INFORMS Journal on Computing

A Set-Covering-Based Heuristic Approach for Bin-Packing Problems

Michele Monaci, Paolo Toth,

INFORMS Journal on Computing
Vol. 18, No. 1, Winter 2006, pp. 71–85
ISSN 1091-9856 | EISSN 1526-5528 | 06 | 1801 | 0071

DOI 10.1287/ijoc.1040.0089

A Set-Covering-Based Heuristic Approach for Bin-Packing Problems

Michele Monaci
Dipartimento di Ingegneria dell'Informazione, University of Padova, Via Gradenigo 6/A, 35131 Padova, Italy, monaci@dei.unipd.it

Paolo Toth
Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, ptoth@deis.unibo.it

Several combinatorial optimization problems can be formulated as large set-covering problems. In this work, we use the set-covering formulation to obtain a general heuristic algorithm for this type of problem, and describe our implementation of the algorithm for solving two variants of the well-known (one-dimensional) bin-packing problem: the two-constraint bin-packing problem and the basic version of the two-dimensional bin-packing problem, where the objects cannot be rotated and no additional requirements are imposed. In our approach, both the "column-generation" and the "column-optimization" phases are heuristically performed. In particular, in the first phase, we do not generate the entire set of columns, but only a small subset of it, by using greedy procedures and fast constructive heuristic algorithms from the literature. In the second phase, we solve the associated set-covering instance by means of a Lagrangian-based heuristic algorithm. Extensive computational results on test instances from the literature show that, for the two considered problems, this approach is competitive, with respect to both the quality of the solution and the computing time, with the best heuristic and metaheuristic algorithms proposed so far.

Key words: two-constraint bin-packing; two-dimensional bin-packing; column-generation; set-covering; heuristics
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received June 2003; revised March 2004; accepted March 2004.

1. Introduction

Several NP-hard combinatorial optimization problems can be formulated as large set-covering (or set-partitioning) problems; this happens for all problems in which one is required to partition a given set $J = \{1, 2, \ldots, n\}$ of *items* into subsets having special features and to minimize the sum of the costs associated with the subsets. For instance, in the (one-dimensional) *bin-packing problem* (1BP), we are given a set of n *objects* (each having a positive weight) to be partitioned into the minimum number of subsets (*bins*) so that the sum of the weights in each subset does not exceed a given capacity. In the *graph-coloring problem* (GCP), we are given a graph $G = (V, E)$, and the aim is to partition the n *vertices* of V into the minimum number of subsets (*colors*) such that no two vertices of the same subset are both endpoints of any edge in E. In the *capacitated vehicle-routing problem* (VRP), we are given a set of n *customers* (each having a positive demand) to be served by a fleet of K identical vehicles having a certain capacity; in this case, the objective is to partition all the customers into K subsets (*routes*), one for each vehicle, in such a way that the overall traveling cost is minimized and the capacity constraint is satisfied for each subset. Finally, in the *crew-scheduling problem* (CSP), we are given a set of n timetabled *trips* (each having specific features) to be partitioned into a minimum-cost set of feasible subsets (*pairings* or *crew duties*) (the cost and feasibility of each subset depending on several rules laid down by union contracts and company regulations).

As mentioned above, a common property of all these problems is that they can be expressed through a *set-covering* (or *set-partitioning*) formulation. No assumption is required either on the structure of the cost of the subsets (in VRP and CSP each subset gives a different contribution to the cost of a solution), or on the constraints imposed on the feasibility of each subset (for instance, in CSP many feasibility constraints for a pairing are non-linear).

In order to define the set-partitioning formulation for these problems, let $\mathcal{F}(\cdot)$ be a *feasibility function* such that, for each item-set $S \subseteq J$, $\mathcal{F}(S) \leq 1$ if and only if all items in S can be assigned to a unique subset, and let

$$\mathcal{S} = \{S \subseteq J\colon \mathcal{F}(S) \leq 1\} \tag{1}$$

be the family of all feasible item-sets. Moreover, for each $S \in \mathcal{S}$, let $c(S)$ denote the cost of subset S (with $c(S) = 0$ if $S = \varnothing$, and $c(S) > 0$ if $S \neq \varnothing$). Thus, an integer linear programming (ILP) model for the problems previously considered is the following:

$$\min \sum_{S \in \mathcal{S}} c(S)\ \sigma_S \tag{2}$$

$$\sum_{S:\, j \in S} \sigma_S = 1 \quad (j \in J) \tag{3}$$

$$\sigma_S \in \{0, 1\} \quad (S \in \mathcal{S}) \tag{4}$$

where σ_S $(S \in \mathcal{S})$ is a binary variable taking value one if and only if item-set S is selected (i.e., it is assigned to a subset). Objective function (2) minimizes the sum of the costs of the selected subsets, while equalities (3) guarantee that each item is inserted in exactly one subset.

There are many relevant applications in which, given a feasible item-set $S \in \mathcal{S}$, for each $j \in S$, set $\bar{S} := S \backslash \{j\}$ (with $\bar{S} \neq \varnothing$) is also feasible and $c(\bar{S}) = c(S)$; this means that removing one item from a feasible subset leaves the residual subset feasible and does not change its cost. In this situation, we can replace constraints (3) by

$$\sum_{S:\, j \in S} \sigma_S \geq 1 \quad (j \in J). \tag{5}$$

Indeed, given a solution satisfying (5) and violating (3), it is always possible to construct another feasible solution, having exactly the same cost, which satisfies (3). Consider an item $j \in J$ for which the corresponding (5) is not tight: by removing j from all subsets but one, we get a new solution that satisfies (3), and whose cost is equal to that of the original solution.

In this case, we refer to the *set-covering formulation* of the problem. Moreover, we can consider $\mathcal{S}$ as the family of all the item *inclusion maximal* sets, i.e.,

$$\mathcal{S} = \big\{S \subseteq J\colon \mathcal{F}(S) \leq 1,\ \mathcal{F}(S \cup \{k\}) > 1\ \forall\, k \in J \backslash S\big\}. \tag{6}$$

On the one hand, this allows us to reduce the cardinality of $\mathcal{S}$, and hence the number of variables to be considered; on the other hand, we do not need to face a set-partitioning problem, but rather a set-covering problem for which more effective algorithms have been proposed in the literature.

The "nice" property of the formulations above is that they are extremely general; we have to define for each problem only the feasibility function $\mathcal{F}(\cdot)$ and the cost of each subset. On the other hand, the difficult part consists in solving the entire ILP model because the number of variables (i.e., the number of feasible item-sets) can be very large; for this reason, these formulations are usually faced by means of column-generation techniques (see Gilmore and Gomory 1965). Effective exact algorithms based on the column-generation technique for the solution of packing and cutting-stock problems have been recently proposed by Vance et al. (1994), Vance (1998), Valerio de Carvalho (1998), Vanderbeck (1999), and Caprara and Toth (2001).

In this paper, we propose a general heuristic technique for solving NP-hard combinatorial optimization problems that can be formulated as set-covering problems, and apply this approach to two variants of 1BP: the *two-constraint bin-packing problem* (*2CBP*) and the *two-dimensional bin-packing problem* (*2DBP*). In Section 2, we introduce the general heuristic approach, while in Sections 3 and 4 we give more details about our implementation of the algorithms for 2CBP and 2DBP, respectively. Section 5 gives computational results for the two problems on a large set of instances from the literature and compares the performance of the proposed algorithms with that of the most effective heuristic and metaheuristic algorithms in the literature.

2. The Heuristic Approach

The proposed approach is related to the set-covering formulation described in the previous section and operates in two phases. In the first phase (*column-generation*), a very large number of feasible item-sets (*columns*) is generated, while in the second phase (*column-optimization*) a feasible solution of the problem is obtained by solving the associated set-covering instance. Note that if we were able to both generate all maximal item-sets in the first phase, and solve the column-optimization phase to optimality, we would obtain an optimal solution to the original problem. However, explicit enumeration of all feasible item-sets can be too expensive in terms of computing time and can lead to set-covering instances of very large size. In the proposed approach, both the column-generation and the column-optimization phases are heuristically performed. In the first phase, only a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ of feasible item-sets is generated, while in the second phase the associated set-covering problem is heuristically solved, thus defining a feasible solution for the original problem. For this reason, in the following we will call the proposed algorithm *Set-Covering Heuristic* (SCH).

Similar approaches have been used for other combinatorial optimization problems. Caprara et al. (1997), and Caprara et al. (2001) obtained good results by using a similar heuristic technique for the CSP arising in railway applications, in which a set of timetabled train services (*trips*) is required to be covered with a minimum-cost set of crew duties (*pairings*). In that case, each set of trips that can be covered by the same crew corresponds to a feasible pairing and represents a column in the associated second phase, while the feasibility function is implicitly defined by the pairing feasibility rules. Each column has associated a cost depending on several features of the corresponding pairing, such as length or overnight working period, and the problem is to determine a minimum-cost set of pairings covering all the trips. As for applications of the approach to airline crew scheduling, see, for instance, Marsten and Shepardson (1981), Bodin et al. (1983), and Wedelin (1995). In a similar way Kelly and Xu (1999) heuristically solved the capacitated VRP by generating a large set of feasible *routes*, and selecting a subset of them to serve all the clients by heuristically solving an associated set-partitioning problem.

A peculiar feature of the proposed approach is that in the first phase (column-generation) the subfamily $\mathscr{S}'$, containing the feasible item-sets, is not determined through an explicit algorithm (generally called "column generator"), but applying in sequence several *greedy procedures* and "fast" *constructive heuristic algorithms* from the literature, possibly considering different parameter sets. Indeed, each feasible item-set in any heuristic solution of the original problem corresponds to a column of subfamily $\mathscr{S}'$.

It is well known that the solution found by a greedy procedure depends on the *order* in which the items are given in input, although the average performance is better if the items are sorted according to some specific criterion. Because the column-generation phase is aimed at generating a large set of different columns, in the proposed approach each greedy procedure is applied several times, in an iterative way, by sorting the items according to different criteria, so that several different sets of columns (possibly derived from "bad" solutions) are generated.

In the second phase (column-optimization) the set-covering problem corresponding to the columns of subfamily $\mathscr{S}'$ is solved through the Lagrangian heuristic algorithm CFT proposed by Caprara et al. (1999). This iterative algorithm can handle very large set-covering instances (up to millions of columns), producing good (possibly optimal) solutions within a reasonable amount of computing time.

In Section 2.1, we describe how the heuristic algorithms from the literature are used in the column-generation phase, and in Section 2.2 we give some details about the general structure of the SCH approach.

2.1. The Column-Generation Phase

As mentioned in the previous section, the column-generation phase aims at generating a large set of columns, which define the set-covering instance used in the following phase. This phase can be performed by applying a set of heuristic algorithms and storing the item-sets of the corresponding solutions in subfamily $\mathscr{S}'$. The aim of the column-generation phase is twofold: we want both to obtain a good feasible solution for the original problem, and to generate a large set of different columns (possibly all the "good" feasible columns). In particular, the aim is to generate a set of columns that are, in some sense, "spread," i.e., such that the items are mixed in the columns in a scrambled way.

The proposed way to get a large number of different columns is to apply some greedy algorithms several times, each time sorting the items according to different (almost random) criteria, so that many different columns are generated. In particular, at the first iteration the items are sorted according to some specified criterion, while the following iterations are performed according to a two-loop procedure: in the external loop at most $n/2$ iterations are performed by partitioning the items into groups of $2t$ consecutive items ($t = 1, \ldots, \lfloor n/2 \rfloor$) and, for each group, the first t items are switched with the last t items (i.e., items j and $t + j$ are switched, for $j = 1, \ldots, t$), while in the internal loop at most $n - 1$ iterations are performed, by shifting the items by one position in a circular way (i.e., the first item becomes the last, the second becomes the first, ..., and the last becomes the next-to-last).

The main drawbacks of this approach are evident:

- a lot of columns that are not maximal, according to definition (6), are generated;
- the same item-set can be generated by different procedures, thus producing many redundant columns.

The first problem often arises when dealing with greedy algorithms. Indeed, the first item-sets in a solution are generally "well-filled" (in the sense that they are maximal), while the same does not occur for the last item-sets of the solution (because only few items are still available). This drawback can be heuristically solved by applying a *heuristic-fill* procedure. In this step, one tries to increase the size of a non-maximal item-set S by adding to S the first item $j \in J \backslash S$ such that $S \cup \{j\}$ is still feasible. Once a new item j is added to S, the procedure is iterated, starting from the item following j in $J \backslash S$, until no more items can be added to S. This procedure can be used in a *generation fashion*, by considering different orders of the items in $J \backslash S$, so as to produce several maximal columns (at most K, K being a parameter of the algorithm) starting from the same item-set. Note that, for some problems, testing whether an item can be inserted into an item-set is an NP-hard problem; in these cases, the test can be performed in a heuristic way.

As for the second problem, a *hashing technique* is used to avoid storing identical columns. To achieve this goal, for each possible *hashing score* v, a list L_v

of columns having score v is defined. Each list L_v is stored through a pointer technique, so as to have no limit on its cardinality. Each feasible column is assigned a score v and is compared with all columns in L_v (if any); if the current column turns out to be identical to one of the columns in L_v, then it is disregarded; otherwise it is stored in L_v and added to subfamily $\mathcal{S}'$. Thus, given a feasible item-set S, the hashing procedure operates as follows:

hashing(S)
begin
1. compute the score v of item-set S;
2. **for each** item-set R in L_v
   if R and S are identical **then** exit
3. **endfor**;
4. insert item-set S in list L_v;
5. complete item-set S by applying the heuristic-fill procedure, thus obtaining item-set S';
6. **if** $S' = S$ **then** store item-set S in $\mathcal{S}'$
   else
7. compute the score v' of item-set S';
8. **for each** item-set R in $L_{v'}$
   if R and S are identical **then** exit
9. **endfor**;
10. insert item-set S' in list $L_{v'}$;
11. store item-set S' in $\mathcal{S}'$
12. **endif**
end

Note that the original item-set S is not stored in subfamily $\mathcal{S}'$ if it is not maximal (i.e., if $S' \neq S$). In addition, the heuristic-fill procedure is not applied to S if it belongs to list L_v.

In order to implement the hashing technique, we represent each item-set S by means of an n-dimensional binary vector a, with $a_j = 1$ iff item $j \in J$ belongs to item-set S. For each item-set S, we compute its hashing score v by using the following hashing function:

$$v(S) = \frac{\sum_{p=1}^{4} H_p(S)}{3}$$

where

$$H_p(S) = \sum_{j=0}^{\alpha-1}\left(a_{p+j\delta}2^{\alpha-j} + \sum_{k=p+j\delta+1}^{p+(j+1)\delta-1} a_k \left\lceil \frac{k}{2}+1 \right\rceil^2\right) \tag{7}$$

for $p = 1, \ldots, 4$, $\alpha = 20$ and $\delta = \lfloor n/\alpha \rfloor$. In (7) all values of k greater than n must be replaced by $k - n$. The overall space complexity of the hashing structure is thus bounded by $O(2^\alpha)$. The hashing procedure gave quite satisfactory results in terms of computing time because it took about 6% of the global generation time for the instances considered in our computational experiments (see Section 5). Note that this hashing technique is "exact" in the sense that it disregards a column if and only if it is equal to a previously inserted one.

Both the column-generation and column-optimization phases can be stopped as soon as a solution that is proved to be optimal is found, i.e., if the cost UB of the best solution found so far is equal to a lower bound for the original problem. For this reason, before starting the column-generation phase, we apply all the "fast" lower-bounding procedures available in the literature and take the maximum of the corresponding values as the best "lower bound" LB.

2.2. The General Structure of the Algorithm

Let us consider any feasible solution $A = \{S_1, S_2, \ldots, S_b\}$ of the original problem found by one of the procedures applied during the column-generation phase, where S_i ($i = 1, 2, \ldots, b$) denotes the ith item-set of the solution. Moreover, let $\bar{c}(A) := \sum_{i=1}^{b} c(S_i)$ be the corresponding cost. For each solution A, the following two steps are performed.

(i) Possibly update the best solution found so far: if $\bar{c}(A) < \text{UB}$ then set $\text{UB} := \bar{c}(A)$ and store A as the best solution found so far (if UB = LB then stop).

(ii) Insert columns $S_1, S_2, \ldots, S_b$ into subfamily $\mathcal{S}'$ (removing possible redundant columns) by applying, for $i = 1, \ldots, b$, procedure *hashing*(S_i).

The general structure of the SCH approach follows.

Heuristic Algorithm SCH

Initialization Phase

1. Apply the fast lower-bounding procedures from the literature, and let LB be the maximum of the corresponding lower bounds.

Column-Generation Phase

2. Apply the *greedy procedures* from the literature (possibly updating UB).

3. Apply the *fast constructive algorithms* from the literature (possibly updating UB).

4. For *each greedy procedure*: Apply the procedure in an iterative way by sorting the items according to different criteria (possibly updating UB).

Column-Optimization Phase

5. Apply heuristic algorithm CFT (with a given time limit) to the set-covering instance corresponding to subfamily $\mathcal{S}'$ (possibly updating UB).

Note that algorithm CFT by Caprara et al. (1999) computes an "internal" lower bound (not valid for the original problem) on the value of the optimal solution of the corresponding set-covering instance. Whenever this lower bound becomes equal to the current UB value, algorithm CFT stops (possibly without reaching the given time limit).

As previously mentioned, in the following sections we will apply algorithm SCH to two bin-packing problems: 2CBP and 2DBP. A typical behavior of the branch-and-bound algorithms proposed for finding the exact solution of this type of problem is that for several instances they are able to prove optimality of the best solution found so far within very short computing times (generally a few seconds), while for the remaining instances the corresponding computing times can be very large (hours or days). For this reason, in the column-generation phase, the set of columns of subfamily $\mathcal{S}'$ is augmented by considering the best solution found, within a small time limit, by some branch-and-bound algorithm from the literature. The aim of this step is possibly to find, in a short computing time, the optimal solution of the original problem, or to generate good columns that are generally different from those obtained by the heuristic procedures. In particular, the following additional step is performed just after Step 3.

3a. Apply (with a small time limit) some *branch-and-bound algorithm* from the literature (possibly updating UB and LB).

3. The Two-Constraint Bin-Packing Problem

The first problem we consider is the two-constraint bin-packing problem (2CBP, also called the *two-dimensional vector-packing problem*), a generalization of 1BP in which each object $j \in J$ (with $n = |J|$) has two attributes, a *weight* w_j and a *volume* v_j (with $w_j \geq 0$, $v_j \geq 0$, and $w_j + v_j > 0$), and bins have weight and volume capacities, denoted W and V, respectively. The n objects have to be packed into the minimum number of bins, so that the sum of the weights and the sum of the volumes in each bin do not exceed W and V, respectively. 2CBP is NP-hard in the strong sense because it generalizes 1BP, arising when $v_j = 1$ ($j \in J$) and $V = n$.

This problem has been widely studied in the literature. Garey et al. (1976) considered the mCBP (the generalization of 2CBP in which objects have m attributes and bins have m capacities), while Spieksma (1994) considered some applications of 2CBP in loading and scheduling contexts, and proposed lower bounds, constructive heuristics, and a branch-and-bound algorithm. Approximation procedures with guaranteed performance ratios have been presented by Garey et al. (1976), Maruyama et al. (1977), Yao (1980), Fernandez de la Vega and Lueker (1981), Chekuri and Khanna (1999), and Kellerer and Kotov (2003). Woeginger (1997) showed that 2CBP has no asymptotic polynomial-time approximation scheme unless $P = \mathrm{NP}$. Extensive studies on 2CBP have been recently presented in Caprara and Toth (2001), where several lower bounds, greedy procedures, and constructive heuristics are embedded into exact enumerative algorithms based on branch-and-bound and branch-and-price techniques, while Caprara et al. (2002) proposed fast greedy procedures for the mCBP.

In the case of 2CBP, the feasibility function can be easily expressed as

$$\mathcal{F}(S) = \max\left\{\frac{\sum_{j\in S} w_j}{W}, \frac{\sum_{j\in S} v_j}{V}\right\}. \tag{8}$$

Lower bounds are obtained by using the fast bounding procedures L_C, L_1, and L_2 proposed by Spieksma (1994) and by Caprara and Toth (2001).

The following greedy procedures from the literature have been used:

- 2FFD (two-constraint first-fit decreasing), proposed by Garey et al. (1976);
- 2BFD (two-constraint best-fit decreasing), proposed by Caprara and Toth (2001);
- one-way decreasing and double-way decreasing, described in Caprara et al. (2002).

The following fast constructive heuristics have been used:

- $2\mathrm{FFD}_\mu$ proposed by Spieksma (1994);
- $2\mathrm{FFD}_\lambda$, $2\mathrm{BFD}_\lambda$, and $2\mathrm{BFD}_\mu$, proposed by Caprara and Toth (2001).

As for H_M, the matching-based heuristic algorithm proposed by Caprara and Toth (2001), we did not use it in the column-generation phase since it may require much computing time, even if it produces solutions that are quite good on some sets of instances (see Caprara and Toth 2001).

An exchange procedure 2REF proposed by Caprara and Toth (2001) is applied to each of the feasible solutions found in the column-generation phase, possibly improving the best solution so far and producing new feasible item-sets.

As an exact algorithm, the branch-and-bound algorithm BB2, proposed by Caprara and Toth (2001), has been used.

Note that, for 2CBP, the problem of testing whether an item-set S is maximal can be solved in linear time because one can consider all items $j \in J\backslash S$ and check if j fits in both the weight and the volume capacities.

4. The Two-Dimensional Bin-Packing Problem

In the two-dimensional bin-packing problem (2DBP), one is required to pack a set of n rectangular objects into identical rectangular bins, in such a way that (i) all objects are packed, with their edges parallel to the edges of the bin; (ii) objects packed in the same bin do not overlap; and (iii) the number of bins used is minimized. This problem has been extensively studied in the literature and several variants have been proposed, depending on the possibility of rotating the objects and on additional requests concerning the way of packing (*cutting*) the objects (guillotine cuts, two-staged packing, and so on). In this section we consider the case in which objects have fixed orientation and no additional constraint is imposed on the cuts. According to the three-field typology introduced by Lodi et al. (1999a), the problem considered in this section may also be denoted as 2BP|O|F. The problem is NP-hard in the strong sense since it generalizes 1BP, arising when objects and bins have only one dimension.

Exact approaches for 2DBP have been proposed in the literature. Christofides and Whitlock (1977) proposed an ILP model based on a discrete representation of the geometric space and solved the problem by using a Lagrangian relaxation of the model. Martello and Vigo (1998) introduced combinatorial lower bounds and embedded them into a branch-and-bound algorithm. Finally, Fekete and Schepers (1997, 2004a and b) proposed a general framework for the exact solution of multi-dimensional packing problems.

Approximate algorithms with asymptotic worst-case performance guarantee have been proposed by Chung et al. (1982) and by Frenk and Galambos (1987). Greedy procedures and constructive heuristics have been proposed by Berkey and Wang (1987). Lodi et al. (1999a, b) proposed fast constructive heuristics and a tabu-search approach that, starting from a feasible solution, tries to recombine the objects packed into a set of k bins, plus one object packed into a *target bin*, until this bin has been emptied. Færø et al. (2003) proposed a guided-local-search technique (see Voudouris and Tsang 1999 for details) that starts from a feasible solution and randomly removes some bins assigning the corresponding objects to the other bins. The new solution is generally infeasible and the objective function is given by the pairwise overlapping area; the associated neighborhood is explored through object shifts, until a feasible solution is found. Recently, Boschetti and Mingozzi (2003a, b) proposed new lower bounds and an effective constructive heuristic (called HBP) that assigns a score to each object, considers the objects according to decreasing values of the corresponding scores, updates the scores by using a specified criterion, and iterates until an optimal solution is found or a maximum number of iterations has been performed. The execution of the algorithm is repeated for a given set of different criteria used for the updating of the object scores.

For recent reviews on two-dimensional packing problems, the reader is referred to Dyckhoff et al. (1997) and to Lodi et al. (2002).

The approach described in Section 2 can clearly be used for solving 2DBP. In this case, computation of the feasibility function $\mathcal{F}(S)$ is difficult because the problem of testing whether an object-set S is feasible is strongly NP-hard (see Martello and Vigo 1998).

As for the lower bounds, we used those proposed by Martello and Vigo (1998) and by Boschetti and Mingozzi (2003a).

The following greedy procedures from the literature have been used:

- *Finite bottom-left*, *finite first-fit*, and *finite best-fit*, proposed by Berkey and Wang (1987);
- *Alternate directions*, proposed by Lodi et al. (1999a).

The following constructive heuristics have been used:

- *Floor ceiling* and *knapsack packing*, proposed by Lodi et al. (1999a, b);
- HBP, proposed by Boschetti and Mingozzi (2003b).

As an exact algorithm, we used the improved version of the branch-and-bound algorithm proposed by Martello and Vigo (1998), as described in Martello and Vigo (2001).

The main difference with respect to 2CBP is that, as previously mentioned, given a feasible object-set S, the problem of testing whether this set is maximal (according to (6)) is strongly NP-hard. In our preliminary computational experiments, we performed several attempts to implement an effective heuristic-fill procedure. We first tried to insert objects $j \in J\setminus S$ by means of the bottom-left strategy (see Baker et al. 1980), but this approach led to large overall computing times for the column-generation phase. On the other hand, the adaptation of simple shelf algorithms, such as finite first-fit and finite best-fit (see Berkey and Wang 1987), did not significantly improve the quality of the columns produced in the first phase. Thus, for 2DBP, we did not use the heuristic-fill procedure, adding to subfamily $\mathcal{S}'$ only object-sets directly produced by the heuristics mentioned above.

However, to obtain good columns from the worst bins in the greedy solutions, we use a more sophisticated generation algorithm; given a feasible solution, we compute a score for each of the b used bins (according to the filling function described in Lodi et al. 1999a) and construct a sub-instance composed by the objects currently packed in the "worst" $b/2$ bins. All the heuristics described above, but the one producing the initial solution, are applied to this new instance, each producing a new set of candidate columns. Computational results showed that this generation algorithm, similar to procedure 2REF mentioned in Section 3, gives better results than the standard way of generating columns.

5. Computational Results

In this section we present computational results of algorithm SCH on a large set of instances from the literature. All instances (and the corresponding best known solution values) are available at `www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`. All procedures used in SCH were coded in Fortran 77 and run on a Digital Alpha 533 MHz. (having 16.1 SPECint95 value). We tested the algorithms with two different time limits: $TL = 30$ seconds and $TL = 100$ seconds. The parameters that entirely describe the behavior of algorithm SCH for a given time limit TL are the following: TG (with $TG < TL$), i.e., the time limit imposed on the column-generation phase, TE (with $TE < TG$), i.e., the time limit imposed on the execution of the exact algorithm, and K, i.e., the maximum number of columns obtained by an object-set in the heuristic-fill procedure.

5.1. Results on the 2CBP Instances

The algorithms for 2CBP were tested on all the instances proposed in the literature. In particular, we considered the set of instances introduced by Spieksma (1994) and by Caprara and Toth (2001). This set of instances contains 10 classes, each composed of 40 instances (10 instances for 4 different values of n), so the codes were tested on 400 instances.

Tables 1 and 2 report computational results for the two time limits $TL = 30$ seconds and $TL = 100$ seconds, respectively. The results concern only 5 classes of instances (in particular, classes 1, 6, 7, 9, and 10) since almost all the instances of the remaining classes are easily solved to proven optimality by simple greedy heuristics. Computational experiments suggested using $TG = 15$ seconds, $TE = 2$ seconds, and $K = 20$ when $TL = 30$ seconds, and $TG = 50$ seconds, $TE = 5$ seconds, and $K = 20$ when $TL = 100$ seconds.

Each table reports, for a given time limit TL, the results obtained by SCH after the application of the initial heuristic algorithms (Steps 2, 3, and 3a of Section 2.2, column "Initial Heuristics"), at the end of the column-generation phase (Step 4, column "Phase 1"), and at the end of the column-optimization phase (Step 5, column "SCH"), and those of the heuristics from the literature (including the matching-based algorithm H_M by Caprara and Toth 2001) with an overall time limit of TL seconds (column "Lit. heurs"), and of the exact algorithm BB2 proposed in Caprara and Toth (2001) with time limit TL (column "BB2"). In particular, for each class and value of n ($n \in \{25, 50, 100, 200\}$ for classes 1, 6, 7, and 9 and $n \in \{24, 51, 99, 201\}$ for class 10), we give:

- class and value of n;
- the sum (with respect to the 10 corresponding instances) of the *best known* lower bounds (column "LB*"); these lower bounds were computed by applying all the lower-bounding procedures from the literature and an exact algorithm for a large computing time;

Table 1 Two-Constraint Bin-Packing Problem: Instances Proposed by Spieksma (1994) and by Caprara and Toth (2001)

				Initial heuristics			Phase 1			SCH					Lit. heurs			BB2		
Class	n	LB*	LB	#o	UB	T	#o	UB	T	$\|\mathcal{S}'\|$	#o	#o*	UB	T	#o*	UB	T	#o*	UB	T
1	25	69	69	10	69	0.07	10	69	0.07	0	10	10	69	0.07	10	69	0.07	10	69	0.01
	50	135	135	10	135	0.88	10	135	0.88	0	10	10	135	0.88	6	139	0.09	10	135	0.18
	100	255	255	3	262	1.75	5	260	8.35	31,889	5	5	260	10.10	2	263	0.35	2	264	24.23
	200	503	503	0	529	2.48	0	526	15.09	15,488	1	1	512	29.63	0	530	3.83	0	530	30.00
	Global	962	962	23	995	1.30	25	990	6.10	20,955	26	26	976	10.17	18	1,001	1.08	22	998	13.60
6	25	101	101	10	101	0.08	10	101	0.08	0	10	10	101	0.08	9	102	0.08	10	101	0.01
	50	214	213	5	218	1.18	7	216	5.39	2,002	8	9	215	5.40	7	217	0.11	7	217	14.26
	100	405	405	0	424	2.24	0	422	15.04	9,545	5	5	410	15.63	1	415	0.42	0	421	30.00
	200	803	803	0	844	2.77	0	844	15.24	9,020	0	0	816	26.01	0	824	8.82	0	843	30.00
	Global	1,523	1,522	15	1,587	1.57	17	1,583	8.94	8,333	23	24	1,542	11.78	17	1,558	2.36	17	1,582	18.57
7	25	96	96	10	96	0.12	10	96	0.12	0	10	10	96	0.12	8	98	0.07	10	96	0.05
	50	196	196	5	201	1.27	5	201	7.72	9,436	9	9	197	7.82	4	202	0.11	7	199	9.86
	100	398	398	3	405	2.23	3	405	11.21	21,072	3	3	405	11.67	3	405	0.46	3	405	21.24
	200	799	799	1	810	2.72	1	810	14.00	9,443	7	7	802	19.98	0	812	3.89	1	809	28.31
	Global	1,489	1,489	19	1,512	1.59	19	1,512	8.26	13,318	29	29	1,500	9.90	15	1,517	1.13	21	1,509	14.87
9	25	73	73	10	73	0.11	10	73	0.11	0	10	10	73	0.11	10	73	0.08	10	73	0.05
	50	144	139	4	145	1.59	4	145	9.33	22,613	4	9	145	9.41	9	145	0.13	9	146	9.87
	100	257	257	0	276	2.18	0	272	15.03	32,516	0	0	268	19.40	0	277	0.44	0	276	30.00
	200	503	503	0	534	2.49	0	533	15.10	15,449	0	0	521	29.05	0	537	4.14	0	534	30.00
	Global	977	972	14	1,028	1.59	14	1,023	9.89	23,667	14	19	1,007	14.49	19	1,032	1.19	19	1,029	17.48
10	24	80	80	10	80	0.08	10	80	0.08	0	10	10	80	0.08	1	89	27.08	10	80	0.00
	51	170	170	3	180	1.52	3	177	10.54	4,846	10	10	170	10.56	1	179	3.12	2	186	24.02
	99	330	330	0	351	2.22	0	350	15.03	17,784	8	8	332	15.26	0	342	0.33	0	357	30.00
	201	670	670	0	698	2.67	0	698	15.14	17,044	0	0	680	28.58	0	680	1.98	0	713	30.00
	Global	1,250	1,250	13	1,309	1.62	13	1,305	10.20	14,156	28	28	1,262	13.62	2	1,290	8.13	12	1,336	21.01
Overall		6,201	6,195	84	6,431	1.53	88	6,413	8.68	15,921	120	126	6,287	11.99	71	6,398	2.78	91	6,454	17.10

Note. CPU seconds on a digital alpha 533 MHz. Values over 10 instances. Time limit for each instance: 30 seconds.

Table 2 Two-Constraint Bin-Packing Problem: Instances Proposed by Spieksma (1994) and by Caprara and Toth (2001)

				Initial heuristics			Phase 1			SCH					Lit. heurs			BB2		
Class	*n*	LB*	LB	#*o*	UB	*T*	#*o*	UB	*T*	$\lvert\mathcal{S}'\rvert$	#*o*	#*o**	UB	*T*	#*o**	UB	*T*	#*o**	UB	*T*
1	25	69	69	10	69	0.06	10	69	0.06	0	10	10	69	0.06	10	69	0.07	10	69	0.01
	50	135	135	10	135	2.08	10	135	2.07	0	10	10	135	2.07	6	139	0.09	10	135	0.19
	100	255	255	3	262	4.15	5	260	26.75	92,693	5	5	260	32.05	2	263	0.35	3	263	74.70
	200	503	503	0	529	5.49	0	525	50.07	69,257	3	3	510	93.69	0	530	3.89	0	530	100.00
	Global	962	962	23	995	2.95	25	989	19.74	77,069	28	28	974	31.97	18	1,001	1.10	23	997	43.72
6	25	101	101	10	101	0.08	10	101	0.08	0	10	10	101	0.08	9	102	0.08	10	101	0.01
	50	214	213	5	218	2.68	7	216	13.37	2,063	8	9	215	13.38	7	217	0.11	8	216	33.70
	100	405	405	0	423	5.24	0	421	50.05	11,076	5	5	410	50.73	1	415	0.42	0	420	100.00
	200	803	803	0	843	5.77	0	843	50.15	24,822	2	2	811	61.49	0	824	22.82	0	841	100.00
	Global	1,523	1,522	15	1,585	3.44	17	1,581	28.41	15,877	25	26	1,537	31.42	17	1,558	5.86	18	1,578	58.43
7	25	96	96	10	96	0.13	10	96	0.12	0	10	10	96	0.12	8	98	0.07	10	96	0.06
	50	196	196	6	200	2.55	6	200	14.20	11,864	9	9	197	14.27	4	202	0.11	7	199	30.86
	100	398	398	3	405	5.13	3	405	36.48	27,393	3	3	405	37.07	3	405	0.46	3	405	70.24
	200	799	799	1	809	5.73	1	809	45.75	60,941	7	7	802	59.93	0	812	3.89	1	809	91.31
	Global	1,489	1,489	20	1,510	3.38	20	1,510	24.14	39,384	29	29	1,500	27.85	15	1,517	1.13	21	1,509	48.12
9	25	73	73	10	73	0.12	10	73	0.11	0	10	10	73	0.11	10	73	0.08	10	73	0.04
	50	144	140	5	145	3.23	5	145	14.53	29,019	5	9	145	14.62	9	145	0.12	9	146	23.79
	100	257	257	0	276	5.17	0	271	50.03	90,882	0	0	267	52.45	0	277	0.44	0	276	100.00
	200	503	503	0	534	5.49	0	532	50.14	68,649	0	0	513	73.17	0	537	4.15	0	534	100.00
	Global	977	973	15	1,028	3.50	15	1,021	28.70	69,616	15	19	998	35.09	19	1,032	1.19	19	1,029	55.96
10	24	80	80	10	80	0.08	10	80	0.08	0	10	10	80	0.08	1	89	90.08	10	80	0.01
	51	170	170	3	180	3.62	3	177	25.42	5,158	10	10	170	25.44	1	179	10.12	2	186	80.02
	99	330	330	0	351	5.21	0	350	50.02	27,201	9	9	331	50.38	0	342	0.32	0	357	100.00
	201	670	670	0	698	5.67	0	698	50.19	60,595	2	2	678	97.49	0	680	1.96	0	712	100.00
	Global	1,250	1,250	13	1,309	3.64	13	1,305	31.43	33,854	31	31	1,259	43.35	2	1,290	25.62	12	1,335	70.01
Overall		6,201	6,196	86	6,427	3.38	90	6,406	26.49	45,121	128	133	6,268	33.94	71	6,398	6.98	93	6,448	55.25

Note. CPU seconds on a digital alpha 533 MHz. Values over 10 instances. Time limit for each instance: 100 seconds.

- the sum of the *best internal* lower bounds (column "LB") found by the fast lower-bounding procedures or by the exact algorithm within the corresponding time limit *TE*;
- for columns "Initial Heuristics," "Phase 1," and "SCH": the number of instances solved to proven optimality with respect to the best internal lower bound (column "#*o*");
- for columns "SCH," "Lit. heurs," and "BB2": the number of instances solved to proven optimality with respect to the best known lower bound (column "#*o**");
- for column "SCH": the average number (with respect to the instances not solved to proven optimality at the end of Phase 1) of columns in subfamily $\mathcal{S}'$ (column "$\lvert\mathcal{S}'\rvert$");
- for each algorithm: the sum of the upper bounds (column "*UB*") and the average computing time expressed in seconds (column "*T*").

In addition, for each class, line "Global" reports (with respect to the corresponding 40 instances) the sum of the best known and of the best internal lower bounds, the average number of columns in $\mathcal{S}'$ for instances not solved to proven optimality at the end of Phase 1, and, for each algorithm, the number of instances solved to proven optimality, the sum of the upper bounds, and the average computing time. The same data, with respect to all the classes, are reported in line "Overall."

Tables 1 and 2 show that, as may be expected, results with $TL = 30$ seconds are worse than those with $TL = 100$ seconds. For the initial heuristics, the overall number of bins is equal to 6,431 and 6,427, respectively (the difference being due to four instances for which better solutions were found in the latter case by the exact algorithm), with a *gap* with respect to the overall sum of the best known lower bounds equal to 230 and 226, respectively. For the first phase, the gap is 212 and 205, respectively, with an overall reduction of the number of bins, with respect to the initial heuristics, equal to 18 and 21, respectively. The column-optimization phase has similar behavior, producing a gap equal to 86 and 67, respectively, with a reduction, with respect to the first phase, of 126 and 138 bins, respectively. Note that these savings have been obtained out of the 112 and 110, respectively, instances not solved to proven optimality at the end of the column-generation phase. Note also that a bigger time limit for the exact algorithm in Step 3a (from 2 to 5 seconds, for $TL = 30$ seconds and $TL = 100$ seconds, respectively) leads to an increase of the sum of the best internal lower bounds from 6,195 to 6,196. As for the number of instances solved to proven optimality by SCH, this is equal to 120 and 128 (out of 200 instances), respectively. As for what affects the average computing times, note that the execution of algorithm CFT in the optimization phase marginally increases the time of Phase 1 (from 8.68 to 11.99 seconds for $TL = 30$ seconds, and from 26.49 to 33.94 seconds for $TL = 100$ seconds).

The tables also show that, for both time limits, algorithm SCH outperforms the previous heuristics from the literature and the exact algorithm proposed by Caprara and Toth (2001). The gap corresponding to the heuristics from the literature is 197 for both time limits, while it is 253 and 247, respectively, for the exact algorithm.

5.2. Results on the 2DBP Instances

For 2DBP we tested all the instances proposed in the literature. As was done for 2CBP, we consider two different time limits: $TL = 30$ seconds and $TL = 100$ seconds. For the former time limit, we set $TG = 10$ seconds and $TE = 2$ seconds, while for the latter time limit, we set $TG = 25$ seconds and $TE = 5$ seconds; in both cases we set $K = 0$, i.e., no heuristic-fill procedure is executed (see the discussion in Section 4). As for algorithm HBP by Boschetti and Mingozzi (2003b), we used it in both Steps 3 and 4 of the column-generation phase (see Section 4). For each criterion used for updating the object scores, the maximum number of iterations was set to 1 in Step 3 (using HBP as a fast heuristic) and, in Step 4, to 100 (as suggested by the authors) and 250 for TG equal to 10 seconds and 25 seconds, respectively. The same numbers of iterations were used for the iterative execution of the other greedy procedures considered in Step 4.

We first consider the instances proposed by Berkey and Wang (1987) and by Martello and Vigo (1998). Each class is composed of 50 instances (10 instances for each value of $n \in \{20, 40, 60, 80, 100\}$), so a set of 500 instances has been considered.

Tables 3 and 4 give the results obtained by algorithm SCH, for the two time limits respectively, after the application of the heuristics from the literature (comprehensive of the exact algorithm proposed by Martello and Vigo 2001, column "Initial Heuristics"), after the iterative execution of algorithm HBP by Boschetti and Mingozzi (2003b) (column "After HBP"), at the end of the column-generation phase (column "Phase 1") and at the end of the column-optimization phase (column "SCH"). The entries in the tables for each class and value of n are analogous to those of Tables 1 and 2.

The overall number of bins (with respect to the 500 instances) needed by the best solution found by the initial heuristics is 7,308 and 7,303 for the two time limits, respectively, with a gap with respect to the overall sum of the best known lower bounds equal to 135 and 130, respectively. The gap after the application of algorithm HBP is 99 and 94, respectively, while the remaining part of the column-generation phase never improves the value of the best solution found. The gap of algorithm SCH is 75 and 70 for the two time limits, respectively. Thus the column-optimization phase led to a saving of 24 bins for both time limits. Note that these savings were obtained out of the 104 and 97, respectively, instances not solved to proven optimality at the end of the column-generation phase.

To test the effectiveness of the approach, we compared the performance of algorithm SCH with that of the best algorithms proposed in the literature for 2DBP. In particular, we considered the exact algorithm by Martello and Vigo (2001) (column "Exact Algorithm"), the tabu-search algorithm by Lodi et al. (1999a) (column "Tabu Search"), the guided-local-search algorithm by Færø et al. (2003) (column "GLS"), and the constructive algorithm by Boschetti and Mingozzi (2003b) (column "HBP"). Since the results of algorithm HBP reported by Boschetti and Mingozzi (2003b) were obtained by performing at most 100 iterations, without imposing any time limit, we also give the results of our implementation of the algorithm with $TL = 30$ seconds and $TL = 100$ seconds (column "HBP(TL)"). The results of our implementation of HBP obtained by performing at most 100 iterations are equivalent to those reported by the authors for what affects the values of the best solution found, while our computing times are on average four times smaller than those given in Boschetti and Mingozzi (2003b).

Tables 5 and 6 give the results of algorithm SCH and of the algorithms mentioned above for $TL = 30$ seconds and $TL = 100$ seconds, respectively. The entries of the tables for each class and value of n are analogous to those of Tables 3 and 4, the only difference being that column "$\#o^*$" replaces column "$\#o$."

To have a "fair" comparison of the algorithms we performed the following experiments. As for the exact algorithm proposed in Martello and Vigo (2001), we ran the corresponding code on our machine for the two time limits. As for the tabu-search algorithm by Lodi et al. (1999a) we used the results reported in the paper for a time limit of 60 seconds; because these results were obtained on a Silicon Graphics INDY R10000sc 195 MHz (which is about half as fast as our machine), they can be compared with the results obtained by algorithm SCH with $TL = 30$ seconds. In the paper, the authors note that the tabu-search algorithm in its best version (obtained by setting the maximum number of distinct tabu lists to 3) gives good results when the time limit is 60 seconds, while

Table 3 Two-Dimensional Bin-Packing Problem: Instances Proposed by Berkey and Wang (1987) and by Martello and Vigo (1998)

				Initial heuristics			After HBP			Phase 1			SCH			
Class	n	LB*	LB	#o	UB	T	#o	UB	T	#o	UB	T	$\lvert\mathscr{S}'\rvert$	#o	UB	T
1	20	71	71	10	71	0.07	10	71	0.07	10	71	0.07	0	10	71	0.07
	40	134	131	5	136	1.06	7	134	1.18	7	134	1.23	5,057	7	134	3.93
	60	197	197	6	201	0.87	6	201	1.15	6	201	1.28	9,326	7	200	2.50
	80	274	274	9	275	0.27	9	275	0.39	9	275	0.45	12,109	9	275	2.50
	100	317	317	5	322	1.15	7	320	1.68	7	320	1.73	13,479	10	317	3.38
	Global	993	990	35	1,005	0.68	39	1,001	0.89	39	1,001	0.95	9,547	43	997	2.48
2	20	10	10	10	10	0.06	10	10	0.06	10	10	0.06	0	10	10	0.06
	40	19	19	9	20	0.27	10	19	0.31	10	19	0.31	0	10	19	0.31
	60	25	25	10	25	0.07	10	25	0.07	10	25	0.07	0	10	25	0.07
	80	31	31	10	31	0.07	10	31	0.07	10	31	0.07	0	10	31	0.07
	100	39	39	9	40	0.28	10	39	0.37	10	39	0.37	0	10	39	0.37
	Global	124	124	48	126	0.15	50	124	0.18	50	124	0.18	0	50	124	0.18
3	20	51	51	10	51	0.07	10	51	0.07	10	51	0.07	0	10	51	0.07
	40	92	92	7	95	0.69	8	94	0.78	8	94	0.81	8,556	8	94	1.10
	60	136	136	6	140	0.91	6	140	1.20	6	140	1.32	12,342	7	139	2.66
	80	187	187	3	195	1.53	6	191	2.04	6	191	2.26	14,453	7	190	6.09
	100	221	221	3	230	1.72	5	226	2.59	5	226	2.67	16,196	8	223	5.10
	Global	687	687	29	711	0.98	35	702	1.34	35	702	1.42	13,685	40	697	3.00
4	20	10	10	10	10	0.06	10	10	0.06	10	10	0.06	0	10	10	0.06
	40	19	19	10	19	0.07	10	19	0.07	10	19	0.07	0	10	19	0.07
	60	23	23	8	25	0.50	8	25	1.45	8	25	1.79	7,435	8	25	1.86
	80	30	30	7	33	0.74	8	32	2.15	8	32	2.48	8,777	8	32	3.15
	100	37	37	9	38	0.36	9	38	1.13	9	38	1.13	4,576	9	38	2.32
	Global	119	119	44	125	0.35	45	124	0.97	45	124	1.11	7,400	45	124	1.49
5	20	65	65	10	65	0.06	10	65	0.06	10	65	0.06	0	10	65	0.06
	40	119	116	7	119	0.85	7	119	0.96	7	119	1.01	4,766	7	119	1.21
	60	179	179	7	182	0.87	9	180	1.00	9	180	1.03	10,150	9	180	1.05
	80	241	241	3	249	1.57	4	247	2.31	4	247	2.66	11,594	4	247	8.82
	100	279	279	3	287	1.66	3	287	2.93	3	287	3.04	9,974	6	283	5.30
	Global	883	880	30	902	1.00	33	898	1.45	33	898	1.56	9,637	36	894	3.29
6	20	10	10	10	10	0.07	10	10	0.07	10	10	0.07	0	10	10	0.07
	40	15	15	6	19	0.93	8	17	2.81	8	17	2.81	1,950	8	17	2.81
	60	21	21	9	22	0.35	10	21	0.35	10	21	0.35	0	10	21	0.35
	80	30	30	10	30	0.23	10	30	0.23	10	30	0.23	0	10	30	0.23
	100	32	32	8	34	0.82	8	34	2.29	8	34	2.29	2,053	8	34	2.75
	Global	108	108	43	115	0.48	46	112	1.15	46	112	1.15	2,001	46	112	1.24
7	20	55	55	10	55	0.12	10	55	0.12	10	55	0.12	0	10	55	0.12
	40	109	109	5	114	1.07	7	112	1.22	7	112	1.27	5,141	8	111	1.41
	60	156	156	5	161	1.11	6	160	1.41	6	160	1.55	9,719	8	158	3.50
	80	224	224	1	233	1.96	2	232	2.90	2	232	3.37	13,163	2	232	15.71
	100	269	269	4	276	1.42	5	274	2.40	5	274	2.48	10,600	8	271	9.86
	Global	813	813	25	839	1.14	30	833	1.61	30	833	1.76	10,630	36	827	6.12
8	20	58	58	10	58	0.06	10	58	0.06	10	58	0.06	0	10	58	0.06
	40	112	112	9	113	0.28	9	113	0.31	9	113	0.33	4,270	9	113	0.49
	60	159	159	5	164	1.10	7	162	1.34	7	162	1.43	8,167	7	162	3.36
	80	223	223	7	226	0.75	8	225	0.99	8	225	1.09	12,730	9	224	3.90
	100	274	274	4	281	1.43	4	280	2.51	4	280	2.59	10,447	5	279	13.30
	Global	826	826	35	842	0.72	38	838	1.04	38	838	1.10	9,743	40	836	4.22
9	20	143	143	10	143	0.06	10	143	0.06	10	143	0.06	0	10	143	0.06
	40	278	278	10	278	0.06	10	278	0.06	10	278	0.06	0	10	278	0.06
	60	437	437	10	437	0.07	10	437	0.07	10	437	0.07	0	10	437	0.07
	80	577	577	10	577	0.08	10	577	0.08	10	577	0.08	0	10	577	0.08
	100	695	695	10	695	0.11	10	695	0.11	10	695	0.11	0	10	695	0.11
	Global	2,130	2,130	50	2,130	0.08	50	2,130	0.08	50	2,130	0.08	0	50	2,130	0.08
10	20	42	42	10	42	0.12	10	42	0.12	10	42	0.12	0	10	42	0.12
	40	74	74	10	74	0.11	10	74	0.11	10	74	0.11	0	10	74	0.11
	60	98	98	5	103	1.12	5	103	1.61	5	103	1.80	14,525	7	101	4.20
	80	123	123	2	131	1.78	3	130	2.80	3	130	3.23	20,638	3	130	14.48
	100	153	153	0	163	2.26	2	161	4.22	2	161	4.37	17,801	3	160	19.07
	Global	490	490	27	513	1.08	30	510	1.77	30	510	1.93	17,975	33	507	7.60
Overall		7,173	7,167	366	7,308	0.67	396	7,272	1.05	396	7,272	1.13	11,617	419	7,248	2.97

Note. CPU seconds on a digital alpha 533 MHz. Values over 10 instances. Time limit for each instance: 30 seconds.

Table 4 Two-Dimensional Bin-Packing Problem: Instances Proposed by Berkey and Wang (1987) and by Martello and Vigo (1998)

				Initial heuristics			After HBP			Phase 1			SCH			
Class	n	LB*	LB	#o	UB	T	#o	UB	T	#o	UB	T	$\lvert\mathscr{S}'\rvert$	#o	UB	T
1	20	71	71	10	71	0.06	10	71	0.06	10	71	0.06	0	10	71	0.06
	40	134	132	8	134	2.15	8	134	2.31	8	134	2.39	11,915	8	134	2.42
	60	197	197	6	201	2.07	6	201	2.77	6	201	3.10	18,062	7	200	7.26
	80	274	274	9	275	0.57	9	275	0.86	9	275	1.00	22,878	9	275	4.63
	100	317	317	5	322	2.65	7	320	3.98	7	320	4.60	41,611	10	317	5.21
	Global	993	991	38	1,003	1.50	40	1,001	2.00	40	1,001	2.23	24,379	44	997	3.91
2	20	10	10	10	10	0.06	10	10	0.06	10	10	0.06	0	10	10	0.06
	40	19	19	9	20	0.57	10	19	0.67	10	19	0.67	0	10	19	0.67
	60	25	25	10	25	0.07	10	25	0.07	10	25	0.07	0	10	25	0.07
	80	31	31	10	31	0.07	10	31	0.07	10	31	0.07	0	10	31	0.07
	100	39	39	9	40	0.58	10	39	0.79	10	39	0.79	0	10	39	0.79
	Global	124	124	48	126	0.27	50	124	0.33	50	124	0.33	0	50	124	0.33
3	20	51	51	10	51	0.07	10	51	0.07	10	51	0.07	0	10	51	0.07
	40	92	92	7	95	1.58	8	94	1.79	8	94	1.86	17,848	8	94	2.66
	60	136	136	6	140	2.10	6	140	2.82	6	140	3.13	25,958	7	139	6.21
	80	187	187	3	195	3.63	6	191	4.82	6	191	5.36	30,227	8	189	8.80
	100	221	221	3	229	3.82	6	225	5.67	6	225	6.44	53,649	8	223	12.80
	Global	687	687	29	710	2.24	36	701	3.03	36	701	3.37	33,931	41	696	6.11
4	20	10	10	10	10	0.06	10	10	0.06	10	10	0.06	0	10	10	0.06
	40	19	19	10	19	0.07	10	19	0.07	10	19	0.07	0	10	19	0.07
	60	23	23	8	25	1.11	8	25	3.71	8	25	4.53	18,570	8	25	6.15
	80	30	30	7	33	1.64	8	32	5.54	8	32	6.11	18,294	8	32	10.35
	100	37	37	9	38	0.66	9	38	2.63	9	38	2.63	9,539	9	38	4.72
	Global	119	119	44	125	0.71	45	124	2.40	45	124	2.68	16,653	45	124	4.27
5	20	65	65	10	65	0.06	10	65	0.06	10	65	0.06	0	10	65	0.06
	40	119	117	8	119	1.70	8	119	1.88	8	119	1.96	8,362	8	119	1.98
	60	179	179	8	181	1.48	9	180	1.78	9	180	1.86	20,029	9	180	1.93
	80	241	241	3	249	3.67	4	247	5.51	4	247	6.38	22,550	4	247	20.66
	100	279	279	3	287	3.76	4	286	6.51	4	286	7.77	35,414	7	282	18.50
	Global	883	881	32	901	2.13	35	897	3.15	35	897	3.61	25,636	38	893	8.63
6	20	10	10	10	10	0.06	10	10	0.06	10	10	0.06	0	10	10	0.06
	40	15	15	6	19	2.13	8	17	6.85	8	17	6.85	2,873	8	17	6.85
	60	21	21	9	22	0.66	10	21	0.66	10	21	0.66	0	10	21	0.66
	80	30	30	10	30	0.23	10	30	0.23	10	30	0.23	0	10	30	0.23
	100	32	32	8	34	1.42	8	34	5.29	8	34	5.29	4,639	8	34	6.29
	Global	108	108	43	115	0.90	46	112	2.62	46	112	2.62	3,756	46	112	2.82
7	20	55	55	10	55	0.13	10	55	0.13	10	55	0.13	0	10	55	0.13
	40	109	109	6	113	2.49	8	111	2.70	8	111	2.78	10,616	8	111	3.02
	60	156	156	5	161	2.61	6	160	3.36	6	160	3.70	19,503	8	158	8.85
	80	224	224	1	233	4.66	2	232	7.01	2	232	8.19	26,537	2	232	54.79
	100	269	269	4	276	3.22	6	273	5.10	6	273	6.02	36,527	8	271	25.06
	Global	813	813	26	838	2.62	32	831	3.66	32	831	4.16	25,425	36	827	18.37
8	20	58	58	10	58	0.06	10	58	0.06	10	58	0.06	0	10	58	0.06
	40	112	112	9	113	0.58	9	113	0.66	9	113	0.70	7,242	9	113	0.96
	60	159	159	5	164	2.60	7	162	3.17	7	162	3.38	15,743	7	162	9.05
	80	223	223	7	226	1.65	8	225	2.25	8	225	2.49	25,937	9	224	11.60
	100	274	274	4	281	3.23	4	280	5.92	4	280	7.04	32,531	5	279	47.13
	Global	826	826	35	842	1.62	38	838	2.41	38	838	2.73	25,128	40	836	13.76
9	20	143	143	10	143	0.06	10	143	0.06	10	143	0.06	0	10	143	0.06
	40	278	278	10	278	0.07	10	278	0.07	10	278	0.07	0	10	278	0.07
	60	437	437	10	437	0.07	10	437	0.07	10	437	0.07	0	10	437	0.07
	80	577	577	10	577	0.08	10	577	0.08	10	577	0.08	0	10	577	0.08
	100	695	695	10	695	0.11	10	695	0.11	10	695	0.11	0	10	695	0.11
	Global	2,130	2,130	50	2,130	0.08	50	2,130	0.08	50	2,130	0.08	0	50	2,130	0.08
10	20	42	42	10	42	0.12	10	42	0.12	10	42	0.12	0	10	42	0.12
	40	74	74	10	74	0.11	10	74	0.11	10	74	0.11	0	10	74	0.11
	60	98	98	5	103	2.62	6	102	3.70	6	102	4.08	33,450	7	101	8.89
	80	123	123	2	131	4.18	3	130	6.75	3	130	7.82	46,068	5	128	38.26
	100	153	153	0	163	5.26	2	161	10.17	2	161	12.14	58,105	4	159	55.77
	Global	490	490	27	513	2.46	31	509	4.17	31	509	4.86	48,480	36	504	20.63
Overall		7,173	7,169	372	7,303	1.46	403	7,267	2.39	403	7,267	2.67	29,711	426	7,243	7.90

Note. CPU seconds on a digital alpha 533 MHz. Values over 10 instances. Time limit for each instance: 100 seconds.

Table 5 Two-Dimensional Bin-Packing Problem: Instances Proposed by Berkey and Wang (1987) and by Martello and Vigo (1998)

			SCH			Exact algorithm			Tabu search			GLS	HBP			HBP(*TL*)		
Class	*n*	LB*	#*o**	UB	*T*	#*o**	UB	*T*	#*o**	UB	*T*	UB	#*o**	UB	*T*	#*o**	UB	*T*
1	20	71	10	71	0.07	10	71	0.01	10	71	24.00	71	10	71	0.02	10	71	3.08
	40	134	10	134	3.93	10	134	4.62	9	135	36.11	134	10	134	0.19	10	134	9.68
	60	197	7	200	2.50	6	201	21.01	6	201	48.93	201	6	201	0.53	6	201	12.12
	80	274	9	275	2.50	9	275	15.01	2	282	48.17	275	9	275	0.22	9	275	3.08
	100	317	10	317	3.38	5	322	24.07	2	326	60.81	321	9	318	0.83	8	319	6.33
	Global	993	46	997	2.48	40	1,003	12.94	29	1,015	43.60	1,002	44	999	0.36	43	1,000	6.86
2	20	10	10	10	0.06	10	10	0.00	10	10	0.01	10	10	10	0.00	10	10	0.06
	40	19	10	19	0.31	9	20	3.00	9	20	0.01	19	10	19	0.30	10	19	0.46
	60	25	10	25	0.07	8	27	6.00	8	27	0.09	25	10	25	0.07	10	25	0.07
	80	31	10	31	0.07	7	34	9.00	8	33	12.00	32	10	31	0.85	10	31	0.48
	100	39	10	39	0.37	9	40	3.00	9	40	6.00	39	10	39	0.14	10	39	0.26
	Global	124	50	124	0.18	43	131	4.20	44	130	3.62	125	50	124	0.27	50	124	0.26
3	20	51	10	51	0.07	10	51	0.01	6	55	54.00	51	10	51	0.08	10	51	6.28
	40	92	8	94	1.10	7	95	12.01	5	97	54.02	95	7	95	0.39	8	94	6.48
	60	136	7	139	2.66	6	140	15.04	6	140	45.67	140	6	140	0.94	6	140	12.14
	80	187	7	190	6.09	3	195	24.01	2	198	54.31	193	6	191	1.75	7	190	9.93
	100	221	8	223	5.10	3	228	27.70	1	236	60.10	229	5	226	3.05	6	225	12.59
	Global	687	40	697	3.00	29	709	15.75	20	726	53.62	708	34	703	1.24	37	700	9.48
4	20	10	10	10	0.06	10	10	0.00	10	10	0.01	10	10	10	0.00	10	10	0.07
	40	19	10	19	0.07	9	20	3.00	10	19	0.01	19	10	19	0.02	10	19	0.08
	60	23	8	25	1.86	6	27	12.00	7	26	0.14	25	8	25	2.67	8	25	6.13
	80	30	8	32	3.15	7	33	9.00	7	33	18.00	33	7	33	5.70	8	32	7.10
	100	37	9	38	2.32	7	40	9.00	9	38	6.00	39	9	38	4.17	9	38	4.04
	Global	119	45	124	1.49	39	130	6.60	43	126	4.83	126	44	125	2.51	45	124	3.48
5	20	65	10	65	0.06	10	65	0.01	9	66	36.02	65	10	65	0.01	10	65	0.10
	40	119	10	119	1.21	10	119	5.39	10	119	27.07	119	10	119	0.75	10	119	9.31
	60	179	9	180	1.05	9	180	15.19	7	182	56.77	181	9	180	1.37	9	180	8.21
	80	241	4	247	8.82	3	249	27.00	2	251	56.18	250	4	248	4.63	4	248	18.80
	100	279	6	283	5.30	4	286	27.01	0	295	60.34	288	3	287	8.42	4	286	18.50
	Global	883	39	894	3.29	36	899	14.92	28	913	47.28	903	36	899	3.04	37	898	10.99
6	20	10	10	10	0.07	10	10	0.00	10	10	0.01	10	10	10	0.00	10	10	0.07
	40	15	8	17	2.81	6	19	12.00	6	19	0.03	18	7	18	6.31	8	17	6.91
	60	21	10	21	0.35	9	22	3.01	9	22	0.04	22	10	21	1.50	10	21	0.16
	80	30	10	30	0.23	10	30	0.01	10	30	0.01	30	10	30	0.19	10	30	0.24
	100	32	8	34	2.75	7	35	9.01	8	34	12.00	34	8	34	11.41	8	34	6.39
	Global	108	46	112	1.24	42	116	4.81	43	115	2.41	114	45	113	3.88	46	112	2.75
7	20	55	10	55	0.12	10	55	0.06	10	55	12.02	55	10	55	0.06	10	55	6.09
	40	109	8	111	1.41	8	111	11.58	5	114	37.01	113	8	111	0.73	7	112	10.25
	60	156	8	158	3.50	4	162	18.00	4	162	36.44	161	6	160	2.03	6	160	13.09
	80	224	2	232	15.71	0	234	30.00	2	232	54.52	233	2	232	6.70	2	232	24.24
	100	269	8	271	9.86	4	276	21.00	4	277	47.43	276	5	274	7.32	6	273	13.01
	Global	813	36	827	6.12	26	838	16.13	25	840	37.48	838	31	832	3.37	31	832	13.34
8	20	58	10	58	0.06	10	58	0.00	10	58	18.04	58	10	58	0.01	10	58	0.07
	40	112	9	113	0.49	9	113	6.00	8	114	18.72	114	9	113	0.26	9	113	3.47
	60	159	7	162	3.36	5	164	15.00	7	162	20.99	163	7	162	1.72	7	162	9.33
	80	223	9	224	3.90	7	226	12.01	7	226	37.95	228	7	226	3.03	8	225	6.36
	100	274	5	279	13.30	4	281	21.00	2	284	52.66	282	4	280	8.59	5	279	15.48
	Global	826	40	836	4.22	35	842	10.80	34	844	29.67	845	37	839	2.72	39	837	6.94
9	20	143	10	143	0.06	10	143	0.00	10	143	0.01	143	10	143	0.01	10	143	0.06
	40	278	10	278	0.06	10	278	0.01	10	278	24.05	278	10	278	0.01	10	278	0.07
	60	437	10	437	0.07	10	437	0.12	9	438	24.26	437	10	437	0.05	10	437	0.08
	80	577	10	577	0.08	10	577	3.51	10	577	54.31	577	10	577	0.10	10	577	0.08
	100	695	10	695	0.11	10	695	12.80	10	695	34.11	695	10	695	0.17	10	695	0.11
	Global	2,130	50	2,130	0.08	50	2,130	3.29	49	2,131	27.35	2,130	50	2,130	0.07	50	2,130	0.08
10	20	42	10	42	0.12	10	42	0.05	9	43	12.00	42	9	43	0.19	10	42	4.78
	40	74	10	74	0.11	10	74	2.28	9	75	25.18	74	10	74	0.41	10	74	6.11
	60	98	7	101	4.20	5	103	16.86	4	104	42.13	102	6	102	2.49	6	102	16.13
	80	123	3	130	14.48	1	132	28.29	3	130	47.30	130	3	130	4.92	3	130	21.26
	100	153	3	160	19.07	0	164	30.00	0	166	60.10	163	1	162	9.72	3	160	26.63
	Global	490	33	507	7.60	26	515	15.50	25	518	37.34	511	29	511	3.55	32	508	14.98
Overall		7,173	425	7,248	2.97	366	7,313	10.49	340	7,358	28.72	7,302	400	7,275	2.01	410	7,265	6.92

Note. Values over 10 instances. Time limit for each instance: 30 seconds.

Table 6 Two-Dimensional Bin-Packing Problem: Instances Proposed by Berkey and Wang (1987) and by Martello and Vigo (1998)

			SCH			Exact algorithm			GLS		HBP(*TL*)		
Class	*n*	LB*	#*o**	UB	*T*	#*o**	UB	*T*	#*o**	UB	#*o**	UB	*T*
1	20	71	10	71	0.06	10	71	0.00	10	71	10	71	10.09
	40	134	10	134	2.42	10	134	11.01	10	134	10	134	32.02
	60	197	7	200	7.26	6	201	70.00	6	201	6	201	40.17
	80	274	9	275	4.63	9	275	50.00	9	275	9	275	10.10
	100	317	10	317	5.21	7	320	77.96	6	321	8	319	20.79
	Global	993	46	997	3.91	42	1,001	41.80	41	1,002	43	1,000	22.64
2	20	10	10	10	0.06	10	10	0.00	10	10	10	10	0.06
	40	19	10	19	0.67	9	20	10.00	10	19	10	19	1.33
	60	25	10	25	0.07	8	27	20.00	10	25	10	25	0.07
	80	31	10	31	0.07	7	34	30.00	10	31	10	31	1.35
	100	39	10	39	0.79	9	40	10.00	10	39	10	39	0.26
	Global	124	50	124	0.33	43	131	14.00	50	124	50	124	0.61
3	20	51	10	51	0.07	10	51	0.01	10	51	10	51	20.74
	40	92	8	94	2.66	7	95	40.01	8	94	8	94	21.38
	60	136	7	139	6.21	6	140	50.03	6	140	6	140	40.19
	80	187	8	189	8.80	3	195	80.00	6	191	7	190	32.72
	100	221	8	223	12.80	3	228	90.71	5	226	6	225	41.51
	Global	687	41	696	6.11	29	709	52.16	35	702	37	700	31.31
4	20	10	10	10	0.06	10	10	0.00	10	10	10	10	0.07
	40	19	10	19	0.07	9	20	10.00	10	19	10	19	0.08
	60	23	8	25	6.15	6	27	40.00	8	25	8	25	20.15
	80	30	8	32	10.35	7	33	30.00	7	33	8	32	21.67
	100	37	9	38	4.72	7	40	30.00	9	38	9	38	12.02
	Global	119	45	124	4.27	39	130	22.00	44	125	45	124	10.80
5	20	65	10	65	0.06	10	65	0.01	10	65	10	65	0.10
	40	119	10	119	1.98	10	119	9.66	10	119	10	119	30.78
	60	179	9	180	1.93	9	180	43.70	8	181	9	180	27.07
	80	241	4	247	20.66	3	249	90.00	3	249	4	248	62.19
	100	279	7	282	18.50	4	286	90.00	2	288	4	286	61.03
	Global	883	40	893	8.63	36	899	46.68	33	902	37	898	36.23
6	20	10	10	10	0.06	10	10	0.00	10	10	10	10	0.07
	40	15	8	17	6.85	6	19	40.00	7	18	8	17	22.69
	60	21	10	21	0.66	9	22	10.00	9	22	10	21	0.16
	80	30	10	30	0.23	10	30	0.00	10	30	10	30	0.23
	100	32	8	34	6.29	7	35	30.01	8	34	8	34	20.42
	Global	108	46	112	2.82	42	116	16.01	44	114	46	112	8.71
7	20	55	10	55	0.13	10	55	0.06	10	55	10	55	20.12
	40	109	8	111	3.02	8	111	32.77	6	113	7	112	33.56
	60	156	8	158	8.85	4	162	60.00	7	159	6	160	43.33
	80	224	2	232	54.79	0	234	100.00	2	232	2	232	80.35
	100	269	8	271	25.06	4	276	70.00	4	275	6	273	42.82
	Global	813	36	827	18.37	26	838	52.57	29	834	31	832	44.03
8	20	58	10	58	0.06	10	58	0.00	10	58	10	58	0.07
	40	112	9	113	0.96	9	113	20.00	8	114	9	113	11.36
	60	159	7	162	9.05	5	164	50.00	6	163	7	162	30.81
	80	223	9	224	11.60	7	226	40.00	8	225	8	225	20.83
	100	274	5	279	47.13	4	281	70.00	3	281	5	279	50.98
	Global	826	40	836	13.76	35	842	36.00	35	841	39	837	22.81
9	20	143	10	143	0.06	10	143	0.00	10	143	10	143	0.06
	40	278	10	278	0.07	10	278	0.00	10	278	10	278	0.07
	60	437	10	437	0.07	10	437	0.13	10	437	10	437	0.07
	80	577	10	577	0.08	10	577	4.88	10	577	10	577	0.09
	100	695	10	695	0.11	10	695	37.85	10	695	10	695	0.11
	Global	2,130	50	2,130	0.08	50	2,130	8.58	50	2,130	50	2,130	0.08
10	20	42	10	42	0.12	10	42	0.05	10	42	10	42	15.73
	40	74	10	74	0.11	10	74	2.62	10	74	10	74	20.14
	60	98	7	101	8.89	5	103	52.21	6	102	6	102	53.39
	80	123	5	128	38.26	1	132	91.49	3	130	3	130	70.35
	100	153	4	159	55.77	1	162	94.64	1	162	3	160	88.00
	Global	490	36	504	20.63	27	513	48.21	30	510	32	508	49.52
Overall		7,173	430	7,243	7.90	369	7,309	33.80	391	7,284	410	7,265	22.67

Note. Values over 10 instances. Time limit for each instance: 100 seconds.

Table 7 Two-Dimensional Bin-Packing Problem: Other Instances from the Literature

				SCH			Exact algorithm			Tabu search			GLS		HBP(*TL*)		
Class	*n*	# inst.	LB*	#*o**	UB	*T*	#*o**	UB	*T*	#*o**	UB	*T*	#*o**	UB	#*o**	UB	*T*
cgcut	16-62	3	27	3	27	0.07	3	27	0.00	3	27	0.01	3	27	3	27	0.09
gcut	10-50	13	104	13	104	0.51	12	105	4.64	9	108	67.22	13	104	13	104	7.33
ngcut	7-22	12	32	12	32	0.13	12	32	0.06	8	36	58.35	12	32	12	32	7.59
beng	20-120	8	54	7	55	1.31	7	55	4.46	6	56	25.02	—	—	7	55	3.83

Note. Time limit for each instance: 30 seconds.

there are very marginal improvements with higher time limits; thus, we report only the results obtained with that time limit (corresponding to $TL = 30$ seconds), disregarding tabu-search in the experiments with $TL = 100$ seconds. Note also that the times in column T associated with tabu-search (Table 5) should be halved in order to be compared with the other time values. As for what affects the guided-local-search by Færø et al. (2003), we used the results reported in the paper; these results were obtained on a Digital 500au workstation with a 500 MHz 21164 CPU (having 15.7 SPECint95 value, i.e., a speed almost equal to that of our machine) with the same time limits we used, thus allowing us to obtain an immediate comparison of the algorithms. The number of known optimal solutions found by the guided-local-search algorithm is given only for the experiments with $TL = 100$ seconds (see Table 6). Indeed, the results for each instance are not available for the experiments with $TL =$ 30 seconds, so we omitted in Table 5 the corresponding number of instances solved to proven optimality. Finally, for algorithm HBP, we give in Table 5 both the results reported in Boschetti and Mingozzi (2003b) and those of our implementation, while in Table 6 only the latter are given. Note that the results reported by the authors were obtained on a Pentium III Intel 933 MHz, which is slightly faster than our machine.

Tables 5 and 6 show that algorithm SCH outperforms, for both time limits, the best exact algorithm in the literature and the tabu-search algorithm proposed by Lodi et al. (1999a). As for the other two algorithms, HBP performs better than GLS for both time limits. When considering $TL = 30$ seconds, the gap for the latter is 129, while for the former it is 102 when at most 100 iterations are performed (column "HBP") and 92 when 30 seconds are given (column "HBP(*TL*)"). As mentioned before, the gap of algorithm SCH with time limit $TL = 30$ seconds is 75, with a saving of 17 bins with respect to algorithm HBP(*TL*). When considering $TL = 100$ seconds, the gap of algorithm GLS is reduced to 111, that of algorithm HBP(*TL*) remains equal to 92, while the gap of algorithm SCH is 70, with a saving of 22 bins with respect to HBP(*TL*).

As for the average computing times, that of algorithm SCH is comparable to that of HBP, and considerably smaller than those of the exact algorithm and of algorithms tabu-search and HBP(*TL*). The average computing times of algorithm GLS are not explicitly reported in Færø et al. (2003). However, a rough estimation, based on the number of instances not solved to proven optimality (hence requiring a computing time equal to the corresponding time limit), shows that the average computing times of GLS are at least four times larger than those of SCH.

Finally, we consider the other instances proposed in the literature: instances `cgcut`, proposed by Christofides and Whitlock (1977), `gcut` and `ngcut`, proposed by Beasley (1985a, b) (all available in the ORLIB library, web site `http://www.ms.ic.ac.uk/info.html`, see Beasley 1990), and `beng`, proposed by Bengtsson (1982). Table 7 reports the corresponding results for algorithm SCH, for the exact algorithm by Martello and Vigo (2001), and for algorithms tabu-search, GLS, and HBP(*TL*), with time limit $TL = 30$ seconds. The results of algorithm tabu-search are those reported in Lodi et al. (1999b), and have been obtained by imposing a time limit of 100 seconds (i.e., about 50 seconds on our machine). As for algorithm GLS, no results on instances of class `beng` are reported in Færø et al. (2003). The table gives, for each class, the range of the corresponding values of n, the number of instances of the class (column "# inst."), the sum of the best known lower bounds, and, for each algorithm, the same information reported in Tables 5 and 6. The results with time limit $TL = 100$ seconds are omitted because no improvement of the solution values occurred with respect to the case with $TL = 30$ seconds (only one more instance of class `gcut` was solved to proven optimality by the exact algorithm). Table 7 shows that for these instances as well, algorithm SCH requires average computing times smaller than those of the other algorithms, with equal or better solution values.

Note that for all the 536 instances considered in our test bed, algorithm SCH found the best upper bound known in the literature.

Acknowledgments

The authors thank the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), and the Consiglio Nazionale delle Ricerche (CNR), Italy, for having partially supported this project. They are grateful to Silvano Martello and Daniele Vigo for having provided them with the codes of their exact algorithm for 2DBP. Thanks also are due to Andrea Lodi, Silvano Martello, Daniele Vigo, Oluf Færø, David Pisinger, Martin Zachariasen, Marco A. Boschetti, and Aristide Mingozzi for having provided the authors the results, for each 2DBP instance, of their algorithms. Finally, they are also grateful to Terenzio Berni and Luca Fabbri for their help in implementing the algorithms and carrying out some preliminary computational tests. The computational experiments were executed at the Laboratory of Operations Research of the University of Bologna (LabOR).

References

Baker, B. S., E. G. Coffman Jr., R. L. Rivest. 1980. Orthogonal packing in two dimensions. *SIAM J. Comput.* **9** 846–855.

Beasley, J. E. 1985a. Algorithms for unconstrained two-dimensional guillotine cutting. *J. Oper. Res. Soc.* **36** 297–306.

Beasley, J. E. 1985b. An exact two-dimensional non-guillotine cutting tree search procedure. *Oper. Res.* **33** 49–64.

Beasley, J. E. 1990. OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** 1069–1072.

Bengtsson, B. E. 1982. Packing rectangular pieces—A heuristic approach. *Comput. J.* **25** 353–357.

Berkey, J. O., P. Y. Wang. 1987. Two dimensional finite bin packing algorithms. *J. Oper. Res. Soc.* **38** 423–429.

Bodin, L., B. Golden, A. Assad, M. Ball. 1983. Routing and scheduling of vehicles and crews: The state of the art. *Comp. Oper. Res.* **10** 63–211.

Boschetti, M. A., A. Mingozzi. 2003a. Two-dimensional finite bin packing problems. Part I: New lower and upper bounds. *4OR* **1** 27–42.

Boschetti, M. A., A. Mingozzi. 2003b. Two-dimensional finite bin packing problems. Part II: New lower and upper bounds. *4OR* **2** 135–147.

Caprara, A., P. Toth. 2001. Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Appl. Math.* **111** 231–262.

Caprara, A., M. Fischetti, P. Toth. 1999. A heuristic method for the set covering problem. *Oper. Res.* **47** 730–743.

Caprara, A., M. Monaci, P. Toth. 2001. A global method for crew planning in railway applications. S. Voss, J. R. Daduna, eds. *Computer-Aided Scheduling of Public Transport. Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Germany, 17–36.

Caprara, A., M. Monaci, P. Toth. 2002. Greedy procedures for the multi-constraint bin packing problem. Technical report OR/02/12, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Bologna, Italy.

Caprara, A., M. Fischetti, P. Toth, D. Vigo, P. L. Guida. 1997. Algorithms for railway crew management. *Math. Programming* **79** 125–141.

Chekuri, C., S. Khanna. 1999. On multi-dimensional packing problems. *Proc. 10th Annual ACM-SIAM Symp. Discrete Algorithms (SODA99)*, ACM Press, Baltimore, Maryland, 185–194.

Christofides, N., C. Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Oper. Res.* **25** 30–44.

Chung, F. K. R., M. R. Garey, D. S. Johnson. 1982. On packing two-dimensional bins. *SIAM J. Algebraic Discrete Methods* **3** 66–76.

Dyckhoff, H., G. Scheithauer, J. Terno. 1997. Cutting and packing (C&P). M. Dell'Amico, F. Maffioli, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization.* John Wiley and Sons, Chichester, U.K., 393–413.

Færø, O., D. Pisinger, M. Zachariasen. 2003. Guided local search for the three-dimensional bin packing problem. *INFORMS J. Comput.* **15** 267–283.

Fekete, S. P., J. Schepers. 1997. On more-dimensional packing III: Exact algorithms. Technical report ZPR/97/290, Mathematisches Institut, Universität zu Köln, Köln, Germany.

Fekete, S. P., J. Schepers. 2004a. A combinatorial characterization of high-dimensional orthogonal packing. *Math. Oper. Res.* **29** 353–368.

Fekete, S. P., J. Schepers. 2004b. A general framework for bounds for higher-dimensional orthogonal packing problems. *Math. Methods Oper. Res.* **60** 311–329.

Fernandez de la Vega, W., G. S. Lueker. 1981. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica* **1** 349–355.

Frenk, J. B., G. G. Galambos. 1987. Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing* **39** 201–217.

Garey, M. R., R. L. Graham, D. S. Johnson, A. C. Yao. 1976. Resource constrained scheduling as generalized bin packing. *J. Combin. Theory (A)* **21** 257–298.

Gilmore, P. C., R. E. Gomory. 1965. Multistage cutting problems of two and more dimensions. *Oper. Res.* **13** 94–119.

Kellerer, H., V. Kotov. 2003. An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing. *Oper. Res. Lett.* **31** 35–41.

Kelly, J. P., J. Xu. 1999. A set-partitioning-based heuristic for the vehicle routing problem. *INFORMS J. Comput.* **11** 161–172.

Lodi, A., S Martello, M. Monaci. 2002. Two-dimensional packing problems: A survey. *Eur. J. Oper. Res.* **141** 241–252.

Lodi, A., S. Martello, D. Vigo. 1999a. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS J. Comput.* **11** 345–357.

Lodi, A., S. Martello, D. Vigo. 1999b. Approximation algorithms for the oriented two-dimensional bin packing problem. *Eur. J. Oper. Res.* **112** 158–166.

Marsten, R., F. Shepardson. 1981. Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications. *Networks* **112** 167–177.

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

Martello, S., D. Vigo. 2001. New computational results for the exact solution of the two-dimensional finite bin packing problem. Technical report OR/01/06, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Bologna, Italy.

Maruyama, K., S. K. Chang, D. T. Tang. 1977. A general packing algorithm for multidimensional resource requirements. *Internat. J. Comput. Inform. Sci.* **6** 131–149.

Spieksma, F. C. R. 1994. A branch-and-bound algorithm for the two-dimensional vector packing problem. *Comput. Oper. Res.* **21** 19–25.

Valerio de Carvalho, J. M. V. 1998. Exact solution of cutting stock problems using column generation and branch-and-bound. *Internat. Trans. Oper. Res.* **5** 35–44.

Vance, P. H. 1998. Branch-and-price algorithms for the one-dimensional cutting stock problem. *Comput. Optim. Appl.* **9** 211–228.

Vance, P. H., C. Barnhart, E. L. Johnson, G. L. Nemhauser. 1994. Solving binary cutting stock problems by column generation and branch-and-bound. *Comput. Optim. Appl.* **3** 111–130.

Vanderbeck, F. 1999. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Programming* **86** 565–594.

Voudouris, C., E. Tsang. 1999. Guided local search and its application to the traveling salesman problem. *Eur. J. Oper. Res.* **113** 469–499.

Wedelin, D. 1995. An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. *Ann. Oper. Res.* **57** 283–301.

Woeginger, G. 1997. There is no asymptotic ptas for two-dimensional vector packing. *Inform. Processing Lett.* **64** 293–297.

Yao, A. C. 1980. New algorithms for bin packing. *J. ACM* **27** 207–227.