A Reinforced Tabu Search Approach for 2D Strip Packing

Giglia Gómez-Villouta, Jean-Philippe Hamiez*, and Jin-Kao Hao
Université d'Angers / LERIA, 2 boulevard Lavoisier, 49045 Angers CEDEX 01, France
{gomez, hamiez, hao}@info.univ-angers.fr

ABSTRACT

*This paper deals with a particular "packing" problem, namely the **two dimensional strip packing problem**, where a finite set of objects have to be located in a strip of fixed width and infinite height. The variant studied here considers regular items (they are rectangular to be more precise) that must be packed without overlap, rotations being not allowed. The objective is to minimize the height of the resulting packing. For this problem, we present a local search algorithm based on the well-known tabu search method. While tabu search is sometimes said to be a **generic** search engine (or "metaheuristic"), we reinforce two important components of our tabu search strategy to try to include problem knowledge: The fitness function incorporates a measure related to the empty spaces and diversification relies on a set of historically "frozen" objects. The resulting reinforced tabu search approach is evaluated on a set of well-known hard benchmark instances and compared with some state-of-the-art algorithms.*

***Keywords**: Tabu search, strip packing, guided diversification*

1. INTRODUCTION

In packing problems, "small" items (also called "boxes", "modules", "objects", or "pieces" e.g.) of various shapes (regular or not) and dimensions have to be packed (i.e. located) without overlap, with rotation and "guillotine" cuts (see Figure 1) allowed or not, in other "larger" items of regular forms or not[1]. These larger objects are usually called "containers" or "pallets" for the three-dimensional cases (3D, all dimensions fixed or infinite height) and "bins", "plates", or "(stock) sheets" (all dimensions fixed) or "strips" (only width fixed, infinite height) in 2D.

Objectives are, for instance, to minimize the number of containers or to maximize the material used (hence to minimize the "trim loss", i.e. the wasted area). A huge number of practical or industrial applications are concerned, such as truck loading, cardboard packing, facilities, fashion, plant, machine, newspaper, or web page layout design, VLSI macro-cell placement, glass, cloth, metal, paper, or wood industries, dynamic memory allocation, meta-computing, multi-processor or publicity scheduling for instance. This may explain why (commercial) software packages exist, sometimes for a long time. See (Dowsland & Dowsland, 1992; Lodi et al., 2002; Wäscher et al., 2007) just to mention a few surveys.

* Corresponding author.

[1] Expressions like "(pallet) loading", "containment", "marker making", "nesting", "(layout) design", "placement", "(resource) allocation", or "(stock) cutting" e.g. are also sometimes employed to refer to this type of problems.

These problems are usually generalizations or restrictions of the well-known NP-hard (or NP-complete for decision variants) quadratic assignment, bin packing, knapsack, or quadratic set covering problems. Packing problems are thus optimization or satisfaction problems (sometimes with multiple objectives) that are NP-hard or NP-complete in the general case (Fowler et al., 1981; Garey & Johnson, 1979).

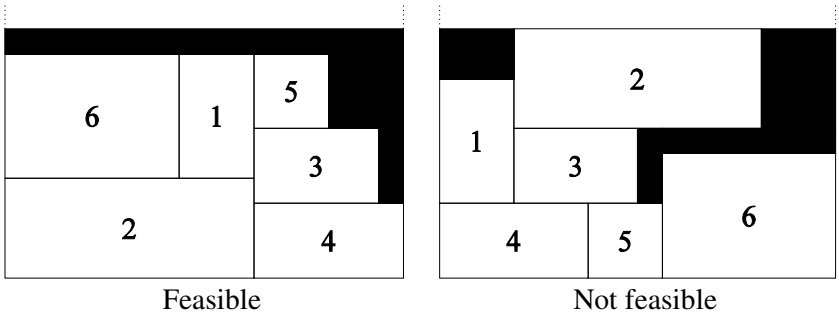


Figure 1. The guillotine constraint imposes a pattern where the items can be extracted by a sequence of "edge-to-edge" cuts, i.e. the cutting tool cannot change of direction within the same cutting step (dark zones map wasted areas).

This paper is dedicated to the NP-hard 2D Strip Packing Problem (2D-SPP) which can be informally stated as follows: Given a finite set of objects, pack all of them without overlap in one strip of an infinite height and fixed width (also called "basis") while minimizing the height of the resulting packing. The guillotine constraint is not considered here. Furthermore, all objects are regular (rectangular to be more precise) and cannot be rotated, i.e. they have a fixed orientation.

In this paper, we introduce CTS (for "Consistent Tabu Search"), a reinforced tabu search algorithm dedicated to the 2D-SPP. Compared with previous algorithms for the 2D-SPP, our CTS has several notable features. First, it handles a *consistent* neighborhood. Second, CTS evaluates packings, possibly partial, using *problem knowledge*. Finally, our algorithm includes a *diversification* mechanism relying on a set of historically "frozen" rectangles. Computational results suggest that CTS may be of great interest to solve the 2D-SPP.

In the two next sections, the 2D-SPP is formally stated and a brief description of various existing methods is given. Section 4 is devoted to the detailed presentation of our dedicated tabu search algorithm for the 2D-SPP. Experimental results are finally shown in Sect. 5 on a set of well-known benchmarks and compared with previous attempts including best performing state-of-the-art algorithms.

2. PROBLEM FORMULATION

A "strip" is a 2D vertical space with fixed width W and infinite height, see Figure 2. The bottom-left (BL) corner of the strip stands for the (0, 0) point of an xy-plane where the x-axis (respectively y-axis) is the direction of the width (resp. height).

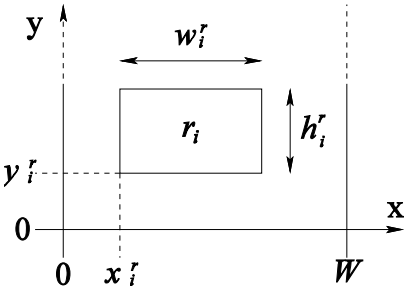


Figure 2. A strip with basic notations.

The set of $n \geq 2$ *R*ectangles to be positioned in the strip is $R = \{r_1,\ldots, r_n\}$ where the weight (resp. height) of each object $r_{1\leq i\leq n}$ is $0 < w_i^r \leq W$ (resp. $h_i^r > 0$).

According to these notations, the 2D-SPP is then to determine the (x_i^r, y_i^r) coordinates of the BL corner of all rectangles (i.e. the location of each $r_i \in R$ in the strip) so as to minimize the height of the resulting packing. This can be formally stated as follows:

$$\text{Minimize } f: \quad \max_{1\leq i\leq n}\{y_i^r + h_i^r\} \tag{1}$$

$$\text{Subject to:} \quad 0 \leq x_i^r \leq W - w_i^r \wedge y_i^r \geq 0 \tag{2}$$

$$\wedge(x_i^r \geq x_j^r + w_j^r \vee x_i^r + w_i^r \leq x_j^r \tag{3}$$

$$\vee y_i^r \geq y_j^r + h_j^r \vee y_i^r + h_i^r \leq y_j^r)\ . \tag{4}$$

where (2) forces each rectangle r_i to be inside the strip and (3–4) specify that any two r_i and $r_{j\neq i}$ objects must not overlap horizontally or vertically.

3. LITERATURE REVIEW

Only a few *exact* methods are available for 2D-SPP or closely related problems (see Sect. 3.1), while a wide range of *approximate* heuristics has been reported (Sect. 3.2). Among these strategies, the *greedy randomized adaptive search procedure* from Alvarez-Valdes et al. (2008) and the hybrid *hyperheuristic + intensification / diversification walk* strategy from Neveu et al. (2008) are probably the best performing ones for 2D-SPP (these two effective methods are briefly summarized in Sect. 3.2).

3.1. Exact methods

These approaches are often based on implicit enumeration of the search space. They are thus usually limited to **small** instances. However, given sufficient time, they can in theory either find a solution (optimal for optimization problems) or prove that none exists (for satisfaction problems).

The *branch-and-bound* algorithm presented by Martello et al. (2003) relies on a 2D-SPP relaxation (basically, cutting each object $r_i \in R$ into items of height lower than h_i^r) that can be solved as a particular NP-hard one-dimensional bin-packing problem (with "side" constraints). This leads to a lower bound on f_{OPT}, the *OPT*imum value of (1), better than those previously proposed.

Recently, two *branch-and-bound* approaches faster than those from Martello et al. (2003) were designed by Kenmochi et al. (2009). Including many components, dynamic programming cuts for instance, the 2D-SPP *optimization* problem (as formally defined in Sect. 2) is reduced here to the "perfect packing problem" which is a *satisfaction* problem (determine if a packing without wasted space exists), sometimes by adding new objects.

Other (recent) exact methods related to 2D-SPP, or bounds, can be found e.g. in (Belov et al., 2009; Clautiaux et al., 2008; Soh et al., 2008).

3.2. Approximate heuristics

These approaches include e.g. "greedy" constructive strategies or "(meta) heuristics". While they loss the completeness of exact methods, they can handle **large** instances and usually obtain good quality

solutions in reasonable time. In this section, we review two best-performing algorithms and discuss about other representative heuristics related to 2D-SPP.

Alvarez-Valdes et al. (2008) proposed a *Greedy Randomized Adaptive Search Procedure* (GRASP) for 2D-SPP. It is a multi-start scheme that iteratively builds a feasible solution in a greedy way following various dynamic pseudo-random selection rules. This solution is then modified to try to correct previous wrong random choices, for use in the next greedy step, by different "simple local search" algorithms.

Neveu et al. (2008) developed an hybrid approach combining a *hyperheuristic* (HH) with the *intensification / diversification walk* strategy (IDW). It starts (HH phase) with a greedy packing, where the selection rule possibly alternates between different criteria, e.g. in a round robin manner. This solution is then perturbed (IDW phase) by iteratively moving an object at the top of the strip below its current location. If such a perturbation generates overlaps, they are repaired using a greedy heuristic (possibly randomly chosen).

Other representative approaches for 2D-SPP (or closely related variants) include, for instance:

- Simulated annealing (Burke et al., 2009; Hopper & Turton, 2001; Soke & Bingul, 2006).
- Tabu search (Alvarez-Valdes et al., 2007; Błażewicz et al., 2004; Hamiez et al., 2009; Iori et al., 2003).
- Iterated local search (Imahori et al., 2005, 2003).
- Genetic and evolutionary algorithms (Beasley, 2004; Bortfeldt, 2006; Gómez-Villouta et al., 2008; Gonçalves, 2007; Hopper & Turton, 2001; Iori et al., 2003; Soke & Bingul, 2006).
- Hybrid (meta)heuristics (Beltrán Cano et al., 2004; Ibaraki et al., 2008; Iori et al., 2003; Mir & Imam, 2001; Neveu et al., 2008; Yeung & Tang, 2004).
- Hyperheuristics (Araya et al., 2008; Garrido & Riff, 2007; Terashima-Marín et al., 2005).

Note that *polynomial-time approximation schemes* with (asymptotic or absolute) performance guarantee are also available for these problems (Harren & van Stee, 2009; Jansen & van Stee, 2005).

4. CTS: A CONSISTENT TABU SEARCH FOR 2D-SPP

We first recall here the fundamentals of tabu search (Sect. 4.1) and how the problem is addressed (Sect. 4.2). Next sections (4.3–4.8) describe then the problem-specific components of our CTS, where all p variables (with subscripts) are *p*arameters whose values will be given in the experimentation part (Sect. 5.1). The general CTS procedure is finally summarized in Sect. 0.

4.1. A brief review of tabu search

Tabu search is an advanced metaheuristic designed for tackling hard combinatorial optimization or satisfaction problems (Glover & Laguna, 1997). It relies on a neighborhood relation as well as some forms of memory and learning strategies to explore effectively a search space. Let (S, f) be our search problem where S and f are respectively the search space and the optimization objective.

A "neighborhood" N over S is any function that associates to each individual $s \in S$ some solutions $N(s) \subset S$. Any solution $s' \in N(s)$ is called a neighboring solution or simply a neighbor of s. For a given neighborhood N, a solution s is a "local optimum" with respect to N if s is the best among the solutions in $N(s)$. The notion of neighborhood can be explained in terms of the "move" operator. Typically applying a move μ to a solution s changes slightly s and leads to a neighboring solution s'.

This transition from a solution to a neighbor is denoted by $s' = s \oplus \mu$. Let $\Gamma(s)$ be the set of all possible moves which can be applied to solution s, then the neighborhood $N(s)$ of s can be defined by: $N(s) = \{s \oplus \mu: \mu \in \Gamma(s)\}$.

A typical tabu search algorithm begins with an initial configuration in S and proceeds iteratively to visit a series of locally best configurations following the neighborhood. At each iteration, a *best* neighbor $s' \in N(s)$ is sought to replace the current configuration s even if s' does not improve s in terms of the cost function.

To avoid the problem of possible cycling and to allow the search to go beyond local optima, tabu search introduces the notion of "tabu list", one of the most important components of the method. A tabu list τ is a special short term memory that maintains a selective history composed of previously encountered solutions or, more generally, pertinent attributes (or moves) of such solutions. A simple strategy based on this short term memory consists in preventing previously visited solutions from being reconsidered for the next p_τ iterations (p_τ, called "tabu tenure", is problem dependent). Now, at each iteration, tabu search searches for a best neighbor from this dynamically modified neighborhood.

4.2. Solving scheme

Let 2D-SPP$_{k>0}$ be the following *satisfaction* problem: Is there a solution s to 2D-SPP such that $f(s) \leq k$? Obviously, 2D-SPP is equivalent to find the lowest k such that 2D-SPP$_k$ holds.

CTS treats the 2D-SPP *optimization* problem (minimizing the height f) as successive 2D-SPP$_k$. Starting from a complete packing s_0 of height $f(s_0)$, e.g. obtained with a greedy method (see Sect. 4.4), CTS tackles 2D-SPP$_k$ with decreasing values of $f(s_0)$ for k. To be more precise, if CTS finds a solution s to 2D-SPP$_k$, it then tries to solve 2D-SPP$_{f(s)-p_f}$ ($p_f > 0$, for decrement of the height).

4.3. Search space: A direct representation

Some approaches for 2D-SPP, or closely related variants, consider a (quite natural or, at least, intuitive) search space S composed of the set of (all) permutations of the objects, see (Gómez-Villouta et al., 2008; Iori et al., 2003; Soke & Bingul, 2006; Yeung & Tang, 2004) for instance.

More precisely, for a given n-set R of objects to be packed, a permutation $s \in S$ of $[1,\ldots, n]$ is built (statically or dynamically) using a *selection heuristic* σ[2] which is followed by a given *placement heuristic* ϕ (or "decoder"). In other words, given a selection operator σ and a ϕ decoder, one can locate all the objects using ϕ and according to the order imposed by σ; see Algorithm 1 where s_ρ is the element at rank ρ in permutation s. The problem is then to find a particular permutation $s_* \in S$ (from the $n!$ available) such that the resulting packing is optimal, i.e. $f(s_*) = f_{OPT}$.

Note that many (usually greedy) selection / placement heuristics have been investigated according to various criteria (Alvarez-Valdes et al., 2008, 2007; Aşik & Özcan, 2009; Burke et al., 2009).

CTS does not code packings with permutations but adopts a *direct representation* where a 2D-SPP$_k$ packing $s \in S$ (optimal or not, possibly partial) is a $\{L, E\}$ set[3]:

[2] σ introduces then an order for all the objects.
[3] Other approaches that do not use permutations to model the problem, or closely related variants, can also be found e.g. in (Bortfeldt, 2006; Hamiez et al., 2009; Soh et al., 2008).

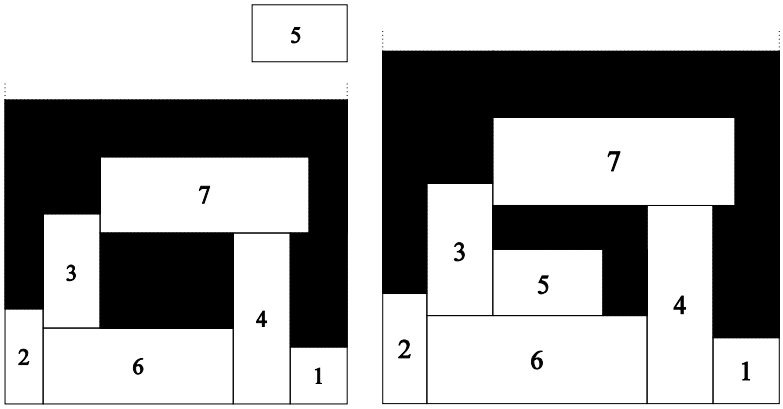
- $L \subseteq R$ is the set of rectangles properly *L*ocated in the strip, i.e. r_i verifies (2) with $y_i^r + h_i^r \leq k \ \forall$ $r_i \in L$ and (r_i, r_j) verifies (3–4) $\forall (r_i, r_{j\neq i}) \in L \times L$. Let the set of "free" objects, i.e. rectangles not (yet) located in the strip, be $\overline{L} = R \setminus L$.
- E is a set of rectangular *E*mpty spaces in the strip. Each empty space $e_i \in E$ is characterized by the coordinates (x_i^e, y_i^e) of its BL corner, a width $0 < w_i^e \leq W$, and a height $0 < h_i^e \leq k$, with $0 \leq x_i^e \leq W - w_i^e$ and $0 \leq y_i^e \leq k - h_i^e$.
  Each empty space $e_i \in E$ is a maximal rectangle[4], i.e. e_i is not included in another empty space e_j: $\forall (e_i, e_{j\neq i}) \in E \times E,\ x_i^e < x_j^e \vee x_i^e + w_i^e > x_j^e + w_j^e \vee y_i^e < y_j^e \vee y_i^e + h_i^e > y_j^e + h_j^e$.

Algorithm 1. The simplest greedy algorithm for 2D-SPP.

Require: A selection operator σ and a placement heuristic ϕ
  $R' \leftarrow R$
  for $\rho = 1$ **to** n **do**
    Select a rectangle $r_i \in R'$ according to σ
    $R' \leftarrow R' \setminus \{r_i\}$
    $s_\rho \leftarrow i$
    Locate the r_i object in the strip according to ϕ
  end for
  return $f(s)$ and s

4.4. Initial configuration

Tabu search needs an initial configuration s_0 that specifies where the search begins in the search space S. CTS uses Algorithm 1 to construct s_0, where the ϕ placement heuristic is the "Bottom Left Fill" procedure (BLF) from Baker et al. (1980) and the σ selection operator orders all rectangles $r_i \in R$ first by decreasing width, secondly by decreasing height (when two objects r_i and $r_{j\neq i}$ have the same width), randomly last if necessary (r_i and $r_{j\neq i}$ have the same width and height).



Before BLF After BLF

Figure 3. Basically, BLF places each object at the left-most and lowest possible free area.

BLF is capable of filling enclosed wasted areas, see Figure 3 where rectangle r_5 has to be packed. Notice that, according to the way BLF is implemented, its worst time complexity goes from $O(n^3)$

[4] The notion of "maximal rectangular empty space" seems to have been independently introduced in (El Hayek et al., 2008) (where it is called "maximal area") and (Neveu et al., 2008) ("maximal hole"). $|E|$ is at most (i.e. in the worst case) in $O(n^2)$ according to El Hayek et al. (2008).

The cost and objective functions, c (5) and f (1) respectively, are used to compare any two packings s_1 and s_2 (possibly partial): With respect to 2D-SPP$_k$, s_1 is said to be "better" than s_2 if the evaluation of s_1 is lower than that of s_2, formally $c(s_1) < c(s_2)$. However, note that the c fitness function is inadequate when s_1 and s_2 are both solutions to 2D-SPP$_k$, i.e. when $c(s_1) = c(s_2) = 0$. In this case, s_1 is better than s_2 if $f(s_1) < f(s_2)$.

Other evaluation functions have been proposed for 2D-SPP. Neveu et al. (2007) consider the number u of "*u*nits filled by rectangles on the highest line of the strip" to define $c_N(s) = (f(s) - 1)W + u$. Hamiez et al. (2009) proposed a similar fitness function, relying on the set $\lceil R \rceil \subseteq R$ of rectangles at the top of the strip, formally defined by (6) where $\lceil R \rceil = \left\{ r_i \in R : y_i^r + h_i^r > H_* - p_H \right\}$, p_H is an integer parameter, and H_* is the best height found, initially the height $f(s_0)$ of the starting solution s_0. One can observe that c_N and c_H compute a measure solely based on rectangles at the top of the strip and do not consider what happens below these rectangles. Our c cost function seems thus more relevant since it precisely includes a measure (the δ component) related to this useful information.

$$c_H(s) = \begin{cases} 0 \text{ if } \lceil R \rceil = \emptyset \\ \sum_{r_i \in \lceil R \rceil} w_i^r * \left(y_i^r + h_i^r - H_* + p_H \right) \text{ otherwise } . \end{cases} \quad (6)$$

4.6. Neighborhood

The neighborhood N is another key element of tabu search. It defines a structure of the search space S and determines the paths the algorithm will follow to explore S.

The main goal of the CTS neighborhood is to empty the set of free objects, i.e. the set $\overline{L}$ of rectangles not yet placed in the strip. Basically, it tries to locate a free rectangle r_i in the strip (r_i moves then from $\overline{L}$ to L), at the BL corner either of an empty space (defining a sub-neighborhood N_E, described in Sect. 4.6.1) or of another placed object (defining N_L, Sect. 4.6.2).

This location for object r_i may generate overlaps with a set $L_i \subseteq L$ of other rectangles already in the strip: $L_i = \left\{ r_{j \neq i} \in L : x_i^r < x_j^r + w_j^r \wedge x_i^r + w_i^r > x_j^r \wedge y_i^r < y_j^r + h_j^r \wedge y_i^r + h_i^r > y_j^r \right\}$. All objects overlapping with the r_i rectangle are thus removed from the strip to repair these overlaps ($\forall\, r_j \in L_i$, r_j moves from L to $\overline{L}$). This principle, known as "ejection chains", is used to make the neighboring configuration consistent with (3–4).

Finally, notice that locating a rectangle in the strip and the possible ejection of all overlapping objects imply updates of the set of empty spaces (E). This is done using the efficient "incremental" procedures detailed in (Neveu et al., 2008).

4.6.1 Neighborhood N_E: Consider the empty spaces

CTS examines two cases here, with two different objectives: To reduce the number of free rectangles, i.e. to minimize $|\overline{L}|$ (defining $N_E^{|\overline{L}|}$), or to make the packing *D*enser (N_E^D).

The case of $N_E^{|\overline{L}|}$. E . All free objects $r_i \in \overline{L}$ are tried to be located in the strip to the BL corner of all empty spaces $e_j \in E$ such that r_i fits entirely in e_j. More formally, r_i and e_j must verify $x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k \wedge w_j^e \geq w_i^r \wedge h_j^r \geq h_i^r$. This generates $|\overline{L}|$ sets $N_E^{|\overline{L}|}(s, i)$ of

neighbors for configuration s (some possibly empty), $N_E^{|\overline{L}|}(s)$ being the union of these sets: $N_E^{|\overline{L}|}(s) = \bigcup_{r_i \in \overline{L}} N_E^{|\overline{L}|}(s,i)$.

Note that $N_E^{|\overline{L}|}(s) \neq \varnothing$ means that there is at least one free rectangle $r_i \in \overline{L}$ and one empty space $e_j \in E$ such that r_i fits in e_j, i.e. that the number of free objects can be reduced. Furthermore, in this case, locating r_i will generate no overlap ($L_i = \varnothing$), hence no repairing is needed.

The case of N_E^D. Here again, all free rectangles $r_i \in \overline{L}$ are tried to be located to the BL corner of all empty spaces $e_j \in E$ but the previous condition on e_j and r_i is relaxed to allow overlaps. More formally, r_i and e_j must only verify $x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k$.

Note that $N_E^D(s) = \varnothing$ means either, $\forall (r_i, e_j) \in \overline{L} \times E$, that the empty space e_j is located rather to the right (top, respectively) of the strip and placing the r_i rectangle in e_j will violate (2) since $x_j^e + w_i^r > W$ (resp. will not follow the definition of 2D-SPP$_k$ since $y_j^e + h_i^r > k$) or that it is forbidden to remove at least one rectangle overlapping with the r_i object (this is due to some restrictions described in Sect. 0).

4.6.2 Neighborhood N_L: Consider all rectangles already packed

From the current configuration s, all free objects $r_i \in \overline{L}$ are tried to be located to the BL corner of all already packed rectangles $r_j \in L$ such that r_i and r_j have different sizes, locating r_i will respect (2), and the resulting packings will follow the definition of 2D-SPP$_k$. More formally, r_i and r_j must verify $\left(w_i^r \neq w_j^r \vee h_i^r \neq h_j^r\right) \wedge x_j^r + w_i^r \leq W \wedge y_j^r + h_i^r \leq k$.

This generates $|\overline{L}|$ sets $N_L(s, i)$ of neighbors for the configuration s with $0 \leq |N_L(s, i)| \leq |L|$: $N_L(s)$ is the union of these sets. Similarly to N_E^D, note that $N_L(s) = \varnothing$ means that all overlapping objects cannot be removed from the strip (see Sect. 0).

4.6.3 Search strategy

Our neighborhood N is composed of the three sub-neighborhoods described above: $N_E^{|\overline{L}|}$ and N_E^D (Sect. 4.6.1), and N_L (Sect. 4.6.2). They are explored by CTS in a hierarchical way: $N_E^{|\overline{L}|}$ is always tried the first, then N_E^D is used only if $N_E^{|\overline{L}|}$ is not applicable (i.e. empty), N_L being (possibly) considered the last. This scheme is more formally detailed in Algorithm 2.

Algorithm 2. The neighborhood is explored in a hierarchical way.

Require: A configuration $s = \{L, E\}$

if $N_E^{|\overline{L}|}(s) \neq \varnothing$ **then return** $N_E^{|\overline{L}|}(s)$
else if $N_E^D(s) \neq \varnothing$ **then return** $N_E^D(s)$
else if $N_L(s) \neq \varnothing$ **then return** $N_L(s)$
else return $\varnothing$
end if

Note that if configuration s has no neighbor, i.e. $N(s) = N_E^{|\overline{L}|}(s) = N_E^D(s) = N_L(s) = \varnothing$, no move is performed and diversification is invoked (see Sect. 4.8).

4.7. Tabu list

At current iteration m, since a CTS move from solution s to a neighbor $s' \in N(s)$ consists in locating one free rectangle $r_i \in \overline{L}$ in the strip, it seems quite natural to forbid object r_i leaving the strip from configuration s'. This "reverse" move will then be stored in the tabu list τ for a duration $0 < p_\tau \leq n$ (integer) to indicate that r_i cannot be removed from the strip at least up to iteration $m + p_\tau$.

Note that the tabu list τ is made empty at the beginning of the search or when CTS finds a solution to the current satisfaction problem considered (2D-SPP$_k$), i.e. if there is no free rectangle at all ($\overline{L} = \varnothing$).

4.8. Diversification

Let s_* be the overall best complete packing, according to (1), found by CTS at iteration m_* (initially s_* is the initial configuration s_0 introduced in Sect. 4.4 with $m_* = 0$).

When a solution s has no neighbor (i.e. $N(s) = \varnothing$) or s_* keeps unchanged for a number $p_* > 0$ of iterations, CTS first resets the tabu list τ and reloads s_* in s ($s \leftarrow s_*$, $\tau \leftarrow \varnothing$). This new current complete packing s is then perturbed according to two different *D*iversification schemes called D_I (for "*I*nterchange", performed with probability p_D) and D_T (for "*T*etris-like", probability $1 - p_D$). After perturbation, p_* supplementary moves are given to CTS to update the overall best complete packing s_*.

4.8.1 D_I : A basic perturbation

L and $\overline{L}$ are first modified according to 2D-SPP$_k$ with $k = f(s_*) - p_f$: $L \leftarrow \left\{ r_i \in R : y_i^r + h_i^r \leq k \right\}$, $\overline{L} \leftarrow R \setminus L$. Diversification D_I considers then all $(r_i, r_j) \in L \times \overline{L}$ such that the r_i packed rectangle and the r_j free object have different sizes, more formally r_i and r_j verify $\left(w_i^r \neq w_j^r \vee h_i^r \neq h_j^r \right) \wedge x_i^r + w_j^r \leq W \wedge y_i^r + h_j^r \leq k$. It simply interchanges two such elements (randomly selected) and makes r_j tabu.

Note that locating the free object r_j at the place of r_i (i.e. swapping r_j from $\overline{L}$ to L and r_i from L to $\overline{L}$) may cause overlaps. In this case, repairing is done like in Sect. 4.6.

4.8.2 D_T : A perturbation based on the history

During the overall search process, CTS keeps for each rectangle $r_i \in R$ the number F_i (for "*F*requency") of times r_i leaved the strip, i.e. the number of times r_i swaps from L to $\overline{L}$ [7].

The D_T diversification scheme considers a π_F permutation that orders all objects $r_i \in R$ first by increasing frequencies, secondly by decreasing widths (when $F_i = F_{j \neq i}$), then by decreasing heights ($w_i^r = w_{j \neq i}^r$), randomly last if necessary ($h_i^r = h_{j \neq i}^r$). Let the set $\lfloor F \rfloor$ be composed of the first p_τ elements of π_F.

[7] $F_{1 \leq i \leq n} = 0$ at the beginning of the search.

All rectangles $r_i \in \lfloor F \rfloor$ are first temporarily removed from the strip and their frequencies are updated[8]. Then, the partial packing is pushed down to the basis of the strip, like in the famous Tetris game. Finally, all objects $r_i \in \lfloor F \rfloor$ are sorted like in Sect. 4.4 and relocated in the strip with BLF, see Figure 5.

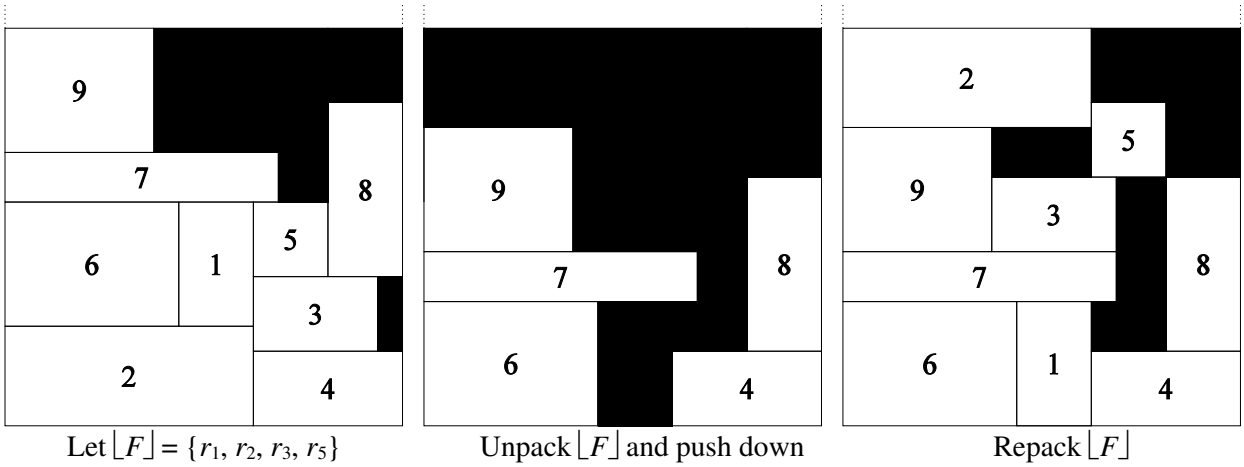


Figure 5. The D_T diversification. Surprisingly, experiments showed that almost all rectangles with lowest frequency (F_i) were located close to the bottom of the strip.

CTS deals now with 2D-SPP$_k$ with $k = f(s)$ - p_f: $L \leftarrow \{r_i \in R : y_i^r + h_i^r \le k\}$, $\overline{L} \leftarrow R \setminus L$. This means that CTS possibly considers 2D-SPP$_k$ problems with $k \ge f(s_0) \ge f(s_*)$.

4.9. CTS: The general procedure

Algorithm 3. An overview of CTS.

Require: A starting configuration $s = \{L, E\}$ such that $L = R$ // Hence $\overline{L} = \varnothing$
1. $m \leftarrow 0$, $s_* \leftarrow s$, $m_* \leftarrow 0$ // Initialization
2. **while** $m \le p_M$ **do**
3. **if** $\overline{L} = \varnothing$ **then**
4. **if** $f(s) < f(s_*)$ **or** ($f(s) = f(s_*)$ **and** some probability $p_\approx$ is verified) **then** $s_* \leftarrow s$, $m_* \leftarrow m$
5. $L \leftarrow \{r_i \in R : y_i^r + h_i^r \le f(s) - p_f\}$, $\overline{L} \leftarrow R \setminus L$
6. $\tau \leftarrow \varnothing$ // Tabu list
7. **if** $N(s) = \varnothing$ **then** *Div* $\leftarrow$ true // To *Div*ersify
8. **else**
9. $m \leftarrow m + 1$
10. Let $\lfloor N(s) \rfloor$ be the set of the *best evaluated* neighbors s' of s according to (5): $\lfloor N(s) \rfloor = \{s_1' \in N(s) : \forall s_2' \in N(s), c(s_1') \le c(s_2')\}$
11. **if** $c(s') > 0 \ \forall \ s' \in \lfloor N(s) \rfloor$ **then**
12. **if** $(m - m_*) \bmod p_* = 0$ **then** *Div* $\leftarrow$ true **else** *Div* $\leftarrow$ false
13. Select $s' \in \lfloor N(s) \rfloor$ at random
14. **else** *Div* $\leftarrow$ false, select $s' \in \lfloor N(s) \rfloor$ minimizing (1) at random
15. **if** *Div* **then** $s \leftarrow s_*$, $\tau \leftarrow \varnothing$, modify s using D_I or D_T according to p_D // Diversification
16. **else** $s \leftarrow s'$, update τ
17. **return** $f(s_*)$ and s_*

[8] $F_i \leftarrow 2F_i$ is used here to avoid considering (almost) the same $\lfloor F \rfloor$ set in next applications of diversification D_T while $F_i \leftarrow F_i + 1$ is applied when performing a move.

The CTS procedure is summarized in Algorithm 3. It requires an initial complete packing (Sect. 4.4). Then CTS proceeds iteratively to solve a series of 2D-SPP$_k$ satisfaction problems. If it finds a solution s to 2D-SPP$_k$ (i.e. $\overline{L} = \varnothing$), it then tries to solve the next 2D-SPP$_{f(s)-p_f}$ problem (see lines 3–6). Note that $f(s) = f(s_*)$ (line 4) may occur only after diversification D_T (see Sect. 4.8) or at the beginning of the search process.

While it is not mentioned here for simplicity, note that CTS can also end before reaching the $p_M \geq 0$ *M*aximum number of allowed moves (line 2). This may occur each time the overall best complete packing s_* is updated (lines 1 and 4) whenever the optimum height f_{OPT} (or an upper bound) is known and $f(s_*) \leq f_{OPT}$.

5. EXPERIMENTATIONS

We use the complete set of the 21 well-known (perfect) instances defined in (Hopper & Turton, 2001)[9] to assess the performance of our CTS algorithm. These instances (or a subset of them) are largely studied in the literature. The main characteristics of these instances, including their known optimal height f_{OPT}, are given in Table 1.

Table 1. Main characteristics of the test problems defined in (Hopper & Turton, 2001). These instances are grouped by "categories" according to the f_{OPT} value.

Category	Instances	W	n	f_{OPT}
C1	C1P1, C1P2, C1P3	20	16, 17, 16	20
C2	C2P1, C2P2, C2P3	40	25	15
C3	C3P1, C3P2, C3P3	60	28, 29, 28	30
C4	C4P1, C4P2, C4P3	60	49	60
C5	C5P1, C5P2, C5P3	60	73	90
C6	C6P1, C6P2, C6P3	80	97	120
C7	C7P1, C7P2, C7P3	160	196, 197, 198	240

Notice that the instances of categories C4–C7 are very difficult because, to our knowledge, none algorithm is known to be able to optimally solve them. Indeed, some studies don't even report computational results for them, perhaps due to the (very) large size of these instances. Furthermore, the difficulty sometimes also hold for small size instances from categories C1–C3. In particular, various studies report more computational effort for C1P2, C2P2, or C3P2 or never reach (or are more distant from) an optimal solution for these three specific instances compared to the other similar instances within the same categories.

5.1. Experimentation conditions

CTS is coded in the C programming language ("gcc" compiler). All computational results were obtained running CTS on a computer equipped with a 2.83 Ghz quad-core Intel® Xeon® E5440 processor and 8 Gb RAM[10]. The values of the CTS parameters are:

- p_f = 1. To build the starting configuration of 2D-SPP$_k$, the current satisfaction problem considered.
- $p_{\approx} \in$ [0.4, …, 0.8]. Probability that a complete packing s replaces the overall best complete packing s_*, according to (1), whenever $f(s) = f(s_*)$.

[9] They are available from the "PackLib2" benchmarks library e.g., see http://www.ibr.cs.tu-bs.de/alg/packlib/xml/ht-eimhh-01-xml.shtml.

[10] For indicative purpose, the mean running time of CTS ranges from a few seconds (for the smallest instances) to about 33 hours (for the largest instances).

- $p_\tau \in$ [2, …, 6]. Tabu tenure.
- $p_* \in$ [200, …, 500]. Maximum number of moves to update s_*.
- $p_D \in$ [0.7, …, 1]. Probability to apply diversification D_l.
- $p_M \in$ [1 000 000, …, 20 000 000]. Maximum number of allowed moves per run.

The comparison is based on the percentage gap γ of a solution s from the optimum or its best bound (f_{OPT}): $\gamma(s) = 100(1 - f_{OPT} / f(s))$. The lower is $\gamma(s)$, the better is the solution s. For CTS, mean gap $\bar{\gamma}$ (resp. best gap γ^*) is averaged over 5 runs (resp. over best runs only).

5.2. Computational results

CTS is compared in Table 2 with five state-of-the-art algorithms that deal with the whole set of instances, or at least categories C1–C6.

We consider two Tabu Search procedures, denoted as TS1 (Iori et al., 2003)[11] and TS2 (Hamiez et al., 2009), one of the most effective Genetic Algorithm (GA) from Bortfeldt (2006), and two best performing state-of-the-art approaches: The *Greedy Randomized Adaptive Search Procedure* (GRASP) from Alvarez-Valdes et al. (2008) and the hybrid *HyperHeuristic + Intensification / Diversification Walk* strategy (HH+IDW) from Neveu et al. (2008).

In Table 2, "–" marks (for HH+IDW, GA, or TS1) mean either that $\bar{\gamma}$ or γ^* cannot be computed or that we did not found the information in (Bortfeldt, 2006; Iori et al., 2003; Neveu et al., 2008). "Mean C*i*" are averaged values on category C*i*. The "C1–C*j*" aggregated lines, reporting averaged values for all instances in categories C1 to C*j*, can be used to identify up to which problem size a particular approach may be effective. The last line shows the number of instances optimally solved.

According to Table 2, TS1 is the worst performing (tabu search) approach for the benchmark tried. Indeed, $\gamma^* = 0$ only for C2P1 and C2P3 while the other methods (possibly except GA) always solved at least 8 instances. Almost all other approaches (except TS1 and GA) solved the C1 and C2 instances, see line "C1–C2" where $\gamma^* = 0.00$ or $\bar{\gamma} = 0.00$.

To our knowledge, the only approaches solving all the 9 instances C1P1–C3P3 are the tabu search from Alvarez-Valdes et al. (2007) and the recent exact procedures described in (Kenmochi et al., 2009; Soh et al., 2008). In Table 2, CTS is the only method reaching the same qualitative results, see line "C1–C3" where $\gamma^* = 0.00$ just for CTS. Furthermore, note that CTS achieves here the lowest $\bar{\gamma}$ value ($0.29 < 0.36 < 0.89 < 1.64 < 2.69$).

Aggregated results show that CTS compares also well with the competitors if one considers instances up to C5. Indeed, line "C1–C4" indicates better CTS values for γ^* ($0.41 < 0.68 \leq 0.68 \leq 0.68 < 2.38 < 5.40$) and $\bar{\gamma}$ ($0.63 < 0.68 < 1.08 < 1.84 < 2.90$). The same observation holds in line "C1–C5" only for γ^* ($0.62 < 0.76 \leq 0.76 \leq 0.76 < 2.20 < 5.31$) but the difference is sharp between the best $\bar{\gamma}$ (0.76 for GRASP) and that of CTS (0.85).

CTS obtains worst γ^* or $\bar{\gamma}$ values than those of the two best-known state-of-the-art approaches considered here (GRASP and HH+IDW) only when adding the largest two categories of instances. For C1–C6, note, however, that the difference is still reasonable considering γ^* (0.83 - 0.77 = 0.06).

[11] TS1 is perhaps the first tabu search approach for 2D-SPP.

Table 2. Mean and best percentage gaps ($\overline{\gamma}$ and γ resp.) on instances from Hopper & Turton (2001).

Instances	CTS		TS1	TS2		GA		GRASP		HH+IDW	
	$\overline{\gamma}$	γ_*	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*	$\overline{\gamma}$	γ_*
C1P1	0.00	0.00	9.09	0.00	0.00	–	–	0.00	0.00	–	–
C1P2	0.00	0.00	9.09	4.76	0.00	–	–	0.00	0.00	–	–
C1P3	0.00	0.00	4.76	0.00	0.00	–	–	0.00	0.00	–	–
Mean C1	0.00	0.00	7.65	1.59	0.00	1.59	1.59	0.00	0.00	0.48	0.00
C2P1	0.00	0.00	0.00	0.00	0.00	–	–	0.00	0.00	–	–
C2P2	0.00	0.00	6.25	0.00	0.00	–	–	0.00	0.00	–	–
C2P3	0.00	0.00	0.00	0.00	0.00	–	–	0.00	0.00	–	–
Mean C2	0.00	0.00	2.08	0.00	0.00	3.33	2.08	0.00	0.00	1.87	0.00
C3P1	0.00	0.00	3.23	0.00	0.00	–	–	0.00	0.00	–	–
C3P2	2.60	0.00	9.09	3.23	3.23	–	–	3.23	3.23	–	–
C3P3	0.00	0.00	9.09	0.00	0.00	–	–	0.00	0.00	–	–
Mean C3	0.87	0.00	7.14	1.08	1.08	3.16	3.16	1.08	1.08	2.58	1.08
C4P1	1.64	1.64	6.25	1.64	1.64	–	–	1.64	1.64	–	–
C4P2	1.64	1.64	4.76	1.64	1.64	–	–	1.64	1.64	–	–
C4P3	1.64	1.64	3.23	1.64	1.64	–	–	1.64	1.64	–	–
Mean C4	1.64	1.64	4.75	1.64	1.64	3.52	2.70	1.64	1.64	2.43	1.64
C5P1	2.17	2.17	5.26	1.10	1.10	–	–	1.10	1.10	–	–
C5P2	1.96	1.10	3.23	1.10	1.10	–	–	1.10	1.10	–	–
C5P3	1.10	1.10	6.25	1.10	1.10	–	–	1.10	1.10	–	–
Mean C5	1.74	1.46	4.91	1.10	1.10	2.03	1.46	1.10	1.10	1.78	1.10
C6P1	1.80	1.64	4.76	1.64	0.83	–	–	1.56	0.83	–	–
C6P2	2.44	2.44	3.23	0.83	0.83	–	–	1.56	0.83	–	–
C6P3	2.28	1.64	3.23	1.64	0.83	–	–	1.56	0.83	–	–
Mean C6	2.17	1.91	3.74	1.37	0.83	1.72	1.64	1.56	0.83	1.75	1.10
C7P1	2.20	2.04	–	1.23	1.23	–	–	1.64	1.64	–	–
C7P2	2.04	2.04	–	1.23	1.23	–	–	1.19	0.83	–	–
C7P3	2.20	2.04	–	1.23	1.23	–	–	1.23	1.23	–	–
Mean C7	2.15	2.04	–	1.23	1.23	1.52	1.23	1.36	1.23	1.42	1.10
C1–C2	0.00	0.00	4.86	0.79	0.00	2.46	1.84	0.00	0.00	1.18	0.00
C1–C3	0.29	0.00	5.62	0.89	0.36	2.69	2.28	0.36	0.36	1.64	0.36
C1–C4	0.63	0.41	5.40	1.08	0.68	2.90	2.38	0.68	0.68	1.84	0.68
C1–C5	0.85	0.62	5.31	1.08	0.76	2.73	2.20	0.76	0.76	1.83	0.76
C1–C6	1.07	0.83	5.04	1.13	0.77	2.57	2.11	0.90	0.77	1.82	0.82
C1–C7	1.22	1.01	–	1.14	0.84	2.41	1.98	0.96	0.84	1.76	0.86
$\#_{f_{OPT}}/21$	9		2	8		–		8		8	

6. CONCLUSIONS

In this paper, we presented CTS, a Consistent Tabu Search algorithm for the 2D Strip Packing Problem (2D-SPP). CTS includes some components already used by (or, at least, similar to) other approaches, such as the direct representation of the problem or the neighborhood mentioned e.g. in (Hamiez et al., 2009). Apart from these traditional components, our CTS approach was reinforced mainly by introducing two novel elements that, to our knowledge, were never tried for 2D-SPP:

- A fitness function including a measure related to the empty spaces. This was motivated by the fact that most of the previous studies on 2D-SPP usually employ evaluation functions solely based on the rectangles. This is the case, for instance, in (Hamiez et al., 2009; Neveu et al., 2008). Such an additional criterion may be helpful to guide more efficiently the search process.

- A diversification scheme based on a frequency measure (D_T). The motivation behind this component of the *generic* tabu search strategy, that can help to escape from local optima e.g., originates from running profile observations. Indeed, preliminary tests (without diversification D_T) shown that some rectangles were almost always in the strip and, so, their location changed rarely. D_T was thus designed to force these "frozen" objects, that may be considered as problematic (since, perhaps, they are not positioned there in optimal solutions), to leave the strip and to be packed at other locations.

These components proved to be quite useful for the effectiveness of the CTS algorithm. We believe that the basic ideas behind these components could be applicable to other optimization problems.

ACKNOWLEDGEMENTS

We would like to thank the reviewers of the paper. This work was partially supported by three grants from the French "Pays de la Loire" region (MILES, RadaPop, and LigeRO projects). The first author is supported by a Chilean CONICIT scholarship.

REFERENCES

Alvarez-Valdes, R., Parreño, F., & Tamarit, J. (2007). A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *European Journal of Operational Research, 183, 1167–1182*.

Alvarez-Valdes, R., Parreño, F., & Tamarit, J. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research, 35, 1065–1083*.

Araya, I., Neveu, B., & Riff, M.-C. (2008). An efficient hyperheuristic for strip-packing problems. In C. Cotta, M. Sevaux, & K. Sörensen (Editors), *Adaptive and Multilevel Metaheuristics* (volume 136 of Studies in Computational Intelligence, pp. 61–76). Berlin: Springer.

Aşik, O., & Özcan, E. (2009). Bidirectional best-fit heuristic for orthogonal rectangular strip packing. *Annals of Operations Research, 172, 405–427*.

Baker, B., Brown, D., & Katseff, H. (1981). A 5/4 algorithm for two-dimensional packing. *Journal of Algorithms, 2, 348–368*.

Baker, B., Jr., E. C., & Rivest, R. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing, 9, 846–855*.

Beasley, J. (2004). A population heuristic for constrained two-dimensional non-guillotine cutting. *European Journal of Operational Research, 156, 601–627*.

Belov, G., Kartak, V., Rohling, H., & Scheithauer, G. (2009). One-dimensional relaxations and LP bounds for orthogonal packing. *International Transactions in Operational Research, 16, 745–766*.

Beltrán Cano, J., Calderón, J., Cabrera, R., Moreno Pérez, J., & Moreno-Vega, J. (2004). GRASP / VNS hybrid for the strip packing problem. In C. Blum, A. Roli, & M. Sampels (Editors), *First International Workshop on Hybrid Metaheuristics* (pp. 79–90).

Błażewicz, J., Moret-Salvador, A., & Walkowiak, R. (2004). Parallel tabu search approaches for two-dimensional cutting. *Parallel Processing Letters, 14, 23–32*.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research, 172, 814–837*.

Burke, E., Kendall, G., & Whitwell, G. (2009). A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock cutting problem. *INFORMS Journal on Computing, 21, 505–516*.

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

Clautiaux, F., Jouglet, A., Carlier, J., & Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research, 35, 944–959*.

Coffman Jr., E., Garey, M., Johnson, D., & Tarjan, R. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing, 9, 808–826*.

Dowsland, K., & Dowsland, W. (1992). Packing problems. *European Journal of Operational Research, 56, 2–14*.

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

Fowler, R., Paterson, M., & Tanimoto, S. (1981). Optimal packing and covering in the plane are NP-complete. *Information Processing Letters, 12, 133–137*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company.

Garrido, P., & Riff, M.-C. (2007). Collaboration between hyperheuristics to solve strip-packing problems. In P. Melin, O. Castillo, L. Aguilar, J. Kacprzyk, & W. Pedrycz (Editors), *Twelfth International Fuzzy Systems Association World Congress* (volume 4529 of Lecture Notes in Computer Science, pp. 698–707). Berlin: Springer.

Glover, F. W., & Laguna M. (1997). *Tabu Search*. Dordrecht: Kluwer.

Gómez-Villouta, G., Hamiez, J.-P., & Hao, J.-K. (2008). A dedicated genetic algorithm for two-dimensional non-guillotine strip packing. *Sixth Mexican International Conference on Artificial Intelligence, Special Session* (pp. 264–274). Los Alamitos: IEEE Computer Society.

Gonçalves, J. (2007). A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European Journal of Operational Research, 183, 1212–1229*.

Hamiez, J.-P., Robet, J., & Hao, J.-K. (2009). A tabu search algorithm with direct representation for strip packing. In C. Cotta, & P. Cowling (Editors), *Ninth European Conference on Evolutionary Computation in Combinatorial Optimization* (volume 5482 of Lecture Notes in Computer Science, pp. 61–72). Berlin: Springer.

Harren, R., & van Stee, R. (2009). Improved absolute approximation ratios for two-dimensional packing problems. In I. Dinur, K. Jansen, J. Naor, & J. Rolim (Editors), *Approximation, Randomization, and Combinatorial Optimization* (volume 5687 of Lecture Notes in Computer Science, pp. 177–189). Berlin: Springer.

Hopper, E., & Turton, B. (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research, 128, 34–57*.

Ibaraki, T., Imahori, S., & Yagiura, M. (2008). Hybrid metaheuristics for packing problems. In C. Blum, M. B. Aguilera, A. Roli, & M. Sampels (Editors), *Hybrid Metaheuristics: An Emerging Approach to Optimization* (volume 114 of Studies in Computational Intelligence, pp. 185–219). Berlin: Springer.

Imahori, S., Yagiura, M., & Ibaraki, T. (2003). Local search algorithms for the rectangle packing problem with general spatial costs. *Mathematical programming, 97, 543–569*.

Imahori, S., Yagiura, M., & Ibaraki, T. (2005). Improved local search algorithms for the rectangle packing problem with general spatial costs. *European Journal of Operational Research, 167, 48–67*.

Imahori, S., Yagiura, M., & Nagamochi, H. (2007). Practical algorithms for two-dimensional packing. In T. Gonzalez (Editor), *Handbook of Approximation Algorithms and Metaheuristics* (volume 13 of Chapman & Hall / CRC Computer & Information Science Series, chapter 36). Boca Raton / London: CRC Press.

Iori, M., Martello, S., & Monaci, M. (2003). Metaheuristic algorithms for the strip packing problem. In P. Pardalos, & V. Korotkikh (Editors), *Optimization and Industry: New Frontiers* (volume 78 of Applied Optimization, pp. 159–179). Berlin: Springer.

Jansen, K., & van Stee, R. (2005). On strip packing with rotations. In H. Gabow, & R. Fagin (Editors), *Thirty-Seventh annual ACM symposium on Theory of computing* (pp. 755–761). New York: ACM Press.

Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., & Nagamochi, H. (2009). Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research, 198, 73–83*.

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

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

Mir, M., & Imam, M. (2001). A hybrid optimization approach for layout design of unequal-area facilities. *Computers & Industrial Engineering, 39, 49–63*.

Neveu, B., Trombettoni, G., & Araya, I. (2007). Incremental move for strip-packing. In N. Avouris, N. Bourbakis, & I. Hatzilygeroudis (Editors), *Nineteenth IEEE International Conference on Tools with Artificial Intelligence* (volume 2, pp. 489–496). Los Alamitos: IEEE Computer Society.

Neveu, B., Trombettoni, G., Araya, I., & Riff, M.-C. (2008). A strip packing solving method using an incremental move based on maximal holes. *International Journal on Artificial Intelligence Tools, 17, 881–901*.

Schiermeyer, I. (1994). Reverse-Fit: A 2-optimal algorithm for packing rectangles. In J. van Leeuwen (Editor), *Second Annual European Symposium on Algorithms* (volume 855 of Lecture Notes in Computer Science, pp. 290–299). Berlin: Springer.

Soh, T., Inoue, K., Tamura, N., Banbara, M., & Nabeshima, H. (2008). A SAT-based method for solving the two-dimensional strip packing problem. In M. Gavanelli, & T. Mancini (Editors), *Fifteenth International RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*. Aachen: RWTH Aachen University.

Soke, A., & Bingul, Z. (2006). Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems. *Engineering Applications of Artificial Intelligence, 19, 557–567*.

Terashima-Marín, H., Flores-Álvarez , E., & Ross, P. (2005). Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems. *2005 Conference on Genetic and Evolutionary Computation* (volume 2, pp. 637–643). New York: ACM Press.

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

Yeung, L., & Tang,W. (2004). Strip-packing using hybrid genetic approach. *Engineering Applications of Artificial Intelligence, 17, 169–177*.