A branch-and-price algorithm for the two-dimensional level strip packing problem

Andrea Bettinelli, Alberto Ceselli, Giovanni Righini
Dipartimento di Tecnologie dell'Informazione,
Università degli Studi di Milano
Via Bramante 65, 26013 Crema, Italy [*]

December 2005

Abstract

The two-dimensional level strip packing problem (2LSPP) consists in packing rectangular items of given size into a strip of given width divided into levels. Items packed into the same level cannot be put on top of one another and their overall width cannot exceed the width of the strip. The objective is to accommodate all the items while minimizing the overall height of the strip. The problem is NP-hard and arises from applications in logistics and transportation. We present a set covering formulation of the 2LSPP suitable for a column generation approach, where each column corresponds to a feasible combination of items inserted into the same level. For the exact optimization of the 2LSPP we present a branch-and-price algorithm, in which the pricing problem is a penalized knapsack problem. Computational results are reported for benchmark instances with some hundreds items.

1 Introduction

In several industrial applications it is required to place a set of rectangular items in standard stock units. In wood and glass manufacturing, for instance, rectangular components must be cut from large pieces of material; in warehouses, the goods must be placed on shelves; in the design of newspapers layout it is needed to arrange articles and advertisements in pages of given size. These difficult combinatorial problems are often modeled as two-dimensional packing or cutting problems. A review on packing and cutting problems and methods can be found in [3], while in [4], [5] and [6] the authors propose general graph-theoretical frameworks for devising bounds on multi-dimensional packing problems.

In production contexts such as clothes or paper manufacturing, a single strip of material is available and a set of items must be obtained from the strip. The

[*] *Correspondence to*: righini@dti.unimi.it

aim is to cut the desired items, while minimizing the height of the strip to be used. This problem is called two-dimensional strip packing (2SPP). Recently a fully polynomial time approximation scheme for the 2SPP has been proposed [10]. Martello et al. [13] presented a branch-and-bound algorithm, which is able to solve 2SPP instances with up to 200 items at optimality in one hour of computing time.

In this paper we study a variation of the 2SPP in which a further restriction is imposed: the items must be organized into horizontal strips, indicated as *levels*; inside each level, the items cannot be put on top of one another. This problem, surveyed by Lodi et al. [11] and referred to as two-dimensional level strip packing problem (2LSPP), is $\mathcal{NP}$-hard in the strong sense, because it contains the bin-packing problem as a special case [7].

The 2LSPP can be approximated with fast heuristics, which provide also an a priori guarantee on the quality of the solution [1] [11]. More recently, Lodi et al. [11] proposed a formulation for the 2LSPP with a polynomial number of variables and constraints; the effectiveness of state-of-the-art general purpose ILP solvers makes this approach particularly appealing. In the survey papers [11] and [12] the authors present models and lower bounds for several packing problems, including the 2LSPP, without computational results.

In this paper we introduce a new formulation for the 2LSPP as a set covering problem. The linear relaxation of this model is optimized with column generation, and the lower bound found in this way is used in a branch-and-price algorithm. A similar approach was followed by Gilmore and Gomory [8] [9] for the cutting stock problem. In Section 2 we present a set covering reformulation of the 2LSPP and we discuss its relationship with the compact formulation of Lodi et al. [11]. In Section 3 we outline the main issues in the design of our branch-and-price algorithm. Finally, in Section 4 we report the outcome of an experimental analysis, in which we show that the branch-and-price algorithm outperforms a state-of-the-art general purpose solver.

2 Problem formulation

In the 2LSPP we are given a strip, whose width is a positive integer W, and a set $\mathcal{N}$, where each item $j \in \mathcal{N}$ has positive integer width and height, denoted by w_j and h_j respectively. The items must be packed into levels: the sum of the widths of the items in the same level cannot exceed the width of the strip. Items in the same level cannot be piled up; hence the height of each level corresponds to the maximum height of an item in that level. We call this particular item the *leading item* and we say that the leading item *initializes* the level. Throughout the paper we assume that the items are sorted by non-decreasing height values: $h_i \leq h_j$ for each $i < j$. Hence, without loss of generality, we can state that no item i can be assigned to a level initialized by item j if $i > j$.

Lodi et al. [11] proposed an integer linear programming (ILP) compact formulation for the 2LSPP, which is reported here below. Each binary variable x_{ij} indicates whether item i is assigned to a level in which j is the leading item;

therefore each binary variable x_{jj} indicates whether item j is a leading item. Because of the ordering of the items, we can fix each x_{ij} variable with $i > j$ to 0 and remove it from the model.

$$\min \sum_{j\in\mathcal{N}} h_j x_{jj} \tag{1}$$

$$\text{s.t.} \sum_{j\geq i} x_{ij} = 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{i<j} w_i x_{ij} \leq (W - w_j)x_{jj} \qquad \forall j \in \mathcal{N} \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \leq j \in \mathcal{N} \tag{4}$$

Constraints (2) impose that each item is assigned to a level. Constraints (3) impose that the sum of the widths of the items assigned to the same level does not exceed the width of the strip. The objective is to minimize the overall height of the strip, that is the sum of the heights of the leading items.

A set covering reformulation

A lower bound for the 2LSPP can be obtained from the model above by replacing the integrality conditions (4) with the inequalities $0 \leq x_{ij} \leq 1$. We sharpen this bound exploiting Dantzig-Wolfe decomposition [14]: let Ω_j be the set of levels complying with the capacity constraints (3), for each $j \in \mathcal{N}$:

$$\Omega_j = \{x_{ij} \in \Re^n_+ | \sum_{i<j} w_i x_{ij} \leq (W - w_j)x_{jj}, 0 \leq x_{ij} \leq 1 \ \forall i \in \mathcal{N}, i \leq j\}.$$

Let K_j be the set of the integer points in Ω_j and let x^k_j be the generic integer point of Ω_j. Each point x_{ij} in the convex hull of Ω_j can be expressed as a convex combination of the integer points in K_j:

$$\text{conv}(\Omega_j) = \{x_{ij} \in \Re^n_+ | x_{ij} = \sum_{k\in K_j} x^k_j z_k, \sum_{k\in K_j} z_k = 1, 0 \leq z_k \leq 1 \ \forall k \in K_j\}. \tag{5}$$

Hence, by substitution from the linear relaxation of model (1)-(4) the following relaxation of the 2LSPP is obtained:

$$\begin{aligned}
&\min \sum_{j\in\mathcal{N}} h_j \sum_{k\in K_j} x^k_j z_k \\
&\text{s.t.} \sum_{j\geq i} \sum_{k\in K_j} x^k_i z_k = 1 && \forall i \in \mathcal{N} \\
&\sum_{k\in K_j} z_k = 1 && \forall j \in \mathcal{N} \\
&0 \leq z_k \leq 1 && \forall j \in \mathcal{N}, \forall k \in K_j.
\end{aligned} \tag{6}$$

Here, all polyhedra Ω_j have been replaced by their convex hulls. Since the Ω_j polyhedra are knapsack problem polyhedra and they do not possess the integrality property, the convexification of the capacity constraints yields a lower bound that dominates that of the linear relaxation of the 2LSPP.

We further elaborate on this model. First of all we remark that each K_j contains a point representing an empty level: these points can be considered implicitly by rewriting constraints (6) as inequalities:

$$
\begin{aligned}
\min & \sum_{j\in\mathcal{N}} h_j \sum_{k\in K_j} z_k \\
\text{s.t.} & \sum_{j\geq i} \sum_{k\in K_j} x_i^k z_k = 1 && \forall i \in \mathcal{N} && (7)\\
& \sum_{k\in K_j} z_k \leq 1 && \forall j \in \mathcal{N} && (8)\\
& 0 \leq z_k \leq 1 && \forall j \in \mathcal{N}, \forall k \in K_j.
\end{aligned}
$$

Furthermore no item is chosen more than once as a leading item in optimal solutions and hence constraints (8) are redundant and can be deleted. Finally the set partitioning constraints (7) can be replaced by set covering constraints, because it is never convenient to pack an item in more than one level.

After these manipulations the resulting model is the following:

$$
\begin{aligned}
\text{MP)}\ \min & \sum_{j\in\mathcal{N}} h_j \sum_{k\in K_j} z_k && && (9)\\
\text{s.t.} & \sum_{j\geq i} \sum_{k\in K_j} x_i^k z_k \geq 1 && \forall i \in \mathcal{N} && (10)\\
& 0 \leq z_k \leq 1 && \forall j \in \mathcal{N},\ \forall k \in K_j && (11)
\end{aligned}
$$

In this linear master problem (MP) the column corresponding to each variable z_k with $k \in K_j$ represents a feasible set of items packed into a same level initialized by item j. An ILP formulation of the 2LSPP, alternative to (1)-(4), is obtained by restoring the integrality conditions $z_k \in \{0, 1\}$ in the model above.

The pricing problem

Model (9) – (11) may have a huge number of columns. Therefore a restricted master problem (RMP) involving a subset of the variables is considered and columns not included in the RMP are iteratively generated when needed.

Let λ be the vector of the non-negative dual variables associated with covering constraints (10) in a RMP optimal solution. The pricing problem we need to solve to identify new columns is the following: $\pi(\lambda) = \min_{j\in\mathcal{N}}\{\pi_j(\lambda)\}$, where for each $j \in \mathcal{N}$

$$
\begin{aligned}
\pi_j(\lambda) = \min\ & h_j x_j - \sum_{i\leq j} \lambda_i x_i \\
\text{s.t.} & \sum_{i<j} w_i x_i \leq (W - w_j) x_j \\
& x_i \in \{0, 1\} && \forall i \leq j.
\end{aligned}
$$

Thus a negative reduced cost column can be generated by solving at most $|\mathcal{N}|$ binary knapsack problems, obtained by setting to 1 one x_j variable at a time.

However solving a large number of knapsack problems to optimality to generate negative reduced cost columns can be unnecessary, since we just need one negative reduced cost column, provided it exists. Therefore we solve a pricing problem in which the leading item is not fixed, but rather it must be chosen in an optimal way, that is we search for the column of minimum reduced cost for all possible choices of the leading item. The pricing problem can be rewritten in an equivalent way as follows:

$$\pi(\lambda) = \min \eta - \sum_{i \in \mathcal{N}} \lambda_i x_i \tag{12}$$

$$\text{s.t.} \sum_{i \in \mathcal{N}} w_i x_i \leq W \tag{13}$$

$$h_i x_i \leq \eta \qquad \forall i \in \mathcal{N} \tag{14}$$

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{N}.$$

Each binary variable x_i is equal to 1 if and only if item i is assigned to the level represented by the new column. The free variable η is a penalty term. The value of η is determined by the height of the leading item of the level. The capacity constraint (13) imposes that the overall width of the level does not exceed the width of the strip.

The objective function (12) can be rewritten in maximization form:

$$\pi'(\lambda) = \max \sum_{i \in \mathcal{N}} \lambda_i x_i - \eta.$$

This pricing problem can be solved with special purpose algorithms for the penalized knapsack problem (PKP) illustrated in Ceselli and Righini [2].

3 Branch-and-price

Branching strategy

We base our branching rule on the x variables of the compact ILP formulation (1)-(4): once an optimal MP solution z^* is obtained, a corresponding (fractional) solution x^* in terms of the original variables can be found exploiting the relation $x^*_{ij} = \sum_{k \in K_j} x^k_{ij} z^*_k$ for each $i, j \in \mathcal{N}$, where x^k_{ij} is the i^{th} component of the integer vector $x^k_j \in K_j$.

We have adopted a two-stage branching strategy: in the first stage search-tree the branching decisions are taken on the x_{jj} variables, i.e. the leading items are chosen; in the second stage search-tree feasibility problems are solved: the non-leading items must be packed into the levels initialized by the leading items selected in the first stage, without violating width and height constraints. In both stages branching is done on the x variable whose value is closest to 0.5 and the branching variable is fixed to 0 in one branch and to 1 in the other branch;

x_{jj} variables are considered in the first stage and x_{ij} variables with $i \neq j$ are considered in the second stage.

These variable fixing operations slightly change the structure of the pricing problem. In the first stage, each time a x_{jj} variable is fixed to 1, j is discarded from the set of items in the PKP optimization, and an additional KP is solved, to compute the best solution in which j is the leading item; when a x_{jj} variable is fixed to 0, it is simply discarded from the set of candidate leading items in the PKP. In the second stage the pricing problem is a KP for each level. Therefore fixing x_{ij} variables only reduces the dimension of these KP instances.

The search trees are explored in a best-bound-first order.

Initialization

In order to obtain an initial set of columns to populate the RMP, we used the well known Best-Fit Decreasing-Height (BFDH) heuristic [11]. The items are iteratively considered from item $|\mathcal{N}|$ down to item 1 and in each iteration the current item is packed into the level with the minimum residual capacity among those that can accommodate it. If an item cannot be accommodated in this way, a new level is initialized. We implemented a simple randomized version of this heuristic (r-BFDH): a preprocessing step is added, in which r items are randomly drawn from a uniform probability distribution and the corresponding levels are initialized.

Besides running the original version of BFDH once, three r-BFDH solutions are computed for each value of r from 1 to $\lceil \sum_{i \in \mathcal{N}} w_i / W \rceil$, that is the number of levels in a fractional solution rounded up (this is a trivial lower bound on the number of levels of an optimal solution). The best solution value found in this way is also kept as an initial upper bound.

Upper bounds

We experimentally observed that the r-BFDH heuristic often provides tight bounds. Nevertheless, we incorporate a fast heuristic rounding algorithm for the set covering problem, in order to search for good integer solutions during the exploration of the search-tree. The heuristic rounding algorithm works as follows: initially, all the items are uncovered, and the columns of the RMP are sorted by non-increasing value of the corresponding z_k variables; following this order, each column k is considered: if column k represents a level containing uncovered items, the corresponding z_k variable is rounded up to 1 and each item in k is marked as covered, otherwise the z_k variable is fixed to 0.

We run this heuristic once for each node of the search tree, when the column generation process is over.

Problem reduction

Consider a generic node $\mathcal{P}$ of the first stage search-tree; let $\mathcal{N}(\mathcal{P})$ be the set of already selected leading items, $v(\mathcal{P})$ be the sum of their heights, and UB be the value of the best incumbent integer solution. For each item $j \in \mathcal{N} \setminus \mathcal{N}(\mathcal{P})$, if $v(\mathcal{P}) + h_j \geq UB$, then j can be discarded from the set of candidate leading items in node $\mathcal{P}$.

Columns deletion and re-insertion

Each time a node of the search-tree is considered, the columns in the RMP with a reduced cost higher than a given threshold are moved into a separate pool. The reduced cost of each column is computed with respect to the optimal dual solution of the ancestor node. In our implementation, the removal threshold is computed as the difference between the best known upper and lower bounds, divided by $\lceil \sum_{i \in \mathcal{N}} w_i / W \rceil$.

The columns pool is scanned at each column generation iteration: whenever a column is found to have a negative reduced cost with respect to the current dual solution, it is re-inserted into the RMP. Each column is kept into the pool for up to 6 consecutive unsuccessful checks; then it is erased.

Lagrangean bounds

The bound obtained by optimizing the master problem can also be obtained by solving a Lagrangean dual problem when the set of constraints (2) is relaxed:

$$\begin{aligned} \max_{\lambda \geq 0} \ \omega(\lambda) = \min & \sum_{j \in \mathcal{N}} h_j x_{jj} - \sum_{i \in \mathcal{N}} \lambda_i (\sum_{j \geq i} x_{ij} - 1) & \\ \text{s.t.} & \sum_{i<j} w_i x_{ij} \leq (W - w_j) x_{jj} & \forall j \in \mathcal{N} \\ & x_{ij} \in \{0,1\}. & \forall i \leq j \in \mathcal{N} \end{aligned}$$

For each set of multipliers λ this problem is analogous to the pricing problem for the set covering formulation of the 2LSPP. In fact, it decomposes into independent subproblems, one for each $j \in \mathcal{N}$:

$$\begin{aligned} \min \omega_j(\lambda) = & h_j x_{jj} - \sum_{i \leq j} \lambda_i x_{ij} & \\ \text{s.t.} & \sum_{i<j} w_i x_{ij} \leq (W - w_j) x_{jj} & \forall j \in \mathcal{N} \\ & x_{ij} \in \{0,1\} & \forall i \leq j \in \mathcal{N}. \end{aligned}$$

Therefore, each subproblem j can be optimized by considering two cases: if the variable x_{jj} is fixed to 1, then the remaining problem is a binary knapsack; this is solved to optimality obtaining a value $\pi_j(\lambda)$. If the variable x_{jj} is fixed to 0, then each variable x_{ij} with $i < j$ must be set to 0; this yields a solution of null value. Hence, for any choice of the λ multipliers, a valid lower bound $\omega(\lambda)$ for the 2LSPP is given by

$$\omega(\lambda) = \sum_{i \in \mathcal{N}} \lambda_i + \sum_{j \in \mathcal{N}} \min\{\pi_j(\lambda), 0\}.$$

However a key property of our pricing routine is actually to implicitly consider these π_j values to avoid the optimization of a large number of knapsack problems. In fact, the one with minimum value is computed by solving a PKP.

Therefore, a lower bound $\underline{\omega}(\lambda)$ on $\omega(\lambda)$ can be obtained by replacing each $\pi_j(\lambda)$ value with a corresponding lower bound $\underline{\pi}_j(\lambda)$.

$$\underline{\omega}(\lambda) = \sum_{i \in \mathcal{N}} \lambda_i + \sum_{j \in \mathcal{N}} \min\{\underline{\pi}_j(\lambda), 0\}.$$

We initially approximate each $\underline{\pi}_j(\lambda)$ with the value of the linear relaxation of the corresponding subproblem. These values are readily available, since they are computed in a preprocessing step by the algorithm for the PKP. Furthermore, whenever a tighter bound is computed during the optimization of the PKP, the corresponding value $\underline{\pi}_j(\lambda)$ is updated and the bound $\underline{\omega}(\lambda)$ is tightened.

Whenever, during the column generation iterations, the difference between the highest $\underline{\omega}(\lambda)$ encountered and the RMP optimal value is less than 10^{-6}, the column generation process is terminated, and the Lagrangean bound is kept as the final lower bound.

Variable fixing

We use the $\underline{\pi}_j(\lambda)$ values to fix variables. Once these values have been computed, the following reduction tests can be checked in linear time: let UB be the value of the incumbent integer solution:

- for each j such that $\underline{\pi}_j(\lambda) < 0$, if $\lceil \underline{\omega}(\lambda) - \underline{\pi}_j(\lambda) \rceil \geq UB$ then j can be fixed as a leading item (i.e. x_{jj} is fixed to 1);
- for each j such that $\underline{\pi}_j(\lambda) > 0$, if $\lceil \underline{\omega}(\lambda) + \underline{\pi}_j(\lambda) \rceil \geq UB$ then j can be discarded from the set of candidate leading items (x_{jj} is fixed to 0).

Combinatorial bound

Finally, we incorporated in our bounding procedure a combinatorial lower bound (called CUT in the remainder) proposed by Lodi et al. [12]. It consists in splitting each item in vertical strips of unit width and in filling the levels by considering these strips in order of non-increasing height. This bound dominates that given by the LP relaxation of the compact formulation (1)-(4), but no dominance relation exists with the set covering LP bound. Since we are assuming that items have been sorted in a preprocessing step, this bound can be computed in linear time.

The CUT bound is computed for each node of the search-tree before the column generation process is started. Whenever the value of an RMP optimal solution is found to be less than the value of the CUT bound, the column generation process is halted, and the CUT bound is kept as the lower bound.

4 Computational results

Our branch-and-price algorithm was implemented in C++ and compiled with a GNU C/C++ compiler version 3.2.2. We solved the restricted linear master problem with the CPLEX 8.1 implementation of the primal simplex algorithm. All our experiments were run on a Linux workstation equipped with a Pentium

IV 1.6 GHz processor and 512 MB of RAM. A time limit of 1 hour was imposed to each test. Furthermore, the program was halted whenever the computation exceeded the amount of physical memory.

In order to assess the effectiveness of our method, we considered two data-sets for two-dimensional packing problems widely used in the literature; they are both described in [12]. The first one consists of 5 classes of instances: BENG (10 instances), CGCUT (3 instances), GCUT (4 instances), HT (9 instances) and NGCUT (12 instances). The second data-set consists of 500 instances, divided into 10 classes of 50 instances, named MV and BW. They contain instances involving up to 200 items with different types of correlation between height and width of the items.

Lower bounds

First we compared three different lower bounds, namely the lower bound given by the linear relaxation of the set covering formulation (9)-(11), indicated hereafter with SC bound, the lower bound given by the linear relaxation of the compact formulation (1)-(4), indicated with LP bound, and the CUT lower bound. As a measure of the duality gap we considered, for each instance, the ratio $(UB - LB)/UB$, where UB is the value of the BFDH heuristic solution and LB is the value of the lower bound considered. In Tables 0(a) and 0(b) we report the average values of the gap for the instances in each class of the first and the second data-set respectively. Each row of these tables corresponds to a class of instances.

The LP bound is weaker than the CUT bound also from an experimental point of view. For the instances of the first data-set, CUT is on the average the tightest bound, while for the instances of the second data-set the SC bound is clearly superior (see, for instance, classes BW03 and BW05). On the other hand, the computation of the SC bound is two orders of magnitude slower than that of the CUT bound.

Finally it is worth noting that the CUT and SC bounds seem to be complementary, since they are tighter for different classes of instances. This observation was one of the motivations for including the computation of both bounds in a unique lower bounding routine to solve the 2LSPP to optimality.

Solving the 2LSPP to proven optimality

We compared the performance of our branch-and-price algorithm with that of CPLEX 8.1, used as a general purpose ILP solver to optimize the compact model (1)–(4). Tables 1(a) and 1(b) contain the results for the first and the second data-set respectively. Each entry of the table represents an average value of the instances in a class, and the class identifiers are indicated in the first column. Each table is made by two blocks; each of them corresponds to the solution method indicated in the first row. In each block we report the number of instances in each class that were solved to proven optimality (column "solved inst."), the average gap between the value of the incumbent primal solution (UB) and the lower bound (LB), defined as $(UB - LB)/UB$, for the instances in which optimality was not proven (column "avg. gap") and the average computing time for the instances that were closed (column "time"). In the last row of each table we report the total number of instances solved to proven optimality.

Branch-and-price solves all the instances in the first data-set, while CPLEX leaves a large gap on 7 of the 10 BENG instances. Moreover, branch-and-price is on the average much faster on the remaining classes. Branch-and-price performs much better than CPLEX also on the instances of the second data-set, solving more problems and consistently requiring less computing time or yielding tighter approximations.

Acknowledgements
This paper is based on the first author's degree thesis, which was awarded the "Camerini-Carraresi" award from the Italian O.R. Society (AIRO) in 2005. The authors acknowledge the kind support of ACSU - Associazione Cremasca Studi Universitari to the "OptLab", where this research has been done.

References

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

[2] A. Ceselli and G. Righini. An optimization algorithm for a penalized knapsack problem. *Operations Research Letters*, in press, available online, 2005.

[3] H. Dickhoff, G. Scheithauer, and J. Terno. *Cutting and packing*, pages 393–413. Wiley, New York, 1997.

[4] S. P. Fekete and J. Schepers. On higher-dimensional packing iii: Exact algorithms. Technical Report 97–290, 1997.

[5] S. P. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29:353–368, 2004.

[6] S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2):311–329, 2004.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

[8] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[9] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem - part ii. *Operations Research*, 11:863–888, 1963.

[10] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25:645–656, 2000.

[11] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 141:241–252, 2002.

[12] A. Lodi, S. Martello, and D. Vigo. Models and bounds for two dimensional packing problems. *Journal of Combinatorial Optimization*, 8:363 – 379, 2004.

[13] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15:310–319, 2003.

[14] R.K. Martin. *Large scale linear and integer optimization.* Kluwer academic, 1998.

(a)

Class	LP bound	CUT bound	SC bound
BENG	6.75%	0.47%	4.69%
GCUT	14.91%	10.57%	0.14%
NGCUT	12.57%	4.21%	5.10%
CGCUT	4.66%	4.66%	6.95%
HT	7.80%	0.40%	4.09%

(b)

Class	LP bound	CUT bound	SC bound
MV 01	8.73%	6.37%	2.29%
MV 02	7.80%	1.00%	5.46%
MV 03	11.95%	8.97%	2.96%
MV 04	7.99%	1.55%	3.80%
BW 01	11.90%	9.40%	2.19%
BW 02	8.68%	1.79%	3.78%
BW 03	14.57%	12.17%	0.69%
BW 04	8.61%	5.42%	4.53%
BW 05	19.18%	17.77%	0.04%
BW 06	9.24%	4.80%	2.56%

Table 1: Comparison of lower bounds

(a)

	B&P			CPLEX 8.1		
Class	solved inst.	avg. gap	time(s)	solved inst.	avg. gap	time(s)
BENG	10	0	0.02	3	4.96%	24.47
GCUT	4	0	3.18	4	0	1.06
NGCUT	12	0	0.02	12	0	0.09
CGCUT	3	0	0.43	3	0	67.01
HT	9	0	0.02	9	0	10.14
Total	38			31		

(b)

	B&P			CPLEX 8.1		
Class	solved inst.	avg. gap	time(s)	solved inst.	avg. gap	time(s)
MV 01	48	0.68%	12.35	48	0.54%	14.09
MV 02	48	0.99%	6.51	25	4.26%	150.01
MV 03	49	0.26%	8.30	47	0.47%	22.49
MV 04	41	1.03%	55.03	21	3.98%	173.26
BW 01	49	0.28%	2.91	48	0.60%	19.29
BW 02	36	1.00%	202.71	21	4.35%	114.12
BW 03	50	0.00%	0.23	50	0.00%	0.16
BW 04	23	1.31%	123.81	17	2.08%	94.71
BW 05	50	0.00%	0.08	50	0.00%	0.04
BW 06	43	1.20%	99.33	34	1.20%	224.74
Total	437			361		

Table 2: Solving the 2LSPP to proven optimality