Training software for orthogonal packing problems

Gianluca Costa(1), Maxence Delorme(1), Manuel Iori(2), Enrico Malaguti(1), Silvano Martello(1)

(1) DEI “Guglielmo Marconi”, University of Bologna, Italy
(2) DISMI, University of Modena and Reggio Emilia, Italy

Abstract

An open source architecture for the interactive solution of packing problems in two dimensions is presented. Although primarily developed for helping engineering students to understand the algorithmic approaches to the solution of difficult combinatorial optimization problems, the application can be useful to practitioners and developers thanks to its visual tools. The paper gives intuitive and formal definitions of the problems at hand, discusses two natural heuristic approaches, provides technical information on the application, and reports the results of classroom experimental testings.

Key words: Training software; Combinatorial optimization; Orthogonal packing; Visualization; Classroom experiments.

1. Introduction

When learning combinatorial optimization algorithms it is helpful that students can use tools to easily understand the features and the difficulty of specific optimization problems. This paper illustrates `TwoBinGame`, an open source visual application for interactively “playing” with (i.e., trying to solve) two-dimensional packing problems. The application was developed at the Universities of Bologna and of Modena and Reggio Emilia with the primary scope of guiding engineering students and assessing human performance in solving hard problems. It can also be useful to practitioners and developers to visualize, test, and evaluate possible exact or heuristic algorithms. In `TwoBinGame`, the user operates in a computer generated environment where it is possible to interact with a graphical interface and look for solutions by manipulating virtual objects. The application was developed in Scala. The reader is referred to `http://scala-lang.org/documentation/books.html` for an overview of recent books on the Scala language.

Two-dimensional packing problems arise in a variety of industrial applications, when it is requested to allocate a given set of rectangular objects (*items*) to rectangular standardized stock units so as to minimize the waste area. In wood or glass industries, large rectangular sheets of material (*bins*) are cut to obtain given sets of rectangular elements. In warehouses, goods have to be allocated to shelves. When paging journals, it is necessary to place articles, photographs, and advertisements in the various pages. In all such cases, the standardized stock units can be seen as “large” rectangles to which smaller rectangles have to be allocated without overlapping. In other industrial applications, such as, e.g., paper or cloth production, the standardized stock unit consists of a roll of material (*strip*) from which one has to obtain the desired rectangular items by minimizing the used roll length. In both cases, two main variants occur in practice: either the items to be packed have a fixed orientation (e.g., when the material is corrugated or decorated), or they can be rotated (usually by 90 degrees). The interactive application we describe is capable of handling the resulting four variants (fixed length stock units or infinite length rolls; oriented or non-oriented items).

The primary scope of this work was to implement a tool to help students developing an algorithmic thinking. In this respect, the computational testing was intended to understand strengths and weaknesses of human thinking in solving combinatorial optimization problems, so as to obtain balanced test beds for future use in classrooms. In the next section a literature review is presented. Section 3 gives a formal definition of the problems at hand. In Section 4 technical information on the developed visual application is provided, and in Section 5 the results of our experimental testing are reported. Conclusions follow in Section 6.

2. Literature review

Besides their industrial relevance, two-dimensional packing problems have considerable theoretical interest in the field of algorithmic combinatorial optimization. Already in 1965, Gilmore and Gomory [22] studied a two-dimensional packing problem and presented a column generation approach for its optimal solution. The analysis of the vast literature produced in the following 50 years is beyond the scope of this paper. We refer the interested reader, e.g., to the book by Dyckhoff and Finke [20], to the more recent surveys by Lodi, Martello, and Monaci [26] and Lodi, Martello, Monaci, and Vigo [27], and to the typologies proposed by Lodi, Martello, and Vigo [28] and Wäscher, Haußner, and Schumann [36]. Exact branch-and-bound approaches have been presented by Martello, Monaci, and Vigo [31], Alvarez-Valdes, Parreño, and Tamarit [1], and Boschetti and Montaletti [8]. Upper bounds have been studied, among others, by Iori, Martello, and Monaci [24] and Burke, Kendall, and Whitwel [10], who presented heuristic and metaheuristic algorithms. Lower bounds have been studied by Martello and Vigo [32] for the oriented case, and by Dell'Amico, Martello, and Vigo [17] for the non-oriented case. More recently, exact approaches have been presented by Côté, Dell'Amico, and Iori [15] and Delorme, Iori, and Martello [18]. For the variant (also considered in the presented software) in which the objective is to maximize the total item area that is packed, we mention the exact approaches proposed by Caprara and Monaci [11] and the upper bounding techniques studied by Egeblad and Pisinger [21].

In order to experiment the developed application, we constructed a set of two-dimensional packing instances, and we asked a set of engineering students to test their skills by using the application to find good-quality feasible solutions within a time limit. Their solutions were compared to optimal and approximate solutions produced by ad-hoc algorithms from the literature.

The literature that presents interactive systems to study and solve decision problems is very varied and multidisciplinary. Although a complete survey is out of the scope of this paper, some interesting contributions are briefly discussed in the following.

A first branch of this literature focused on the way interactive systems can be used in teaching. An early discussion was given in Asfahl, Swayze, Lee, and Safford [3], who emphasized the fact that computer training programs can help capturing the attention of the audience and teaching non-conventional subjects. A few years later, Llaugel and Confesor [25] presented a computerized interactive program to teach quality control and quality improvement to undergraduate students. The program was based on the simulation of the process of filling medicine bottles, where over filling and under filling have a cost, and was assigned to groups of students who competed with each other to get lowest cost solutions. Crumpton and Harden [16] discussed the outcome of a test conducted on 20 students, who were asked to interact with a virtual reality tool simulating a *pick and pack problem* in the cereal industry: cereal boxes were proceeding down a conveyor and the operator was required to grab them, orient them, and then pack them into a larger box. The results were discussed mainly from an ergonomics point of view. Bodin and Gass [7] discussed key aspects in the teaching of the *analytic hierarchy process*. They developed a series of educational tests by using the Expert Choice Software and assigned them as classroom exercises to groups of students. Several pedagogical insights were discussed for the attempted tests. More recently, Costa, D'Ambrosio, and Martello [13, 14] discussed Java tools developed for the teaching of graph theory, including applications to solve a number of optimization problems such as, e.g., shortest spanning trees, shortest paths, and maximum flows. We also mention the special issue of *INFORMS Transactions on Education* edited by Griffin [23], entirely devoted to investigating the use of classroom games in management science and operations research, as well as the study on how board puzzles may help in teaching operations research by DePuy and Don Taylor [19], and the recent classroom game on healthcare by Vliegen and Zonderland [35].

Another branch of this literature focused on the comparison between computerized and human behavior in the solution of optimization problems. Large attention was devoted to the well-known *Traveling Salesman Problem* (TSP), which requires to find a minimal cost hamiltonian cycle in a weighted graph. Mac Gregor and Ormerod [30] showed that humans are efficient in solving Euclidean TSP instances when compared to basic heuristics. They also discussed the practical difficulty of TSP instances as a function of the number of non-boundary points. A related discussion can be found in Chapter 4 of the TSP book by Applegate, Bixby, Chvatal, and Cook [2]. A review of this area of research is provided by MacGregor and Chu [29]. Recently, Miyata, Watanabe, and Minagawa [33] studied the performance of young children on the TSP using a city-block metric. They showed that children tended to use strategies such as traveling straight to the farthest goal first, whereas adults relied more on nearest neighbor attempts. Very recently, Pacaux-Lemoine, Trentesaux, Zambrano Rey, and Millot [34] discussed the design of intelligent manufacturing systems through human-machine cooperation mechanisms. We also mention that the *Genetic and Evolutionary Computation Conference* (GECCO) supports annual awards for human-competitive results ("Humies" Awards). Among the works that participated with success to this competition, we mention the genetic programming system for the two dimensional strip packing problem by Burke, Hyde, Kendall, and Woodward [9], and the large-scale experiment comparing human and genetic programming solutions by Bartoli, De Lorenzo, Medvet, and Tarlao [5].

3. Orthogonal packing problems

Let n be the number of items to pack, and denote as w_j and h_j the width and height of item j ($j = 1, 2, \dots, n$). When packing in a *strip*, the traditional representation is to see it as a *vertical* band, having fixed width and infinite height. In order to obtain a better display on a monitor, we decided instead of adopting an *horizontal* view, i.e., our strip has fixed height and infinite width.

Let H be the height of the bin or strip, and W be either the width of the bin or any upper bound on the maximum strip width. We consider two optimization problems, each in two variants. In the *strip* case, the objective is to pack all the n items by minimizing the width at which the strip is used. In the *bin* case, the objective is to pack a subset of items having the largest total area. In both cases the items are either *oriented* (they cannot be rotated) or they can be rotated by 90° degrees. We denote the resulting four variants as:

- [**SO:**] the stock unit is a strip and the items are oriented;
- [**SR:**] the stock unit is a strip and the items can be rotated by 90°;
- [**BO:**] the stock unit is a bin and the items are oriented;
- [**BR:**] the stock unit is a bin and the items can be rotated by 90°.

All problems we consider are strongly $\mathcal{NP}$-hard and can be modeled in different ways. In the following we adopt, for the sake of clarity, the modeling approach originally developed by Beasley [6], where the variables represent the coordinates at which the items are packed in the bin/strip. As it will be clear later, such variables correspond to the decisions that the user has to take when using our visual tool. We assume in the following that all numerical data are positive integers. Consider a Cartesian system restricted to non-negative integer coordinates with origin $(0, 0)$ in the bottom-left corner of the bin/strip.

We first consider problem SO. The following *Integer Linear Programming* (ILP) model makes use of a pseudo-polynomial number of binary decision variables

$$x^j_{pq} = \begin{cases} 1 & \text{if item } j \text{ is packed with its bottom-left corner at } (p, q); \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

for $j = 1, \dots, n$, $p \in W_j$, $q \in H_j$, where $W_j = \{0, 1, \dots, W - w_j\}$ and $H_j = \{0, 1, \dots, H - h_j\}$ denote all positions where the bottom-left corner of item j may be placed. The ILP model is:

$$\min\ z \tag{2}$$

$$\sum_{p \in W_j} \sum_{q \in H_j} x^j_{pq} = 1 \qquad (j = 1, \dots, n) \tag{3}$$

$$\sum_{j=1}^{n} \sum_{\substack{p=r-w_j+1\\ p \in W_j}}^{r} \sum_{\substack{q=s-h_j+1\\ q \in H_j}}^{s} x^j_{pq} \leq 1 \qquad (r = 0, \dots, W-1; s = 0, \dots, H-1) \tag{4}$$

$$\sum_{p \in W_j} \sum_{q \in H_j} (p + w_j) x^j_{pq} \leq z \qquad (j = 1, \dots, n) \tag{5}$$

$$x^j_{pq} \in \{0, 1\} \qquad (j=1, \dots, n; p \in W_j; q \in H_j). \tag{6}$$

The objective function (2) minimizes the width z at which the strip is used. Equations (3) impose that each item is packed in exactly one position. Inequalities (4) impose that at most one item occupies any unit square of the strip. Constraints (5) set the value of the objective function. Note that, for each item j, definitions (6) only consider variables corresponding to feasible positions for the bottom-left corner of the item, so no packed item can exceed the height of the strip.

In order to model problem SR, we add to (1) a twin set of variables,

$$y^j_{pq} = \begin{cases} 1 & \text{if item } j \text{ is packed, rotated, with its (new) bottom-left corner at } (p, q); \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

for $j = 1, \dots, n$, $p \in \overline{W}_j$, $q \in \overline{H}_j$, where $\overline{W}_j = \{0, 1, \dots, W - h_j\}$ and $\overline{H}_j = \{0, 1, \dots, H - w_j\}$ denote all positions where the bottom-left corner of the rotated item may be placed. The resulting ILP model for SR has the same objective function as SO, while constraints become

$$\sum_{p \in W_j} \sum_{q \in H_j} x^j_{pq} + \sum_{p \in \overline{W}_j} \sum_{q \in \overline{H}_j} y^j_{pq} = 1 \qquad (j = 1, \dots, n) \tag{8}$$

$$\sum_{j=1}^{n} \Bigg(\sum_{\substack{p=r-w_j+1\\ p \in W_j}}^{r} \sum_{\substack{q=s-h_j+1\\ q \in H_j}}^{s} x^j_{pq} + \sum_{\substack{p=r-h_j+1\\ p \in \overline{W}_j}}^{r} \sum_{\substack{q=s-w_j+1\\ q \in \overline{H}_j}}^{s} y^j_{pq} \Bigg) \leq 1 \quad (r=0, \dots, W-1; s=0, \dots, H-1) \tag{9}$$

$$\sum_{p \in W_j} \sum_{q \in H_j} (p + w_j)\, x^j_{pq} + \sum_{p \in \overline{W}_j} \sum_{q \in \overline{H}_j} (p + h_j)\, y^j_{pq} \leq z \qquad (j = 1, \dots, n) \tag{10}$$

$$x^j_{pq} \in \{0, 1\} \quad (j=1, \dots, n; p \in W_j; q \in H_j) \tag{11}$$

$$y^j_{pq} \in \{0, 1\} \quad (j=1, \dots, n; p \in \overline{H}_j; q \in \overline{W}_j) \tag{12}$$

to impose feasibility with respect to the two possible orientations.

An ILP model for problem BO (pack oriented items in a bin) can be immediately derived from (2)-(6) by: (i) eliminating constraints (5), as definitions (6) guarantee that no packed item can exceed the borders of the bin; (ii) replacing objective function (2) with

$$\max \sum_{j=1}^{n} w_j\, h_j \Big(\sum_{p \in W_j} \sum_{q \in H_j} x^j_{pq} \Big), \tag{13}$$

that maximizes the packed area; (iii) replacing the '=' sign with '$\leq$' in (3), as not all items must be packed.

Finally it is easily seen that the non-oriented bin packing version BR can be modeled, by introducing the twin variables (7), as

$$\max \sum_{j=1}^{n} w_j\, h_j \Big(\sum_{p \in W_j} \sum_{q \in H_j} x^j_{pq} + \sum_{p \in \overline{W}_j} \sum_{q \in \overline{H}_j} y^j_{pq} \Big) \tag{14}$$

$$\sum_{p \in W_j} \sum_{q \in H_j} x^j_{pq} + \sum_{p \in \overline{W}_j} \sum_{q \in \overline{H}_j} y^j_{pq} \leq 1 \qquad (j = 1, \ldots, n) \tag{15}$$

$$(9), (10), (11), (12).$$

As already mentioned, the purpose of this paper is not to present the state-of-the-art of algorithms for the exact or approximate solution of two-dimensional packing problems, for which the interested reader is referred to the surveys and the articles mentioned in Section 1. In order to be self-contained, we give however a brief description of two classical and intuitive heuristic algorithms that we used to evaluate the solutions produced by the students who took part in the classroom tests. (The exact solutions were obtained through the exact approaches proposed by Côté, Dell'Amico, and Iori [15] and Delorme, Iori, and Martello [18].)

The classical *Bottom-Left* algorithm was introduced by Baker, Coffman, and Rivest [4] for problem SO, in the (equivalent) version in which the strip is vertical (see Section 3). It preliminary sorts the items according to a prefixed policy (non increasing width, or non increasing height, or non increasing area), and packs one item at a time, in the lowest possible position, left justified. Its worst-case performance is 3, i.e., it is guaranteed to produce a vertical strip whose hight does not exceed by more than three times the hight of the optimal solution. The algorithm can be implemented so as to run in $O(n^2)$ time (see Chazelle [12]). In our case (horizontal strip), the item is packed in the leftmost possible position, top justified.

As the classroom tests were performed both on problems SO and SR, we also implemented a simple variation that preliminarily sorts the items according to a given policy (non-increasing max(width,height) or non-increasing area) and, at every iteration, packs the current item in the leftmost-top position: if both orientations are feasible, the item is packed by selecting the smallest side as the width.

Another stream of heuristics (*Best-fit algorithms*, see Burke, Kendall, and Whitwell [10]) first finds, in the current packing, *holes* (empty orthogonal spaces) at which the next item may be packed (initially, the bottom of the strip/bin is the unique hole). It selects the bottommost hole and inserts in it the largest item that fits. If the item width is smaller than that of the hole, it is packed according to a prefixed policy (leftmost, or close to the tallest neighbor, or close to the smallest neighbor). If no item can be packed in the selected hole, the hole is "closed" (filled with empty space). When rotation is allowed, the algorithm considers both orientations when checking if an item fits in a hole, and selects the largest item that fits, selecting the highest one in case of tie.

For both families of algorithms, our approach performs a separate execution for each of the three prefixed policies, and returns the best solution. For the cases with rotation, a so called *tower reduction* post process is executed, that tries to rotate, if possible, the items whose top edges touch the top of the strip, in order to reduce the strip height.

An instance of any problem variant can obviously include a number of identical items. In order to obtain a compact definition and visual rendering, our representation of an instance handles it without duplicating the items, but defining the number of copies of each item *type*.

4. Architecture and tools

In this section, we detail `TwoBinPack`, the open source Scala architecture that was developed to support `TwoBinGame`. Scala is a general-purpose programming language that combines ideas from both the functional and the object-oriented paradigm. It runs on the Java platform and is released under a BSD licence. Full documentation can be found at `http://scala-lang.org/documentation`.

The application we describe is published under the GPLv3 license. A self-contained version is available for free download, as a compressed file, from `http://www.or.deis.unibo.it/staff_pages/martello/Tools/T.html`. Instructions, additional information, and (future) enhanced versions can be found at `http://gianlucacosta.info/TwoBinPack/`. A visual tutorial can be seen online at `https://youtu.be/SS6mJxugyxc`.

The architecture includes three main components:

1. `TwoBinManager`, that manages the problem instances (creation, modification, removal, import, export) and the solutions (import, visualization);
2. `TwoBinGame`, that loads the instances created by `TwoBinManager` and enables the user to interactively solve them;
3. `TwoBinKernel`, a Scala library referenced by the two previous components.

In addition, `TwoBinPack` is based on four modules of the general-purpose library Helios (see `https://www.facebook.com/pages/Helios/206962992779275`): Helios-core, Helios-fx, Helios-jpa, and Helios-reflection. The overall architecture is summarized in Figure 1.

TwoBinKernel is the core of TwoBinPack's architecture. It provides the ScalaFX components for rendering two-dimensional packing problems, as well as model concepts such as dimensions, frames, coordinate system, templates, problem, and user solution. `TwoBinKernel` is also an open source Scala library, released under the GPLv3 license,

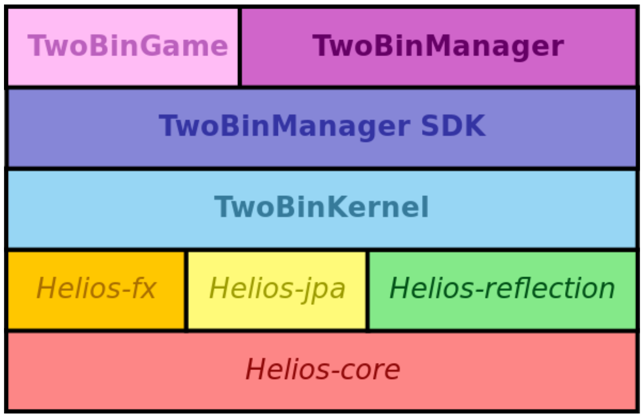


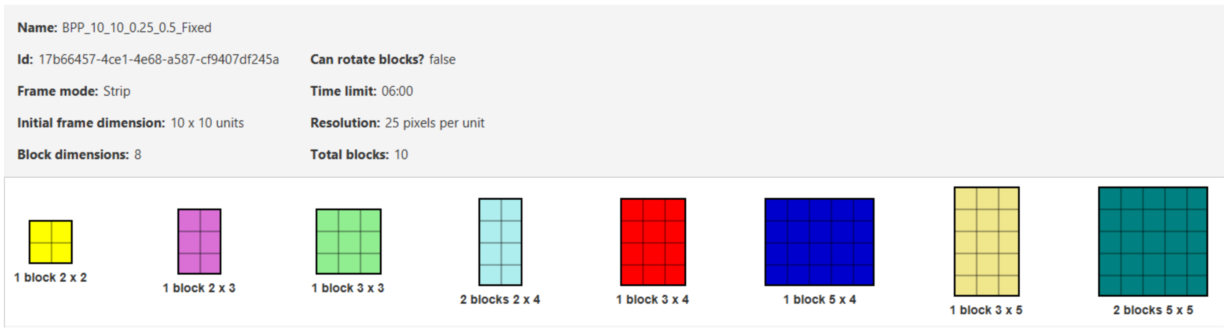
Figure 1: The architecture stack.

available for the creation of new applications (see `https://github.com/giancosta86/TwoBinKernel`). In the next sections we provide some details on the two other components of the architecture.

TwoBinManager

`TwoBinManager` is a ScalaFX application designed to manage instances, bundles, and solutions through a local HyperSQL database residing in the user's home directory.

An instance corresponds to a certain problem variant, it allows or prevents rotation, it has a total number of items (called *blocks* in the application), a gallery of item types, each having a quantized amount of items, and the amount of items of each type. It might also have a time limit for the user to find a solution. Figure 2 shows a strip packing instance with 10 items of 8 item types, where rotation is not allowed and 6 minutes are given to find a solution.


Name: BPP_10_10_0.25_0.5_Fixed

Id: 17b66457-4ce1-4e68-a587-cf9407df245a **Can rotate blocks?** false

Frame mode: Strip **Time limit:** 06:00

Initial frame dimension: 10 x 10 units **Resolution:** 25 pixels per unit

Block dimensions: 8 **Total blocks:** 10

Figure 2: A strip packing instance.

An instance can be generated within the program, or it can be read from a standard text file or from a *bundle* file (a set of one or more instances) previously generated by

TwoBinManager. A few parameters of the instance can be modified in TwoBinManager (e.g., the time limit).

TwoBinGame

TwoBinGame is a ScalaFX application enabling users to interactively solve two-dimensional packing problems. It reads problem bundles (for example, created by TwoBinManager) and returns a file that contains the best solution found by the user for each instance in the bundle as well as the time required to obtain it. Figure 3 depicts the solution given by a student for the instance shown in Figure 2, that packs the items into a strip of width 14, found in 3 minutes and 17 seconds. Upon reading a bundle file, the user tries to solve, in

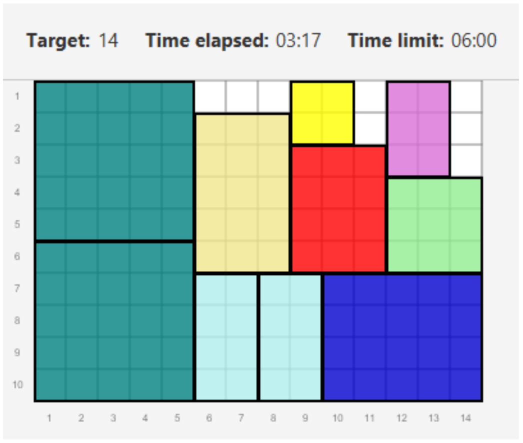


Figure 3: A solution for the instance of Figure 2.

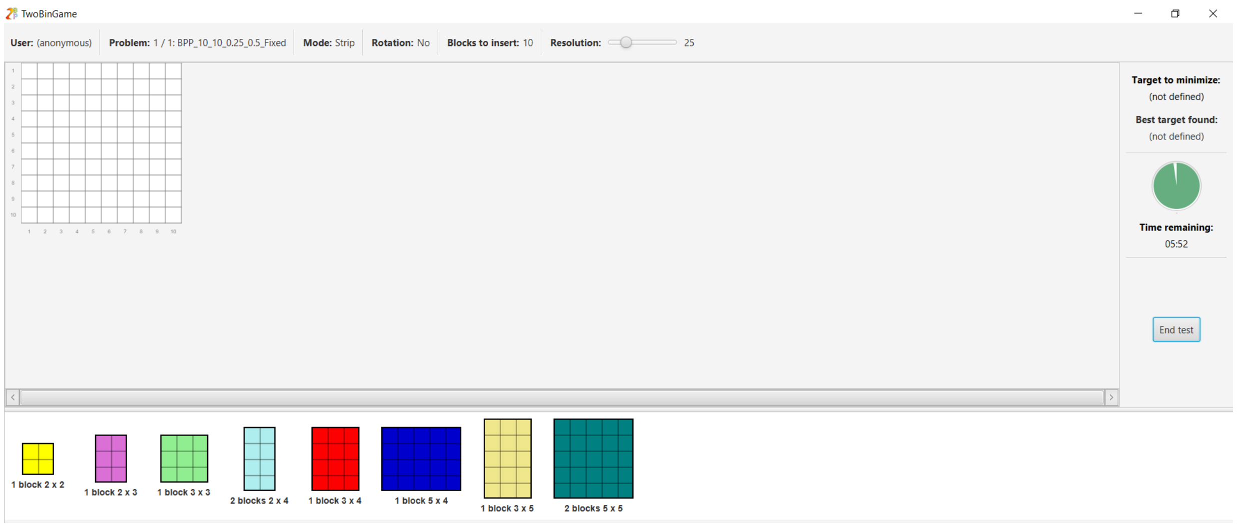
sequential order, all the instances in the bundle. As soon as the time limit has expired(or if the user decides to pass to the next instance), TwoBinGame stores the best solution found for the current instance. When all instances of the bundle have been tried, the user can save the obtained results. TwoBinGame provides two different ways for building a solution: usual drag and drop, or a mouse-wheel and click approach (for faster interaction).

Figure 4 shows the visual interface of TwoBinGame for the instance of Figure 2.

Figure 5 shows a possible workflow for TwoBinPack: problem bundles created via TwoBinManager can be imported into TwoBinGame and the corresponding solutions are usually imported back into TwoBinManager. At the same time, standard problems provide interoperability with third-party software.

5. Experiments

We used TwoBinGame to perform a series of classroom tests on the SO and SR variants with students of Engineering (Degrees in Management Engineering) of the Universities of Bologna and of Modena and Reggio Emilia. Classroom tests were optional and competitive.


TwoBinGame

User: (anonymous)
Problem: 1 / 1: BPP_10_10_0.25_0.5_Fixed
Mode: Strip
Rotation: No
Blocks to insert: 10
Resolution: 25

Target to minimize: (not defined)
Best target found: (not defined)
Time remaining: 05:52

1 block 2 x 2
1 block 2 x 3
1 block 3 x 3
2 blocks 2 x 4
1 block 3 x 4
1 block 5 x 4
1 block 3 x 5
2 blocks 5 x 5

Figure 4: The user view for the instance of Figure 2.

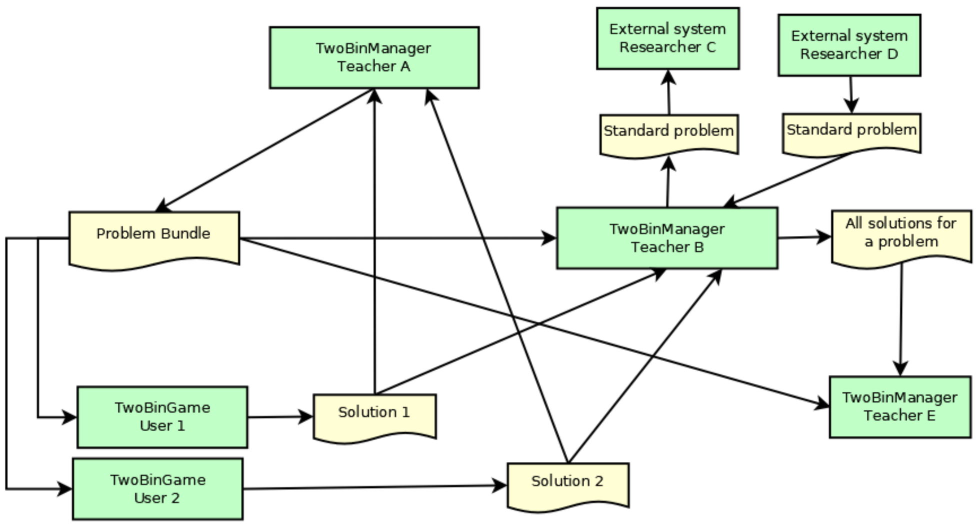


Figure 5: Workflow.

Each student that accepted to participate received a small bonus in her/his final mark. The students achieving the best solutions received a larger bonus. In the next section we describe in details the setup of the tests, whereas in Section 5.2 we discuss the results that were obtained.

5.1 Setup

We decided to focus on SO and SR random instances having the following characteristics:

1. $n \in \{10, 13, 17, 20\}$;
2. $H \in \{10, 15, 20\}$;
3. w_j and h_j values uniformly randomly distributed in $[H/4, H/2], [1, 2H/3]$;
4. rotation either allowed or forbidden.

For each quadruplet (n, H, range, rotation) one instance was generated, producing in total 48 instances. After a manual check, we removed the instances that could be trivially solved, and generated others with the same parameters. We fairly distributed the 48 instances into 8 bundles of 6 *standard* instances each. For statistical purposes, we added to each bundle an additional instance, having either medium or high difficulty. The one of medium difficulty had 13 items to be packed into a strip of height 15 without rotation, while the difficult one had 20 items to be packed into a strip of height 20 allowing rotation. According to the presumed difficulty, each instance was allowed a time limit of x minutes, with $x \in \{4, 5, 6, 7, 8\}$.

Each student was assigned a bundle. Each test lasted about one hour: 20 initial minutes were used to instruct the students on how to download the software, solve a toy instance, and learn on a standard instance with no time limit. The remaining 40 minutes were used to solve the seven instances in the assigned bundle. At the end of the test, each student sent by email the file containing the best solution obtained for each instance.

Four classroom tests were performed. At the University of Bologna, 65 students of the second year of the Bachelor Degree performed the test in the university lab, while 18 students from the same class performed the test at home using their own PCs. In the latter case, links and instructions on how to download `TwoBinGame`, as well as the bundle number, were communicated to the students by email. At the University of Modena and Reggio Emilia, two tests were performed in the university lab: the former one involved 72 students of the first year of the Master Degree, while the latter involved 58 students of the third year of the Bachelor Degree. Overall, 213 students performed the test: 195 in the lab and 18 online.

The outcome of the tests showed no remarkable difference in the performance of students of different courses so, in the next section, we evaluate the results through aggregate information.

5.2 Results

We report in the following the outcome of 201 tests out of the 213 that were performed. The results of the remaining 12 tests were disregarded because either incomplete information was provided via email or the student did not reach a minimum of 5 feasible solutions out of 7. We evaluate in Tables 1–5 the $201 \times 6 = 1206$ solutions of the six standard instances in the bundles, while in Table 6 and Figure 6 we comment on the 201 solutions of the additional, more difficult, instances.

The tables aggregate the instances into subgroups, one per line, according to different characteristics. Let z_{opt} be the optimal solution value. The tables provide, for each line, the number of tests performed on the corresponding instances and, respectively for the humans and the heuristics,

- average percentage of optimal solutions (avg. perc. opt.);
- average absolute gap (avg. abs. gap) between solution value and optimal value;
- average relative gap (avg. rel. gap), computed as (solution value - z_{opt})/z_{opt}.

The last line of each table provides the overall average values.

Table 1 evaluates the solution quality when varying the number n of items. As it could be expected, the students found good quality solutions for instances with a small number of items and worse solutions when this number was larger. A similar behavior cannot be clearly established for the heuristics. Overall, the students beat the heuristics in finding proven optimal solutions (22.5% vs 18%), but they resulted slightly worse in terms of average solution quality (1.83 vs 1.65, and 6.1% vs 5.7%).

Table 1: Evaluation by varying n

n	# tests	avg. perc. opt.		avg. abs. gap		avg. rel. gap	
		human	heuristic	human	heuristic	human	heuristic
10	304	28%	9.2%	1.35	1.41	7%	7.5%
13	299	25.4%	29.1%	1.62	1.48	5.9%	5%
17	299	19.1%	11%	2.12	2.05	6.4%	6.5%
20	304	17.4%	22.7%	2.23	1.66	5.2%	3.9%
overall	1206	22.5%	18%	1.83	1.65	6.1%	5.7%

Table 2 ranks the solutions according to the range of the optimal solution value (strip length). The students performed very well on the 168 tests made on instances for which $z_{opt} \in [5; 14]$, optimally solving almost 40% of them. Their performance decreased consistently when the range increased, and no instance with $z_{opt} \geq 55$ could be solved to optimality. A similar behavior can be noticed for the heuristics, confirming that the range of the optimal solution value has considerable impact on the difficulty of an instance. It is interesting to observe that the students clearly beat the heuristics in finding optimal

Table 2: Evaluation by varying the optimum solution range

z_{opt} range	# tests	avg. perc. opt.		avg. abs. gap		avg. rel. gap	
		human	heuristic	human	heuristic	human	heuristic
[5; 14]	168	39.3%	35.1%	0.73	0.74	5.6%	5.7%
[15; 24]	289	32.2%	23.5%	1.15	1.03	5.9%	5.5%
[25; 34]	462	20.8%	16.7%	1.58	1.73	5.5%	6%
[35; 44]	124	9.7%	10.5%	2.96	2.19	7.5%	5.3%
[45; 54]	90	4.4%	0%	4.16	3.11	8.3%	6.2%
[55; 64]	73	0%	0%	4.25	3.05	7.3%	5.3%
overall	1206	22.5%	18%	1.83	1.65	6.1%	5.7%

solutions, probably because the visualization gives good opportunities for a clever post-processing of near-optimal solutions.

Table 3 takes the variation of the strip height into account. Both students and heuristics found better solutions for instances with small strip height. The strong impact of the strip height on the instance difficulty is also shown by the increase in the absolute and relative gaps when H increases. The reason for this behavior is probably that a small strip height gives few possibilities for the vertical placement of an item.

Table 3: Evaluation by varying the strip height H

H	# tests	avg. perc. opt.		avg. abs. gap		avg. rel. gap	
		human	heuristic	human	heuristic	human	heuristic
10	402	39.8%	39.3%	0.78	0.66	4.2%	3.8%
15	402	21.4%	14.7%	1.56	1.64	6.2%	6.7%
20	402	6.2%	0%	3.22	2.65	8%	6.7%
overall	1206	22.5%	18%	1.83	1.65	6.1%	5.7%

Table 4 shows the results for the two ranges adopted for the items dimensions: in the former range the items have comparable dimensions, while in the latter they are quite dissimilar from each other. This parameter only marginally affected the performance of the students who, however, performed slightly better for the wider range. The fact that such range appears to produce easier instances is confirmed by the heuristics, whose performance is clearly better for it.

Table 5 concerns the possibility of rotating the items (by 90 degrees). On average, allowing rotation helps the students in finding optimal or good-quality solutions. A similar behavior cannot be observed for the heuristics: when rotation is allowed they find less optimal solutions, but at the same time they provide better absolute and relative gaps. This could be produced by the tower-reduction post processing: when rotation is allowed, the algorithm repositions some long and thin items packed at the top of the strip, which helps in reducing the gap but not in reaching optimality.

Table 4: Evaluation by varying the item dimensions' range

item range	# tests	avg. perc. opt.		avg. abs. gap		avg. rel. gap	
		human	heuristic	human	heuristic	human	heuristic
$[H/4; H/2]$	603	21.4%	14.6%	1.97	2.01	6.2%	6.5%
$[1; 2H/3]$	603	23.5%	21.4%	1.69	1.29	6%	4.9%
overall	1206	22.5%	18%	1.83	1.65	6.1%	5.7%

Table 5: Evaluation by allowing/preventing rotation

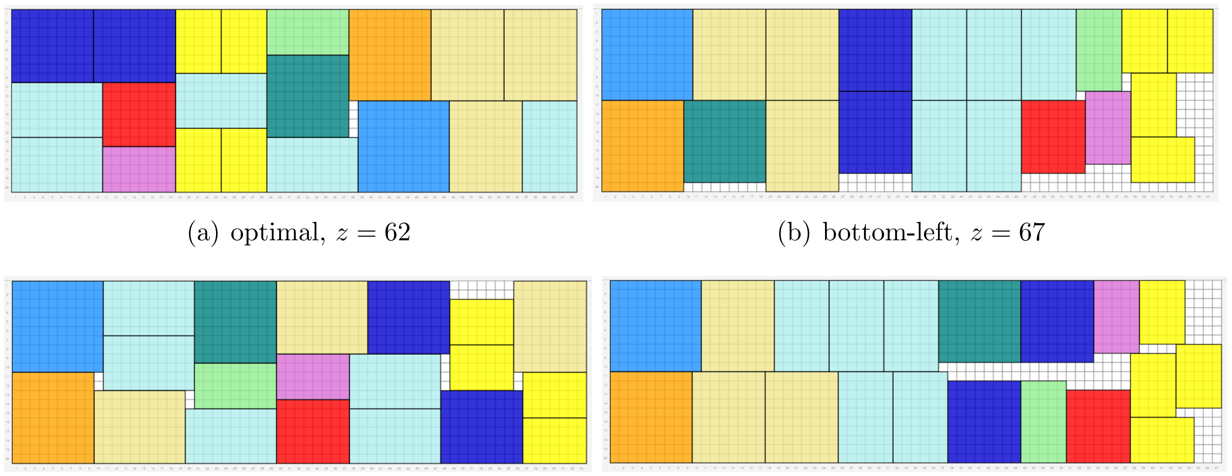
rotation	# tests	avg. perc. opt.		avg. abs. gap		avg. rel. gap	
		human	heuristic	human	heuristic	human	heuristic
not allowed	603	19.2%	18.4%	1.85	1.8	6.8%	6.6%
allowed	603	25.7%	17.6%	1.81	1.51	5.4%	4.9%
overall	1206	22.5%	18%	1.83	1.65	6.1%	5.7%

Table 6 presents results for the two additional instances that were included in the bundles. The instance having a supposed medium difficulty was attempted 71 times by the students, whereas the difficult one was attempted 130 times. Just one of these attempts resulted in an optimal solution (for the medium difficulty instance). The table shows the impact of the allowed time limit (from 4 to 7 minutes). For the medium difficulty instance, the time appears to be a relevant factor to decrease the gaps. For the difficult instance, instead, it does not allow to produce better solutions, probably because the instance was "too difficult".

Table 6: Solution quality for the difficult shared instances by varying time

time limit	Medium instance			Difficult instance		
	# tests	avg. abs. gap	avg. rel. gap	# tests	avg. abs. gap	avg. rel. gap
4	16	2.9	9.8%	33	5.0	7.3%
5	19	2.5	8.7%	32	3.7	5.5%
6	15	1.9	6.6%	31	4.9	7.1%
7	21	1.9	6.7%	34	4.9	7.2%
overall	71	2.3	7.9%	130	4.6	6.8%

The medium difficulty instance had $z_{opt} = 26$, which was found just by one student (in 4:44 minutes), whereas the heuristics found a strip of length 28. The difficult instance had $z_{opt} = 62$: the best student found a solution with $z = 63$ in 7:02 minutes, while the heuristic solutions (both the one found by bottom-left and the one found by best-fit) had $z = 67$. The four solutions obtained for this instance are shown in Figure 6, which gives some insight in the packing processes. The best student solution is very good: some small waste portions of the strip are accepted even at an early stage of the packing, but this results in a small final waste of just 24 units out of 1240. The possibility of rotating the items has been conveniently exploited (see the light blue items). The heuristic solutions are instead quite bad. Best-fit managed to produce a very dense packing at the beginning of the strip, but this resulted in a large waste towards the end. Bottom-left had a similar behavior. Note that both heuristics left for the final portion of the strip the yellow, red, and purple items. In particular, the yellow items appear to be difficult to pack, and the optimal solution is the only one that managed to conveniently pack them together with a light blue item.


(a) optimal, $z = 62$

(b) bottom-left, $z = 67$

(c) best human, $z = 63$

(d) best-fit, $z = 67$

Figure 6: Solutions of the difficult instance

5.3 Discussion

The main reason for the experiments was to evaluate the human performance on a difficult algorithmic task. The results have shown that human operators can perform quite well on two dimensional packing problems when compared with heuristic approaches. As highlighted by the discussion on the solution structure, humans have the capability to keep a global view of the solution construction. In this way, they obtain solutions that are good on average, in the sense that we do not observe a poor space utilization when the last items are inserted, as it happens for the heuristic algorithms. Indeed most heuristic (greedy) paradigms are based on the idea of inserting the next best item in the best position, which can lead to the need of inserting items with poor fitting at the last iterations. Clearly, humans have a limited capacity of keeping under control complexity, and hence their performance deteriorates when the instance to be solved becomes more complex in terms of number of items, length of the optimal strip, and height of the strip. Again, the advantage of human intuition is exploited when rotation is allowed, which instead does not appear to help heuristic algorithms.

6. Conclusions

This paper has presented `TwoBinGame`, a visual application developed in order to guide engineering students through the process of tackling difficult optimization problems via a heuristic approach. Although mainly conceived for didactical purposes, the application can be conveniently used in professional contexts. Practitioners can find it useful to build good quality solutions for real world two-dimensional packing instances arising, e.g., in the glass, steel or paper industry. On the other hand, `TwoBinGame` can help developers in the design of effective algorithms for the solution of such problems. The reported classroom experiments proved to be useful in testing students' skills versus exact and heuristic approaches.

References

[1] R. Alvarez-Valdes, F. Parreño, and J.M. Tamarit. A branch and bound algorithm for the strip packing problem. *OR Spectrum*, 31(2):431–459, 2009.

[2] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook. *The Traveling Salesman Problem - A Computational Study.* Princeton University Press, Princeton, NJ, 2006.

[3] C.R. Asfahl, S. Swayze, J. Lee, and R.R. Safford. An interactive computer training programming for industry. *Computers & Industrial Engineering*, 25(1):57–60, 1993.

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

[5] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. Can a machine replace humans in building regular expressions? a case study. *IEEE Intelligent Systems*, 31(6):15–21, 2016.

[6] J.E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33(1):49–64, 1985.

[7] L. Bodin and S.I. Gass. On teaching the analytic hierarchy process. *Computers & Operations Research*, 30(10):1487–1497, 2003.

[8] M.A. Boschetti and L. Montaletti. An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, 58(6):1774–1791, 2010.

[9] E.K. Burke, M. Hyde, G. Kendall, and J. Woodward. A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. *IEEE Transactions on Evolutionary Computation*, 14(6):942–958, 2010.

[10] E.K. Burke, G. Kendall, and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671, 2004.

[11] A. Caprara and M. Monaci. On the two-dimensional knapsack problem. *Operations Research Letters*, 32(1):5–14, 2004.

[12] B. Chazelle. The bottom-left bin packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, 32:697–707, 1983.

[13] G. Costa, C. D'Ambrosio, and S. Martello. A free educational java framework for graph algorithms. *Journal of Computer Science*, 6(1):87–91, 2010.

[14] G. Costa, C. D'Ambrosio, and S. Martello. GraphsJ 3: A modern didactic application for graph algorithms. *Journal of Computer Science*, 10(7):1115–1119, 2014.

[15] J.F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders' cuts for the strip packing problem. *Operations Research*, 62(3):643–661, 2014.

[16] L.L. Crumpton and E.L. Harden. Using virtual reality as a tool to enhance classroom instruction. *Computers & Industrial Engineering*, 33(1):217–220, 1997.

[17] M. Dell'Amico, S. Martello, and D. Vigo. A lower bound for the non-oriented two-dimensional bin packing problem. *Discrete Applied Mathematics*, 118:13–24, 2002.

[18] M. Delorme, M. Iori, and S. Martello. Logic based benders' decomposition for orthogonal stock cutting problems. *Computers & Operations Research*, 78:290–298, 2017.

[19] W.G. DePuy and G. Don Taylor. Using board puzzles to teach operations research. *INFORMS Transactions on Education*, 7(2):160–171, 2007.

[20] H. Dyckhoff and U. Finke. *Cutting and Packing in Production and Distribution.* Physica Verlag, Heidelberg, 1992.

[21] J. Egeblad and D. Pisinger. Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36(4):1026–1049, 2009.

[22] P. C. Gilmore and R. E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–119, 1965.

[23] P. Griffin. The use of classroom games in management science and operations research. *INFORMS Transactions on Education*, 8(1):1–2, 2007.

[24] M. Iori, S. Martello, and M. Monaci. Metaheuristic algorithms for the strip packing problem. In P. Pardalos and V. Korotkich, editors, *Optimization and Industry: New Frontiers*, pages 159–179. Kluwer, Boston, 2003.

[25] F. Llaugel and S. Confesor. Computer-aided statistical quality control learning. *Computers & Industrial Engineering*, 33(1–2):125–128, 1997.

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

[27] A. Lodi, S. Martello, M. Monaci, and D. Vigo. Two-dimensional bin packing problems. In V. Th. Paschos, editor, *Paradigms of Combinatorial Optimization: Problems and New Approaches*, chapter 5, pages 107–129. ISTE and John Wiley & Sons, 2014.

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

[29] J.N. MacGregor and Y. Chu. Human performance on the traveling salesman and related problems: A review. *The Journal of Problem Solving*, 3(2):1–19, 2011.

[30] J.N. MacGregor and T. Ormerod. Human performance on the traveling salesman problem. *Perception & Psychophysics*, 58(4):527–539, 1996.

[31] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319, 2003.

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

[33] H. Miyata, S. Watanabe, and Y. Minagawa. Performance of young children on traveling salesperson navigation tasks presented on a touch screen. *PLOS ONE*, 9(12):1–19, 2014.

[34] M.-P. Pacaux-Lemoine, D. Trentesaux, G. Zambrano Rey, and P. Millot. Designing intelligent manufacturing systems through human-machine cooperation principles: A human-centered approach. *Computers & Industrial Engineering*, 2017 (forthcoming).

[35] I.M.H. Vliegen and M.E. Zonderland. Game – the bedgame – a classroom game based on real healthcare challenges. *INFORMS Transactions on Education*, 2017 (fothcoming).

[36] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.