Minimum Tiling of a Rectangle by Squares

Michele Monaci
Università degli Studi di Padova
Via Gradenigo 6/A, 35131 Padova, Italia
monaci@dei.unipd.it

André Gustavo dos Santos
Universidade Federal de Viçosa
Av. P. H. Rolfs s/n, 36570-900 Viçosa, MG
andre@dpi.ufv.br

ABSTRACT

We consider a Two-Dimensional problem in which one is required to split a given rectangular *bin* into the smallest number of *items*. The resulting items must be squares, to be packed, without overlapping, into the bin so as to cover all the given rectangle. We present a mathematical model and a heuristic algorithm that is proved to find the optimal solution in some special cases. Then, we introduce a relaxation of the problem and present two different exact approaches based on this relaxation. Finally, we give some preliminary computational experiments on the performances of the algorithms on a set of randomly generated instances.

KEYWORDS. Two-Dimensional Packing, Mathematical Models, Computational Experiments.

Main Area: OC - Combinatorial Optimization, PM - Mathematical Programming

1. Introduction

We consider the problem of splitting a given rectangular *bin* into the smallest number of smaller square *items* having integer sides. These items have to be packed, without overlapping, with their edges parallel to one of the edges of the bin, so as to cover all the given rectangle. Similar problems have been addressed in the computational geometry literature, where a covering of a given region is called a *tiling*, and the smaller items are called *tiles*. Thus, the problem we consider will be denoted as the *Minimum Tiling of a Rectangle by Squares* (MTRS).

Several papers in the literature consider the case in which all the produced tiles must be different from each other. If this is the case, the rectangle is said to be *perfect*. Brooks et al. (1940) considered the problem of finding (if any) a perfect tiling of a given rectangle, and showed that a correspondence exists between feasible solutions of this problem and a certain class of planar electrical networks. Beaumont et al. (2002) considered the problems of covering a square into a set of p (say) rectangles that have a given area, so as to minimize either the sum of the perimeters of the rectangles or the maximum among the p perimeters. For these problems, motivated in heterogeneous parallel computing, they gave a proof of NP-completness and introduced approximation algorithms. Kenyon (1996) addressed MTRS and gave a tight logarithmic bound on the optimal solution value, where Walters (2009) proved polylogarithmic lower and upper bounds for the generalization of the problem to a higher dimension. Recently, Kurz (2012) considered the problem of covering a given $n \times n$ square with the minimum number of squares having side at most $n - 1$, and proposed an Integer Linear Programming (ILP) for this problem.

MTRS is strictly related to the *two dimensional bin packing* problem (2BP) as well. In this problem, one is required to allocate a given set of rectangular items to a minimum number of larger bins; the reader is referred, e.g., to Lodi et al. (2010) for a recent survey on this topic. However, two main differences arise between the two problems: (i) in MTRS we are required to define the dimensions of each items, whereas in classical packing problems there are an input of the problem; (ii) all the items to be packed must be squares.

MTRS arises as a subproblem of more complex packing problems, and has some practical applications, e.g., in telecommunications. Consider, for example, the IEEE 802.16-2009 standard which is the basis of Mobile WiMAX; in this protocol, data packets have to be transmitted from a base station to mobile users, and transmission is implemented using different time slots and different frequencies. This can be modelled as a two-dimensional packing problem in which a rectangular *data bin* is used to transmit some rectangular *data packets*; in this model, widths and heights represent time slots and frequencies, respectively, see Lodi et al. (2011) for more details. Each data packet that is transmitted requires additional information to be stored (and coded/decoded), that is possibly reduced in case the packet is sent as a square instead of a rectangle. To maximize the throughput of the system, one is thus interested in filling the entire data bin with the smallest number of square items.

The paper is organized as follows. In Section 2 we introduce an Integer Linear Programming (ILP) model for MTRS, while Section 3 gives a heuristic algorithm that provides the optimal solution in some special cases. In Section 4 we introduce a mathematical model for a relaxation of the problem, and strengthen this relaxation by adding some valid inequalities; two different exact algorithms based on this relaxation are then presented. Finally, Section 5 presents a preliminary computational analysis of the proposed algorithms on a set of randomly generated instances, and Section 6 draws some conclusions.

Throughout the paper we assume, without loss of generality, that H and W are positive integers. Noting that the cases $W = 1$ (or $H = 1$) and $W = H$ would lead to trivial optimal solutions (with value H and 1, respectively), we will further assume that $1 < W < H$, possibly rotating the rectangle by 90 degrees if necessary.

2. Problem formulation

In this section we give a formal description of the problem we consider, and introduce a mathematical formulation for its solution.

We are given a rectangular bin having integer width W and integer height H. The *Minimum Tiling of a Rectangle by Squares* (MTRS) problem requires to define n square items, so that:

- each item j has an integer side a_j;
- items are orthogonally allocated to the bin without overlapping;
- the set of square items entirely covers the given bin; and
- the number of produced items is a minimum.

To provide an Integer Linear Programming (ILP) formulation for MTRS, we make use of a Cartesian system having axes x and y and assume that the bottom left corner of the bin is at coordinate $(0, 0)$. Then, we note that the size of each produced square in any feasible solution is bounded by W (as we assume $W < H$), and define the following set of decisional variables:

$$\alpha_{ijt} = \begin{cases} 1 & \text{if an item of size } t \text{ is placed in position } (i, j); \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

for $t = 1, \ldots, W$, $i \in W_t = \{1, \ldots, W - t + 1\}$ and $j \in H_t = \{1, \ldots, H - t + 1\}$.

Thus, an ILP model for MTRS is as follows

$$\min \sum_{t=1}^{W} \sum_{i=1}^{W_t} \sum_{j=1}^{H_t} \alpha_{ijt} \tag{2}$$

$$\sum_{t=1}^{W} \sum_{u=\min(0,i-t+1)}^{\max(i,W-t+1)} \sum_{j=\min(0,j-t+1)}^{\max(j,H-t+1)} \alpha_{uvt} = 1 \quad i = 1, \ldots, W; j = 1, \ldots, H \tag{3}$$

$$\alpha_{ijt} \in \{0, 1\} \quad t = 1, \ldots, W; i \in W_t; j \in H_t. \tag{4}$$

The objective function (2) minimizes the number of items that are produced; constraints (3) impose that any unit square of the bin, with bottom left corner, say, at coordinate (i, j), is occupied by exactly one item.

The formulation above is similar to the ILP model proposed for the two-dimensional bin packing problem by Beasley (1985), and by Kurz (2012) for the problem of splitting an $n \times n$ square into square items having size at most $n - 1$. The model has $W^2\, H$ variables and $W\, H$ constraints; this may prevent the possibility of directly using this model in practice even for reasonable values of the W and H parameters. On the contrary, this model can be solved in an efficient way for small values of W and H, possibly producing approximate solutions for MTRS in case the optimum cannot be computed.

3. Heuristic Solution of MTRS

In this section we present a solution approach for MTRS based on a dynamic programming scheme; this algorithm is particularly suited for those instances for which the optimal solution corresponds to a pattern that is guillotinable, i.e., in which each item can be cut with a sequence of edge to edge cuts parallel to the edges of the bin (see the left packing in Figure 1). This is a relevant special case of the problem, and of Two-Dimensional packing problems as well, which has received considerable attention due to relevant real-world applications. For example, many automatic machines do not require the presence of a human employee, but can only produce cuts that go from one side to the other of the bin, at the expense of deteriorating the quality of the produced solutions.

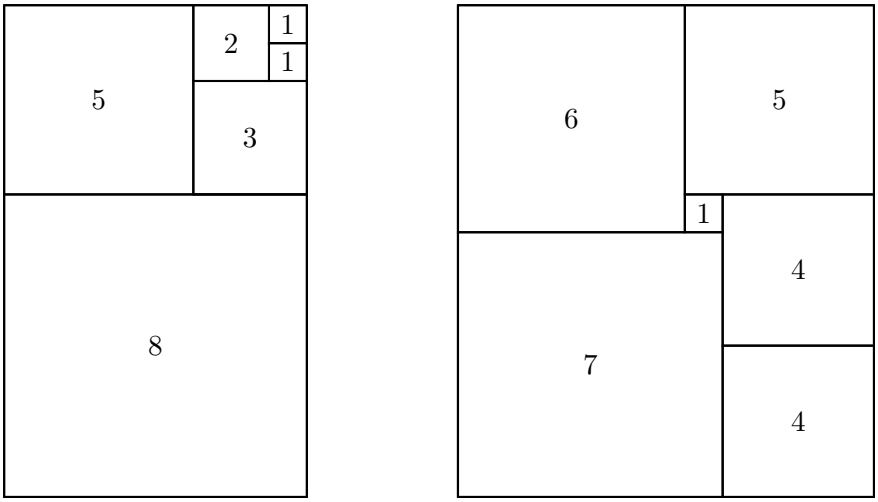

Figure 1: Example of guillotine and non guillotine patterns (the numbers denote the sizes of the items).

For a given rectangle $W \times H$, the dynamic programming algorithm computes the optimal solution value $F(W, H)$ by means of the following recursion scheme:

$$F(W,H) = \begin{cases} F(H,W) & \text{if } W > H; \\ 0 & \text{if } W \leq 0; \\ 1 & \text{if } W = H; \\ \min(\min\{F(k,H) + F(W-k,H); k = 1,\ldots,W\}, & \\ \quad \min\{F(W,k) + F(W,H-k); k = 1,\ldots,H\}) & \text{otherwise.} \end{cases} \quad (5)$$

with the obvious initialization $F(k, 0) = F(0, k) = 0, \ \forall k$. Thought it requires the computation of all the $W \times H$ entries of F, the algorithm is very fast in practice.

The following theorem states that the algorithm provides an optimal solution to MTRS, if an optimal solution satisfying guillotine constraint exists.

Theorem 1 *The recursion scheme given in (5) produces the optimal MTRS solution for guillotinable among those that satisfy guillotine constraint.*

Proof. Without loss of generality, consider the smallest (W, H) instance for which the solution found is not optimal, i.e., the recursion scheme is exact for all $\overline{W} \times \overline{H}$ rectangles such that $\overline{W} \leq W$, $\overline{H} \leq H$ and one of the two inequalities is strict. As the solution found is guillotinable, it is produced by a first cut which divides the bin into two smaller parts, and $F(W, H)$ is the sum of the associated optimal values. Thus, either a better solution exists that uses a different first cut, or the solutions of the smaller parts are not optimal. This latter case is ruled out because these parts are both smaller than (W, H). The first case cannot occur as the algorithm considers all possible vertical and horizontal positions for applying the first cut. □

Unfortunately, there are situations in which the optimal solution corresponds to a non guillotinable pattern (see, e.g., the right packing in Figure 1); in these cases, the algorithm produces a heuristic (non necessarily optimal) solution, i.e., an upper bound on the optimal solution value.

4. Exact solution of MTRS

In this section we examine three exact approaches to the solution of MTRS. The first one applies an ILP solver to the formulation of Section 2, whereas the remaining two schemes are based on a simple relaxation that is possibly strengthened by the addition of valid inequalities in two different ways.

4.1. Approach 1: Direct use of an ILP solver

An immediate way for solving MTRS is to run any ILP solver on the mathematical model (2)–(4) given in Section 2. To take full advantage of the internal heuristics that are commonly embedded in commercial ILP solvers one can warm start the model with a heuristic solution. For example, one can define a "dummy" solution in which the rectangular bin is split into $W \times H$ unit-square items or, even better, derive a heuristic solution by executing the heuristic algorithm described in Section 3.

As already anticipated, this simple formulation may be solved in a very efficient way for small values of W and H. For larger instances, the solver may not be able to provide an optimal solution, but can possibly produce an improved heuristic solution. However, large values of W and H produce a huge number of variables and constraints, which makes it impossible to solve the associated model and, in some cases, even to define it due to memory requirement.

4.2. Approach 2: Enumerate-and-cut algorithm

In this section we describe an alternative approach based on the definition of a simple ILP model that describes a relaxation of the problem; valid inequalities are added on the fly to the formulation, possibly in an iterative fashion, until a feasible solution is produced.

4.2.1. A one-dimensional relaxation

A simple lower bound on the optimal solution value of a MTRS instance can be obtained by solving the following ILP model

$$\min \sum_{t=1}^{W} x_t \tag{6}$$

$$\sum_{t=1}^{W} t^2 x_t = W\,H \tag{7}$$

$$x_t \geq 0 \text{ integer} \qquad t = 1, \ldots, W \tag{8}$$

$$\sum_{k=t}^{W} x_k \leq U_t \qquad t = 1, \ldots, W \tag{9}$$

$$\sum_{t>W/2} t\,x_t \leq H \tag{10}$$

$$\sum_{t>H/2} t\,x_t \leq W \tag{11}$$

where each variable x_t indicates the number of $t \times t$ items in the solution, $\widetilde{z}$ denotes the value of a feasible solution for MTRS and

$$U_t := \min(\widetilde{z}, \lfloor \frac{W}{t} \rfloor \lfloor \frac{H}{t} \rfloor) \tag{12}$$

represents the maximum number of $t \times t$ items to be considered (see below).

The model defined by (6)–(8) corresponds to the 1-dimensional relaxation of the problem in which only the area of each candidate item (and of the bin) is taken into account. A similar relaxation was addressed for the two-dimensional knapsack problem by Caprara and Monaci (2004), who established the worst-case performance analysis of the resulting bound. The resulting model corresponds to a Change-Making Problem, which has been proved be NP-hard in the general case, though efficient algorithms for its solution are available, see, e.g., Martello and Toth (1990).

As to constraints (9), note that in any feasible solution the maximum number of $t \times t$ items is bounded by $\lfloor \frac{W}{t} \rfloor \lfloor \frac{H}{t} \rfloor$. As we assume a feasible solution of value $\widetilde{z}$ is available, each variable x_t can be bounded by U_t, defined by (12). This immediately leads to inequalities (9) through a lifting operation.

Finally, inequality (10) bounds the total maximum height of "large" items, i.e., items that are larger than half of the width of the bin and cannot be packed side by side; similarly, (11) gives an upper bound on the total width of "tall" items.

4.2.2. Strengthening the relaxation

Let us add to the ILP relaxation of the previous model the following additional variables

$$y_{tk} = \begin{cases} 1 & \text{if } x_t = k; \\ 0 & \text{otherwise} \end{cases} \qquad (t = 1, \ldots, W; k = 1, \ldots, U_t) \tag{13}$$

that are linked to the x variables as follows

$$x_t = \sum_{k=1}^{U_t} k\, y_{tk} \quad t = 1, \ldots, W \tag{14}$$

and should satisfy immediate constraints that impose at most one such variable be selected for each possible size t, i.e.,

$$\sum_{k=1}^{U_t} y_{tk} \leq 1 \quad t = 1, \ldots, W. \tag{15}$$

Using the y variables, it is possible to add to the formulation a number of constraints, as stated by the following theorems.

Theorem 2 *The following inequalities*

$$\sum_{p=1}^{W-t} p^2\, x_p \geq t\,(W-t)\, x_t \quad t = \lfloor \frac{W}{2} \rfloor + 1, \ldots, W-1 \tag{16}$$

and

$$\sum_{p=1}^{H-t} p^2\, x_p \geq t\,(H-t)\, x_t \quad t > \lfloor \frac{H}{2} \rfloor + 1, \ldots, W \tag{17}$$

are valid for any feasible packing.

Proof. Let us prove the validity of (16) for a given t value. First observe that the inequality is redundant if either $t = W$ (in which case the constraint is omitted) or $x_t = 0$. Otherwise, all the $t \times t$ must be packed one above the other, leaving a free space having width $W - t$ and height $t\, x_t$. This free space must be entirely filled with items whose side is at most $W - t$, which yields (16). Equations (17) can be derived with a similar reasoning, swapping the role of W and H, and can be imposed also for $t = W$. □

Theorem 3 *Let $k \in \{1, \ldots, U_W\}$ and define $\overline{W} = \min(W, H - k\,W)$ and $\overline{H} = \max(W, H - k\,W)$. Then, the following inequalities*

$$\sum_{p=1}^{\overline{W}-t} p^2\, x_p \geq t(\overline{W}-t)x_t - M(1-y_{Wk}) \quad t = \lfloor \frac{\overline{W}}{2} \rfloor + 1, \ldots, \overline{W}-1 \tag{18}$$

and

$$\sum_{p=1}^{\overline{H}-t} p^2\, x_p \geq t(\overline{H}-t)x_t - M(1-y_{Wk}) \quad t = \lfloor \frac{\overline{H}}{2} \rfloor, \ldots, \overline{W} \tag{19}$$

where M is a "sufficiently large" value, are valid for any feasible packing.

Proof. For each value of k, all the associated inequalities are deactivated if $y_{Wk} = 0$, i.e., in case the number of $W \times W$ items in the solution is different from k. Otherwise, all such items must be packed one above the other, leaving a residual bin with width W and height $\overline{H} = W - k\,W$. Applying Theorem 2 to this residual bin, one immediately gets (18) and (19). □

The following Theorem characterizes inequalities similar to those provided by Theorem 2, but addressing "small" items, i.e., having width not larger than $W/2$.

Theorem 4 *The following inequalities*

$$\sum_{p=1}^{W-t} p^2\, x_p \geq t(W-t)y_{t1} \quad t = 1, \ldots, \lfloor \frac{W}{2} \rfloor \tag{20}$$

and

$$\sum_{p=1}^{H-t} p^2\, x_p \geq t(H-t)y_{t1} \quad t = 1, \ldots, \lfloor \frac{H}{2} \rfloor \tag{21}$$

are valid for any feasible packing.

Proof. The proof is similar to that of Theorem 2; the only difference is that we cannot use x_t variables in the right hand side, as $t \times t$ items can be packed side by side, and have to weaken the constraint by using variables y_{t1}. □

4.2.3. Exact solution of MTRS

Given a solution of the model, one can define the corresponding set of squares by introducing k identical $t \times t$ items for each variable y_{tk} that takes value 1. We say that such a set of items is *packable* if it is possible to allocate all the items, without overlapping, to the given rectangular bin. Note that, due to constraint (7), the produced set of items always covers the entire rectangular bin, thus infeasibility can be due to two-dimensional aspects only. To cut infeasible solutions, one can add to the formulation the following clique inequalities

$$\sum_{(t,k)\in C} y_{tk} \leq |C| - 1 \quad C \in \mathcal{C} \tag{22}$$

where $\mathcal{C}$ denotes the set of (exponentially many) subset of y variables associated with sets of items C that turn out to be not packable. This approach can be seen as a Benders' decomposition, see Benders (1962), in which the feasibility check for the slave is an NP-hard problem, as it corresponds to the problem of checking whether a given set of items fits into a bin or not. Indeed this feasibility checks, described in the next section, turns out to be by far the most time consuming part of the computation for our approach.

4.2.4. Checking feasibility

The problem of checking the feasibility of a given set of items can be solved in two different ways:

- pack all items into the minimum number of $W \times H$ bins, and check if a solution exists that uses only one bin;
- pack all items into a unique strip of width W and infinite height, and check if a packing exists with total height not larger than H.

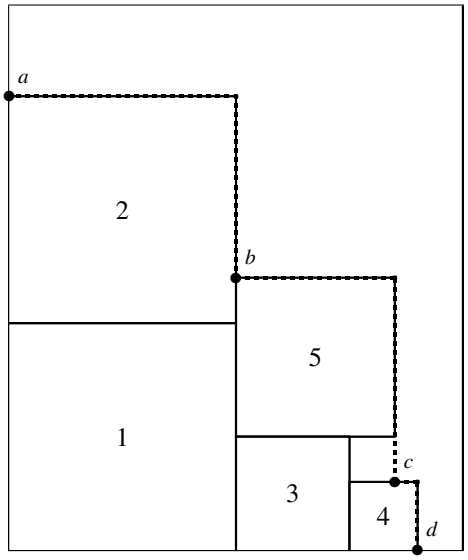

Figure 2: Envelope and corner points associated with item set $S = \{1, 2, 3, 4, 5\}$.

The former corresponds to solving a 2BP instance, whereas the latter asks for the solution of a *two dimensional strip packing* (2SPP) problem. In our algorithm we used the latter option, and implemented an enumeration scheme that packs one item at a time according to the concepts of *envelope* and *corner points* (see Figure 2). At each node of the enumerative tree, denoting by S the set of items that are actually allocated, we compute the associated corner points and generate a number of descendant nodes, each associated to the placement of each item $j \notin S$ in each corner point, see Martello et al. (2003) for details.

To speed up enumeration, at each node we use only simple fathoming criteria, based on the consideration that any feasible solution cannot leave uncovered areas. Thus, a backtracking occurs if one of the following conditions holds:

- the area below the current envelope is strictly smaller than the area of the items is S;
- a corner point exists in which no item $j \notin S$ can be allocated;
- an item $j \notin S$ exists that cannot be allocated to any corner point.

The algorithm is halted as soon as a feasible solution is found, i.e., when all items have been allocated without overlapping to the given rectangle.

4.3. Approach 3: Branch-and-cut algorithm

A third exact algorithm can be obtained by integrating the feasibility check within the ILP solver. In particular, we assume that the solver can be controlled through a callback function invoked each time the incumbent is going to be updated—as it happens in many modern solvers. We can thus run the solver on the relaxed model described in sections 4.2.1 and 4.2.2, using the feasibility check described in Section 4.2.4 every time a candidate set of items is found. The resulting algorithm, that resembles the scheme proposed by Miliotis (1976) for the Travelling Salesman Problem, may turn out to be advantageous as it explores a single tree, including the (generally time consuming) root node that involves preprocessing, cut generation, and so on. On the other hand, the use of callback functions may deactivate some properties of the solver, which can turn in a worsening of the algorithm's performances.

Intermediate schemes that check for feasibility with a given frequency can be devised, though their design is out of the scope of the present paper.

5. Computational experiments

All algorithms were coded in C language and run on an Intel Xeon E3-1220 V2 in single-thread mode with a time limit of 1,800 seconds per instance. All the ILP models were solved by using IBM-ILOG Cplex 12.5.1 (CPLEX in the following), possibly using callback functions.

To test the effectiveness of our algorithms, we randomly generated a small set of instances, as follows: the height H is randomly chosen as an integer from a uniform distribution in the interval $[2, \overline{H}]$, where $\overline{H}$ is a parameter. Then, the width W of the rectangle is generated as a random integer in two different ways, so as to define two different classes:

- R *rectangular* instances: for a given H, the value of W is uniformly distributed in $[H/4, 3H/4]$, i.e., the height of the bin is quite larger than its width;
- Q *quasi-square* instances: for a given H, the value of W is uniformly distributed in $[H - 5, H - 1]$. For these problems there is a small difference between H and W, which produces bins that are quite similar to a square.

As to the maximum height of the bin, we defined:

- *small* instances ($\overline{H} \leq 50$),
- *medium* instances ($50 < \overline{H} \leq 100$), and
- *large* instances ($100 < \overline{H} \leq 250$).

For each class and value of $\overline{H}$, we randomly generated 10 instances; thus a benchmark of 60 instances was obtained. Table 1 reports the outcome of our preliminary computational experiments for each class and range for $\overline{H}$. To evaluate the performances of the heuristic algorithm of Section 3 we computed, for each instance, the ratio between the solution provided by the algorithm and the optimal solution value (in case the optimal was not available, the best known solution was used). The average value of such ratio is reported in column "ratio", whereas column "#best" gives the number of cases in which the heuristic algorithm produced the (either optimal or) best known solution for a given instance. As to the exact algorithms of Section 4, we give the number of instances solved to proven optimality and the average computing time in seconds (columns "#opt" and "time", respectively). In addition, for the first exact approach we report the total number of cases in which CPLEX could not be started due to excessive size of the model ("#fail"), whereas for the remaining two algorithms we give the average number of feasibility executions of the two-dimensional packing routine ("#check").

Instances		Heuristic		CPLEX			Enumerate-and-cut			Branch-and-cut		
		ratio	#best	#opt	time	#fail	#opt	time	#check	#opt	time	#check
H	small	1.162	5	10	77	0	7	549	734	7	542	1826
	medium	1.050	9	1	1333	2	0	1800	1347	1	1648	6192
	large	1.000	10	0	0	10	0	1800	632	0	1800	1910
R	small	1.000	10	10	0	0	10	0	1	10	0	1
	medium	1.000	10	7	817	0	4	1177	1198	7	660	2967
	large	1.000	10	0	360	8	1	1797	910	3	1448	3573

Table 1: Initial heuristic and exact algorithms on PSS instances

Computational experiments show that the heuristic algorithm provides very good solutions, which can be improved only in a few cases by the exact algorithms. The first exact algorithm

is very effective for small instances, whereas is fails due to memory requirement for most of the large instances (namely, in 18 cases out of 20). As to the remaining two algorithms, branch-and-cut proves to perform better than enumerate-and-cut, at least on our small testbed, as it solves a larger number of instances and has an average smaller computing time, although the number of calls to the routines that check packing feasibility is considerably larger.

6. Conclusions

In this paper we introduced a Two-Dimensional packing problem in which one is required to split a given rectangular bin into a minimum number of non-overlapping square items. For this problem we proposed an ILP model and a heuristic algorithm that is proved to produce the optimal solution in relevant situations. Then, we proposed a mathematical model for a relaxation of the problem, which is embedded into two enumerative schemes. Preliminary computational experiments are provided to tested the effectiveness of the algorithms on a small set of randomly generated instances. Our preliminary results suggest that the direct use of an ILP solver is not suitable when large instances are addressed. In addition, the initial relaxation embedded into the remaining exact algorithms has to be strengthened to face with hard instances, which requires the definition of new valid inequalities. Future work should also consist in creating a larger set of instances, possibly including problems with some known characteristics (e.g., instances for which the optimal solution is always guillotinable) so as to draw more accurate conclusions on the performances of the algorithms above.

Acknowledgment

This project was partially supported by CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Grant 030479/2013-01, PVE / Ciência sem Fronteiras).

References

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

Beaumont, O., Boudet, V., Rastello, F. e Robert, Y. (2002), Partitioning a Square into Rectangles: NP-Completeness and Approximation Algorithms, *Algorithmica*, 34, 217–239.

Benders, J.F. (1962), Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik*, 4, 238–252.

Brooks, R., Smith, C., Stone, A. e Tutte, W. (1940), The Dissection of Rectangles into Squares, *Duke Mathematics journal*, 7, 312–340.

Caprara, A. e Monaci, M. (2004), On the two-dimensional knapsack problem, *Operations Research Letters*, 32, 5–14.

Kenyon, R. (1996), Tiling a Rectangle with the Fewest Squares, *Journal of Combinatorial Theory*, 76, 272–291.

Kurz, S. (2012), Squaring the Square with Integer Linear Programming, *Journal of Information Processing*, 20, 680–685.

Lodi, A., Martello, S., Monaci, M., Cicconetti, C., Lenzini, L., Mingozzi, E., Eklund, C. e Moilanen, J. (2011), Efficient two-dimensional packing algorithms for mobile WiMAX, *Management Science*, 57, 2130–2144.

Lodi, A., Martello, S., Monaci, M. e Vigo, D., Two-dimensional bin packing problems, V.Th.Paschos (Ed.) *Paradigms of Combinatorial Optimization*, Wiley/ISTE, 107–129, 2010.

Martello, S., Monaci, M. e Vigo, D. (2003), An exact approach to the strip packing problem, *INFORMS Journal on Computing*, 15, 310–319.

Martello, S. e Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, 1990.

Miliotis, P. (1976), Integer programming approaches to the travelling salesman problem, *Mathematical Programming*, 10, 367–378.

Walters, M. (2009), Rectangles as sum of squares, *Discrete Mathematics*, 309, 2913–2921.