Chapter 1

METAHEURISTIC ALGORITHMS FOR THE STRIP PACKING PROBLEM

Manuel Iori, Silvano Martello, Michele Monaci
Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna
Viale Risorgimento, 2 - 40136 - Bologna (Italy)
{miori, smartello, mmonaci }@deis.unibo.it

Abstract Given a set of rectangular items and a strip of given width, we consider the problem of allocating all the items to a minimum height strip. We present a Tabu search algorithm, a genetic algorithm and we combine the two into a hybrid approach. The performance of the proposed algorithms is evaluated through extensive computational experiments on instances from the literature and on randomly generated instances.

Keywords: packing, cutting, metaheuristics

Introduction

In the *Two-Dimensional Strip Packing Problem* (2SP) we are given a set of n rectangular *items* $j = 1, \ldots, n$, each having a *width* w_j and *height* h_j, and a *strip* of width W and infinite height. The objective is to find a pattern that allocates all the items to the strip, without overlapping, by minimizing the height at which the strip is used. We assume that the items have fixed orientation, i.e., they have to be packed with their base parallel to the base of the strip. Consider the following numerical example: $n = 6$, $w_1 = 6$, $h_1 = 3$, $w_2 = 5$, $h_2 = 2$, $w_3 = 2$, $h_3 = 4$, $w_4 = 3$, $h_4 = 4$, $w_5 = 3$, $h_5 = 3$, $w_6 = 2$, $h_6 = 1$. Feasible (and optimal) strip packings of height 7 are depicted in Figure 1.1 (b) and (c).

Problem 2SP has several real-world applications: cutting of paper or cloth from standardized rolls, cutting of wood, metal or glass from standardized stock pieces, allocating memory in computers, to mention the most relevant ones.

The problem is related to the *Two-Dimensional Bin Packing Problem* (2BP), in which, instead of the strip, one has an unlimited number of identical rectangular *bins* of width W and height H, and the objective is to allocate all the items to the minimum number of bins. Recent surveys on two-dimensional packing problems have been presented by Lodi et al., 1999c, and Lodi et al., 2001, while Dyckhoff et al., 1997, have published an annotated bibliography on cutting and packing.

Both 2SP and 2BP are NP-hard in the strong sense. Consider indeed the strongly NP-hard *One-Dimensional Bin Packing Problem* (1BP): partition n elements $j = 1, \ldots, n$, each having a size w_j, into the minimum number of subsets so that the total size in no subset exceeds a given capacity W. It is easily seen that both 2SP and 2BP generalize 1BP.

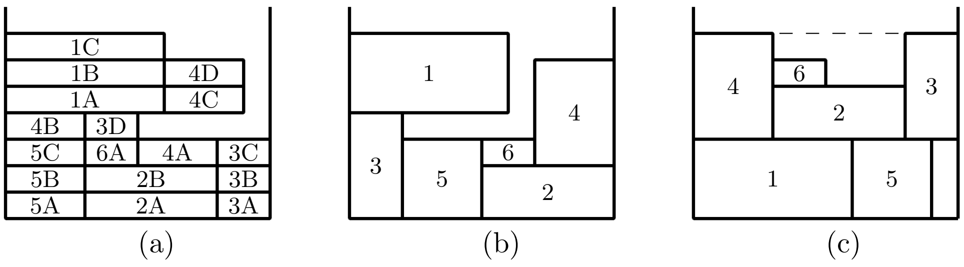

Figure 1.1. (a) optimal solution for the 1BP relaxation; (b) 2SP feasible solution found by algorithm BUILD; (c) 2SP feasible solution found by algorithm TP_{2SP}.

Approximation algorithms for two-dimensional strip packing problems have been presented by Coffman et al., 1980, Baker et al., 1980, Sleator, 1980, Brown, 1980, Golan, 1981, Baker et al., 1981, Baker and Schwarz, 1983, Høyland, 1988, Jakobs, 1996, and Steinberg, 1997. A general framework for the exact solution of multi-dimensional packing problems has been recently proposed by Fekete and Schepers, 1997a, Fekete and Schepers, 1997b, Fekete and Schepers, 1997c, while lower bounds, approximation algorithms and an exact branch-and-bound approach for 2SP have been given by Martello et al., 2000.

In this paper we examine metaheuristic approaches to 2SP (see, e.g., Reeves, 1993, and Davis, 1991, for general introductions to metaheuristics). Section 1 deals with lower bounds from the literature and deterministic approximation algorithms. We present a Tabu search algorithm in Section 2, and a genetic algorithm in Section 3. A hybrid algorithm that combines the two approaches is obtained in Section 4. The results of extensive computational tests are finally presented in Section 5.

We assume in the following, without loss of generality, that all input data are positive integers, and that $w_j \le W$ $(j = 1, \dots, n)$.

1. Lower and upper bounds

In this section we discuss lower bounds and deterministic heuristics that are used in the metaheuristic algorithms introduced in the next sections.

Martello et al., 2000, recently proposed the following lower bound for 2SP. Consider a relaxation in which each item j is "cut" into h_j unit-height *slices* of width w_j. The lower bound given by the minimum height of a strip of width W that packs the resulting $\sum_{j=1}^{n} h_j$ slices can then be determined by solving a *One-Dimensional Contiguous Bin Packing Problem* (1CBP), i.e., a 1BP instance having capacity W, with the additional requirement that for, each original item j, the h_j unit-height slices of width w_j be packed into h_j contiguous one-dimensional bins. As an example, consider again the instance introduced in Section 1: the optimal 1CBP solution is shown in Figure 1.1(a), so a valid lower bound value for the instance is 7.

Given the solution to the 1CBP relaxation, a feasible solution to the original 2SP instance can be obtained through an algorithm, BUILD (see Martello et al., 2000), that re-joins the slices as follows. Consider the first slice of each item, according to non-decreasing one-dimensional bin, breaking ties by decreasing item height. The corresponding two-dimensional item is packed in the lowest position where it fits. It is left (resp. right) justified, if its left (resp. right) edge can touch either the left (resp. right) side of the strip or the right (resp. left) edge of a previous item whose top edge is not lower than that of the current item; otherwise, it is left or right justified, with its edge touching the previous item whose top edge is the tallest one. For the previous example, the items are packed according to the sequence (3, 5, 2, 4, 6, 1), as shown in Figure 1.1(b). Different solutions can be obtained by breaking bin ties according to different policies: decreasing item width or decreasing item area.

A different heuristic was obtained by adapting to 2SP algorithm $\mathrm{TP}_{\mathrm{RF}}$ (*Touching Perimeter*), proposed by Lodi et al., 1999b, for a variant of 2BP. Let L be the lower bound value produced by the 1CBP relaxation. The algorithm, called $\mathrm{TP}_{\mathrm{2SP}}$, initializes the strip at height L, and considers the items according to a given ordering. (When executed from scratch, the items are sorted according to non-increasing area, breaking ties by non-increasing $\min\{w_j, h_j\}$ values.) The first item is packed in the bottom-left corner. Let X (resp. Y) denote the set of x-coordinates (resp. y-coordinates) corresponding to corners of already packed items. Each subsequent item is then packed with its bottom-left corner at a coordinate (x, y) $(x \in X,\ y \in Y)$ that maximizes the *touching fraction*, i.e., the fraction of the item perimeter that touches either the sides of already packed items or the sides of the strip (including the "ceiling" initially placed at height L). If no feasible packing position exists for an item, the strip ceiling is heightened by packing the item in the position for which the increase in the strip height is a minimum. For the previous example, the items are packed according to the sequence (1, 4, 2, 5, 3, 6), as shown in Figure 1.1(c), where the dashed line represents the ceiling. Different solutions can be obtained by sorting the items according to different strategies.

Our *initialization phase*, common to the three metaheuristic algorithms presented in Sections 2, 3 and 4, also includes use of a post-optimization procedure (algorithm POST-OPT, to be described in the next section). It can be outlined as follows:

1 compute lower bound L;

2 execute algorithm BUILD, followed by POST-OPT;

3 execute algorithm $\mathrm{TP}_{\mathrm{2SP}}$, followed by POST-OPT;

4 select the best solution as the incumbent.

2. Tabu search

A Tabu search scheme, 2BP_TS, recently proposed by Lodi et al., 1998, Lodi et al., 1999a, Lodi et al., 1999b, proved to be very effective for two-dimensional bin packing problems. It was thus quite natural to use it as a starting point for the strip packing problem.

The core of our approach can be outlined as follows. Let z^* denote the incumbent solution value for 2SP, initially determined, e.g., through algorithms BUILD and $\mathrm{TP}_{\mathrm{2SP}}$ of Section 1. Let S_{2SP} be an initial feasible solution to 2SP, of value not less than z^*. Let $H \geq \max_j\{h_j\}$ be a given threshold value for an instance of 2BP induced by the item set of the 2SP instance, with bin size $W \times H$. The following procedure returns, for the 2SP instance, a new feasible solution of value z, by exploring 2SP solutions derived from 2BP solutions explored by 2BP_TS. A solution to a 2SP instance is described by the coordinates (x_j, y_j) at which the bottom-left corner of item j is placed $(j = 1, \ldots, n)$, by assuming a coordinate system with origin in the bottom-left corner of the strip. Given a solution S, we denote by $z(S) = \max_j\{y_j + h_j\}$ the corresponding strip height.

```
function EXPLORE($S_{2SP}$,$H$):
   derive from $S_{2SP}$ a feasible solution $S_{2BP}$ for the induced 2BP
     instance;
   execute 2BP_TS starting from $S_{2BP}$;
   let $\Sigma_{2BP}$ be the set of explored 2BP solutions;
   $z := \infty$;
   for each $\overline{S}_{2BP} \in \Sigma_{2BP}$
      derive from $\overline{S}_{2BP}$ a feasible solution $\overline{S}_{2SP}$ for the 2SP instance;
      apply a post-optimization algorithm to $\overline{S}_{2SP}$;
      $z := \min(z, z(\overline{S}_{2SP}))$;
   end for
end
```

We next detail the main steps of EXPLORE. Initially, given an S_{2SP} strip packing, we derive a feasible solution for the induced bin packing instance as follows. The given strip is "cut" into $m = \lceil z(S_{2SP})/H \rceil$ bins, starting from the bottom. For each resulting bin i ($i = 1, \ldots, m-1$), each item crossing the border that separates i from $i+1$ is assigned to a new bin, while the items entirely packed within bin i (if any) remain assigned to i. The whole operation requires $O(n)$ time since, for each item j, we can establish in constant time whether it crosses the border at height $(\lfloor y_j/H \rfloor + 1)H$. The resulting bin packing is clearly not optimized. The reason for this choice is that the computational experiments performed on 2BP_TS (see Lodi et al., 1999b) showed that the algorithm has the best performance when initialized with solutions very far from the optimum.

In the "for each" loop of EXPLORE, we first derive, from a bin packing, a feasible strip packing as follows. We number the bins used in the packing as $1, \ldots, m$, and initially construct a strip of height mH by simply placing the base of bin i at height $(i-1)H$. The resulting strip packing is then improved through item shifting, by considering the items according to non-decreasing y_j value: the current item is shifted down and then left as much as possible. This step can be performed in $O(n^2)$ time by examining, for each item j, the items k with $y_k + h_k \leq y_j$ and establishing, in constant time, whether intervals $(x_j, x_j + w_j)$ and $(x_k, x_k + w_k)$ intersect. (A similar operation is then executed to shift left item j.) A second solution is obtained by considering the bins according to their filling (through the *filling function* introduced in Lodi et al., 1999b), and constructing a strip as follows. We start by placing at the bottom the most filled bin, then we add the least filled one in bottom-up position (i.e., rotated by 180 degrees) then the second most filled one, and so on. The best of the two obtained solutions is finally selected.

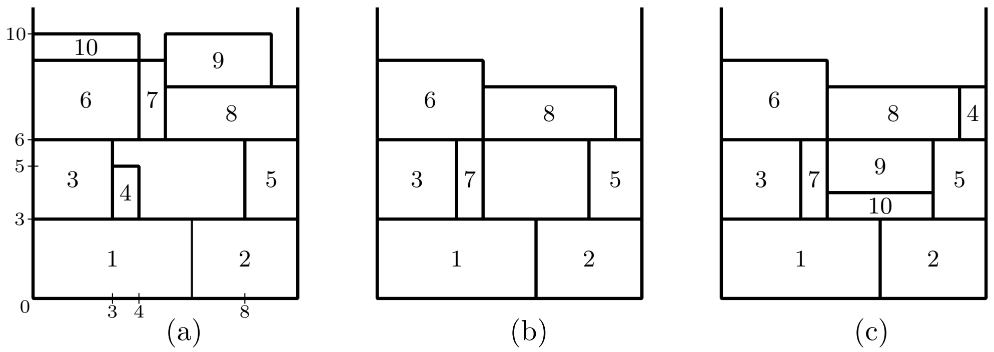

Figure 1.2. (a) initial solution; (b) partial solution after step 4; (c) final solution.

The resulting strip packing is further improved by a post-optimization algorithm, POST-OPT, that iteratively executes the following five steps (illustrated through the example in Figure 1.2).

1 Determine *holes* in the packing, i.e., inner rectangles $[\underline{x}, \underline{y}, \overline{x}, \overline{y}]$ (bottom-left corner in $(\underline{x}, \underline{y})$, top-right corner in $(\overline{x}, \overline{y})$) that contain no (part of) item. In Figure 1.2(a), we have holes $[4, 3, 8, 6]$ and $[3, 5, 8, 6]$.

2 Consider the pairs of holes $([\underline{x}, \underline{y}, \overline{x}, \overline{y}], [\underline{x}', \underline{y}', \overline{x}', \overline{y}'])$ having some intersection, and determine the corresponding potential new hole $[\min(\underline{x}, \underline{x}'), \min(\underline{y}, \underline{y}'), \max(\overline{x}, \overline{x}'), \max(\overline{y}, \overline{y}')]$: choose the pair for which the potential hole has the maximum area, and remove all items overlapping it. In Figure 1.2(a), the only choice is hole $[3, 3, 8, 6]$, so item 4 is removed.

3 Remove from the packing those items j that are placed at height $y_j > \alpha z(\overline{S}_{2\mathrm{SP}})$ (α a prefixed parameter). By assuming $\alpha = 0.7$, in Figure 1.2(a), we remove items 9 and 10.

4 Shift down and left the remaining items (see above). In Figure 1.2(b), items 7 and 8 are shifted.

5 Consider the removed items by non-increasing y_j value, breaking ties by non-increasing item area: pack the current item in the lowest feasible position, left justified. In Figure 1.2(c), items 10, 9 and 4 (in this order) are re-packed.

The time complexity of the post-optimization algorithm is $O(n^2)$. Indeed, Step 1 requires $O(n^2)$ time, by considering, for each item j, all items surrounding it, in order to detect holes touching j. As there are at most $O(n)$ holes, Step 2 too requires $O(n^2)$ time. Step 3 clearly needs $O(n)$ time, while Step 4 requires $O(n^2)$ time, as previously shown. Finally, Step 5 can be implemented so as to require $O(n^2)$ time, as shown by Chazelle, 1983.

We have so far illustrated the core, EXPLORE, of our approach. The overall algorithm, 2SP_TS, performs a search over values of bin height H to be used within the 2BP Tabu search. At each iteration, the range $[H_1, H_2]$ of H values to be tested is halved:

```
algorithm 2SP_TS:
  execute algorithms BUILD and TP$_{2SP}$ of Section 1;
  let $S_{2SP}$ be the best solution found, and $z^*$ its value;
  compute a lower bound $L$ by solving the 1CBP relaxation;
  if $z^* = L$ then stop;
  $H_1 := \max_j\{h_j\}$, $H_2 := z^* - 1$;
  $z_1 :=$ EXPLORE($S_{2SP}$,$H_1$);
  $z_2 :=$ EXPLORE($S_{2SP}$,$H_2$);
  while $H_1 < H_2 - 1$ do
      $H := \lfloor (H_1 + H_2)/2 \rfloor$;
      $z :=$ EXPLORE($S_{2SP}$,$H$);
      $z^* := \min(z^*, z)$;
      if $z^* = L$ then stop;
      if $z_1 \le z_2$ then $z_2 := z$, $H_2 := H$;
      else $z_1 := z$, $H_1 := H$;
  end while;
end
```

3. Genetic algorithm

We developed a genetic algorithm, 2SP_GA, based on a permutation data structure representing the order in which the items are packed into the strip. The population size is constant and new individuals are obtained through elitist criteria and reproduction. Diversification is obtained through immigration and mutation. The research is intensified through local search. The history of the evolution is considered, in order to intensify the search in promising areas, and to escape from poor local minima.

Most of our parameters are evaluated according to the recent evolution of the algorithm. Namely, we distinguish between *improving* and *non-improving phase*: we are in a non-improving phase if the incumbent solution was not improved by the latest Q individuals generated. Computational experiments showed that a good value for distinguishing the two phases is $Q = 750$.

3.1 Data structure

Genotypes

The genotype of an individual is represented by a permutation $\Pi = (\pi_1, \ldots, \pi_n)$ of the items, that gives the order in which the items are packed. This data structure, adopted, for various packing problems, by several authors (see, e.g., Reeves, 1996, Jakobs, 1996, Gomez and de la Fuente, 2000), has the advantage of an easy creation of new permutations and the absence of infeasible solutions. In our approach, the current sequence is given as input to algorithm $\mathrm{TP}_{\mathrm{2SP}}$ of Section 1, that produces the corresponding packing.

Fitness

An individual is evaluated through a fitness function $f : \Pi \to R$, computed as follows. Let v^* denote the incumbent solution value, v_0 the worse solution value in the current population, and $v(\Pi)$ the solution value produced by $\mathrm{TP}_{\mathrm{2SP}}$ for Π. The fitness of Π is then:

$$f(\Pi) = \begin{cases} \max(0,\ \beta' v^* - v(\Pi)) & \text{in an improving phase} \\ \max(0,\ \beta''(v^* + v_0)/2 - v(\Pi)) & \text{in a non-improving phase} \end{cases}$$

with $\beta', \beta'' > 1$. In this way, given two individuals, Π^a and Π^b, the percentage difference between their fitness is higher if we are in an improving phase. Hence, in an improving phase there is a high probability of selecting promising individuals, while in a non-improving phase there is a high probability of escaping from a local minimum. Good values for the parameters were experimentally established as $\beta' = 1.2$ and $\beta'' = 1.5$.

3.2 Evolution process

Population

A constant population size s is adopted. Initially, a first population of s random individuals is created, by ensuring that it includes no "duplicated" individuals. An individual is said to be *duplicated* if his item sequence can be obtained from the sequence of a different individual just by swapping identical items, i.e., distinct items having the same width and height. Sizes in the range $[50, 100]$ computationally proved good efficiency, while higher values produced a strong increase in the computing time needed by the algorithm. Hence, we adopted for s the value

$s = \max(50, \min(n, 100))$.

Evolution

An elitist model (see De Jong, 1975) is adopted, i.e., the e individuals with highest fitness in the current generation directly pass to the new generation, with $e = s/5$ for an improving phase and $e = 1$ for a non-improving phase. An additional individual is added through intensification on the current generation (see below). The new population is then completed through immigration and generation.

The number of immigrants is $i = s/20$ for an improving phase and $i = s/10$ for a non-improving phase. Immigrants are created by ensuring that their initial triplets (π_1, π_2, π_3) are the less frequent among those of all previously generated individuals.

The remaining $s-e-i-1$ individuals are generated through exchange of information between two parents. The parents are selected following the classical *proportional selection* approach (see Holland, 1975), i.e., an individual having fitness $f(\Pi^k)$ has a probability $f(\Pi^k)/\sum_{j=1}^{s} f(\Pi^j)$ of being selected. In addition, we prohibit mating between duplicated parents.

Finally, during non-improving phases, we apply mutation by randomly selecting 10% of the individuals of the new generation: for each selected chromosome, two random elements of the permutation are interchanged.

Crossover

The exchange of information between the two parents is obtained through crossover operator OX3 by Davis, 1991. Let Π^a and Π^b be the selected parents. We generate two random numbers p, q $(p < q)$ in the interval $[1, n]$. The offspring is composed by two individuals, Π^c and Π^d, generated as follows. The crossover copies $\pi_p^a, \ldots, \pi_q^a$ to the same positions in Π^c, and $\pi_p^b, \ldots, \pi_q^b$ to the same positions in Π^d. Π^c and Π^d are then filled up by selecting the missing elements from Π^b and Π^a, respectively, in the same order. For example if $\Pi^a = (2,1,3,7,6,4,5)$, $\Pi^b = (4,3,1,6,2,7,5)$, $p = 3$ and $q = 4$, we obtain $\Pi^c = (4,1,3,7,6,2,5)$ and $\Pi^d = (2,3,1,6,7,4,5)$. The generated individual having the highest fitness is then selected for the new population.

This crossover was successfully compared, through computational experiments, with other crossovers proposed in the literature for similar problems, namely crossovers C1 (*Crossover One*, see Reeves, 1993), PMX (*Partially Mapped Crossover*, see Goldberg and Lingle, 1985), Jakobs (see Jakobs, 1996), OX (*Order Crossover*, see Davis, 1985), UX2 (*Union Crossover 2*, see Poon and Carter, 1995).

3.3 Intensification

Local search is employed in order to more deeply explore promising solution regions.

The following algorithm iteratively operates on a given solution S in two phases: a first attempt to re-pack the item that is most high in S, and a second attempt to rearrange the packing of items placed around the holes (see Section 2 for the definition of hole). The adopted approach, RE-PACK, can be outlined as follows.

```
1 Select as a target the item $t$ that is packed most high in solution
  $S$, i.e., the one for which $y_t + h_t$ is a maximum in $S$.

2 Execute the following variant of algorithm $\mathrm{TP}_{\mathrm{2SP}}$ of Section 1.
  While item $t$ is not packed, at each iteration, evaluate the position
  with highest touching fraction not only for the current item but
  also for $t$, and select for packing the item and the position produc-
  ing the maximum touching fraction. Once item $t$ has been packed,
  the execution proceeds as in $\mathrm{TP}_{\mathrm{2SP}}$. Let $S'$ denote the resulting
  solution.

3 Store in $S$ the best solution between $S$ and $S'$.

4 for each hole in the packing pattern corresponding to $S$ do
      consider the items that touch the hole's sides;
      for each pair of such items do
          swap the items in the permutation;
          execute algorithm $\mathrm{TP}_{\mathrm{2SP}}$;
      end for
  end for

5 Let $S'$ denote the best solution obtained at step 4.
  If $z(S') < z(S)$ store $S'$ in $S$ and go to step 1; otherwise terminate.
```

Recall that a different re-packing approach, still operating on holes, was used in the post-optimization phase of the Tabu search approach (see algorithm POST-OPT of Section 2). This approach too turned out to be useful for the genetic algorithm, and the best results were obtained by alternating the two processes as follows. Assume that the generations are numbered by integers $1, 2, \ldots$. At each odd (resp. even) generation, the individual with largest fitness not previously used for intensification (if any) is selected, and the corresponding solution S is explored through RE-PACK (resp. POST-OPT). As previously mentioned, the best individual obtained by intensification is directly added to the next generation.

4. A Hybrid approach

We performed extensive computational experiments both on instances from the literature and randomly generated instances (see Section 5). Both the Tabu search approach 2SP_TS (see Section 2) and the genetic algorithm 2SP_GA (see Section 3) were executed after the computation of upper and lower bounds through the deterministic approaches discussed in Section 1. The experiments showed a good empirical behavior of both metaheuristics. It turned out in particular that the genetic algorithm has a better performance for small size instances and for cases with a small value of the strip size W, while the opposite holds for the latter. Indeed, the CPU time requested by the iterated execution of $\mathrm{TP}_{2\mathrm{SP}}$ within the genetic algorithm strongly increases with such values. In addition, large values of n imply large populations, so a lesser number of them can be generated within a given time limit.

We thus implemented a third approach, based on a combination of the two algorithms. This hybrid algorithm too starts with the computation of deterministic upper and lower bounds. Let T_{GA} and T_{TS} be prefixed time limits assigned, respectively, to 2SP_GA and 2SP_TS. Algorithm 2SP_GA is first executed, for T_{GA} time units, exactly as described in Section 3. The time check is performed at the beginning of each new generation. If the assigned time has expired, the current generation is obtained as follows. After the selection of the first $e+1$ individuals (see Section 3.2), the solution corresponding to the individual obtained from intensification is given as initial solution to 2SP_TS, and the Tabu search is performed for T_{TS} time units. When this limit has been reached, a new individual, associated with the best solution found by 2SP_TS, is added to the current population (and the incumbent solution is possibly updated by such solution). The population is then completed through i immigrants and $s-e-i-2$ generated individuals. Algorithm 2SP_GA is then executed for T_{GA} time units, and so on, until an overall time limit is reached.

The computational experiments showed that the performance of the hybrid algorithms is quite sensitive to values T_{GA} and T_{TS}. By also considering the outcome of the experiments on the two approaches alone (see above), a reasonable definition of such values was determined as follows. Let T be the time limit assigned to each iteration (2SP_GA plus 2SP_TS) of the hybrid algorithm. Then $T_{\mathrm{GA}} = \gamma T$ and $T_{\mathrm{TS}} = (1-\gamma)T$, with

$$\gamma = \begin{cases} 4/5 & \text{if } \min(n, W) \le 20 \\ 1/5 & \text{if } n \ge 80 \text{ and } W > 20 \\ 1 - n/100 & \text{otherwise} \end{cases} \tag{1.1}$$

where the third value corresponds to the segment that interpolates the two extremes.

5. Computational Experiments

The algorithms of the previous sections were coded in FORTRAN 77 and run on a Pentium III 800 MHz both on test instances from the literature and on randomly generated instances. All considered instances have $n \leq 100$ and $W \leq 500$.

Table 1.1 refers to original instances of the strip packing problem. In particular, instances `jack01`–`jack02` were proposed by Jakobs, 1996, instances `ht01`–`ht18` by Hopper and Turton, 1999, and instances `hifi01`–`hifi25` by Hifi, 1999. The instances considered in Table 1.2, originally introduced for other two-dimensional cutting problems, were transformed into strip packing instances by using the item sizes and bin width. In particular, instances `cgcut01`–`cgcut03` were proposed by Christofides and Whitlock, 1977, instances `beng01`–`beng07` by Bengtsson, 1982, while instances `ngcut01`–`ngcut12` and `gcut01`–`gcut08` by Beasley, 1985a, and Beasley, 1985b. Instances `jack01`–`jack02` are described in Jakobs, 1996, instances `hifi01`–`hifi25` are available on the web at `ftp://panoramix.univ-paris1.fr/pub/CERMSEM/hifi/SCP`, while all other instances can be found in the ORLIB library (see Beasley, 1990, web site `http://www.ms.ic.ac.uk/info.html`).

The initialization phase had a total time limit of 45 CPU seconds (15 seconds for the lower bound, and 30 seconds for the upper bound computation). Each metaheuristic algorithm had a time limit of 15 seconds for the lower bound computation plus 300 seconds for the exploration of the solution space (30 of which, at most, for the initial upper bound computation). The check on the elapsed CPU time was executed after each heuristic in the initialization phase, at each execution of EXPLORE in the Tabu search (and hybrid) algorithm, and at each population in the genetic (and hybrid) algorithm. As a consequence, the times in the tables can exceed the given time limit. For each instance, Tables 1.1 and 1.2 give:

- problem name and values of n and W;
- value of the lower bound (LB);
- best upper bound value (UB) found by the heuristic algorithms of Section 1, with corresponding percentage gap (*%gap*, computed as $(UB - LB)/UB$) and elapsed CPU time (T);
- percentage gap obtained by the genetic algorithm (resp. Tabu search, resp. hybrid algorithm) and corresponding CPU time (*time*)

Table 1.1. Strip packing instances from the literature. CPU seconds of a Pentium III 800 MHz.

Instance			Initial bounds				2SP_GA		2SP_TS		hybrid	
Name	n	W	LB	UB	%*gap*	T	%*gap*	*time*	%*gap*	*time*	%*gap*	*time*
`jack01`	25	40	15	16	6.25	16.58	0.00	132.73	6.25	16.59	0.00	93.93
`jack02`	50	40	15	16	6.25	16.25	6.25	16.26	6.25	16.24	6.25	16.25
`ht01`	16	20	20	22	9.09	16.67	0.00	21.88	9.09	16.67	0.00	21.88
`ht02`	17	20	20	23	13.04	16.66	4.76	27.03	9.09	116.11	4.76	27.09
`ht03`	16	20	20	21	4.76	16.66	0.00	17.46	4.76	16.66	0.00	17.45
`ht04`	25	40	15	16	6.25	16.67	0.00	28.74	0.00	116.12	0.00	28.45
`ht05`	25	40	15	16	6.25	3.05	6.25	3.05	6.25	3.05	6.25	3.05
`ht06`	25	40	15	16	6.25	13.82	0.00	13.82	0.00	13.82	0.00	13.82
`ht07`	28	60	30	31	3.23	16.68	3.23	16.68	3.23	16.68	3.23	16.68
`ht08`	29	60	30	33	9.09	16.69	6.25	17.77	9.09	16.69	3.23	88.43
`ht09`	28	60	30	33	9.09	16.68	0.00	37.50	9.09	16.68	0.00	37.11
`ht10`	49	60	60	64	6.25	16.77	6.25	16.77	6.25	16.77	6.25	16.77
`ht11`	49	60	60	65	7.69	16.78	6.25	37.23	4.76	315.03	4.76	265.80
`ht12`	49	60	60	63	4.76	16.78	4.76	16.78	3.23	91.34	3.23	36.02
`ht13`	73	60	90	95	5.26	17.06	5.26	17.06	5.26	17.06	4.26	30.40
`ht14`	73	60	90	93	3.23	17.07	3.23	17.07	3.23	17.07	3.23	17.07
`ht15`	73	60	90	96	6.25	17.15	5.26	145.94	6.25	17.15	4.26	31.21
`ht16`	97	80	120	126	4.76	17.87	4.76	17.87	4.76	17.87	4.76	17.87
`ht17`	97	80	120	124	3.23	17.59	3.23	17.59	3.23	17.59	3.23	17.59
`ht18`	97	80	120	125	4.00	17.42	4.00	17.42	3.23	125.02	4.00	17.42
`hifi01`	10	5	13	13	0.00	0.26	0.00	0.27	0.00	0.26	0.00	0.26
`hifi02`	11	4	40	40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`hifi03`	15	6	14	14	0.00	15.00	0.00	15.00	0.00	15.01	0.00	15.01
`hifi04`	11	6	19	20	5.00	15.08	5.00	15.09	5.00	15.09	5.00	15.08
`hifi05`	8	20	20	20	0.00	0.56	0.00	0.58	0.00	0.56	0.00	0.56
`hifi06`	7	30	38	38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`hifi07`	8	15	14	14	0.00	0.09	0.00	0.10	0.00	0.09	0.00	0.09
`hifi08`	12	15	17	17	0.00	2.01	0.00	2.07	0.00	1.98	0.00	1.99
`hifi09`	12	27	68	68	0.00	1.66	0.00	1.66	0.00	1.66	0.00	1.67
`hifi10`	8	50	80	80	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00
`hifi11`	10	27	48	48	0.00	0.16	0.00	0.17	0.00	0.16	0.00	0.17
`hifi12`	18	81	34	34	0.00	0.63	0.00	0.65	0.00	0.62	0.00	0.61
`hifi13`	7	70	50	50	0.00	0.20	0.00	0.21	0.00	0.19	0.00	0.20
`hifi14`	10	100	60	70	14.29	15.17	13.04	20.76	14.29	15.17	13.04	20.75
`hifi15`	14	45	34	38	10.53	15.13	0.00	22.54	10.53	15.10	0.00	22.52
`hifi16`	14	6	32	33	3.03	15.09	3.03	15.08	3.03	15.08	3.03	15.08
`hifi17`	9	42	34	34	0.00	0.43	0.00	0.43	0.00	0.43	0.00	0.42
`hifi18`	10	70	89	101	11.88	15.09	11.88	15.11	11.88	15.10	11.88	15.10
`hifi19`	12	5	25	25	0.00	2.75	0.00	2.84	0.00	2.73	0.00	2.72
`hifi20`	10	15	19	20	5.00	15.17	5.00	15.14	5.00	15.17	5.00	15.16
`hifi21`	11	30	140	140	0.00	12.33	0.00	12.64	0.00	12.33	0.00	12.33
`hifi22`	22	90	34	38	10.53	15.08	0.00	30.78	10.53	15.08	0.00	30.57
`hifi23`	12	15	34	34	0.00	4.19	0.00	4.27	0.00	4.15	0.00	4.16
`hifi24`	10	50	103	111	7.21	15.60	6.36	30.39	7.21	15.61	6.36	30.32
`hifi25`	15	25	35	39	10.26	4.97	0.00	7.07	5.41	108.68	0.00	6.91

Table 1.2. Adapted packing instances from the literature. CPU seconds of a Pentium III 800 MHz.

Instance			Initial bounds				2SP_GA		2SP_TS		hybrid	
Name	n	W	LB	UB	$\%gap$	T	$\%gap$	$time$	$\%gap$	$time$	$\%gap$	$time$
cgcut01	16	10	23	23	0.00	1.19	0.00	1.19	0.00	1.19	0.00	1.19
cgcut02	23	70	63	69	8.70	15.68	3.08	293.33	8.70	15.67	3.08	61.05
cgcut03	62	70	636	676	5.92	16.47	5.92	16.48	5.92	16.48	5.92	16.48
beng01	20	25	30	31	3.23	15.77	3.23	15.76	3.23	15.76	3.23	15.76
beng02	40	25	57	59	3.39	16.43	1.72	231.61	3.39	16.43	1.72	38.56
beng03	60	25	84	87	3.45	16.43	2.33	94.71	3.45	16.43	2.33	32.76
beng04	80	25	107	112	4.46	15.10	2.73	36.55	2.73	17.03	2.73	23.72
beng05	100	25	134	137	2.19	15.09	2.19	15.09	1.47	77.03	1.47	28.76
beng06	40	40	36	38	5.26	15.42	0.00	308.24	5.26	15.41	2.70	92.27
beng07	80	40	67	71	5.63	15.09	4.29	53.89	2.90	17.04	2.90	23.23
ngcut01	10	10	23	23	0.00	0.28	0.00	0.28	0.00	0.28	0.00	0.28
ngcut02	17	10	30	30	0.00	6.66	0.00	6.66	0.00	6.66	0.00	6.62
ngcut03	21	10	28	29	3.45	16.66	0.00	21.30	3.45	16.66	0.00	21.28
ngcut04	7	10	20	20	0.00	0.02	0.00	0.02	0.00	0.01	0.00	0.02
ngcut05	14	10	36	36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ngcut06	15	10	29	31	6.45	16.65	6.45	16.65	6.45	16.66	6.45	16.65
ngcut07	8	20	20	20	0.00	0.82	0.00	0.82	0.00	0.82	0.00	0.82
ngcut08	13	20	32	33	3.03	16.66	3.03	16.66	3.03	16.66	3.03	16.66
ngcut09	18	20	49	55	10.91	16.68	2.00	20.14	9.26	16.75	2.00	20.12
ngcut10	13	30	80	80	0.00	1.07	0.00	0.99	0.00	1.02	0.00	1.08
ngcut11	15	30	50	54	7.41	16.66	3.85	121.33	7.41	16.66	3.85	157.99
ngcut12	22	30	87	87	0.00	0.25	0.00	0.26	0.00	0.25	0.00	0.25
gcut01	10	250	1016	1016	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
gcut02	20	250	1133	1245	9.00	26.08	7.36	222.58	6.44	173.57	6.13	251.28
gcut03	30	250	1803	1803	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gcut04	50	250	2934	3130	6.26	45.09	6.26	45.09	6.26	45.09	6.26	45.09
gcut05	10	500	1172	1284	8.72	45.77	8.72	45.77	8.72	45.77	7.93	128.29
gcut06	20	500	2514	2757	8.81	45.84	8.52	318.09	7.37	79.48	6.02	246.65
gcut07	30	500	4641	4796	3.23	51.50	3.23	51.50	3.23	51.50	2.46	267.74
gcut08	50	500	5703	6257	8.85	54.74	8.85	54.74	7.97	83.69	8.61	377.23

elapsed when the last improvement of the incumbent solution occurred.

In Tables 1.3 and 1.4 we give the outcome of other computational tests, performed on ten classes of randomly generated instances originally proposed in the literature for 2BP. In this case too the instances were adapted to 2SP, by disregarding the bin heights.

The first four classes were proposed by Martello and Vigo, 1998, and are based on the generation of items of four different types:

type 1 : w_j uniformly random in $[\frac{2}{3}W, W]$,
h_j uniformly random in $[1, \frac{1}{2}W]$;

type 2 : w_j uniformly random in $[1, \frac{1}{2}W]$,
h_j uniformly random in $[\frac{2}{3}W, W]$;

type 3 : w_j uniformly random in $[\frac{1}{2}W, W]$,
h_j uniformly random in $[\frac{1}{2}W, W]$;

type 4 : w_j uniformly random in $[1, \frac{1}{2}W]$,
h_j uniformly random in $[1, \frac{1}{2}W]$.

Class k ($k \in \{1, 2, 3, 4\}$) is then obtained by generating an item of type k with probability 70%, and of the remaining types with probability 10% each. The strip width is always $W = 100$.
The next six classes have been proposed by Berkey and Wang, 1987:

Class 5 : $W = 10$, w_j and h_j uniformly random in $[1, 10]$;

Class 6 : $W = 30$, w_j and h_j uniformly random in $[1, 10]$;

Class 7 : $W = 40$, w_j and h_j uniformly random in $[1, 35]$;

Class 8 : $W = 100$, w_j and h_j uniformly random in $[1, 35]$;

Class 9 : $W = 100$, w_j and h_j uniformly random in $[1, 100]$;

Class 10 : $W = 300$, w_j and h_j uniformly random in $[1, 100]$.

For each class and value of n ($n \in \{20, 40, 60, 80, 100\}$), ten instances were generated. The time limits per instance were the same as in the previous tables. All these instances (and the corresponding generator code) are available at `http://www.or.deis.unibo.it/ORinstances/2BP/`.
For each class and value of n, Tables 1.3 and 1.4 give:

- class and values of n and W;
- average lower bound value (LB);

Table 1.3. Random instances proposed by Martello and Vigo. CPU seconds of a Pentium III 800 MHz. Average values over 10 instances.

Instance			Initial bounds				2SP_GA		2SP_TS		hybrid	
Class	n	W	LB	UB	$\%gap$	T	$\%gap$	$time$	$\%gap$	$time$	$\%gap$	$time$
	20	10	60.3	61.3	1.46	10.26	1.33	11.02	1.33	37.17	1.33	10.82
	40	10	121.6	122.5	0.85	10.37	0.20	22.36	0.67	36.54	0.30	17.65
1	60	10	187.4	189.2	0.97	12.59	0.86	34.45	0.97	12.63	0.86	23.94
	80	10	262.2	263.0	0.32	7.38	0.23	7.84	0.28	13.42	0.23	8.16
	100	10	304.4	305.7	0.44	15.49	0.41	15.89	0.37	24.52	0.37	18.78
av.			187.2	188.3	0.81	11.22	0.61	18.31	0.74	24.86	0.62	15.87
	20	30	19.7	21.0	6.13	16.61	1.52	23.43	4.32	26.39	0.99	41.37
	40	30	39.1	41.4	5.73	18.40	2.00	72.68	3.48	40.29	2.19	51.86
2	60	30	60.1	62.3	3.53	18.52	2.70	24.03	2.77	25.82	2.44	35.33
	80	30	83.2	85.3	2.43	17.20	2.33	17.20	1.63	64.92	1.76	29.34
	100	30	100.5	102.2	1.68	18.22	1.58	18.22	1.36	24.04	1.27	19.77
av.			64.5	66.6	5.13	21.11	2.33	35.80	3.58	41.57	1.93	43.81
	20	40	157.4	167.9	6.23	20.09	4.81	79.11	5.04	64.92	4.42	94.23
	40	40	328.8	341.3	3.79	16.88	2.69	81.22	3.34	54.62	3.08	81.28
3	60	40	500.0	518.0	3.68	18.94	3.09	118.53	3.31	92.32	3.48	72.91
	80	40	701.7	721.1	2.73	19.10	2.19	106.19	2.31	71.13	2.49	50.02
	100	40	832.7	848.8	1.98	18.82	1.90	48.06	1.77	95.38	1.94	36.16
av.			504.1	519.4	3.68	18.77	2.93	86.62	3.15	75.67	3.08	66.92
	20	100	61.4	70.2	12.45	18.41	6.76	66.47	8.29	35.48	6.33	78.22
	40	100	123.9	137.9	10.38	18.58	5.54	141.50	5.93	60.32	5.79	59.83
4	60	100	193.0	208.1	7.26	18.89	6.12	87.33	4.60	36.12	4.53	60.31
	80	100	267.2	284.9	6.21	19.41	6.14	37.15	4.09	46.25	4.19	27.77
	100	100	322.0	342.2	5.91	20.33	5.89	33.80	3.08	81.87	3.14	57.69
av.			193.5	208.7	8.44	19.12	6.09	73.25	5.20	52.01	4.80	56.76

Table 1.4. Random instances proposed by Berkey and Wang. CPU seconds of a Pentium III 800 MHz. Average values over 10 instances.

Instance			Initial bounds				2SP_GA		2SP_TS		hybrid	
Class	*n*	*W*	*LB*	*UB*	*%gap*	*T*	*%gap*	*time*	*%gap*	*time*	*%gap*	*time*
	20	100	512.2	549.5	7.18	19.54	4.48	103.15	5.37	67.58	4.51	72.87
	40	100	1053.8	1096.5	4.16	19.61	3.37	92.97	3.60	88.77	2.95	117.96
5	60	100	1614.0	1675.1	3.83	21.86	3.54	43.06	3.23	66.73	3.32	41.46
	80	100	2268.4	2311.5	1.88	24.93	1.81	57.51	1.57	73.17	1.73	81.70
	100	100	2617.4	2699.6	3.17	27.43	3.00	33.28	2.91	97.65	3.07	56.63
av.			1613.2	1666.4	4.04	22.67	3.24	65.99	3.33	78.78	3.12	74.12
	20	300	159.9	188.0	15.00	19.98	9.15	148.47	10.46	51.68	8.56	145.96
	40	300	323.5	366.0	11.59	24.93	8.26	155.94	7.04	88.96	6.52	68.91
6	60	300	505.1	554.8	8.95	29.33	8.17	147.78	4.86	75.82	5.04	65.08
	80	300	699.7	757.3	7.59	36.83	7.44	63.27	4.43	45.11	4.43	63.00
	100	300	843.8	910.7	7.39	42.98	7.30	60.92	3.55	55.32	3.57	85.65
av.			506.4	555.4	10.10	30.81	8.06	115.28	6.07	63.38	5.62	85.72
	20	100	490.4	501.9	2.45	11.19	2.45	11.21	2.45	11.21	2.45	11.21
	40	100	1049.7	1063.1	1.34	12.18	1.00	36.41	1.18	41.24	1.03	28.38
7	60	100	1515.9	1530.4	0.94	9.18	0.89	10.76	0.90	22.01	0.89	10.18
	80	100	2206.1	2230.0	1.10	13.40	0.96	44.32	1.03	41.82	0.82	72.69
	100	100	2627.0	2651.3	0.90	15.93	0.78	80.36	0.78	55.68	0.73	85.08
av.			1577.8	1595.3	1.35	12.38	1.22	36.61	1.27	34.39	1.18	41.51
	20	100	434.6	480.5	9.52	19.34	7.06	194.66	8.67	43.62	7.66	128.18
	40	100	922.0	994.8	7.42	20.91	6.56	108.80	6.06	112.37	5.85	160.03
8	60	100	1360.9	1442.2	5.63	22.97	5.53	87.81	5.23	99.89	5.27	48.12
	80	100	1909.3	2019.8	5.46	26.11	5.46	26.11	4.88	85.97	5.23	57.77
	100	100	2362.8	2485.8	4.94	29.41	4.94	29.41	4.62	116.37	4.85	35.23
av.			1397.9	1484.6	6.59	23.75	5.91	89.36	5.89	91.64	5.77	85.87
	20	100	1106.8	1106.8	0.00	1.11	0.00	1.11	0.00	1.11	0.00	1.11
	40	100	2189.2	2190.6	0.07	1.88	0.07	1.88	0.07	1.88	0.07	1.88
9	60	100	3410.4	3410.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	80	100	4578.6	4588.1	0.20	6.06	0.20	6.06	0.20	6.06	0.20	6.06
	100	100	5430.5	5434.9	0.08	5.04	0.08	5.04	0.08	5.04	0.08	5.04
av.			3343.1	3346.2	0.07	2.82	0.07	2.82	0.07	2.82	0.07	2.82
	20	100	337.8	359.2	6.45	18.63	4.88	82.07	5.35	65.45	4.88	99.29
	40	100	642.8	685.4	6.23	19.78	4.80	135.25	5.13	142.45	4.65	135.39
10	60	100	911.1	960.3	5.16	21.68	4.94	29.73	4.45	128.92	4.48	100.37
	80	100	1177.6	1236.1	4.70	21.91	4.69	38.80	4.23	93.30	4.61	76.07
	100	100	1476.5	1542.3	4.29	24.60	4.29	24.60	3.97	134.26	4.29	24.60
av.			909.2	956.7	5.37	21.32	4.72	62.09	4.63	112.88	4.58	87.14

- average upper bound value (UB) found by the heuristic algorithms of Section 1, with corresponding average percentage gap ($\%gap$) and elapsed CPU time (T);
- average percentage gap obtained by the genetic algorithm (resp. Tabu search, resp. hybrid algorithm) and corresponding average CPU time (*time*) elapsed when the last improvement of the incumbent solution occurred.

In addition, for each class we give the same average values over the 50 instances. The tables show a satisfactory performance of the three metaheuristic approaches. Algorithms 2SP_TS and 2SP_GA have a "complementary" behavior: 2SP_GA performs better in the case of small size instances (both for what concerns number of items and strip width), while the opposite holds for 2SP_TS. The hybrid algorithm was thus structured so as to take advantage of this phenomenon (see equation (1.1)).

The performance of the hybrid algorithm is generally (although not always) the best one: for the instances from the literature, it produced the best solution in 72 cases out of 75. Half of these instances were solved to proved optimality (i.e., by determining a solution of value equal to the lower bound). The gap obtained by the hybrid algorithm was below 5% in 62 out of 75 instances. By considering the instances not solved to proved optimality during the initialization phase, it improved the initial solution in 32 instances out of 49.

Concerning the random instances of Tables 1.3 and 1.4, the average percentage gap of the hybrid algorithm was below 5% in 41 cases out of 50, and below 6% in 46 cases. It produced the best average percentage gap in 26 cases out of 50, and the best overall average percentage gap in 8 classes out of 10. With the exception of Class 9, for which most of the instances were optimally solved by the initialization phase, the hybrid algorithm improved the average gap in almost all cases (43 out of 45).

Acknowledgments

We thank the Ministero dell'Università e della Ricerca Scientifica e Tecnologica (M.U.R.S.T.) and the Consiglio Nazionale delle Ricerche (C.N.R.), Italy, for the support given to this project. The computational experiments have been executed at the Laboratory of Operations Research of the University of Bologna (Lab.O.R.).

References

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

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

Baker, D. S. and Schwarz, J. S. (1983). Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12:508–525.

Beasley, J. E. (1985a). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36:297–306.

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

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Bengtsson, B. E. (1982). Packing rectangular pieces – a heuristic approach. *The Computer Journal*, 25:353–357.

Berkey, J. O. and Wang, P. Y. (1987). Two dimensional finite bin packing algorithms. *Journal of the Operational Research Society*, 38:423–429.

Brown, D. J. (1980). An improved BL lower bound. *Information Processing Letters*, 11:37–39.

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

Christofides, N. and Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations Research*, 25:30–44.

Coffman, Jr., E. G., Garey, M. R., Johnson, D. S., and Tarjan, R. E. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:801–826.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 162–64. Morgan Kaufmann.

Davis, L., editor (1991). *A Handbbok of Genetic Algorithms.* Van Nostrand Reinhold.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan.

Dyckhoff, H., Scheithauer, G., and Terno, J. (1997). Cutting and Packing (C&P). In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons, Chichester.

Fekete, S. and Schepers, J. (1997a). On more-dimensional packing I: Modeling. Technical Report ZPR97-288, Mathematisches Institut, Universität zu Köln.

Fekete, S. and Schepers, J. (1997b). On more-dimensional packing II: Bounds. Technical Report ZPR97-289, Mathematisches Institut, Universität zu Köln.

Fekete, S. and Schepers, J. (1997c). On more-dimensional packing III: Exact algorithms. Technical Report ZPR97-290, Mathematisches Institut, Universität zu Köln.

Golan, I. (1981). Performance bounds for orthogonal oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 10:571–582.

Goldberg, D. E. and Lingle, R. L. (1985). Alleles, loci and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pages 154–159. Erlbaum.

Gomez, A. and de la Fuente, D. (2000). Resolution of strip-packing problems with genetic algorithms. *Journal of the Operational Research Society*, 51:1289–1295.

Hifi, M. (1999). The strip cutting/packing problem: incremental substrip algorithms-based heuristics. *Pesquisa Operacional, Special Issue on Cutting and Packing Problems*, 19(2):169–188.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Michigan Press.

Hopper, E. and Turton, B. (1999). An empirical investigation of metaheuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*. (to appear).

Høyland, S. (1988). Bin-packing in 1.5 dimension. In Karlsson, R. and Lingas, A., editors, *Lecture Notes in Computer Science*, pages 129–137. Springer-Verlag, Berlin.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88:165–181.

Lodi, A., Martello, S., and Monaci, M. (2001). Two-dimensional packing problems: a survey. *European Journal of Operational Research*. (to appear).

Lodi, A., Martello, S., and Vigo, D. (1998). Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In Voss, S., Martello, S., Osman, I., and Roucairol, C., editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 125–139. Kluwer Academic Publishers, Boston.

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

Lodi, A., Martello, S., and Vigo, D. (1999b). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11:345–357.

Lodi, A., Martello, S., and Vigo, D. (1999c). Recent advances on two-dimensional bin packing problems. Technical Report OR/99/2, DEIS - Università di Bologna.

Martello, S., Monaci, M., and Vigo, D. (2000). An exact approach to the strip packing problem. Technical Report OR/00/4, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna.

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

Poon, P. W. and Carter, J. N. (1995). Genetic algorithm crossover operators for ordering applications. *Computers and Operations Research*, 22(1):135–147.

Reeves, C. (1996). Hybrid genetic algorithms for the bin-packing and related problems. *Annals of Operations Research*, 63:371–396.

Reeves, C. R., editor (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications.

Sleator, D. (1980). A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10:37–40.

Steinberg, A. (1997). A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26:401–409.