Tabu Search with Consistent Neighbourhood for Strip Packing

Giglia Gómez-Villouta, Jean-Philippe Hamiez[⋆], and Jin-Kao Hao

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

Abstract. This paper introduces a new tabu search algorithm for a strip packing problem. It integrates several key features: A *consistent* neighborhood, a fitness function including *problem knowledge*, and a diversification based on the *history* of the search. The neighborhood only considers valid, sometimes partial, packings. The fitness function incorporates measures related to the empty spaces. Diversification relies on a set of historically "frozen" objects. Experimental results are shown on a set of well-known hard instances and compared with previously reported tabu search algorithms as well as the best performing algorithms.

Keywords: Tabu search, strip packing, consistent neighborhood.

1 Introduction

This paper is dedicated to the regular, non-guillotine, and without rotation two-dimensional Strip Packing Problem (2D-SPP): Given a finite set of *rectangular* objects, pack **all** of them in a strip of fixed width while minimizing its height. 2D-SPP is a NP-hard problem with a number of practical applications [1,2,3,4].

Given the NP-hard nature of 2D-SPP, many (meta)heuristic procedures have been tried: Greedy randomized adaptive search procedure (GRASP) [5], intensification / diversification walk (IDW) [6], simulated annealing [7,8], tabu search (TS) [7,9,10,11], genetic algorithms [8,11,12], hybrid (meta)heuristics [8,11], and hyper-heuristics [13]. Exact algorithms have also been considered but they are usually limited to "small" instances [14,15].

In this paper, we present CTS (for "Consistent Tabu Search"), a new TS algorithm dedicated to 2D-SPP. Computational results suggest that CTS may be of great interest to efficiently solve 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 finally shown in Sect. 4.

2 Problem Formulation

A strip is a 2D vertical space with fixed width W and infinite height. The bottom-left (BL) corner of the strip stands for the $(0, 0)$ point of an xy-plane

⋆ Contact author.

© Springer-Verlag Berlin Heidelberg 2010

where the x-axis (respectively y-axis) is the direction of the width (resp. height). The set R, for "Rectangles", of $n \geq 2$ objects to be positioned in the strip is $R = \{r_1, \ldots, r_n\}$ where the $w$eight (resp. $h$eight) of each $r_{1 \leq i \leq n}$ is $0 < w_i^r \leq W$ (resp. $h_i^r > 0$). The 2D-SPP is then to determine the (x_i^r, y_i^r) coordinates of the BL corner of all $r_i \in R$ so as to minimize the height H of the resulting packing:

$$\begin{aligned} \text{Minimize: } & H = \max_{1 \leq i \leq n} \{y_i^r + h_i^r\} && (1) \\ \text{Subject to: } & 0 \leq x_i^r \leq W - w_i^r \wedge y_i^r \geq 0 && (2) \\ & \wedge (x_i^r \geq x_j^r + w_j^r \vee x_i^r + w_i^r \leq x_j^r && (3) \\ & \vee y_i^r \geq y_j^r + h_j^r \vee y_i^r + h_i^r \leq y_j^r) \ . && (4) \end{aligned}$$

where (2) forces each 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 neither horizontally nor vertically, respectively.

3 CTS: A Consistent Tabu Search for 2D-SPP

Tabu search is an advanced metaheuristic designed for tackling hard combinatorial optimization problems [16]. We first introduce here the way the problem is addressed (Sect. 3.1). Next sections (3.2–3.7) describe then the problem-specific components of CTS, where all p variables (with subscripts) are $p$arameters whose values will be given in the experimentation part (Sect. 4.1). The general CTS procedure is finally summarized in Sect. 3.8.

3.1 Solving Scheme

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

CTS treats the 2D-SPP *optimization* problem (minimizing H) as *successive* 2D-SPP$_k$. Starting from a complete packing s_0 of height $H(s_0)$, e.g. obtained with a greedy method (see Sect. 3.3), CTS tackles 2D-SPP$_k$ with decreasing values of $H(s_0)$ for k. To be more precise, if CTS finds a solution s to 2D-SPP$_k$, it then tries to solve 2D-SPP$_{H(s)-p_H}$ ($p_H > 0$, for decrement of the height).

3.2 Search Space: A Direct Representation

Many approaches for the 2D-SPP consider a search space S composed of the set of all permutations of the objects, see [11] for instance. More precisely, a permutation π of $[1, \ldots, n]$ is used to introduce an order for all the objects which is followed by a given placement heuristic ϕ (or "decoder"). In other words, given (π, 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. "Bottom Left Fill" (BLF, shortly described in next section) is such a heuristic [17].

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:

- $L \subseteq R$ 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) $\forall (r_i, r_{j \neq i}) \in L \times L$. So, let $\overline{L} \leftarrow R \backslash L$ be the set of "free" objects, i.e. rectangles not (yet) located.
- E is a set of *rectangular E*mpty spaces in the strip. Each $e_i \in E$ 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 $e_i \in E$ is a *maximal* rectangle[1], i.e. $\forall (e_i, e_{j \neq i}) \in E \times E, x_i^e < x_j^e \vee x_i^e + w_i^e > x_j^e + w_j^e \vee y_i^e < y_j^e \vee y_i^e + h_i^e > y_j^e + h_j^e$ (e_i is not included in e_j).

3.3 Initial Configuration

CTS uses the BLF procedure [17] to construct an initial configuration $s_0 \in S$, where the π permutation orders all $r_i \in R$ first by decreasing width, secondly by decreasing height (when $w_i^r = w_{j \neq i}^r$), randomly last if necessary ($h_i^r = h_{j \neq i}^r$).

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 O(n^3) [19] to O(n^2) [20] for a permutation of n objects. We employed this decoder / order since previous experiments suggested that the BLF placement algorithm usually outperforms other decoders, see [19,21] for instance.

s_0 is a solution to 2D-SPP$_k$ $\forall k \geq H(s_0)$. So, s_0 provides a trivial upper bound for 2D-SPP: $H_{OPT} \leq H(s_0)$, where H_{OPT} is the *OPT*imum value of (1).

3.4 Fitness Function

This measure, also called "evaluation" or "cost" function, is a key component of TS because it guides the choices of the algorithm at each iteration. CTS uses the following f function (for "*f*itness", to be minimized) to evaluate a (possibly partial) 2D-SPP$_k$ packing $s \in S$:

$$f(s) = \begin{cases} M_w M_h \alpha \text{ if } E = \emptyset \\ \frac{M_w M_h \alpha \delta}{M_a} \text{ otherwise} \end{cases} . \tag{5}$$

where $M_w = \max_{r_i \in \overline{L}} w_i^r$ (*M*aximum *w*idth of free rectangles), $M_h = \max_{r_i \in \overline{L}} h_i^r$ (*M*aximum *h*eight of free rectangles), $\alpha = |\{r_i \in \overline{L} / w_i^r = M_w\}|$ (number of free objects with width M_w), $\delta = \sum_{e_j \in E} (W - x_j^e)(k - y_j^e)^2$, and $M_a = \max_{e_j \in E} w_j^e h_j^e$ (*M*aximum *a*rea of empty spaces).

Roughly speaking, the $f(s)$ value measures the quality of s with respect to 2D-SPP$_k$, the current satisfaction problem considered:

[1] The notion of "maximal rectangular empty space" is called "maximal area" in [18] and "maximal hole" in [7]. $|E|$ is at most in O(n^2) [18].

[2] δ measures the density of s. Indeed, a "small" δ value indicates that (almost) all e_j are concentrated close to the top-right corner of the strip.

- $f(s) = 0$ (or $\overline{L} = \emptyset$ equivalently) signifies that s is a solution to 2D-SPP$_k$, i.e. that $H(s) \leq k$. In this case, s is an *optimal* solution to 2D-SPP (i.e. 2D-SPP$_{k'}$ admits no solution $\forall k' < k$) if $E = \emptyset$: $H_{OPT} = k$.
- $f(s) > 0$ (or $\overline{L} \neq \emptyset$) indicates a *partial* packing. Here, $E = \emptyset$ means that 2D-SPP$_{k'}$ have no solution $\forall k' \leq k$ and that a trivial *lower bound* has been found for 2D-SPP: $H_{OPT} > k$.

f is used to compare any $(s_1, s_2) \in S \times S$: With respect to 2D-SPP$_k$, s_1 is better than s_2 if $f(s_1) < f(s_2)$. However, note that it is inadequate when s_1 and s_2 are both solutions to 2D-SPP$_k$, i.e. when $f(s_1) = f(s_2) = 0$. In this case, s_1 is better than s_2 if $H(s_1) < H(s_2)$.

3.5 Neighborhood and Search Strategy

The neighborhood N is another key element of TS. It defines a structure of the search space S and determines the paths the algorithm will follow to explore S. 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 fitness function f. To be more precise, a neighborhood N over S is any function that associates to each individual $s \in S$ some solutions $N(s) \subset S$. A solution s is a "local optimum" if s is the best (with respect to N and f) among the solutions $s' \in N(s) \cup \{s\}$. 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 $s' \in N(s)$. This transition from a solution s to a neighbor s' is denoted by $s' = s \oplus \mu$. Let $\Gamma(s)$ be the set of all possible moves which can be applied to s, then the neighborhood $N(s)$ of s can formally be defined by: $N(s) = \{s \oplus \mu / \mu \in \Gamma(s)\}$.

The main goal of the `CTS` neighborhood is to empty $\overline{L}$. Basically, it moves one rectangle r_i from $\overline{L}$ to L, at the BL corner either of an empty space $e_j \in E$ (defining a sub-neighborhood N_E) or of another $r_j \in L$ (defining N_L). This location for r_i may generate overlaps with a set $L_i \subseteq L$ of other rectangles: $L_i = \{r_{j \neq i} \in L / x^r_i < x^r_j + w^r_j \wedge x^r_i + w^r_i > x^r_j \wedge y^r_i < y^r_j + h^r_j \wedge y^r_i + h^r_i > y^r_j\}$. All $r_j \in L_i$ are thus removed from L and added to $\overline{L}$ to repair these overlaps. This principle, known as "ejection chains", is used to make s' *consistent* with respect to (3–4). Finally, notice that locating r_i in the strip and the possible deletion of all $r_j \in L_i$ imply updates of E. This is done using the efficient "incremental" procedures introduced in [7].

Let s_* be the overall *best* complete packing, according to (1), found by `CTS` at iteration m_* (initially $s_* \leftarrow s_0$ and $m_* \leftarrow 0$). Each time a move is performed from s to s', at iteration m, s_* and m_* are updated ($s_* \leftarrow s'$ and $m_* \leftarrow m$) once $\overline{L} = \emptyset \wedge f(s') \leq f(s_*)$.[3]

N_E: Consider First the Empty Spaces. `CTS` examines two cases here, with two different objectives.

[3] $f(s') = f(s_*)$ may occur only after diversification D_T described in Sect. 3.7. In this particular case, s_* and m_* are updated according to some probability $p_{\approx}$.

$N_E^{|\overline{L}|}$: *Reduce* $|\overline{L}|$. **All** $r_i \in \overline{L}$ are tried to be located in the strip to the BL corner of **all** $e_j \in E / x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k \wedge w_j^e \geq w_i^r \wedge h_j^e \geq h_i^r$ (r_i fits *entirely* in e_j). This generates $|\overline{L}|$ sets $N_E^{|\overline{L}|}(s,i)$ of neighbors for s, some possibly empty: $N_E^{|\overline{L}|}(s) = \cup_{r_i \in \overline{L}} N_E^{|\overline{L}|}(s,i)$.

Note that $N_E^{|\overline{L}|}(s) \neq \emptyset$ means that there is at least a $(r_i, e_j) \in \overline{L} \times E$ such that r_i fits in e_j, i.e. that $|\overline{L}|$ can be reduced. Furthermore, in this case, locating r_i will generate no overlap ($L_i = \emptyset$), hence no repair is needed.

Let $\lfloor N_E^{|\overline{L}|}(s) \rfloor \subseteq N_E^{|\overline{L}|}(s)$ be the set of the *best evaluated* neighbors s' of s: $\lfloor N_E^{|\overline{L}|}(s) \rfloor = \{s'_1 \in N_E^{|\overline{L}|}(s) / \forall s'_2 \in N_E^{|\overline{L}|}(s), f(s'_1) \leq f(s'_2)\}$. If $\lfloor N_E^{|\overline{L}|}(s) \rfloor \neq \emptyset \wedge \overline{L} = \emptyset \ \forall s' \in \lfloor N_E^{|\overline{L}|}(s) \rfloor$, select randomly one $s' \in \lfloor N_E^{|\overline{L}|}(s) \rfloor$ minimizing (1) to become the new starting configuration for the next 2D-SPP$_{H(s')-p_H}$ problem (s' is a solution to 2D-SPP$_k$) and possibly update s_* and m_*. Otherwise, if $\lfloor N_E^{|\overline{L}|}(s) \rfloor \neq \emptyset$, select randomly a $s' \in \lfloor N_E^{|\overline{L}|}(s) \rfloor$ minimizing (5) and make $s \leftarrow s'$.

N_E^D: *Make the Packing Denser.* This case occurs only if $N_E^{|\overline{L}|}$ is not applicable from s, i.e. when $N_E^{|\overline{L}|}(s) = \emptyset$ ($|\overline{L}|$ cannot be reduced).

Here again, **all** $r_i \in \overline{L}$ are tried to be located to the BL corner of **all** $e_j \in E$ but the previous condition on e_j and r_i is relaxed to $x_j^e + w_i^r \leq W \wedge y_j^e + h_i^r \leq k$ since we now know that no r_i can fit *entirely* in a e_j (so, overlaps will temporarily appear and be repaired).

Note that $N_E^D(s) = \emptyset$ means either, $\forall (r_i, e_j) \in \overline{L} \times E$, that all e_j are located rather to the right of the strip with $x_j^e + w_i^r > W$ or to its top with $y_j^e + h_i^r > k$, or that it is forbidden to remove all rectangles overlapping with r_i (see Sect. 3.6).

Let $\lfloor N_E^D(s) \rfloor$ be defined similarly to $\lfloor N_E^{|\overline{L}|}(s) \rfloor$. If $\lfloor N_E^D(s) \rfloor \neq \emptyset$, select randomly one $s' \in \lfloor N_E^D(s) \rfloor$ minimizing (5) to become the new "current" configuration for the next iteration ($s \leftarrow s'$).

***N_L*: Possibly Consider All *$r_j \in L$*.** This second sub-neighborhood is explored only if N_E is not applicable from s, i.e. when $N_E^{|\overline{L}|}(s) = N_E^D(s) = \emptyset$.

From the current s, **all** $r_i \in \overline{L}$ are tried to be located to the BL corner of **all** $r_j \in L / (w_i^r \neq w_j^r \vee h_i^r \neq h_j^r) \wedge x_j^r + w_i^r \leq W \wedge y_j^r + h_i^r \leq k$ (r_i and r_j have different sizes). This generates $|\overline{L}|$ sets $N_L(s,i)$ of neighbors for 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 $N_L(s) = \emptyset$ means that all $r_j \in L_i$ cannot be removed from the strip (see Sect. 3.6).

Let $\lfloor N_L(s) \rfloor \subseteq N_L(s)$ be the set of the *best evaluated* neighbors of s according to (5): $\lfloor N_L(s) \rfloor = \{s'_1 \in N_L(s) / \forall s'_2 \in N_L(s), f(s'_1) \leq f(s'_2)\}$. If $N_L(s) \neq \emptyset$, choose $s' \in \lfloor N_L(s) \rfloor$ at random for the next iteration.

A Worst Case. If neither N_E nor N_L is applicable from s, i.e. $N(s) = N_E^{|\overline{L}|}(s) = N_E^D(s) = N_L(s) = \emptyset$, apply diversification (see Sect. 3.7).

3.6 Tabu List

To avoid the problem of possible cycling and to allow the search to go beyond local optima, TS introduces the notion of "tabu list", one of the most important components of the method. A tabu list 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 TS strategy based on this short term memory consists in preventing solutions previously visited from being reconsidered for the next p_τ iterations (integer p_τ, called "tabu tenure", is problem dependent). Now, at each iteration, TS searches for a best neighbor from this dynamically modified neighborhood.

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

Note that τ is made empty at the beginning of the search or when CTS finds a solution s for 2D-SPP$_k$, i.e. if $\overline{L} = \emptyset$.

3.7 Diversification

When $N(s) = \emptyset$ (s has no neighbor) or s_* keeps unchanged for a number $p_* > 0$ of iterations (integer p_*), CTS first resets τ and reloads s_*: $\tau \leftarrow \emptyset, s \leftarrow s_*$. 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 s_*.

D_I: A Basic Perturbation. L and $\overline{L}$ are first built according to 2D-SPP$_k$ with $k = H(s) - p_H$: $L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq k\}, \overline{L} \leftarrow R \backslash L$. D_I considers then all $(r_i, r_j) \in L \times \overline{L} / (w_i^r \neq w_j^r \vee h_i^r \neq h_j^r) \wedge x_i^r + w_j^r \leq W \wedge y_i^r + h_j^r \leq k$ (r_i and r_j have different sizes). It simply interchanges two such rectangles (randomly selected) and makes r_j tabu. Note that swapping r_i with r_j may cause overlaps for r_j. In this case, L_j is repaired like in Sect. 3.5.

D_T: A Perturbation Based on the History. During the overall search process, CTS keeps for each $r_i \in R$ the number F_i (for "*F*requency") of times r_i leaved the strip.[4] D_T considers a π_F permutation that orders all $r_i \in R$ 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.

All $r_i \in \lfloor F \rfloor$ are first temporarily removed from the strip and their frequencies are updated.[5] Then, the *partial* packing is pushed down to the basis of the strip,

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

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

like in the famous Tetris game. Finally, all $r_i \in \lfloor F \rfloor$ are sorted like in Sect. 3.3 and relocated in the strip with BLF.

CTS deals now with 2D-SPP$_{H(s)-p_H}$: $L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq H(s) - p_H\}, \overline{L} \leftarrow R \backslash L$. This means that CTS possibly considers 2D-SPP$_k$ with $k \geq H(s_0) \geq H(s_*)$.

3.8 CTS: The General Procedure

The CTS algorithm begins with an initial *complete* packing (Sect. 3.3). Then it proceeds iteratively to solve a series of 2D-SPP$_k$ *satisfaction* problems. If CTS finds a solution s to 2D-SPP$_k$, it then tries to solve 2D-SPP$_{H(s)-p_H}$.

While it is not mentioned here for simplicity, note that CTS can also end (see Step 3 below) before reaching the Maximum number of allowed moves $p_M \geq 0$ (integer). This may occur each time s_* is updated whenever the optimum height H_{OPT} (or an upper bound) is known and $H(s_*) \leq H_{OPT}$.

1. *Initialization.* Build s using BLF, $m \leftarrow 0, s_* \leftarrow s, m_* \leftarrow m$.
2. *Generation of the starting configuration for the next 2D-SPP$_k$.*
   $L \leftarrow \{r_i \in R / y_i^r + h_i^r \leq H(s) - p_H\}, \overline{L} \leftarrow R \backslash L$.
3. *Stop condition.* If $m = p_M$ Then: Return $H(s_*)$ and s_*.
4. *Exploration of the neighborhood.* If $N(s) = \emptyset$ Then: Go to step 5.
   $m \leftarrow m + 1$. Update s according to $N(s)$.
   If s_* is replaced by s or $\overline{L} = \emptyset$ Then: Go to step 2.
   If $(m - m_*) \bmod p_* \neq 0$ Then: Go to step 3.
5. *Diversification.* Modify s using D_I or D_T according to p_D. Go to step 3.

4 Experimentations

We used the set of 21 well-known hard instances defined in [19].[6] The main characteristics of this benchmark are given in the 4 first columns of Table 1, each of the 7 categories "Cat." being composed of 3 different instances.

4.1 Experimentation Conditions

The comparison is based on the percentage gap γ of a solution s from the optimum: $\gamma(s) = 100 * (1 - H_{OPT}/H(s))$. For CTS, mean gap $\overline{\gamma}$ (resp. best gap γ^*) is averaged over a number of 5 runs (resp. over best runs only) per instance.

The CTS parameters are: $p_H = 1$ (to build the starting configuration of 2D-SPP$_k$, the current satisfaction problem considered), $p_{\approx} = [0.4, \ldots, 0.8]$ (probability that a complete packing s replaces s_* whenever $f(s) = f(s_*)$), $p_{\mathcal{T}} = [2, \ldots, 6]$ (tabu tenure), $p_* = [200, \ldots, 500]$ (maximum number of moves to update s_*), $p_D \in [0.7, \ldots, 1]$ (probability to apply diversification D_I), $p_M \in [1\,000\,000, \ldots, 20\,000\,000]$ (maximum number of allowed moves per run).

CTS is coded in the c programming language (gcc compiler). All computational results were obtained running CTS on a Bull NovaScale R422 server (2.83 Ghz quad-core Intel® Xeon® E5440 processor, 8 Gb RAM).

[6] They are available e.g. from the "PackLib^2" benchmarks library, see `http://www.ibr.cs.tu-bs.de/alg/packlib/xml/ht-eimhh-01-xml.shtml`

4.2 Computational Results

CTS is compared in Table 1 with the previously reported TS algorithms, denoted as TS1 [11], TS2 [10], and TS3 [9], and the best performing approaches: GRASP [5] and IDW [6].[7]

In Table 1, "–" marks and the absence of $\overline{\gamma}$ or γ^* values for TS1, TS2, and IDW mean either that $\overline{\gamma}$ or γ^* cannot be computed or that the information is not given in [6,10,11].

Table 1. Mean and best percentage gaps ($\overline{\gamma}$ and γ^* resp.) on instances from [19]

Instances				CTS		TS1 [11]	TS2 [10]	TS3 [9]		GRASP [5]		IDW [6]
Cat.	W	n	H_{OPT}	$\overline{\gamma}$	γ^*	γ^*	γ^*	$\overline{\gamma}$	γ^*	$\overline{\gamma}$	γ^*	$\overline{\gamma}$
C1	20	16–17	20	0	0	7.65	0	1.59	0	0	0	0
C2	40	25	15	0	0	2.08	0	0	0	0	0	0
C3	60	28–29	30	0.87	0	7.14	0	1.08	1.08	1.08	1.08	2.15
C4	60	49	60	1.64	1.64	4.75	–	1.64	1.64	1.64	1.64	1.09
C5	60	73	90	1.74	1.46	4.91	–	1.1	1.1	1.1	1.1	0.73
C6	80	97	120	2.17	1.91	3.74	–	1.37	0.83	1.56	0.83	0.83
C7	160	196–197	240	2.15	2.04	–	–	1.23	1.23	1.36	1.23	0.41

According to Table 1, TS1 is the worst performing (TS) approach for the benchmark tried. Indeed, all other approaches (except TS1) solved the C1 and C2 instances, see lines C1–C2 where $\gamma^* = 0$ or $\overline{\gamma} = 0$.

To our knowledge, only TS2 (and the exact algorithm from [14]) solved **all** the 9 smallest instances (C1–C3). CTS is the first method reaching the same qualitative results, see lines C1–C3 where γ^* is **always** 0 just for TS2 and CTS. Furthermore, note that CTS achieves here the lowest $\overline{\gamma}$ values compared with GRASP, IDW, and TS3.

CTS compares also well with the competitors if one considers the 3 instances from category C4. Indeed, line C4 indicates the same γ^* values (1.64) for CTS, TS3, and GRASP.

CTS obtains worst γ^* or $\overline{\gamma}$ values than those of the best-known approaches (GRASP and IDW) on the largest 3 categories of instances (C5–C7).

5 Conclusions

In this paper, we presented CTS, a Consistent Tabu Search algorithm for a 2D Strip Packing Problem. CTS treats the initial 2D-SPP *optimization* problem (minimizing the height H) as a succession of 2D-SPP$_{k>0}$ *satisfaction* problems: Is there a solution s to 2D-SPP such that $H(s) \leq k$? Starting from a complete packing s_0, CTS tackles 2D-SPP$_k$ with decreasing values of $H(s_0)$ for k.

[7] For indicative purpose, the mean running time of CTS ranges from less than a second (for the smallest instances) to about 33 hours (for the largest instances). The mean computation time of the competing methods varies from less than a second to about 45 minutes.

The key features of `CTS` include a direct representation of the search space which permits inexpensive basic operations, a consistent neighborhood, a fitness function including problem knowledge, and a diversification based on the history of the search. The performance of `CTS` was assessed on a set of 21 well-known hard instances. The computational experiments showed that `CTS` is able to reach the optimal values for the first 9 problem instances (categories C1–C3) and to match the best results for the next 3 instances (C4). Nevertheless, `CTS` does not compete well with the best performing algorithms on the largest problems (C5–C7), which constitutes the topic for future investigations.

Acknowledgments. 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. We would like to thank the reviewers of the paper for their useful comments.

References

1. Dowsland, K., Dowsland, W.: Packing Problems. Eur. J. Oper. Res. 56(1), 2–14 (1992)
2. Fowler, R., Paterson, M., Tanimoto, S.: Optimal Packing and Covering in the Plane are NP-Complete. Inf. Process. Lett. 12(3), 133–137 (1981)
3. Wäscher, G., Haußner, H., Schumann, H.: An Improved Typology of Cutting and Packing Problems. Eur. J. Oper. Res. 183(3), 1109–1130 (2007)
4. Garey, M., Johnson, D.: Computers and Intractability – A Guide to the Theory of NP-Completness. W.H. Freeman and Company, San Francisco (1979)
5. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive GRASP for the Strip-Packing Problem. Comput. Oper. Res. 35(4), 1065–1083 (2008)
6. Neveu, B., Trombettoni, G.: Strip Packing Based on Local Search and a Randomized Best-Fit. In: 5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems – 1st Workshop on Bin Packing and Placement Constraints (2008)
7. Neveu, B., Trombettoni, G., Araya, I.: Incremental Move for Strip-Packing. In: Avouris, N., Bourbakis, N., Hatzilygeroudis, I. (eds.) ICTAI 2007, vol. 2, pp. 489–496. IEEE Computer Society, Los Alamitos (2007)
8. 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)
9. Hamiez, J.P., Robet, J., Hao, J.K.: A Tabu Search Algorithm with Direct Representation for Strip Packing. In: Cotta, C., Cowling, P. (eds.) EvoCOP 2009. LNCS, vol. 5482, pp. 61–72. Springer, Heidelberg (2009)
10. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: A Tabu Search Algorithm for a Two-Dimensional Non-Guillotine Cutting Problem. Eur. J. Oper. Res. 183(3), 1167–1182 (2007)
11. Iori, M., Martello, S., Monaci, M.: Metaheuristic Algorithms for the Strip Packing Problem. In: Pardalos, P., Korotkikh, V. (eds.) Optimization and Industry – New Frontiers. Appl. Optim., vol. 78, pp. 159–179. Springer, Heidelberg (2003)
12. Bortfeldt, A.: A Genetic Algorithm for the Two-Dimensional Strip Packing Problem with Rectangular Pieces. Eur. J. Oper. Res. 172(3), 814–837 (2006)

13. 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. Stud. Comput. Intell, vol. 136, pp. 61–76. Springer, Heidelberg (2008)
14. Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., Nagamochi, H.: Exact Algorithms for the Two-Dimensional Strip Packing Problem with and without Rotations. Eur. J. Oper. Res. 198(1), 73–83 (2009)
15. Martello, S., Monaci, M., Vigo, D.: An Exact Approach to the Strip Packing Problem. INFORMS J. Comput. 15(3), 310–319 (2003)
16. Glover, F., Laguna, M.: Tabu Search. Kluwer, Dordrecht (1997)
17. Baker, B., Coffman Jr., E., Rivest, R.: Orthogonal Packings in Two Dimensions. SIAM J. Comput. 9(4), 846–855 (1980)
18. El Hayek, J.: Le Problème de Bin-Packing en Deux-Dimensions, le Cas Non-Orienté : Résolution Approchée et Bornes Inférieures (in French). PhD thesis, Université de Technologie de Compiègne, France (2006)
19. Hopper, E., Turton, B.: An Empirical Investigation of Meta-Heuristic and Heuristic Algorithms for a 2D Packing Problem. Eur. J. Oper. Res. 128(1), 34–57 (2001)
20. Chazelle, B.: The Bottom-Left Bin-Packing Heuristic – An Efficient Implementation. IEEE Trans. Comput. 32(8), 697–707 (1983)
21. 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 Comput. & Inf. Sc. Ser, vol. 13, ch. 36. CRC Press, Boca Raton (2007)