	281	**277**	0.06	278	0.05	**277**	0.25
Class IX																
20	143	**143**	0.06	**143**	0.00	**143**	0.06	**143**	0.01	**143**	**143**	0.19	**143**	0.00	**143**	0.00
40	278	**278**	0.07	**278**	0.00	**278**	0.07	**278**	24.05	**278**	279	0.04	**278**	0.00	**278**	0.00
60	437	**437**	0.07	**437**	0.10	**437**	0.07	438	24.26	**437**	438	0.12	**437**	0.00	**437**	0.00
80	577	**577**	0.08	**577**	0.00	**577**	0.09	**577**	54.31	**577**	**577**	0.16	**577**	0.00	**577**	0.00
100	695	**695**	0.11	**695**	0.00	**695**	0.11	**695**	34.11	**695**	**695**	0.23	**695**	0.00	**695**	0.00
Class X																
20	42	**42**	0.12	**42**	0.00	**42**	15.73	43	12.00	**42**	43	0.29	**42**	0.05	**42**	0.02
40	74	**74**	0.11	**74**	0.00	**74**	20.14	75	25.18	**74**	**74**	0.21	**74**	0.00	**74**	0.00
60	98	101	8.89	**100**	4.50	102	53.39	104	42.13	102	102	0.16	101	2.61	101	0.71
80	123	**128**	38.26	129	9.40	130	70.35	130	47.30	130	129	5.42	129	3.15	**128**	0.06
100	153	**159**	55.77	**159**	9.20	160	88.00	166	60.10	162	**159**	9.26	160	0.14	160	0.08
Summary	7173	7243	7.90	7241	2.21	7265	22.67	7358	28.72	7284	7253	2.39	7247	1.02	**7239**	1.77

2BP.

In the future we plan to investigate additional ways in which the probabilistic version of LGFi might be exploited. For example, an ant colony optimization approach might be better suited than an evolutionary algorithm for learning input sequences for LGFi. Moreover, we plan to add a local search procedure to our algorithms for improving the constructed solutions.

Acknowledgements

This work was supported by the binational grant *Acciones Integradas* ES16-2009 (Austria) and MEC HA2008-0005 (Spain), and by grant TIN2007-66523 (FORMALISM) of the Spanish government. In addition, Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Government of which he is a research fellow.

References

[1] B. S. Baker, E. G. Coffman Jr., and R. L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.

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

[3] M. A. Boschetti and A. Mingozzi. Two-dimensional finite bin packing problems. part II: New lower and upper bounds. *4OR*, 2:30–44, 2003.

[4] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3(1):66–76, 1982.

[5] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[6] T. G. Crainic, G. Perboli, and R. Tadei. Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on Computing*, 20(3):368–384, 2008.

[7] K. A. Dowsland and W. B. Dowsland. Packing problems. *European Journal of Operational Research*, 56(1):2–14, 1992.

[8] O. Faroe, D. Pisinger, and M. Zachariasen. Guided local search for the three-dimensional bin-packing problem. *Informs Journal on Computing*, 15(3):267–283, 2003.

[9] J. B. G. Frenk and G. Galambos. Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing*, 39(3):201–217, 1987.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[11] E. Hopper and B. Turton. A genetic algorithm for a 2d industrial packing problem. *Computers and Industrial Engineering*, 37(1–2):375–378, 1999.

[12] L. S. Lee. A genetic algorithm for two-dimensional bin packing problem. *MathDigest*, 2(1):34–39, 2008.

[13] A. Lodi. *Algorithms for Two-Dimensional Bin Packing and Assignment Problems*. PhD thesis, Università degli Studio di Bologna, 1999.

[14] A. Lodi, S. Martello, and D. Vigo. Approximation algorithms for the oriented two-dimensional bin packing problem. *European Journal of Operational Research*, 112(1):158–166, 1999.

[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(4):345–357, 1999.

[16] A. Lodi, S. Martello, and D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1-3):379–396, 2002.

[17] A. Lodi, S. Martello, and D. Vigo. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[18] K.-H. Loh, B. Golden, and E. Wasil. *A Weight Annealing Algorithm for Solving Two-dimensional Bin Packing Problems*, volume 47 of *Research/Computer Science Interfaces*, pages 121–146. Springer, New York, NY, 2009.

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

[20] M. Monaci and P. Toth. A set-covering-based heuristic approach for bin-packing problems. *Informs Journal on Computing*, 18(1):71–85, 2006.

[21] F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit. A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. *Annals of Operations Research*, 179(1):203–220, 2010.

[22] D. Pisinger and M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007.

[23] J. Puchinger and G. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2007.

[24] P. E. Sweeney and E. R. Paternoster. Cutting and packing problems: A categorized, application-orientated research bibliography. *The Journal of the Operational Research Society*, 43(7):691–706, 1992.

[25] L. Wong and L. S. Lee. Heuristic placement routines for two-dimensional bin packing problem. *Journal of Mathematics and Statistics*, 5(4):334–341, 2009.

Appendix A

This appendix contains 10 tables, one for the 50 problem instances of each instance class. Each table provides the results of MS-LGFi and EA-LGFi for each instance. The structure of the tables is as follows. The first column contains the number of items. The second column provides the instance number (numbered from 1 to 500). The next two columns contain information about the currently best known lower and upper bound values for each instance. Finally, the results of MS-LGFi, as well as the results of EA-LGFi, are presented in three columns. The first one of these three columns (with heading **res**) provides the number of bins used in the best found solution. In case a value in this column is shown with a gray background, the upper bound for the corresponding problem instances was improved. On the other side, in case a value is shown within a frame of white background, the best known upper bound for the corresponding instance was not reached. The second column (with heading **eval**) indicates after how many solution evaluations the best solution was found, while the third column (with heading **time**) provides the computation time (in seconds) after which the best solution was found.

Table 6: Detailed results for the 50 problem instances of Class I.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	1	8	8	8	1	0.0	8	1	0.0
	2	5	5	5	2	0.0	5	1	0.0
	3	9	9	9	1	0.0	9	1	0.0
	4	6	6	6	1	0.0	6	1	0.0
	5	6	6	6	327	0.0	6	360	0.0
	6	9	9	9	1	0.0	9	1	0.0
	7	6	6	6	1	0.0	6	1	0.0
	8	6	6	6	29	0.0	6	68	0.0
	9	8	8	8	1	0.0	8	1	0.0
	10	8	8	8	1	0.0	8	1	0.0
40	11	10	10	10	1	0.0	10	1	0.0
	12	12	12	12	1	0.0	12	1	0.0
	13	17	17	17	1	0.0	17	1	0.0
	14	14	14	14	5	0.0	14	2	0.0
	15	15	15	15	3	0.0	15	1	0.0
	16	14	14	14	1	0.0	14	1	0.0
	17	11	11	11	511	0.0	11	1220	0.0
	18	19	19	19	1	0.0	19	1	0.0
	19	11	11	11	1	0.0	11	3	0.0
	20	11	11	11	1	0.0	11	1	0.0
60	21	22	23	23	2	0.0	23	1	0.0
	22	18	19	19	1	0.0	19	1	0.0
	23	21	21	21	1	0.0	21	2	0.0
	24	22	22	22	1	0.0	22	1	0.0
	25	19	19	19	1	0.0	19	1	0.0
	26	17	17	17	63	0.0	17	2465	0.1
	27	15	16	16	1	0.0	16	1	0.0
	28	21	21	21	7	0.0	21	10	0.0
	29	18	18	18	1	0.0	18	9	0.0
	30	24	24	24	1	0.0	24	1	0.0
80	31	24	25	25	1	0.0	25	1	0.0
	32	26	26	26	1	0.0	26	1	0.0
	33	27	27	27	1	0.0	27	1	0.0
	34	27	27	27	2	0.0	27	2	0.0
	35	26	26	26	1	0.0	26	2	0.0
	36	28	28	28	4	0.0	28	67	0.0
	37	31	31	31	1	0.0	31	1	0.0
	38	29	29	29	1	0.0	29	1	0.0
	39	30	30	30	1	0.0	30	2	0.0
	40	26	26	26	1	0.0	26	4	0.0
100	41	28	28	28	3	0.0	28	3	0.0
	42	31	31	31	24	0.0	31	116	0.0
	43	29	29	29	1	0.0	29	1	0.0
	44	30	30	30	11	0.0	30	103	0.0
	45	32	32	32	1	0.0	32	2	0.0
	46	37	37	37	1	0.0	37	1	0.0
	47	28	28	28	6	0.0	28	23	0.0
	48	33	33	33	1	0.0	33	5	0.0
	49	31	31	31	227	0.0	31	181	0.0
	50	38	38	38	2	0.0	38	1	0.0

Table 7: Detailed results for the 50 problem instances of Class II.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	51	1	1	1	1	0.0	1	1	0.0
	52	1	1	1	1	0.0	1	1	0.0
	53	1	1	1	1	0.0	1	1	0.0
	54	1	1	1	1	0.0	1	1	0.0
	55	1	1	1	1	0.0	1	1	0.0
	56	1	1	1	1	0.0	1	1	0.0
	57	1	1	1	1	0.0	1	1	0.0
	58	1	1	1	1	0.0	1	1	0.0
	59	1	1	1	1	0.0	1	1	0.0
	60	1	1	1	1	0.0	1	1	0.0
40	61	1	1	1	166	0.0	1	81	0.0
	62	2	2	2	1	0.0	2	1	0.0
	63	2	2	2	1	0.0	2	1	0.0
	64	2	2	2	1	0.0	2	1	0.0
	65	2	2	2	1	0.0	2	1	0.0
	66	2	2	2	1	0.0	2	1	0.0
	67	2	2	2	1	0.0	2	1	0.0
	68	2	2	2	1	0.0	2	1	0.0
	69	2	2	2	1	0.0	2	1	0.0
	70	2	2	2	1	0.0	2	1	0.0
60	71	3	3	3	1	0.0	3	1	0.0
	72	2	2	2	6	0.0	2	16	0.0
	73	3	3	3	1	0.0	3	1	0.0
	74	3	3	3	1	0.0	3	1	0.0
	75	2	2	2	2	0.0	2	1	0.0
	76	2	2	2	1	0.0	2	1	0.0
	77	2	2	2	1	0.0	2	1	0.0
	78	3	3	3	1	0.0	3	1	0.0
	79	2	2	2	1	0.0	2	4	0.0
	80	3	3	3	1	0.0	3	1	0.0
80	81	3	3	3	1	0.0	3	1	0.0
	82	3	3	3	1	0.0	3	1	0.0
	83	3	3	3	1	0.0	3	1	0.0
	84	3	3	3	1	0.0	3	1	0.0
	85	3	3	3	1	0.0	3	1	0.0
	86	3	3	3	1	0.0	3	1	0.0
	87	3	3	3	4	0.0	3	13	0.0
	88	3	3	3	2	0.0	3	1	0.0
	89	4	4	4	1	0.0	4	1	0.0
	90	3	3	3	1	0.0	3	1	0.0
100	91	4	4	4	1	0.0	4	1	0.0
	92	4	4	4	1	0.0	4	1	0.0
	93	3	3	3	41	0.0	3	8	0.0
	94	4	4	4	1	0.0	4	1	0.0
	95	4	4	4	1	0.0	4	1	0.0
	96	4	4	4	1	0.0	4	1	0.0
	97	4	4	4	1	0.0	4	1	0.0
	98	4	4	4	1	0.0	4	1	0.0
	99	4	4	4	1	0.0	4	1	0.0
	100	4	4	4	1	0.0	4	1	0.0

Table 8: Detailed results for the 50 problem instances of Class III.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	101	6	6	6	1	0.0	6	1	0.0
	102	3	3	3	3675	0.0	3	11741	0.2
	103	6	6	6	1	0.0	6	1	0.0
	104	4	4	4	4	0.0	4	3	0.0
	105	4	4	4	36	0.0	4	49	0.0
	106	7	7	7	1	0.0	7	1	0.0
	107	5	5	5	1	0.0	5	1	0.0
	108	4	4	4	5	0.0	4	7	0.0
	109	5	5	5	2	0.0	5	3	0.0
	110	7	7	7	1	0.0	7	1	0.0
40	111	6	6	6	358	0.0	6	91	0.0
	112	8	8	8	2327	0.1	8	583	0.0
	113	11	11	11	3	0.0	11	1	0.0
	114	10	10	10	14	0.0	10	1	0.0
	115	12	12	12	1	0.0	12	1	0.0
	116	10	10	10	1	0.0	10	1	0.0
	117	8	8	8	7831	0.2	8	3549	0.1
	118	13	13	13	6	0.0	13	38	0.0
	119	7	8	8	2	0.0	8	1	0.0
	120	7	8	8	3	0.0	8	4	0.0
60	121	16	16	16	9	0.0	16	21	0.0
	122	12	13	13	5	0.0	13	9	0.0
	123	13	14	14	2	0.0	14	3	0.0
	124	14	15	15	3	0.0	15	2	0.0
	125	12	12	12	27	0.0	12	331	0.0
	126	12	12	12	6170	0.3	12	52890	2.7
	127	11	11	11	90	0.0	11	11	0.0
	128	15	15	15	4	0.0	15	1	0.0
	129	13	13	13	18	0.0	13	9	0.0
	130	18	18	18	1	0.0	18	2	0.0
80	131	17	17	$\boxed{18}$	2	0.0	17	1932	0.2
	132	18	18	18	3936	0.3	18	3902	0.3
	133	17	18	18	2	0.0	18	7	0.0
	134	18	18	18	902	0.1	18	153	0.0
	135	16	17	17	49	0.0	17	57	0.0
	136	20	20	20	29	0.0	20	2	0.0
	137	20	20	$\boxed{21}$	8	0.0	20	2500617	192.3
	138	22	22	22	2	0.0	22	2	0.0
	139	21	21	21	496	0.0	21	767	0.1
	140	18	18	18	38251	2.7	18	184395	14.0
100	141	19	19	19	6585	0.6	19	5573	0.6
	142	22	22	$\boxed{23}$	14	0.0	$\boxed{23}$	5	0.0
	143	18	19	19	19949	2.0	19	1406	0.1
	144	20	20	$\boxed{21}$	1	0.0	20	2480794	260.5
	145	22	22	22	73503	7.3	22	2882	0.3
	146	27	27	27	3	0.0	27	2	0.0
	147	20	20	20	3159	0.3	20	933	0.1
	148	23	23	23	761	0.1	23	11	0.0
	149	21	22	22	107	0.0	22	64	0.0
	150	29	29	29	1	0.0	29	3	0.0

Table 9: Detailed results for the 50 problem instances of Class IV.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	151	1	1	1	1	0.0	1	1	0.0
	152	1	1	1	1	0.0	1	1	0.0
	153	1	1	1	1	0.0	1	1	0.0
	154	1	1	1	1	0.0	1	1	0.0
	155	1	1	1	1	0.0	1	1	0.0
	156	1	1	1	2	0.0	1	1	0.0
	157	1	1	1	1	0.0	1	1	0.0
	158	1	1	1	1	0.0	1	1	0.0
	159	1	1	1	1	0.0	1	1	0.0
	160	1	1	1	1	0.0	1	1	0.0
40	161	1	1	1	1	0.0	1	1	0.0
	162	2	2	2	1	0.0	2	1	0.0
	163	2	2	2	1	0.0	2	1	0.0
	164	2	2	2	1	0.0	2	1	0.0
	165	2	2	2	1	0.0	2	1	0.0
	166	2	2	2	1	0.0	2	1	0.0
	167	2	2	2	1	0.0	2	1	0.0
	168	2	2	2	1	0.0	2	1	0.0
	169	2	2	2	1	0.0	2	1	0.0
	170	2	2	2	1	0.0	2	1	0.0
60	171	3	3	3	1	0.0	3	1	0.0
	172	2	2	2	3	0.0	2	1	0.0
	173	2	3	$\boxed{2^*}$	1422800	71.1	$\boxed{2^*}$	1729156	96.4
	174	2	3	$\boxed{2^*}$	972745	46.5	$\boxed{2^*}$	456033	25.4
	175	2	2	2	1	0.0	2	1	0.0
	176	2	2	2	1	0.0	2	1	0.0
	177	2	2	2	1	0.0	2	1	0.0
	178	3	3	3	1	0.0	3	1	0.0
	179	2	2	2	1	0.0	2	2	0.0
	180	3	3	3	1	0.0	3	1	0.0
80	181	3	3	3	1	0.0	3	1	0.0
	182	3	3	3	1	0.0	3	1	0.0
	183	3	3	3	1	0.0	3	1	0.0
	184	3	3	3	1	0.0	3	1	0.0
	185	3	3	3	1	0.0	3	1	0.0
	186	3	3	3	1	0.0	3	1	0.0
	187	3	3	3	242	0.0	3	84	0.0
	188	3	3	3	69	0.0	3	16	0.0
	189	3	4	4	1	0.0	4	1	0.0
	190	3	3	3	1	0.0	3	1	0.0
100	191	3	3	3	6	0.0	3	5	0.0
	192	4	4	4	1	0.0	4	1	0.0
	193	3	3	3	1	0.0	3	1	0.0
	194	4	4	4	1	0.0	4	1	0.0
	195	4	4	4	1	0.0	4	1	0.0
	196	4	4	4	1	0.0	4	1	0.0
	197	3	4	$\boxed{3^*}$	1080	0.1	$\boxed{3^*}$	277	0.0
	198	4	4	4	1	0.0	4	1	0.0
	199	4	4	4	1	0.0	4	1	0.0
	200	4	4	4	1	0.0	4	1	0.0

Table 10: Detailed results for the 50 problem instances of Class V.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	201	8	8	8	1	0.0	8	1	0.0
	202	5	5	5	1	0.0	5	1	0.0
	203	7	7	7	1	0.0	7	3	0.0
	204	5	5	5	1	0.0	5	1	0.0
	205	5	5	5	341	0.0	5	1268	0.0
	206	9	9	9	1	0.0	9	1	0.0
	207	6	6	6	1	0.0	6	1	0.0
	208	5	5	5	1	0.0	5	1	0.0
	209	7	7	7	1	0.0	7	1	0.0
	210	8	8	8	1	0.0	8	1	0.0
40	211	8	8	8	92	0.0	8	29	0.0
	212	10	10	10	30	0.0	10	117	0.0
	213	15	15	15	1	0.0	15	1	0.0
	214	13	13	13	1	0.0	13	1	0.0
	215	14	14	14	1	0.0	14	1	0.0
	216	12	12	12	1	0.0	12	1	0.0
	217	10	10	10	63	0.0	10	103	0.0
	218	17	17	17	1	0.0	17	1	0.0
	219	10	10	10	1	0.0	10	1	0.0
	220	10	10	10	19332	0.5	10	8046	0.3
60	221	20	20	20	10303	0.6	20	8882	0.5
	222	17	17	17	1	0.0	17	1	0.0
	223	19	19	19	2	0.0	19	5	0.0
	224	20	20	20	1	0.0	20	1	0.0
	225	15	15	15	384	0.0	15	433	0.0
	226	15	16	16	2	0.0	16	1	0.0
	227	14	14	14	5	0.0	14	21	0.0
	228	19	19	19	2	0.0	19	11	0.0
	229	16	16	16	1722	0.1	16	16211	0.9
	230	24	24	24	1	0.0	24	1	0.0
80	231	22	22	22	102	0.0	22	101	0.0
	232	22	23	23	13	0.0	23	10	0.0
	233	24	25	25	1	0.0	25	2	0.0
	234	25	25	25	1	0.0	25	1	0.0
	235	22	23	23	1	0.0	23	14	0.0
	236	25	26	26	1	0.0	26	1	0.0
	237	27	27	27	11	0.0	27	29	0.0
	238	26	26	26	11164	0.9	26	3376	0.3
	239	26	27	27	226	0.0	27	19	0.0
	240	22	23	23	244	0.0	23	447	0.0
100	241	23	24	$\boxed{25}$	5	0.0	$\boxed{25}$	8	0.0
	242	28	29	29	28	0.0	29	12	0.0
	243	24	24	24	135288	15.3	24	632	0.1
	244	26	26	$\boxed{27}$	67	0.0	26	1309655	152.7
	245	28	28	$\boxed{29}$	68	0.0	28	834082	93.8
	246	34	34	34	1	0.0	34	2	0.0
	247	25	25	$\boxed{26}$	1	0.0	25	236926	26.7
	248	29	29	$\boxed{30}$	265	0.0	$\boxed{30}$	106	0.0
	249	27	27	$\boxed{28}$	42	0.0	$\boxed{28}$	61	0.0
	250	35	35	35	4	0.0	35	1	0.0

Table 11: Detailed results for the 50 problem instances of Class VI.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	251	1	1	1	1	0.0	1	1	0.0
	252	1	1	1	1	0.0	1	1	0.0
	253	1	1	1	1	0.0	1	1	0.0
	254	1	1	1	1	0.0	1	1	0.0
	255	1	1	1	1	0.0	1	1	0.0
	256	1	1	1	1	0.0	1	1	0.0
	257	1	1	1	1	0.0	1	1	0.0
	258	1	1	1	1	0.0	1	1	0.0
	259	1	1	1	1	0.0	1	1	0.0
	260	1	1	1	1	0.0	1	1	0.0
40	261	1	1	1	1	0.0	1	1	0.0
	262	1	2	2	1	0.0	2	1	0.0
	263	2	2	2	1	0.0	2	1	0.0
	264	2	2	2	1	0.0	2	1	0.0
	265	2	2	2	1	0.0	2	1	0.0
	266	1	2	2	1	0.0	2	1	0.0
	267	2	2	2	1	0.0	2	1	0.0
	268	2	2	2	1	0.0	2	1	0.0
	269	1	1	1	114	0.0	1	96	0.0
	270	1	1	1	3013	0.1	1	6880	0.3
60	271	2	2	2	272	0.0	2	112	0.0
	272	2	2	2	1	0.0	2	1	0.0
	273	2	2	2	1	0.0	2	1	0.0
	274	2	2	2	1	0.0	2	1	0.0
	275	2	2	2	1	0.0	2	1	0.0
	276	2	2	2	1	0.0	2	1	0.0
	277	2	2	2	1	0.0	2	1	0.0
	278	2	2	2	1	0.0	2	1	0.0
	279	2	2	2	1	0.0	2	1	0.0
	280	3	3	3	1	0.0	3	1	0.0
80	281	3	3	3	1	0.0	3	1	0.0
	282	3	3	3	1	0.0	3	1	0.0
	283	3	3	3	1	0.0	3	1	0.0
	284	3	3	3	1	0.0	3	1	0.0
	285	3	3	3	1	0.0	3	1	0.0
	286	3	3	3	1	0.0	3	1	0.0
	287	3	3	3	1	0.0	3	1	0.0
	288	3	3	3	1	0.0	3	1	0.0
	289	3	3	3	1	0.0	3	1	0.0
	290	3	3	3	1	0.0	3	1	0.0
100	291	3	3	3	1	0.0	3	1	0.0
	292	3	3	3	13896	1.5	3	37796	4.6
	293	3	3	3	1	0.0	3	1	0.0
	294	3	3	3	1	0.0	3	1	0.0
	295	3	3	3	2	0.0	3	3	0.0
	296	4	4	4	1	0.0	4	1	0.0
	297	3	3	3	1	0.0	3	1	0.0
	298	3	4	$\boxed{3^*}$	2615111	294.2	$\boxed{3^*}$	9555	1.2
	299	3	3	3	2	0.0	3	1	0.0
	300	4	4	4	1	0.0	4	1	0.0

Table 12: Detailed results for the 50 problem instances of Class VII.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	301	5	5	5	3	0.0	5	6	0.0
	302	5	5	5	1	0.0	5	1	0.0
	303	5	5	5	4	0.0	5	4	0.0
	304	7	7	7	1	0.0	7	1	0.0
	305	6	6	6	1	0.0	6	1	0.0
	306	6	6	6	1	0.0	6	1	0.0
	307	4	4	4	1	0.0	4	2	0.0
	308	7	7	7	1	0.0	7	1	0.0
	309	6	6	6	21	0.0	6	32	0.0
	310	4	4	4	2	0.0	4	4	0.0
40	311	10	10	10	1	0.0	10	1	0.0
	312	12	12	12	71	0.0	12	1108	0.0
	313	9	10	10	1	0.0	10	1	0.0
	314	14	14	14	7	0.0	14	2	0.0
	315	10	10	10	1	0.0	10	1	0.0
	316	11	11	11	42	0.0	11	29	0.0
	317	11	12	12	1	0.0	12	1	0.0
	318	11	11	11	5645	0.1	11	1931	0.1
	319	8	8	8	5	0.0	8	1	0.0
	320	13	13	13	1	0.0	13	1	0.0
60	321	17	17	$\boxed{18}$	1	0.0	$\boxed{18}$	1	0.0
	322	14	14	14	1	0.0	14	1	0.0
	323	17	17	17	1	0.0	17	1	0.0
	324	15	15	15	1	0.0	15	2	0.0
	325	14	15	15	1	0.0	15	1	0.0
	326	15	15	15	1	0.0	15	2	0.0
	327	15	15	15	1	0.0	15	1	0.0
	328	17	17	17	141	0.0	17	1	0.0
	329	14	14	14	4	0.0	14	57	0.0
	330	18	19	19	1	0.0	19	5	0.0
80	331	20	21	21	5	0.0	21	6	0.0
	332	25	25	25	1	0.0	25	3	0.0
	333	20	21	21	1	0.0	21	1	0.0
	334	21	22	22	4	0.0	22	16	0.0
	335	23	24	24	1	0.0	24	1	0.0
	336	22	23	23	1	0.0	23	1	0.0
	337	24	25	25	1	0.0	25	1	0.0
	338	22	23	23	1	0.0	23	1	0.0
	339	23	24	24	2	0.0	24	1	0.0
	340	24	24	24	2	0.0	24	4	0.0
100	341	27	27	27	248	0.0	27	52	0.0
	342	27	27	27	1	0.0	27	14	0.0
	343	24	25	25	1	0.0	25	1	0.0
	344	26	27	27	1	0.0	27	1	0.0
	345	25	25	25	1	0.0	25	1	0.0
	346	28	28	28	2791	0.3	28	413	0.0
	347	27	27	27	1	0.0	27	1	0.0
	348	29	29	29	2	0.0	29	1	0.0
	349	25	25	25	18	0.0	25	32	0.0
	350	31	31	31	52488	5.0	31	501	0.1

Table 13: Detailed results for the 50 problem instances of Class VIII.

#				**MS-LGFi**			**EA-LGFi**		
items	**inst**	**LB**	**UB**	**res**	**eval**	**time**	**res**	**eval**	**time**
20	351	6	6	6	2	0.0	6	1	0.0
	352	7	7	7	1	0.0	7	1	0.0
	353	5	5	5	14387	0.1	5	20148	0.3
	354	7	7	7	1	0.0	7	2	0.0
	355	6	6	6	1	0.0	6	1	0.0
	356	6	6	6	1	0.0	6	1	0.0
	357	5	5	5	1	0.0	5	1	0.0
	358	5	5	5	22	0.0	5	32	0.0
	359	7	7	7	1	0.0	7	1	0.0
	360	4	4	4	1	0.0	4	2	0.0
40	361	11	12	12	1	0.0	12	2	0.0
	362	13	13	13	26	0.0	13	403	0.0
	363	11	11	11	2	0.0	11	1	0.0
	364	12	12	12	1	0.0	12	1	0.0
	365	9	9	9	1	0.0	9	2	0.0
	366	12	12	12	4	0.0	12	2	0.0
	367	11	11	11	3662	0.1	11	452	0.0
	368	11	11	11	4	0.0	11	10	0.0
	369	9	9	9	2	0.0	9	4	0.0
	370	13	13	13	10	0.0	13	2	0.0
60	371	17	17	17	10	0.0	17	29	0.0
	372	17	17	17	1	0.0	17	1	0.0
	373	16	17	17	4	0.0	17	2	0.0
	374	15	15	15	41	0.0	15	78	0.0
	375	14	14	14	4	0.0	14	2	0.0
	376	15	15	15	1	0.0	15	3	0.0
	377	14	14	14	4	0.0	14	1	0.0
	378	17	17	17	339	0.0	17	2890	0.2
	379	16	17	17	1	0.0	17	1	0.0
	380	18	18	18	2	0.0	18	3	0.0
80	381	22	22	22	4	0.0	22	4	0.0
	382	24	24	24	1	0.0	24	1	0.0
	383	20	21	21	1	0.0	21	1	0.0
	384	20	20	20	1	0.0	20	1	0.0
	385	26	26	26	3	0.0	26	3	0.0
	386	22	22	22	1	0.0	22	1	0.0
	387	22	22	22	1	0.0	22	1	0.0
	388	22	22	22	60	0.0	22	5	0.0
	389	21	21	21	1184	0.1	21	153	0.0
	390	24	24	24	12	0.0	24	76	0.0
100	391	26	27	27	1	0.0	27	1	0.0
	392	27	27	27	15	0.0	27	14	0.0
	393	24	24	24	1	0.0	24	1	0.0
	394	30	30	31	8	0.0	30	22536	2.5
	395	29	29	29	28	0.0	29	14	0.0
	396	26	27	27	1	0.0	27	1	0.0
	397	25	26	26	3	0.0	26	34	0.0
	398	28	28	28	4815	0.5	28	61	0.0
	399	27	27	27	2	0.0	27	2	0.0
	400	32	32	32	17	0.0	32	2	0.0

Table 14: Detailed results for the 50 problem instances of Class IX.

#				MS-LGFi			EA-LGFi		
items	**inst**	**LB**	**UB**	**res**	**eval**	**time**	**res**	**eval**	**time**
20	401	19	19	19	1	0.0	19	1	0.0
	402	13	13	13	1	0.0	13	1	0.0
	403	14	14	14	2	0.0	14	1	0.0
	404	16	16	16	1	0.0	16	1	0.0
	405	16	16	16	1	0.0	16	1	0.0
	406	14	14	14	1	0.0	14	1	0.0
	407	9	9	9	1	0.0	9	1	0.0
	408	14	14	14	1	0.0	14	1	0.0
	409	15	15	15	1	0.0	15	1	0.0
	410	13	13	13	1	0.0	13	1	0.0
40	411	25	25	25	1	0.0	25	1	0.0
	412	32	32	32	1	0.0	32	1	0.0
	413	29	29	29	1	0.0	29	1	0.0
	414	31	31	31	1	0.0	31	1	0.0
	415	27	27	27	1	0.0	27	1	0.0
	416	29	29	29	2	0.0	29	1	0.0
	417	24	24	24	1	0.0	24	1	0.0
	418	26	26	26	1	0.0	26	1	0.0
	419	21	21	21	1	0.0	21	1	0.0
	420	34	34	34	1	0.0	34	1	0.0
60	421	46	46	46	14	0.0	46	19	0.0
	422	45	45	45	1	0.0	45	1	0.0
	423	46	46	46	1	0.0	46	1	0.0
	424	44	44	44	1	0.0	44	1	0.0
	425	41	41	41	1	0.0	41	1	0.0
	426	37	37	37	1	0.0	37	1	0.0
	427	41	41	41	1	0.0	41	1	0.0
	428	47	47	47	1	0.0	47	1	0.0
	429	45	45	45	1	0.0	45	1	0.0
	430	45	45	45	1	0.0	45	1	0.0
80	431	59	59	59	1	0.0	59	1	0.0
	432	58	58	58	1	0.0	58	1	0.0
	433	57	57	57	1	0.0	57	1	0.0
	434	53	53	53	1	0.0	53	1	0.0
	435	62	62	62	1	0.0	62	1	0.0
	436	62	62	62	1	0.0	62	1	0.0
	437	59	59	59	1	0.0	59	1	0.0
	438	58	58	58	1	0.0	58	1	0.0
	439	49	49	49	1	0.0	49	1	0.0
	440	60	60	60	1	0.0	60	1	0.0
100	441	71	71	71	17	0.0	71	3	0.0
	442	64	64	64	1	0.0	64	1	0.0
	443	68	68	68	1	0.0	68	1	0.0
	444	78	78	78	1	0.0	78	1	0.0
	445	65	65	65	1	0.0	65	1	0.0
	446	71	71	71	1	0.0	71	1	0.0
	447	66	66	66	1	0.0	66	1	0.0
	448	74	74	74	1	0.0	74	1	0.0
	449	66	66	66	1	0.0	66	1	0.0
	450	72	72	72	1	0.0	72	1	0.0

Table 15: Detailed results for the 50 problem instances of Class X.

# items	inst	LB	UB	MS-LGFi res	MS-LGFi eval	MS-LGFi time	EA-LGFi res	EA-LGFi eval	EA-LGFi time
20	451	6	6	6	1	0.0	6	1	0.0
	452	3	3	3	2	0.0	3	2	0.0
	453	4	4	4	6	0.0	4	4	0.0
	454	5	5	5	1	0.0	5	1	0.0
	455	4	4	4	1	0.0	4	1	0.0
	456	4	4	4	1	0.0	4	1	0.0
	457	5	5	5	1	0.0	5	1	0.0
	458	3	3	3	1	0.0	3	1	0.0
	459	5	5	5	41546	0.5	5	15244	0.2
	460	3	3	3	1	0.0	3	1	0.0
40	461	8	8	8	1	0.0	8	1	0.0
	462	8	8	8	1	0.0	8	1	0.0
	463	9	9	9	5	0.0	9	7	0.0
	464	6	6	6	1	0.0	6	3	0.0
	465	6	6	6	1	0.0	6	1	0.0
	466	6	6	6	1	0.0	6	1	0.0
	467	7	7	7	1	0.0	7	2	0.0
	468	7	7	7	2	0.0	7	1	0.0
	469	8	8	8	2	0.0	8	2	0.0
	470	9	9	9	1	0.0	9	2	0.0
60	471	11	12	12	1	0.0	12	2	0.0
	472	12	12	12	23812	1.1	12	6833	0.4
	473	11	11	11	523309	24.9	11	121233	6.7
	474	7	8	8	1	0.0	8	1	0.0
	475	8	8	8	4	0.0	8	2	0.0
	476	13	13	13	5	0.0	13	1	0.0
	477	10	10	10	2	0.0	10	1	0.0
	478	10	10	10	33	0.0	10	3	0.0
	479	8	9	9	3	0.0	9	1	0.0
	480	8	8	8	49	0.0	8	11	0.0
80	481	12	13	13	4	0.0	13	5	0.0
	482	10	11	11	3	0.0	11	2	0.0
	483	11	11	11	1	0.0	11	4	0.0
	484	13	14	14	18	0.0	14	11	0.0
	485	14	14	14	2	0.0	14	8	0.0
	486	13	13	13	430266	31.5	13	4257	0.3
	487	14	14	$\boxed{15}$	1	0.0	14	3082	0.2
	488	10	10	10	1	0.0	10	10	0.0
	489	12	13	13	2	0.0	13	2	0.0
	490	14	15	15	2	0.0	15	3	0.0
100	491	14	15	15	1	0.0	15	1	0.0
	492	15	16	16	1	0.0	16	5	0.0
	493	15	16	16	34	0.0	16	24	0.0
	494	17	18	18	1	0.0	18	6	0.0
	495	17	18	18	2	0.0	18	1	0.0
	496	13	13	13	183	0.0	13	172	0.0
	497	13	14	14	1	0.0	14	1	0.0
	498	18	18	$\boxed{19}$	1	0.0	$\boxed{19}$	1	0.0
	499	16	16	16	13486	1.3	16	6475	0.7
	500	15	15	15	254	0.0	15	481	0.1