A Tabu Search Algorithm with Direct Representation for Strip Packing

Jean-Philippe Hamiez*, Julien Robet, and Jin-Kao Hao

LERIA, Université d'Angers, 2 Bd. Lavoisier, 49045 Angers, France
{hamiez,robet,hao}@info.univ-angers.fr

Abstract. This paper introduces a new tabu search algorithm for a two-dimensional (2D) Strip Packing Problem (2D-SPP). It integrates several key features: A *direct* representation of the problem, a *satisfaction-based* solving scheme, two different *complementary* neighborhoods, a *diversification* mechanism and a *particular* tabu structure. The representation allows inexpensive basic operations. The solving scheme considers the 2D-SPP as a succession of satisfaction problems. The goal of the combination of two neighborhoods is (to try) to reduce the height of the packing while avoiding solutions with (hard to fill) tall and thin wasted spaces. Diversification relies on a set of historically "interesting" packings. The tabu structure avoids visiting similar packings. To assess the proposed approach, experimental results are shown on a set of well-known benchmark instances and compared with previously reported tabu search algorithms as well as the best performing algorithms.

Keywords: Tabu search, strip packing, direct representation, multi-neighborhoods.

1 Introduction

Packing (and cutting) problems are optimization problems which are NP-hard in the general case. "Small" objects of various shapes (regular or not) and dimensions have to be packed *without overlap*, with rotation and "guillotine" cuts[1] allowed or not, into other larger objects. These larger objects are usually called "containers" for the 3D cases (all dimensions fixed or infinite height) and "bins" (all dimensions fixed) or "strips" (only width fixed, infinite height) in 2D. Objectives are, for instance, to minimize the number of containers and / or to maximize the material used (hence to minimize the wasted area). The most studied category of such problems seems to be in the 2D space.

This paper is dedicated to the 2D (non-guillotine and without rotation) Strip Packing Problem (2D-SPP) which can be informally stated as follows. Given a set of rectangular objects, pack them into a strip of an infinite height and fixed width while minimizing the height of the packing. 2D-SPP is a NP-hard combinatorial

* Contact author.

[1] The guillotine constraint imposes a sequence of *edge-to-edge* cuts.

C. Cotta and P. Cowling (Eds.): EvoCOP 2009, LNCS 5482, pp. 61–72, 2009.

optimization problem with a number of practical applications such as cardboard packing, glass and metal cutting or publicity scheduling for instance [1,2,3,4,5].

Given the NP-hard nature of 2D-SPP, many (meta)heuristic procedures have been tried: Greedy Randomized Adaptive Search Procedure (GRASP) [6], Intensification / Diversification Walk (IDW) [7], Simulated Annealing [8,9,10,11], Tabu Search (TS) [8,12,13], Genetic Algorithm [9,10,11,13,14,15,16,17,18,19], hybrid (meta)heuristic [11,13,20], HyperHeuristic [21]. Exact algorithms have also been considered but they are usually limited to "small instances" (up to 200 objects) [22,23,24,25]. Among these procedures, the approximate GRASP and IDW approaches [6,7] are probably the best performing ones.

In this paper, we introduce a new TS algorithm dedicated to the 2D-SPP (TSD for "Tabu Search with Direct representation"). Compared with previous algorithms for the 2D-SPP, our TSD has several notable features. First, it uses a *direct representation* of the problem (location of the objects on the strip) while many previous attempts manipulate permutations of the objects. Second, TSD treats the optimization problem (minimizing the height of the packing) as successive *satisfaction* problems: Starting from a packing s_0 (obtained with a greedy method e.g.) of height $H(s_0)$, TSD tries to solve the 2D-SPP with decreasing values of $H(s_0)$. Finally, our algorithm includes two different *complementary neighborhoods*, a *diversification* mechanism and handles a *particular tabu structure*. Preliminary computational results suggest that TSD may be of great interest to efficiently solve the 2D-SPP.

In the next section, the 2D-SPP is formally stated. Section 3 is devoted to the detailed presentation of our dedicated TS algorithm for the 2D-SPP. Experimental results are shown in Sect. 4 on a set of well-known benchmarks and compared with previous TS attempts and the best performing state-of-the-art algorithms. We finally discuss possible extensions in Sect. 5 before concluding.

2 Problem Formulation

Let P (for "Plane") be a 2D vertical space with fixed width W and infinite height. The bottom-left corner of P 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) of P. The set of $n > 1$ objects (Rectangles) to be positioned in P is $R = \{r_1, \ldots, r_n\}$ where the $w$eight (resp. $h$eight) of each r_i $(1 \leq i \leq n)$ 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 bottom-left corner of all $r_i \in R$ (i.e. the location of each r_i in P) so as to minimize the $y_i^r + h_i^r$ value of the highest object in P, see (1). This can be formally stated as follows:

$$\text{Minimize: } H = \max_{1 \leq i \leq n} (y_i^r + h_i^r) \tag{1}$$

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

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

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

where (2) forces each r_i to be inside P and (3–4) specify that any two r_i and r_j objects ($i \neq j$) must not overlap horizontally and vertically, respectively.

3 TSD: A Tabu Search with Direct Representation

TS is an advanced local search method using general mechanisms and rules as guidelines for smart search [26]. In Sect. 3.1–3.6, we first describe the components of our TSD algorithm for the 2D-SPP where all p variables (with subscripts) are *p*arameters whose values will be given in the experimentation part (Sect. 4.1). The general procedure is finally summarized in Sect. 3.7.

3.1 Search Space: A Direct Representation

Many approaches for the 2D-SPP consider a search *S*pace S composed of the set of all permutations of the objects, see [13,14] for instance. More precisely, for a given n-set of objects to be packed, a permutation of $[1 \dots n]$ is used to introduce an order for all the objects which is followed by a given placement heuristic (or "decoder"). In other words, given a particular permutation π and placement heuristic ϕ, one can pack all the objects using ϕ and according to the order indicated by π. Based on this permutation representation, several greedy placement heuristics have been investigated for the 2D-SPP. BLF (Bottom Left Fill) is such a heuristic [27]. Basically, BLF places each object at the left-most and lowest possible free area. It is capable of filling enclosed wasted areas. Notice that, according to the way BLF is implemented, its worst time complexity goes from $\mathrm{O}(n^3)$ [28] to $\mathrm{O}(n^2)$ [29] for a permutation of n objects.

TSD does not code packings with permutations but adopts a *direct* representation where a "*s*olution" $s \in S$ (optimal or not) is a $\{L, E\}$ set:

- L, for "*L*ocation" (of the rectangular objects to be positioned in P), is an n-vector. It indicates the coordinates (x_i^r, y_i^r) of the bottom-left corner of each rectangle $r_i \in R$ in P.
- E is a set of *rectangular* "*E*mpty" spaces in P. Each $e_i \in E$ is characterized by the coordinates (x_i^e, y_i^e) of its bottom-left corner, a width $0 < w_i^e \leq W$ and a height $0 < h_i^e \leq H(s)$ with $0 \leq x_i^e \leq W - w_i^e$ and $0 \leq y_i^e \leq H(s) - h_i^e$. Each $e_i \in E$ is a *maximal* rectangle, i.e. $\forall (e_i, e_j) \in E \times E / i \neq j, x_i^e < x_j^e \vee x_i^e + w_i^e > x_j^e + w_j^e \vee y_i^e < y_j^e \vee y_i^e + h_i^e > y_j^e + h_j^e$ (e_i is not included into e_j). Note that the notion of "maximal rectangular empty space" seems to have been independently introduced in [30] (where it is called "maximal area") and [8] ("maximal hole"). In particular, it was proved in [30] that $|E|$ is at most in $\mathrm{O}(n^2)$.

3.2 Initial Solution

In local search algorithms, the initial solution s_0 specifies where the search begins in S. TSD uses the BLF procedure [27] to construct s_0, where the π permutation

orders the $r_i \in R$ first by decreasing width, and, second, by decreasing height if necessary (randomly last). We employed this decoder / order since previous experiments suggested that the `BLF` placement algorithm usually outperforms other ϕ decoders, see [28,31] for instance.

3.3 Fitness Function

To evaluate a solution $s \in S$, `TSD` uses the following $f$itness (or "evaluation") function f to be minimized:

$$f(s) = \begin{cases} 0 \text{ if } \lceil R \rceil = \emptyset \\ \sum_{r_i \in \lceil R \rceil} w_i^r * (y_i^r + h_i^r - H^* + p_H) \text{ otherwise .} \end{cases} \quad (5)$$

where $\lceil R \rceil \subseteq R$ is the set of rectangles $r_i / y_i^r + h_i^r > H^* - p_H$ (integer $0 < p_H < H^*$, for decrement of the Height) and H^* is the best height found, initially the height $H(s_0)$ of the starting solution s_0 introduced in the previous section.

Roughly speaking, the value $f(s)$ is the area of rectangles exceeding $H^* - p_H$ in P with $f(s) = 0$ meaning $H(s) < H^*$, see Fig. 1 for an example. In other words, f measures the quality of s with respect to the current satisfaction problem considered, defined by H^* and p_H: Is there a solution $s \in S / H(s) \leq H^* - p_H$? f is used to compare any $(s, s') \in S \times S$: s is better than s' if $f(s) < f(s')$. `TSD` maintains a set S^* of best solutions according to (5) with $|S^*| \leq p_*$ and $S^* = \{s_0\}$ at the beginning of the search. S^* is used for the diversification process described in Sect. 3.6.

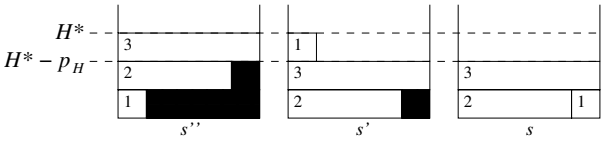

Fig. 1. Let r_1 be the unit square. $f(s'') = W$ since $\lceil R \rceil = \{r_3\}$ for s''. Similarly, $f(s') = 1$ and $f(s) = 0$.

3.4 Neighborhoods and Their Exploitation

A Neighborhood $N : S \rightarrow S$ is an application used to explore S (and to guide the search process) such as $\forall s \in S, s' \in N(s)$ if s and s' only differ by a particular operation called a "move" (noted μ). `TSD` integrates two different *complementary* neighborhoods called N_1 (performed with probability p_N) and N_2 (probability $1 - p_N$), both of them are based on the principle of ejection chains. The goal of this combination is to reduce the height of the packing while avoiding solutions with (hard to fill) tall and thin wasted spaces.

Each time a move is performed from s to s' (at iteration m), S^* and H^* are updated if necessary, and only whenever $s' \notin S^*$, with the following rules where $\lceil s^* \rceil$ (resp. $\lfloor s^* \rfloor$) is the worst (resp. a best, found at iteration m^*) element in S^* according to (5) (consider the $\lceil s^* \rceil$ introduced the most recently):

- $f(s') < f(\lfloor s^* \rfloor) \Rightarrow \delta \leftarrow m - m^*$, $p_D \leftarrow m + p_I * \delta$, $m^* \leftarrow m$. δ (number of moves required to improve $\lfloor s^* \rfloor$), p_I (for "*I*ncrement") and p_D are used for the *D*iversification process detailed in Sect. 3.6
- $f(s') = 0 \Rightarrow H^* \leftarrow H(s')$
- $|S^*| < p_* \Rightarrow S^* \leftarrow S^* \cup \{s'\}$
- $|S^*| = p_* \wedge f(s') < f(\lceil s^* \rceil) \Rightarrow S^* \leftarrow S^* \setminus \lceil s^* \rceil \cup \{s'\}$

N_1 and N_2 are based on the ejection chain principle and share a common characteristic: They move (at least) one rectangle r_i to another location. This new location for r_i may generate overlaps with a set $R_{r_i} \subset R$ of other rectangles: $R_{r_i} = \left\{r_j \in R / j \neq i \wedge x_i^r < x_j^r + w_j^r \wedge x_i^r + w_i^r > x_j^r \wedge y_i^r < y_j^r + h_j^r \wedge y_i^r + h_i^r > y_j^r\right\}$. To repair the overlaps between r_i and all $r_j \in R_{r_i}$ (i.e. to insure $s' \in S$), all $r_j \in R_{r_i}$ are removed from P, sorted like in Sect. 3.2 and, then, relocated with `BLF`.

Finally, notice that changing the location of r_i and the deletion or repositioning of all $r_j \in R_{r_i}$ (possibly) imply updates of E. This is done using the efficient "incremental" procedures introduced in [8,20].

N_1: Reduce the Height of the Packing. This is done by moving a $r_i \in \lceil R \rceil$ **below** its current location (x_i^r, y_i^r), at the bottom-left corner either of an empty space $e_j \in E$ (defining a sub-neighborhood N_1^E) or of another $r_j \in R$ (defining N_1^R).

Start with N_1^E. From the current s, **all** $r_i \in \lceil R \rceil$ (considered from the highest and left-most to the lowest and right-most) are first tried to be relocated to the $\left(x_j^e, y_j^e\right)$ coordinates of **all** $e_j \in E / y_j^e < y_i^r \wedge x_j^e + w_i^r \leq W$. This generates $|\lceil R \rceil|$ sets $N_1^E(s,i)$ of neighbors: $N_1^E(s) = \cup_{r_i \in \lceil R \rceil} N_1^E(s,i)$. Let $\lfloor N_1^E(s) \rfloor \subseteq N_1^E(s)$ be the set of the *best evaluated* neighbors of s according to N_1^E and (5).

If $f(s') = 0 \; \forall s' \in \lfloor N_1^E(s) \rfloor$, select randomly one $s' \in \lfloor N_1^E(s) \rfloor$ minimizing (1) to become the new "current" solution for the next iteration ($s \leftarrow s'$). Otherwise, if $f(s') < f(s) \forall s' \in \lfloor N_1^E(s) \rfloor$ make $s \leftarrow s'$ (select s' randomly if $\lfloor N_1^E(s) \rfloor$ contains more than one such element).

Possibly Apply N_1^R. If N_1^E cannot improve s according to (5), i.e. if $f(s') \geq f(s) \forall s' \in \lfloor N_1^E(s) \rfloor$, try to do so with N_1^R. In this case, $N_1^R(s)$ is explored in a **random** order and the **first** improving move (if available) is performed.

From the current s, a random $r_i \in \lceil R \rceil$ is first selected. Then, all $r_j \in R / j \neq i \wedge y_j^r < y_i^r \wedge x_j^r + w_i^r \leq W$ are considered in a random order. If transferring r_i at $\left(x_j^r, y_j^r\right)$ leads to an improved solution s', the exploration of $N_1^R(s)$ halts and the search continues from s' ($s \leftarrow s'$). Otherwise, another such r_j is selected. If all r_j have been tried, the exploration of $N_1^R(s)$ continues with another r_i.

In the Worst Case, Make a Best Non-Improving Move. Let $\lfloor N_1(s) \rfloor$ be the set of the *best evaluated* neighbors of s according to N_1^E, N_1^R and (5). If N_1 cannot improve s, i.e. $f(s') \geq f(s) \forall s' \in \lfloor N_1(s) \rfloor$, a random **non-improving** neighbor $s' \in \lfloor N_1(s) \rfloor$ is selected for the next iteration: $s \leftarrow s'$.

N_2: Avoid Solutions with Tall and Thin Wasted Spaces. This second neighborhood relies on the following empirical observation: Some empty spaces (usually tall and thin) have a low area / perimeter ratio. Intuitively, since they are often located on the borders of the strip, they cannot be used with N_1^E. Indeed, preliminary computational experiments, using only N_1, has shown that some of them (those with a maximum perimeter) were persistent and hard to fill. N_2 has thus been designed to (try to) avoid these situations, i.e. to concentrate on these particular empty spaces to limit their number.

Let $\lceil E \rceil \subseteq E$ be the set of empty spaces with maximum perimeter: $\lceil E \rceil = \left\{ e_{j'} \in E / w^e_{j'} + h^e_{j'} \geq w^e_{j''} + h^e_{j''} \forall e_{j''} \in E \right\}$. Choose the empty space $e_j \in \lceil E \rceil$ located the highest and the left-most in P, i.e. $y^e_j \geq y^e_{j'} \wedge x^e_j \leq x^e_{j'} \forall e_{j'} \in \lceil E \rceil$. e_j is said to be "not adjacent" to $r_k \in R$ (shortly noted "$e_j \cap r_k = \emptyset$") if $x^r_k > x^e_j + w^e_j \vee x^r_k + w^r_k < x^e_j \vee y^r_k > y^e_j + h^e_j \vee y^r_k + h^r_k < y^e_j \vee \left(\left(x^r_k = x^e_j + w^e_j \vee x^r_k + w^r_k = x^e_j \right) \wedge \left(y^r_k + h^r_k = y^e_j \vee y^r_k = y^e_j + h^e_j \right) \right)$. Let $\lceil R_{e_j} \rceil \subset R$ be the set of rectangles with a greater area than that of e_j and that are not adjacent to e_j: $\lceil R_{e_j} \rceil = \left\{ r_k \in R / w^r_k * h^r_k > w^e_j * h^e_j \wedge e_j \cap r_k = \emptyset \right\}$.

From the current s, **all** $r_i \in \lceil R_{e_j} \rceil$ (considered from the highest and left-most to the lowest and right-most) are first tried to be relocated to the four corner of e_j. To be more precise, the bottom-left corner of each $r_i \in \lceil R_{e_j} \rceil$ is positioned at (x^e_j, y^e_j) if $x^e_j + w^r_i \leq W$, $(x^e_j + w^e_j - w^r_i, y^e_j)$ if $x^e_j + w^e_j - w^r_i \geq 0$, $(x^e_j, y^e_j + h^e_j - h^r_i)$ if $y^e_j + h^e_j - h^r_i \geq 0 \wedge x^e_j + w^r_i \leq W$ and $(x^e_j + w^e_j - w^r_i, y^e_j + h^e_j - h^r_i)$ if $y^e_j + h^e_j - h^r_i \geq 0 \wedge x^e_j + w^e_j - w^r_i \geq 0$. This generates $| \lceil R_{e_j} \rceil |$ sets $N_2(s,i)$ of neighbors with $1 \leq |N_2(s,i)| \leq 4$: $N_2(s)$ is the union of these sets. Let $\lfloor N_2(s) \rfloor$ be the set of the *best evaluated* neighbors of s according to N_2 and (5): $\lfloor N_2(s) \rfloor = \{ s' \in N_2(s) / \forall s'' \in N_2(s), f(s') \leq f(s'') \}$. Similarly to N_1, if $f(s') = 0 \ \forall s' \in \lfloor N_2(s) \rfloor$, select randomly one $s' \in \lfloor N_2(s) \rfloor$ minimizing (1). Otherwise, choose $s' \in \lfloor N_2(s) \rfloor$ at random for the next iteration.

3.5 Tabu List

One fundamental component of TS is a special short-term memory that maintains a selective history of the search. It is composed of previously encountered solutions or, more generally, pertinent attributes of such solutions. The aims of a *T*abu *L*ist (shortly "TL") are to avoid cycling and to go beyond local optima.

At current iteration m, since a `TSD` move μ_m from s to a neighbor $s' \in N(s)$ consists in relocating (at least) one $r_i \in R$ from (x^r_i, y^r_i) to another location (x'^r_i, y'^r_i), i.e. $x'^r_i \neq x^r_i \vee y'^r_i \neq y^r_i$, it seems quite natural to forbid r_i to return to (x^r_i, y^r_i) from s'. This "reverse" move (noted μ_m^{-1}), that can be characterized by $\mu_m^{-1} = (i, x^r_i, y^r_i)$, will then be stored in TL (shortly $TL \leftarrow TL \cup \left\{ \mu_m^{-1} \right\}$) for a duration TT (called the "*T*abu *T*enure") to indicate that μ_m^{-1} is forbidden, at least up to iteration $m + TT$. In `TSD`, TT is a random integer number from $[p_{\min TT}, \ldots, p_{\max TT}]$.

This strategy has one main drawback. Assume that μ_m has been performed and that the next move μ_{m+1} relocates a $r_j \in R$ to (x^r_i, y^r_i) such that $j \neq i$ and μ_{m+1} is not tabu ($\mu_{m+1} \notin TL$). If r_j and r_i are of the same dimensions,

i.e. $w_i^r = w_j^r \wedge h_i^r = h_j^r$, μ_{m+1} may return the search to s (already visited) or to solutions (already visited or not) too close to s.

To avoid these situations, TSD does not record (i, x_i^r, y_i^r) but $(w_i^r, h_i^r, x_i^r, y_i^r)$. Note that the following simple "aspiration criterion" (that removes a tabu status) is available in TSD: A tabu move is always accepted if it leads to a solution s' that is better than the best solution $\lfloor s^* \rfloor$ ever found ($f(s') < f(\lfloor s^* \rfloor)$). Furthermore, if **all** potential moves are tabu, we select the first one with the lowest tabu duration that would be performed if all tabu status were (temporarily) removed.

3.6 Diversification

The TSD algorithm maintains a set S^* of high quality solutions obtained from the beginning of the search (see Sect. 3.3 and 3.4). These elite solutions are used as candidates for diversification.

When the current search cannot be improved for a number of iterations, TSD picks one solution $s^* \in S^*$ at random. This solution is then slightly perturbed by moving a random $r_i \in R$ to $(x, H^* - p_H - h_i^r)$, where x is chosen randomly from $[0 \dots W - w_i^r]$. Rectangles overlapping with r_i are relocated like in Sect. 3.4. This perturbed solution becomes finally the new current solution ($s \leftarrow s^*$) with the tabu list reset to empty (except the move used for perturbation).

3.7 TSD: An Overview

The TSD algorithm begins with an initial solution (Sect. 3.2). Then it proceeds iteratively to visit a series of (locally best) solutions following the neighborhoods (Sect. 3.4). At each iteration, the current solution s is replaced by a neighboring solution s' even if s' does not improve s.

While it is not mentioned here for simplicity, note that TSD can also end (see Step 2 below) before reaching the maximum time *L*imit p_L. This may occur each time S^* is updated whenever the optimum height H_{OPT} (or an upper bound) is known and $H(\lfloor s^* \rfloor) \leq H_{OPT}$.

1. *Initialization.* $m \leftarrow 0$ (current number of iterations), build s using BLF. $H^* \leftarrow H(s)$, $S^* \leftarrow \{s\}$, $m^* \leftarrow 0$, $TL \leftarrow \emptyset$.
2. *Stop condition.* If elapsed time has reached p_L Then: Return H^* and $\lfloor s^* \rfloor$.
3. *Diversification.* If $m > p_D$ Then:
   - Choose at random $s \in S^*$, $r_i \in R$ and $x \in [0 \dots W - w_i^r]$.
   - Update s by relocating r_i to $(x, H^* - p_H - h_i^r)$, defining a move μ.
   - $TL \leftarrow \{\mu^{-1}\}$, $m^* \leftarrow m$, $p_D \leftarrow m + p_I * \delta$. Possibly update S^* or H^*.
4. *Exploration of the neighborhood.* Let N be N_1 or N_2 according to p_N. Update s according to $N(s)$, defining a move μ. $m \leftarrow m + 1$. $TL \leftarrow TL \cup \{\mu^{-1}\}$. Possibly update S^*, H^* or p_D. Go to step 2.

4 Experimentations

We used a set of 21 well-known benchmark instances [28] available from http://mo.math.nat.tu-bs.de/packlib/xml/ht-eimhh-01-xml.shtml to compare

Table 1. Main characteristics of the test problems from [28]

Category	Instances	W	n	H_{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, 196	240

`TSD` with previously reported TS algorithms as well as the best performing approaches. The main characteristics of these instances are given in Tab. 1. Note that these benchmarks have a known optimal height H_{OPT}.

4.1 Experimentation Conditions

The comparison is based on the percentage gap γ of a solution s from the optimum or its best bound (H_{OPT}): $\gamma(s) = 100 * (H(s) - H_{OPT}) / H(s)$. Similarly to the best-known approaches considered in Tab. 2, mean gap $\overline{\gamma}$ (resp. best gap γ^*) is averaged over a number of 10 runs (resp. over best runs only), each run being *L*imited to p_L seconds.

The `TSD` parameters are: $p_H = 1$ (to define the current satisfaction problem to solve), $p_* = 30$ (maximum size of S^*), $p_N = 0.65$ (probability to explore neighborhood N_1), $p_{\min TT} = 5$ and $p_{\max TT} = 15$ (minimum and maximum *T*abu *T*enure), $p_D = 10$ and $p_I = 3$ (for diversification), $p_L \in [60 \ldots 2\,700]$ (time *L*imit, in seconds). `TSD` is coded in `C++` and all computational results were obtained running `TSD` on a 2 Ghz Dual Core PC.

4.2 Computational Results

`TSD` is compared in Tab. 2 with the previously reported TS algorithms denoted as "`TS1`" [13] and "`TS2`" [12] and the best performing approaches: `GRASP` [6] and `IDW` [7]. Note that TS was also tried in [8], achieving "good performance", but no numerical results were reported. While the stopping criterion per run for `GRASP`, `IDW` and `TS1` is also a time *L*imit p_L (60, 100 and 360 seconds resp.), `TS2` used a maximum number of iterations (1 500).

In Table 2, "-" marks mean unknown information, "Mean C*i*" are averaged values on category C*i*, the last three lines reporting averaged values for the largest (and hardest) instances and all the 21 instances, and (minimum) number of instances optimally solved. No $\overline{\gamma}$ or γ^* is mentioned for `IDW`, `TS2` and `TS1` since this information is not given in [7,12,13].

According to Tab. 2, `TS1` is the worst performing (TS) approach for the benchmark tried. Indeed, $\gamma^* = 0$ only for C2P1 and C2P3 while other methods *always* solved at least 6 instances.

On the "smallest" (easiest) instances C1–C3, `TS2` is the best method since it always solved **all** the 9 instances. However, `TSD` (and `GRASP`) compares well with `TS2` since only one instance was not solved optimally ($H^* = 31$ for C3P2).

The largest instances (C4–C7) are quite challenging since no approach reached H_{OPT} to our knowledge on these instances (except IDW perhaps but this is not clearly mentioned in [7]). Note that, while IDW achieved here the smallest $\overline{\gamma}$ value, TSD is slightly better than GRASP on C6 ($1.37 < 1.56$) and C7 ($1.23 < 1.36$), see also line "Mean C4–C7" where $1.33 < 1.41$.

Table 2. Mean and best percentage gap ($\overline{\gamma}$ and γ^* resp.) on instances from [28]

Instances	TSD		TS1 [13]	TS2 [12]	GRASP [6]		IDW [7]
	$\overline{\gamma}$	γ^*	γ^*	γ^*	$\overline{\gamma}$	γ^*	$\overline{\gamma}$
C1P1	0	0	9.09	0	0	0	-
C1P2	4.76	0	9.09	0	0	0	-
C1P3	0	0	4.76	0	0	0	-
Mean C1	1.59	0	7.65	0	0	0	0
C2P1	0	0	0	0	0	0	-
C2P2	0	0	6.25	0	0	0	-
C2P3	0	0	0	0	0	0	-
Mean C2	0	0	2.08	0	0	0	0
C3P1	0	0	3.23	0	0	0	-
C3P2	3.23	3.23	9.09	0	3.23	3.23	-
C3P3	0	0	9.09	0	0	0	-
Mean C3	1.08	1.08	7.14	0	1.08	1.08	2.15
C4P1	1.64	1.64	6.25	-	1.64	1.64	-
C4P2	1.64	1.64	4.76	-	1.64	1.64	-
C4P3	1.64	1.64	3.23	-	1.64	1.64	-
Mean C4	1.64	1.64	4.75	-	1.64	1.64	1.09
C5P1	1.1	1.1	5.26	-	1.1	1.1	-
C5P2	1.1	1.1	3.23	-	1.1	1.1	-
C5P3	1.1	1.1	6.25	-	1.1	1.1	-
Mean C5	1.1	1.1	4.91	-	1.1	1.1	0.73
C6P1	1.64	0.83	4.76	-	1.56	0.83	-
C6P2	0.83	0.83	3.23	-	1.56	0.83	-
C6P3	1.64	0.83	3.23	-	1.56	0.83	-
Mean C6	1.37	0.83	3.74	-	1.56	0.83	0.83
C7P1	1.23	1.23	-	-	1.64	1.64	-
C7P2	1.23	1.23	-	-	1.19	0.83	-
C7P3	1.23	1.23	-	-	1.23	1.23	-
Mean C7	1.23	1.23	-	-	1.36	1.23	0.41
Mean C4–C7	1.33	1.2	-	-	1.41	1.2	0.76
Mean C1–C7	1.14	0.84	-	-	0.96	0.84	0.4
$\#H_{OPT}/21$	8		≥ 2	9	8		9

5 Possible Extensions

The TSD approach reported in this paper is in fact the first version of an ongoing study. In this section, we discuss possible extensions which are worthy of further study and would help to improve the performance of TSD: Diversification, combined utilization of neighborhoods and evaluation function. All these points merit certainly more investigations and constitute our ongoing work.

Diversification: The diversification technique described in Sect. 3.6 is based on a random perturbation strategy. This strategy can be reinforced by a more elaborated strategy using useful information extracted from high quality solutions. For instance, it would be possible to identify a set of *critical* objects

that prevent the search from converging toward a good packing and then focus on these objects in order to realize a *guided* diversification.

Combined utilization of neighborhoods: Two neighborhoods are proposed in Sect. 3.4. They are used in a particular manner, applying N_1 with probability p_N or N_2 with probability $1 - p_N$. These two neighborhoods can also be employed in other combined ways, for instance, by the union of N_1 and N_2 ($N_1 \cup N_2$) or sequentially (token-ring, $N_1 \rightarrow N_2 \rightarrow N_1 \ldots$).

Evaluation function: The current evaluation function (see Sect. 3.3) is unable to distinguish two solutions with the same height. However, such a situation occurs often during a search process. To overcome this difficulty, it would be useful to introduce an additional criterion into the evaluation function. For instance, the free surface under H^* may be such a potential criterion. As such, we can say a solution s' is better than another solution s if s' has a larger total free area under H^* than s does even if both solution have the same height. Indeed, it would be easier to improve s' than s.

6 Conclusions

In this paper, we presented `TSD`, a Tabu Search algorithm for the 2D Strip Packing Problem. `TSD` uses a solution strategy which traits the initial *optimization* problem as a succession of *satisfaction* problems: Starting from a packing s_0 of height H, `TSD` tries to solve the 2D-SPP with decreasing values of H.

`TSD` uses a direct representation of the search space which permits inexpensive basic operations. Two *complementary* neighborhoods using ejection chains are explored in a combined way by `TSD`. The goal of this combination is to reduce H^* (hence to solve the current satisfaction problem) while avoiding solutions with (hard to fill) tall and thin wasted spaces. A *specific* fitness function f is designed to guide the search. A *diversification* mechanism, relying on a set of historically best packings, helps to direct the search to promising and unexplored regions. The tabu structure includes knowledge of the problem to avoid visiting similar packings.

Preliminary computational results were reported on a set of 21 well-known benchmark instances, showing competitive results in comparison with two recent best performing algorithms. The results on the largest and hardest instances are particularly promising. Several issues were identified for further improvements.

Acknowledgments. We would like to thank the reviewers of the paper for their useful comments. This work was partially supported by two grants from the French "Pays de la Loire" region (MILES and RadaPop projects).

References

1. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research 183(3), 1109–1130 (2007)
2. Dowsland, K., Dowsland, W.: Packing problems. European Journal of Operational Research 56(1), 2–14 (1992)

3. Sweeney, P., Ridenour Paternoster, E.: Cutting and packing problems: A categorized, application-orientated research bibliography. Journal of the Operational Research Society 43(7), 691–706 (1992)
4. Fowler, R., Paterson, M., Tanimoto, S.: Optimal packing and covering in the plane are NP-complete. Information Processing Letters 12(3), 133–137 (1981)
5. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completness. W.H. Freeman and Company, San Francisco (1979)
6. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive GRASP for the strip-packing problem. Computers & Operations Research 35(4), 1065–1083 (2008)
7. Neveu, B., Trombettoni, G.: Strip packing based on local search and a randomized best-fit. In: Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: First Workshop on Bin Packing and Placement Constraints (CPAIOR: BPPC), Paris, France, May 22 (2008)
8. Neveu, B., Trombettoni, G., Araya, I.: Incremental move for strip-packing. In: Avouris, N., Bourbakis, N., Hatzilygeroudis, I. (eds.) Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), vol. 2, pp. 489–496. IEEE Computer Society, Los Alamitos (2007)
9. Soke, A., Bingul, Z.: Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems. Engineering Applications of Artificial Intelligence 19(5), 557–567 (2006)
10. Zhang, D., Kang, Y., Deng, A.: A new heuristic recursive algorithm for the strip rectangular packing problem. Computers & Operations Research 33(8), 2209–2217 (2006)
11. Zhang, D., Liu, Y., Chen, S., Xie, X.: A meta-heuristic algorithm for the strip rectangular packing problem. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 1235–1241. Springer, Heidelberg (2005)
12. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: A tabu search algorithm for a two-dimensional non-guillotine cutting problem. European Journal of Operational Research 183(3), 1167–1182 (2007)
13. Iori, M., Martello, S., Monaci, M.: Metaheuristic algorithms for the strip packing problem. In: Pardalos, P.M., Korotkikh, V. (eds.) Optimization and Industry: New Frontiers. Applied Optimization, vol. 78, pp. 159–179. Springer, Heidelberg (2003)
14. Gómez-Villouta, G., Hamiez, J.P., Hao, J.K.: A dedicated genetic algorithm for two-dimensional non-guillotine strip packing. In: Proceedings of the 6th Mexican International Conference on Artificial Intelligence, Special Session, MICAI, Aguascalientes, Mexico, pp. 264–274. IEEE Computer Society, Los Alamitos (2007)
15. Bortfeldt, A.: A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research 172(3), 814–837 (2006)
16. Mukhacheva, E., Mukhacheva, A.: The rectangular packing problem: Local optimum search methods based on block structures. Automation and Remote Control 65(2), 248–257 (2004)
17. Yeung, L., Tang, W.: Strip-packing using hybrid genetic approach. Engineering Applications of Artificial Intelligence 17(2), 169–177 (2004)
18. Leung, T., Chan, C., Troutt, M.: Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. European Journal of Operational Research 145(3), 530–542 (2003)
19. Gomez, A., de la Fuente, D.: Solving the packing and strip-packing problems with genetic algorithms. In: Mira, J., Sánchez-Andrés, J. (eds.) IWANN 1999. LNCS, vol. 1606, pp. 709–718. Springer, Heidelberg (1999)

20. Neveu, B., Trombettoni, G., Araya, I., Riff, M.C.: A strip packing solving method using an incremental move based on maximal holes. International Journal on Artificial Intelligence Tools 17(5), 881–901 (2008)
21. Araya, I., Neveu, B., Riff, M.C.: An efficient hyperheuristic for strip-packing problems. In: Cotta, C., Sevaux, M., Sörensen, K. (eds.) Adaptive and Multilevel Metaheuristics. Studies in Computational Intelligence, vol. 136, pp. 61–76. Springer, Heidelberg (2008)
22. Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., Nagamochi, H.: Exact algorithms for the 2-dimensional strip packing problem with and without rotations. European Journal of Operational Research (2008) (to appear)
23. Bekrar, A., Kacem, I., Chu, C.: A comparative study of exact algorithms for the two dimensional strip packing problem. Journal of Industrial and Systems Engineering 1(2), 151–170 (2007)
24. Lesh, N., Marks, J., McMahon, A., Mitzenmacher, M.: Exhaustive approaches to 2D rectangular perfect packings. Information Processing Letters 90(1), 7–14 (2004)
25. Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip packing problem. INFORMS Journal on Computing 15(3), 310–319 (2003)
26. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
27. Baker, B., Coffman Jr., E., Rivest, R.: Orthogonal packings in two dimensions. SIAM Journal on Computing 9(4), 846–855 (1980)
28. Hopper, E., Turton, B.: An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research 128(1), 34–57 (2001)
29. Chazelle, B.: The bottom-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers 32(8), 697–707 (1983)
30. El Hayek, J.: Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures. Ph.D thesis, Université de Technologie de Compiègne, France (2006) (in French)
31. Imahori, S., Yagiura, M., Nagamochi, H.: Practical algorithms for two-dimensional packing. In: Gonzalez, T. (ed.) Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/CRC Computer & Information Science Series, ch. 36, vol. 13. CRC Press, Boca Raton (2007)