A New Constraint Programming Approach for the Orthogonal Packing Problem

François Clautiaux, Antoine Jouglet, Jacques Carlier, Aziz Moukrim
Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne,
BP 20529, 60205 Compiègne, France
{francois.clautiaux, antoine.jouglet, jacques.carlier, aziz.moukrim}@hds.utc.fr

April 13, 2006

Abstract

The two-dimensional orthogonal packing problem ($2OPP$) consists in determining if a set of rectangles can be packed in a larger rectangle of fixed size. We propose an exact method for $2OPP$, based on a new constraint-based scheduling model. We provide a generalization of energetic reasoning techniques for the problem under investigation. Feasibility tests requiring the solution of subset-sum problems are described. Computational results confirm the efficiency of our method compared to others in the literature.

1 Introduction

The *two-dimensional orthogonal packing problem* ($2OPP$) consists in determining if a set of rectangles (items) can be packed into one rectangle of fixed size (bin). This problem occurs in industry when rectangular pieces of steel, wood, or paper have to be cut from a larger rectangle. It belongs to the well-known family of *Cutting and Packing* problems, and can be characterized either as a *two-dimensional open-dimension problem*, or as a *two-dimensional knapsack problem* [26]. It is *NP-complete* [19] and is at the root of numerous packing problems. Although Caprara and Monaci [8] and Clautiaux *et al.* [12] have shown that it is possible to avoid solving $2OPP$ for the *two-dimensional knapsack problem* ($2KP$) and for the *two-dimensional bin-packing problem* ($2BP$) when good lower bounds are used, considerable practical difficulties arise in cases where solving $2OPP$ is unavoidable.

A $2OPP$ instance D is a pair (I, B) in which $I = \{1, \dots, n\}$ is the set of items i which have to be packed, and $B = (W, H)$ is a bin of width W and height H. If there is a solution for this instance, then we say that it is feasible. An item i has a width w_i and a height h_i ($w_i, h_i \in \mathbb{N}$). We consider the version of the problem in which the items cannot be rotated. The position of item i, denoted by (x_i, y_i), corresponds to the coordinates of its bottom left-hand corner.

Several exact methods are described in the literature for $2OPP$. Four ways of solving $2OPP$ are used. The first consists in packing items one by one in the bin [20, 24]. The second relies on constraint programming techniques [4, 25]. The third uses a graph-theoretical model [16, 17]. Finally, the fourth [13] deals first with a relaxed problem, and then looks for a corresponding solution for $2OPP$. The latter two methods [13, 16] are the best known so far for the bin-packing problem. To our knowledge, the technique of [4] has never been used for the specific problem we address. The graph-theoretical model of Fekete and Schepers [16, 17] allows non-feasible problems to be quickly detected. The relaxation proposed by Clautiaux *et al.* [13] is interesting, since it transpires that for many non-feasible test cases, the relaxed problem is not feasible either. The relaxation of [13] consists in cutting each item (w_i, h_i) into h_i strips of width w_i. All strips related to a given item have to be packed at the same x-coordinate, even if all parts are not contiguous. Using this relaxation, the problem consists in finding a suitable set of x-coordinates for the items. The obtained relaxed problem is similar to a cumulative scheduling problem.

To our knowledge, the method of Fekete and Schepers [16] and the $TSBP$ method of Clautiaux *et al.* [13] are the most efficient. It would appear that these two procedures are complementary. The graph-theoretical method behaves well when there are subsets of reduced size which are not feasible. The $TSBP$ method behaves well when it has to detect hard non-feasible configurations with a total area close to the area of the bin. However, these methods remain inefficient in several hard instances. The goal of this paper is to propose a new constraint programming branch-and-bound procedure, allowing non-feasible sets to be detected more efficiently.

Previous results in the literature for $2OPP$ are recalled in some detail in Section 2. Constraint programming has been shown to be efficient for solving many optimization problems, including cumulative scheduling problems, which are close to the orthogonal packing problem. In Section 3 we describe a new constraint programming model relying on cumulative scheduling problems, which correspond to the relaxation proposed by Clautiaux *et al.* [13]. We then describe a branch-and-bound algorithm based on this model to solve $2OPP$, and present new techniques which can significantly reduce the search space and the amount of time required by the method. In Section 4, we describe the concept of two-dimensional energetic reasoning, which generalizes classical energetic reasoning from scheduling problems to orthogonal packing problems. We also use the fact that previous lower bounds [9] can be used to improve energetic reasoning deductions. We show in Section 5 how solving subset-sum problems can be useful to perform feasibility tests. Finally, in Section 6 we report computational experiments testing our new method against the randomly-generated benchmarks described in [13]. We then test the efficiency of our techniques and compare our procedure to the methods described Fekete and Schepers [16] and Clautiaux *et al.* [13]. We also test the method of Beldiceanu and Carlsson [4] against the instances proposed in [13].

2 Literature Review

2.1 Exact Methods

Several methods in the literature [11, 20, 24] build a configuration step by step using the *leftmost-downward* strategy. The coordinates considered for an item i are the normal pattern coordinates (see [11]): i has its left edge adjacent either to the right edge of a previously packed item or to the left edge of the bin, and its bottom edge is adjacent either to a packed item or to the bottom edge of the bin.

Pisinger *et al.* [25] propose a constraint programming approach for the orthogonal packing problem. For each pair of items i and j, the algorithm decides whether i is above, below, to the left, or to the right of j. Beldiceanu and Carlsson [4] describe a method derived from an idea called "sweep" (see for instance [5]), which is a generic pruning technique for problems whose constraints share some variables. This algorithm has been shown to be particularly suited to propagate efficiently the non-overlapping rectangles constraint. To our knowledge, it has never been used to solve $2OPP$, even if it could improve the obtained results.

Fekete and Schepers [16, 17] propose a new model for the feasibility problem. They show that a pair of interval graphs can be associated with any *packing class* (*i.e.*, a set of packings with common properties). The interest of this concept is that a large number of symmetries are removed, since only one packing is enumerated per class. A graph $G_d = (V, E_d)$ is associated with each dimension, where $|V| = n$ is the number of items in the bin and $d \in \{w, h\}$ is the dimension considered. In the two graphs each vertex v_i is associated with an item i. An edge is added in the graph G_w (respectively G_h) between two vertices v_i and v_j if the projections of items i and j on the horizontal (respectively vertical) axis overlap. The authors provide a branch-and-bound [16] to look for a pair of interval graphs with suitable properties, and then deduce a feasible packing. In comparison with classical methods, this method avoids a large number of redundancies, and outperforms the method of Martello and Vigo [24].

Clautiaux *et al.* [13] describe the branch-and-bound procedure $TSBP$ for $2OPP$. It is a two-step branch-and-bound based on a new relaxation of the problem (see Figure 1). In the first step (*outer* branch-and-bound method), all solutions of a relaxed problem are enumerated. For each solution found, a second enumerative method is run (*inner* branch-and-bound method) to seek a solution to the initial problem corresponding to the current solution. The relaxation consists in cutting each item (w_i, h_i) into h_i strips of width w_i, A constraint states that all strips pertaining to a given item have to be packed at the same x-coordinate, even if they are not vertically contiguous. Thus, the resulting problem cannot be solved as a $1BP$. Using this relaxation, the problem consists in seeking a suitable set of x-coordinates for the items (*outer* branch-and-bound method). When a solution is found for the relaxed problem (*i.e.*, a valid list of x-coordinates for the items), the *inner* branch-and-bound method searches for a set of y-coordinates to obtain a solution for the initial

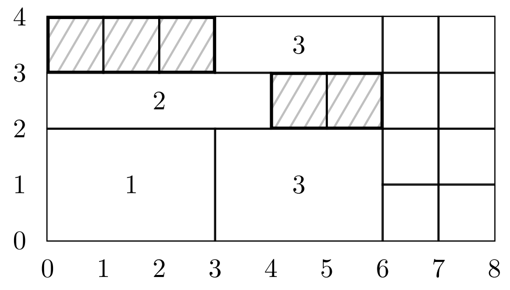

Figure 1: Relaxation of [13]

problem. It transpires that for non-feasible instances of the initial problem, the relaxed problem rarely has a solution.

2.2 Feasibility tests

As an exact resolution may need a large amount of computing time, proposing effective feasibility tests is essential for a method to be competitive. Several lower bounds have been proposed for $2BP$. All of them can be used for detecting non-feasible instances of $2OPP$. Indeed, $2OPP$ can be seen as a $2BP$ instance with only one bin.

Effective polynomial-time-computable lower bounds are described by Carlier *et al.* [9]. The authors use the framework proposed by Fekete and Schepers [18] to compute lower bounds for this problem, using a discretization of so-called dual-feasible functions (DFF). Moreover, Carlier *et al.* [9] have introduced the concept of *Data-Dependent* DFF (DDFF). These functions behave like DFF for the specific instance considered. They introduced two families of DFF [9], similarly to Fekete and Schepers [18], and a family of DDFF [9]. A more complete description of the DDFF and the bounds of Carlier *et al.* can be found in [9], or [13] for $2OPP$. In many cases DFF lead to excellent bounds for $2OPP$. These lower bounds have been improved by Clautiaux *et al.* [14] by using these DFF and the functions described by Carlier and Néron [10]. Recently, Caprara *et al.* [7] proposed an exact method, based on a bilinear programming model, to find the best pair of DFF for a given two-dimensional bin-packing instance.

In a recent paper Clautiaux *et al.* [13] have described a feasibility test for a partial solution of $2OPP$. This bound is based on the application of dual-feasible functions on a modified instance. Consider an enumeration process, such that the x-coordinate and then the y-coordinate of items are fixed. The partial or total packings provide information which can be used to update the lower bounds and the reduction procedures. The idea is to aggregate items which are packed side by side to create new instances more constrained than the initial instance.

3 A Constraint-Based Scheduling Branch-and-Bound Algorithm

Constraint programming is a programming paradigm aimed at solving combinatorial optimization problems that can be described by a set of variables, a set of possible values for each variable, and a set of constraints between the variables. The set of possible values of a variable is called the *variable domain.* A constraint between variables expresses which combination of values for the variables are allowed. The question to be answered is whether there exists an assignment of values to variables, such that all constraints are satisfied. The power of constraint programming method lies mainly in the fact that constraints can be used in an active process termed "constraint propagation" where certain deductions are performed, in order to reduce computational effort. Constraint propagation removes values from the domains, deduces new constraints, and detects inconsistencies.

The aim of this section is to propose a branch-and-bound algorithm with constraint propagation techniques to solve $2OPP$. We describe a basic constraint programming model for $2OPP$. We then propose an original constraint-based scheduling model. Effective propagation techniques will improve the solution process. Next, we describe our constraint programing branch-and-bound algorithm.

3.1 A basic constraint programming model

In constraint programming, $2OPP$ can be classically encoded in terms of variables and constraints. Two variables X_i and Y_i are associated with each item i. They represent the coordinates of i in the bin. We denote as $D(X_i) = [X_i^{min}, X_i^{max}]$ and $D(Y_i) = [Y_i^{min}, Y_i^{max}]$, respectively the domains of variables X_i and Y_i, in which X_i^{min}, X_i^{max}, Y_i^{min} and Y_i^{max} are the lower and upper bounds of the domains. Initially, the domains of these variables are respectively set to $[0, \ldots, W - w_i]$ and $[0, \ldots, H - h_i]$. For each pair of items i and j, we associate the following constraint:

$$[X_i + w_i \leq X_j] \text{ or } [X_j + w_j \leq X_i] \text{ or } [Y_i + h_i \leq Y_j] \text{ or } [Y_j + h_j \leq Y_i]$$

which expresses the fact that items i and j cannot overlap in the bin. This model is sufficient to ensure that the solution is valid, once the domains of variables have been reduced to only one value such that all constraints are satisfied. We will refer to this model as the "basic model". Generally, this model is practically ineffective to solve the problem. However, note that the use of the "sweep" algorithm of Beldiceanu and Carlsson [4], which propagates efficiently the above-described constraint, gives competitive results.

3.2 A constraint-based scheduling model

The principles of constraint programming have been widely applied in the area of scheduling. With the aim of improving the efficiency of the constraint programming model above, we propose a new constraint-based scheduling model relying on a non-preemptive cumulative-scheduling problem associated with two relaxations of $2OPP$. This model can be described independently of the scheduling problems. However, it is interesting to see the relations between the orthogonal packing and the cumulative scheduling problems. Moreover, most of the described constraints, along with the associated propagation, are taken from the constraint-based scheduling literature. This also provides an insight into how our method can be implemented using constraint-based scheduling tools.

Baker [1] defines scheduling as the problem of allocating resources to activities over time. In some cases, a scheduling problem consists of a set of n activities $\{A_1, \ldots, A_n\}$ and a set of resources $\{R_1, \ldots, R_m\}$. Each activity A_i has a processing time, requires a particular amount of a resource R_k and has to be executed within a time window $[est_i, let_i)$, where est_i and let_i are respectively defined as the earliest start time and the latest end time of activity i. Resources have a given capacity that cannot be exceeded at any point in time. The problem to be solved consists in deciding when each activity is executed, while respecting the resource constraints. In a cumulative scheduling problem, the resource can execute several activities, provided that the resource capacity is not exceeded. In non-preemptive scheduling, each activity must be executed without interruption from its start time to its end time.

It is easy to see that the relaxation of $2OPP$ proposed by Clautiaux *et al.* [13] (see Section 1) can be seen as a non-preemptive cumulative scheduling problem. Note that the same observation may be made in relation to the problem obtained by exchanging the roles of the width and the height. We model the two obtained relaxed problems as a non-preemptive cumulative scheduling problem with two resources, as follows. The two considered resources are termed R_w and R_h. The resource capacity of R_w is equal to H and the resource capacity of R_h is equal to W. We define a set $\{A_1^w, \ldots, A_n^w\}$. Each activity A_i^w has a processing time w_i, requires an amount h_i of the resource R_w, and has to be executed within the time window $[0, W)$. Similarly, we define a set $\{A_1^h, \ldots, A_n^h\}$ of activities. Each activity A_i^h has a processing time h_i, requires an amount w_i of the resource R_h and has to be executed within the time window $[0, H)$.

We introduce a variable $start(A)$ for each activity A, representing the start time of A. Initially, the domain of variables $start(A_i^w)$ is set to $[0, \ldots, W - w_i]$ and the domain of variables $start(A_i^h)$ is set to $[0, \ldots, H - h_i]$. Resource constraints represent the fact that activities require some amount of a resource throughout their execution. In our non-preemptive cumulative case, the resource constraints can be expressed as follows:

$$\forall t \in [0, \ldots, W], \quad \sum_{A_i^w / start(A_i^w) \leq t < start(A_i^w) + w_i} h_i \leq H.$$

$$\forall t \in [0, \ldots, H], \quad \sum_{A_i^h / start(A_i^h) \leq t < start(A_i^h) + h_i} w_i \leq W.$$

In other words, the sum of resource requirement of activities A_i^w (respectively

A_i^h) executed at time t has to be lower than or equal to the resource capacity H (respectively W) of resource R_w (respectively R_h).

Finally, the scheduling problem is linked to the constraint programming model of the original $2OPP$ (see previous section) with the following constraints: for each item i, $[start(A_i^w) = X_i]$ and $[start(A_i^h) = Y_i]$. It is easy to see that once the variables $start(A_i^w)$ and $start(A_i^h)$ are instantiated in such a way that all constraints are satisfied, the corresponding solution is valid.

3.3 Solving the problem

Several methods can be used to solve the problem. One may decide the order in which variables are chosen to be instantiated and at which value. Particularly, several branch-and-bound algorithms described for $2OPP$ (see Section 1) can be easily used with our constraint programming models.

To describe our branch-and-bound algorithm, we use the $start(A_i^w)$ and $start(A_i^h)$ variables. We use a schedule-or-postpone method, which works as follows: at each step of the procedure, we choose an unscheduled activity and we schedule it as early as the previous activities scheduled on the same resource will allow. Note that each time such a job is scheduled, the constraint propagation techniques are triggered to reduce domains of variables and thus to reduce the search space. If the chosen start time leads to a failure, *i.e.*, no solution can be found with respect to this start time, the activity is postponed until the domain of its start variable has been modified. This modification can occur as a result of a combination of decisions and constraint propagations. The schedule is built using a depth-first strategy.

We obtained the best experimental results by working first on a given resource exclusively and then on the other. Without loss of generality, in the sequel, we consider first the resource R_w and then the resource R_h : among activities, the non-scheduled activity A_i^w of minimum earliest start time is chosen and scheduled on resource R_w. Once all $start(A_i^w)$ variables are instantiated, the non-scheduled activity A_i^h of minimal earliest start time is chosen and scheduled on resource R_h. Practically speaking, considering resource R_h first can lead to better results. We have tested many heuristics to choose the order in which we consider the resources. None appears to be relevant. In our experiments we arbitrarily work on resource R_w first.

Note that this branching scheme is similar to that used in the two-step branch-and-bound algorithm of Clautiaux *et al.* [13] (see Section 2.1). Indeed, the relaxation of the problem described by [13] corresponds to the set $\{A_1^w, \dots, A_n^w\}$ of activities to be scheduled on resource R_w. The first step (*outer* branch-and-bound method) of the method of [13] corresponds to the stage in which we choose the activities A_i^w to be scheduled and the second step (*inner* branch-and-bound method) corresponds to the stage in which we choose the activities A_i^h to be scheduled. The main difference with respect to the algorithm of Clautiaux *et al.* [13] is that decisions made during our branch-and-bound procedure and constraint propagation tighten the domains of variables. Although we work in two stages, all variables remain linked by the constraints. Consequently, variables in the domain of $start(A_i^h)$ can then be tightened or instantiated, and inconsistencies can be detected on resource R_h even during the first stage of our branch-and-bound method.

Since $2OPP$ has been modeled as a scheduling problem, it is now possible to use powerful constraint-based scheduling propagation techniques specific to non-preemptive scheduling problems (see for instance [2]). These techniques allow us to tighten the domains of variables and to detect inconsistencies during the procedure. To propagate the resource constraints, we rely on the data structure termed "Time-Table" to maintain information about resource utilization and resource availability over time (see for instance [22]). Resource constraints are propagated in two directions: from resources to activities, to update the time windows of activities according to the availability of resources; and from activities to resources to update the Time-Tables according to the time windows of activities. We use the propagation of the disjunctive constraint, which compares the temporal characteristics of pairs of activities: two activities A_i^w and A_j^w such that $h_i + h_j > H$ cannot overlap in time since they require the same resource R_w and since scheduling both tasks at the same time requires a greater amount of resources than the capacity H of R_w. Hence, either A_i^w precedes A_j^w, or A_j^w precedes A_i^w, *i.e.*, the disjunctive constraint holds between these activities. We use the same reasoning for activities $A_1^h, \dots, A_n^h$ and resource R_h. Finally, we could use the edge-finding propagation techniques [2], which are also able to adjust the time-windows of activities according to the resource constraints. Nevertheless, our experimental studies show that edge-finding techniques are not useful in cases where other techniques described in the following sections are used.

4 Two-Dimensional Energetic Reasoning

In this section, we describe the concept of energetic reasoning, originally developed by Erschler *et al.* [15, 23] to solve cumulative scheduling problems. We suggest a generalization of energetic reasoning, which allows the feasibility of orthogonal packing patterns to be tested, and new adjustments to be found.

For scheduling problems, deductions made using energetic reasoning are based on the consumption of resources by activities during given time intervals. For a given time interval $[\alpha, \beta)$, $\alpha < \beta$, energy is supplied by a resource and consumed by an activity. The energy supplied by a resource of capacity C in this interval is equal to $(\beta - \alpha) \times C$, and the energy consumed by an activity of demand c_i is equal to $c_i \times \Delta_i$, where Δ_i is the part of activity i scheduled in $[\alpha, \beta)$. If the starting time of the activity is not yet fixed, we determine the mandatory energy consumption in interval $[\alpha, \beta)$. It is obtained by considering the positions in which the processing of the activity is minimal in $[\alpha, \beta)$. By considering the quantities of energy supplied and consumed within given intervals, the energetic approach aims at developing satisfiability tests and time-bound adjustments to ensure that either a given schedule is not feasible or to derive some necessary conditions that any feasible schedule must satisfy.

Unlike activities in scheduling problems, the position of an item has to be fixed with respect to both the horizontal and vertical dimensions. We therefore suggest the following generalization of energetic reasoning. Instead of considering an interval $[\alpha, \beta)$, we consider a rectangular window. We define the rectangular window $[\alpha, \beta, \gamma, \delta)$, $\alpha < \beta$ and $\gamma < \delta$, in which $[\alpha, \beta) \times [\gamma, \delta)$ is the area under consideration.

Energy is now supplied by the bin and consumed by items. Energy supplied by the bin in window $[\alpha, \beta, \gamma, \delta)$ is equal to $(\beta - \alpha) \times (\delta - \gamma)$. The energy consumed by an item can be computed considering the bottom left and top right positions according to the domains of its coordinate variables, in which the item's consumption is minimal (see Figure 2). Let $\widehat{w}_i(\alpha, \beta)$ and $\widehat{h}_i(\gamma, \delta)$ be respectively the width and the height of the mandatory part of item i in the window $[\alpha, \beta, \gamma, \delta)$ (see Figure 2). We have:

$$\widehat{w}_i(\alpha, \beta) = \max\left(0, \min\left\{w_i, \beta - \alpha, X_i^{min} + w_i - \alpha, \beta - X_i^{max}\right\}\right)$$

and

$$\widehat{h}_i(\gamma, \delta) = \max\left(0, \min\left\{h_i, \delta - \gamma, Y_i^{min} + h_i - \gamma, \delta - Y_i^{max}\right\}\right).$$

Energy consumed by item i in window $[\alpha, \beta, \gamma, \delta)$ is then

$$\widehat{E}_i(\alpha, \beta, \gamma, \delta) = \widehat{w}_i(\alpha, \beta) \times \widehat{h}_i(\gamma, \delta).$$

Therefore, the total energy consumed by all items in window $[\alpha, \beta, \gamma, \delta)$ is

$$\widehat{E}(\alpha, \beta, \gamma, \delta) = \sum_{i \in I} \widehat{E}_i(\alpha, \beta, \gamma, \delta).$$

4.1 Feasibility tests and bounds adjustments

As in basic energetic reasoning, the following proposition holds:

Proposition 4.1. *If there is a feasible packing, then $\forall \alpha, \beta \in [0, W)$, $\forall \gamma, \delta \in [0, H)$, such that $\alpha < \beta$ and $\gamma < \delta$, we have $\widehat{E}(\alpha, \beta, \gamma, \delta) \leq (\beta - \alpha) \times (\delta - \gamma)$.*

This means that for every possible window, energy supplied by the bin has to be at least as large as minimal energy consumed by items. To perform feasibility tests, we can test this inequality at each node of the search tree algorithm for all relevant windows (see Section 4.2) in the bin. If there exists a window for which the inequality does not hold, then the considered node cannot lead to a feasible solution and can consequently be pruned.

The values of $\widehat{E}(\alpha, \beta, \gamma, \delta)$ can also be used to adjust domain variable bounds of X_i and Y_i. For an item i and a window $[\alpha, \beta, \gamma, \delta)$, we define the following lengths:

- $w_i^+(\alpha) = \max(0, w_i - \max(0, \alpha - X_i^{min}))$, the width of the mandatory part of i after α if i is packed furthest to the left with respect to the domain of X_i.

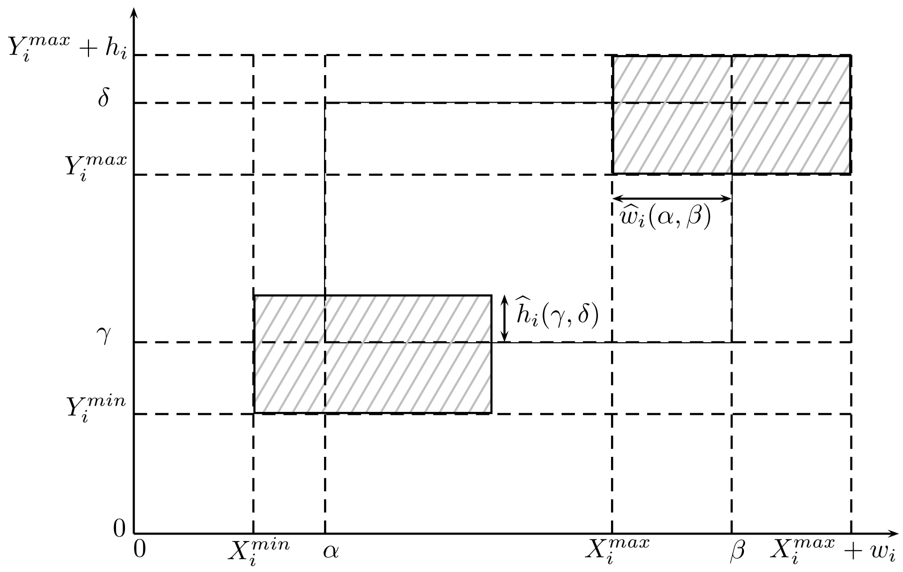

Figure 2: Mandatory part of item i in window $[\alpha, \beta, \gamma, \delta)$ considering the bottom left and top right positions.

- $w_i^-(\beta) = \max(0, w_i - \max(0, X_i^{max} + w_i - \beta))$, the width of the mandatory part of i before β if i is packed furthest to the right with respect to the domain of X_i.
- $h_i^+(\gamma) = \max(0, h_i - \max(0, \gamma - Y_i^{min}))$, the height of the mandatory part of i over γ if i is packed lowest with respect to the domain of Y_i.
- $h_i^-(\delta) = \max(0, h_i - \max(0, Y_i^{max} + h_i - \delta))$, the height of the mandatory part of i under δ if i is packed uppermost with respect to the domain of Y_i.
- $\check{w}_i(\alpha, \beta) = \max(0, \min(w_i, \beta - \alpha, \beta - X_i^{min}, X_i^{max} - \alpha))$, the width of the maximum part of i in interval $[\alpha, \beta)$.
- $\check{h}_i(\gamma, \delta) = \max(0, \min(h_i, \delta - \gamma, \delta - Y_i^{min}, Y_i^{max} - \gamma))$, the height of the maximum part of i in interval $[\gamma, \delta)$.

Let i be an item and let $[\alpha, \beta, \gamma, \delta)$ be a window such that $\beta < X_i^{max} + w_i$. We verify whether i can be fully packed before β, *i.e.*, if i can be packed at a coordinate X_i such that $X_i + w_i \leq \beta$. If i is fully packed before β, then its energy consumption is obtained by considering its leftmost position, and is equal to $w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta)$. Therefore, the total energy consumed by all items in window $[\alpha,\beta,\gamma,\delta)$ is $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^+(\alpha) \times \widehat{h}_i(\gamma,\delta)$. If this total energy consumption is greater than $(\beta-\alpha)\times(\delta-\gamma)$, it means that item i cannot be fully packed before β. Moreover, it means that at least $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^+(\alpha)\times\widehat{h}_i(\gamma,\delta) - (\beta-\alpha)\times(\delta-\gamma)$ units of energy of i are after β. The height of the mandatory part of i in $[\alpha,\beta,\gamma,\delta)$ is at most $\check{h}_i(\gamma,\delta)$. The lower bound of the domain of X_i (*i.e.*, X_i^{min}) can be then adjusted to $\beta - w_i + \frac{1}{\check{h}_i(\gamma,\delta)} \times (\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^+(\alpha)\times\widehat{h}_i(\gamma,\delta) - (\beta-\alpha)\times(\delta-\gamma))$. The same reasoning can be applied if we try to fully pack i after α, under δ or over γ. Consequently the following proposition holds:

Proposition 4.2. *Consider an item i and a window $[\alpha,\beta,\gamma,\delta)$.*

- *If $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^+(\alpha)\times\widehat{h}_i(\gamma,\delta) > (\beta-\alpha)\times(\delta-\gamma)$, then $X_i^{min} \geq \beta - w_i + \frac{1}{\check{h}_i(\gamma,\delta)} \times (\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^+(\alpha)\times\widehat{h}_i(\gamma,\delta) - (\beta-\alpha)\times(\delta-\gamma))$.*

- *If $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^-(\beta)\times\widehat{h}_i(\gamma,\delta) > (\beta-\alpha)\times(\delta-\gamma)$, then $X_i^{max} \leq \alpha - \frac{1}{\check{h}_i(\gamma,\delta)} \times (\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + w_i^-(\beta)\times\widehat{h}_i(\gamma,\delta) - (\beta-\alpha)\times(\delta-\gamma))$.*

- *If $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + h_i^+(\gamma)\times\widehat{w}_i(\alpha,\beta) > (\beta-\alpha)\times(\delta-\gamma)$, then $Y_i^{min} \geq \delta - h_i + \frac{1}{\check{w}_i(\alpha,\beta)} \times (\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + h_i^+(\gamma)\times\widehat{w}_i(\alpha,\beta) - (\beta-\alpha)\times(\delta-\gamma))$.*

- *If $\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + h_i^-(\delta)\times\widehat{w}_i(\alpha,\beta) > (\beta-\alpha)\times(\delta-\gamma)$, then $Y_i^{max} \leq \gamma - \frac{1}{\check{w}_i(\alpha,\beta)} \times (\widehat{E}(\alpha,\beta,\gamma,\delta) - \widehat{E}_i(\alpha,\beta,\gamma,\delta) + h_i^-(\delta)\times\widehat{w}_i(\alpha,\beta) - (\beta-\alpha)\times(\delta-\gamma))$.*

Such adjustments can be done on all relevant windows (see Section 4.2) at each node of the search tree. Algorithm 1 can be used to perform all feasibility tests and bound adjustments. Variable S represents the slack available to fully pack item i before β, after α, under δ or over γ.

4.2 Characterization of relevant and irrelevant intervals

Baptiste *et al.* [3] studied the cumulative scheduling problem and showed that it is sufficient to calculate energies for intervals belonging to a characterized set in order to find all possible deductions. Using our notation, their result can be stated as follows.

Proposition 4.3. *Let O_1 and O_2 be two sets defined as:*

1. $O_1 = \bigcup_{i\in I}\{X_i^{min}, X_i^{max}, X_i^{min} + w_i\}$

2. $O_2 = \bigcup_{i\in I}\{X_i^{max} + w_i, X_i^{min} + w_i, X_i^{max}\}$

Algorithm 1: Feasibility tests and bound adjustments.

```
for each relevant window $[\alpha, \beta, \gamma, \delta)$ do
    $\widehat{E}_{items} \leftarrow \sum_{i \in I} \widehat{E}_i(\alpha, \beta, \gamma, \delta)$;
    $\widehat{E}_{bin} = (\beta - \alpha) \times (\delta - \gamma)$;
    if $\widehat{E}_{items} > \widehat{E}_{bin}$ then
        there is no feasible packing: trigger a backtrack;
    else
        for $i \in I$ do
            $S \leftarrow \widehat{E}_{bin} - \widehat{E}_{items} + \widehat{E}_i(\alpha, \beta, \gamma, \delta)$;
            if $S < w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta)$ then
                $X_i^{min} \leftarrow \max(X_i^{min}, \beta - w_i + \left\lceil \frac{w_i^+(\alpha) \times \widehat{h}_i(\gamma,\delta) - S}{\widehat{h}_i(\gamma,\delta)} \right\rceil)$;
            if $S < w_i^-(\beta) \times \widehat{h}_i(\gamma, \delta)$ then
                $X_i^{max} \leftarrow \min(X_i^{max}, \alpha - \left\lceil \frac{w_i^-(\beta) \times \widehat{h}_i(\gamma,\delta) - S}{\widehat{h}_i(\gamma,\delta)} \right\rceil)$;
            if $S < h_i^+(\gamma) \times \widehat{w}_i(\alpha, \beta)$ then
                $Y_i^{min} \leftarrow \max(Y_i^{min}, \delta - h_i + \left\lceil \frac{h_i^+(\gamma) \times \widehat{w}_i(\alpha,\beta) - S}{\widehat{w}_i(\alpha,\beta)} \right\rceil)$;
            if $S < h_i^-(\delta) \times \widehat{w}_i(\alpha, \beta)$ then
                $Y_i^{max} \leftarrow \min(Y_i^{max}, \gamma - \left\lceil \frac{h_i^-(\delta) \times \widehat{w}_i(\alpha,\beta) - S}{\widehat{w}_i(\alpha,\beta)} \right\rceil)$;
```

If for any interval $[\alpha, \beta)$ such that $\alpha \in O_1$ and $\beta \in O_2$ and $\alpha < \beta$, we have $\widehat{E}(\alpha, \beta, 0, H) \leq (\beta - \alpha)H$, then there are no values α' and β' such that $\widehat{E}(\alpha', \beta', 0, H) > (\beta' - \alpha')H$.

It is obvious that the result of Baptiste *et al.* also holds for energetic reasoning techniques applied to $2OPP$. It can also be generalized for two-dimensional energetic reasoning techniques.

Proposition 4.4. *Consider a given partial solution for a $2OPP$ instance and the four following sets of x- and y-coordinates.*

1. $O_1^W = \bigcup_{i \in I} \{X_i^{min}, X_i^{max}, X_i^{min} + w_i\}$
2. $O_2^W = \bigcup_{i \in I} \{X_i^{max} + w_i, X_i^{min} + w_i, X_i^{max}\}$
3. $O_1^H = \bigcup_{i \in I} \{Y_i^{min}, Y_i^{max}, Y_i^{min} + h_i\}$
4. $O_2^H = \bigcup_{i \in I} \{Y_i^{max} + h_i, Y_i^{min} + h_i, Y_i^{max}\}$

If for all $\alpha \in O_1^W$, $\beta \in O_2^W$, $\gamma \in O_1^H$, $\delta \in O_2^H$ such that $\alpha < \beta$ and $\gamma < \delta$ we have $\widehat{E}(\alpha, \beta, \gamma, \delta) \leq (\beta - \alpha)(\delta - \gamma)$, then there are no coordinates $\alpha', \beta', \gamma', \delta'$ such that $\widehat{E}(\alpha', \beta', \gamma', \delta') > (\beta' - \alpha')(\delta' - \gamma')$.

Proof. When only a horizontal strip in the vertical interval $[\gamma, \delta)$ is considered, a new problem is obtained, where each item i has the same width w_i and the same domain $[X_i^{min}, X_i^{max}]$ as in the original problem. Only the values relating to the height are modified. So when the proposition of Baptiste *et al.* [3] is applied to this new problem, the same set of relevant x-coordinates is found, whatever the values of γ and δ. So the values in O_1^W, O_2^W are sufficient.

This proof also holds for any vertical strip by exchanging the width and the height. It means that the values in O_1^H and O_2^H are sufficient.

We deduce that the set of relevant x-coordinates and y-coordinates that can be computed initially are sufficient for the energetic reasoning in two dimensions. $\square$

Therefore, it is sufficient to calculate energies for all windows $[\alpha, \beta, \gamma, \delta)$ in the Cartesian product of O_1^W, O_2^W, O_1^H and O_2^H. However, considering all these relevant intervals is inefficient in practice, since it requires a large amount of time to be computed. In our current implementation, we use instead the following modified intervals: $O_1'^W = \{X_i^{min}, i \in I\}$, $O_2'^W = \{X_i^{max} + w_i, i \in I\}$, $O_1'^H = \{Y_i^{min}, i \in I\}$, $O_2'^H = \{Y_i^{max} + h_i, i \in I\}$. In practice, slightly fewer deductions are performed, but the technique is more efficient since it requires significantly less time to be executed.

4.3 Using DDFF in Energetic Reasoning

Consider virtual items $(\widehat{w}_i(\alpha, \beta), \widehat{h}_i(\gamma, \delta))$ obtained from the widths and the heights of items in window $[\alpha, \beta, \gamma, \delta)$. The set of items corresponding to the mandatory parts of the items of A in $[\alpha, \beta, \gamma, \delta)$ is denoted by $\widehat{A}(\alpha, \beta, \gamma, \delta)$.

Proposition 4.5. *Let (A, B) be a $2OPP$ problem. For four integer values α, β, γ and δ, and a domain variable for the x- and y-coordinates of the items of A, if the $2OPP$ problem defined by $\widehat{A}(\alpha, \beta, \gamma, \delta)$ and $\widehat{B} = (\beta - \alpha, \delta - \gamma)$ has no solution, then there is no solution for the original $2OPP$ with the current domain variables.*

Several methods can be used to show that a particular $2OPP$ problem obtained is not feasible, the quickest being to check that the continuous lower bound does not exceed one. This corresponds to the feasibility test described in previous section. In our case, we use the lower bounds of [9,13], which are based on DDFF. We could use preprocessing [13] or an exact method. Nevertheless, experimentation has shown that the computation time required for this latter method is too great with respect to the reduction of the search space.

5 Pruning the Search Tree by Solving Subset-Sum Problems

In this section we show how reasoning can be done for pruning the search tree using the solution of subset-sum problems. The subset-sum algorithm can be used to perform feasibility tests at each node of the search tree. It has been used by Boschetti and Mingozzi [6] as a preprocessing, and by Fekete and Schepers [16, 18] in their exact method. We now describe a new feasibility test based on a subset-sum algorithm. The main idea is to make use of the information that is given by the domains of the variables in our constraint programming model.

The subset-sum problem is a particular case of the knapsack problem. Let C be an integer value, and $L = \{c_1, c_2, \ldots, c_z\}$ a list of z integer values less than C. The subset-sum problem is the following:

$$\max \left\{ \sum_{i=1}^{z} c_i \xi_i : \sum_{i=1}^{z} c_i \xi_i \leq C, \xi_i \in \{0, 1\} \right\}$$

It consists in determining the sub-list of L for which the sum of elements is the largest, under the constraint that this sum is lower than or equal to C. The subset-sum algorithm is pseudo-polynomial in $O(zC)$, but is executed rapidly when C is small.

5.1 General Principle

Let P be a partial solution for a $2OPP$ instance, and x a given x-coordinate. We denote by H_x the sum of the heights of the items whose variable has been fixed ($X_i^{min} = X_i^{max}$) and such that $x \in [X_i, X_i + w_i)$. If $H_x < H$, additional items can be packed in $[x, x+1)$. However, if there is no unpacked item j such that $h_j < H - H_x$, then necessarily some area has been lost. Consequently, the value $H - H_x$ can be added to the total area of the items to strengthen any reasoning based on the remaining area.

This idea can be generalized when items can be packed in $[x, x+1)$ without the possibility of perfectly fitting the free space. Let I_x be the subset of items i whose variable X_i has not been fixed ($X_i^{min} < X_i^{max}$) and such that $x \in [X_i^{min}, X_i^{max} + w_i)$, *i.e.*, a piece of i could be packed in $[x, x+1)$. We want to determine the minimum height loss in interval $[x, x+1)$. For this purpose, we consider the following subset-sum problem: $C = H - H_x$, and $L = \{h_i / i \in I_x\}$. Let h^* be the optimal solution of the subset-sum problem. The height loss in $[x, x+1)$ is then at least $\delta_x = C - h^*$. The sum of the values δ_x computed for each coordinate $x \in [0, W)$ can be added to the total area of the items. If this value is greater than the total area of the bin, then there is no solution including the current configuration. Thus, if $\sum_{x \in [0,W)} \delta_x + \sum_{i \in I} w_i h_i > W \times H$, a cut is performed.

5.2 Considering intervals instead of all coordinates: SS_1

It should be noted that if all coordinates $x \in [0, W)$ were to be considered, substantial computing time would be required. Instead of considering each possible coordinate x, we only consider intervals $[\alpha, \beta)$ in which $\forall x \in [\alpha, \beta)$ $H_x = H_\alpha$. Let $I_{[\alpha,\beta)}$ be the subset of items i whose variable X_i has not been fixed and such that the interval $[X_i^{min}, X_i^{max} + w_i)$ intersects with $[\alpha, \beta)$. We then consider the following subset-sum problem: $C_1 = H - H_\alpha$ and $L_1 = \{h_i / i \in I_{[\alpha,\beta)}\}$. If h^* is the optimal solution of the corresponding subset-sum problem, then the loss for this interval is at least $\delta_{[\alpha,\beta)} = (C_1 - h^*) \times (\beta - \alpha)$.

We now discuss how these intervals $[\alpha, \beta)$ may be chosen according to the considered partial solution. Let $Z = \{x_1, x_2, \dots, x_z\}$ be a set of z x-coordinates such that $x_1 < x_2 < \dots < x_z$. We define the function ρ as the function that associates Z with a set $\rho(Z) = \{[0, x_1), [x_1, x_2), \dots [x_{z-1}, x_z), [x_z, W)\}$ of intervals.

The method consists in building the set $\rho(Z_1)$ of disjoint intervals related to the set of coordinates Z_1 at which the value H_x moves. Note that $|\rho(Z_1)|$ is in $O(n)$. Now, if $\sum_{[\alpha,\beta) \in \rho(Z_1)} \delta_{[\alpha,\beta)} + \sum_{i \in I} w_i h_i > W \times H$ then there is no feasible solution including the current configuration, and a cut is performed. From now on we shall refer to the technique considering intervals belonging to $\rho(Z_1)$ as technique SS_1.

5.3 Performing all possible deductions: SS_2

Note that considering only intervals related to set Z_1 leads to fewer deductions than if we were to consider all coordinates $x \in [0, W)$. Indeed, an interval $[\alpha, \beta)$ being given, if $\alpha < X_i^{min} < \beta$ (respectively $\alpha < X_i^{max} < \beta$) then an item i can be packed only on a part of this interval. In this case, it might be better to consider two intervals $[\alpha, t)$ and $[t, \beta)$ instead of $[\alpha, \beta)$ where $t = X_i^{min}$ (respectively $t = X_i^{max}$). Indeed, item i can participate in only one subset-sum problem among the two associated with these two intervals. Consequently, the sum of losses for the two intervals (*i.e.*, $\delta_{[\alpha,t)} + \delta_{[t,\beta)}$) can be greater than $\delta_{[\alpha,\beta)}$.

Proposition 5.1. *Let $Z_2 = Z_1 \cup \{X_i^{min}, X_i^{max} / i \in I_{[\alpha,\beta)}\}$ be the union of set Z_1 and the set of coordinates corresponding to the X_i^{min} and X_i^{max} values of unpacked items. We have $\sum_{x \in [0,W)} \delta_x = \sum_{[\alpha,\beta) \in \rho(Z_2)} \delta_{[\alpha,\beta)}$.*

Proof. Let $[\alpha, \beta)$ be an interval belonging to $\rho(Z_2)$. We have $\forall x \in [\alpha, \beta)$ $I_x = I_{[\alpha,\beta)}$. It follows that $\sum_{x \in [\alpha,\beta)} \delta_x = \delta_{[\alpha,\beta)}$. $\square$

Proposition 5.1 means that all deductions done if we consider all coordinates $x \in [O, W)$ can be equivalently done by only considering all intervals related to Z_2. The subset-sum problem to consider for each interval $[\alpha, \beta) \in Z_2$ is: $C_2 = H - H_\alpha$ and $L_2 = \{h_i / i \in I_{[\alpha,\beta)}\}$. Note that $|\rho(Z_2)|$ is in $O(n)$. From now on we shall refer to the technique that considers intervals related to Z_2 as technique SS_2.

5.4 Taking into account the mandatory parts of items: SS_3

We now propose an improvement on technique SS_2, which relies on the mandatory parts of items in certain intervals. Consider an interval $[\alpha, \beta)$. Suppose that the width of the mandatory part of i in $[\alpha, \beta)$ (*i.e.*, $\widehat{w}_i(\alpha, \beta)$ as defined in Section 4) is exactly equal to $\beta - \alpha$. We deduce that h_i should necessarily participate in the optimal solution of the subset-sum problem associated with interval

$[\alpha, \beta)$. We then subtract h_i from the value C_2 and remove it from the list L_2 of heights to be considered in the solution of the subset-sum problem. Let $I'_{[\alpha,\beta)}$ be the subset of items i whose variable X_i has not been fixed ($X_i^{min} < X_i^{max}$) and such that $\widehat{w}_i(\alpha, \beta) = \beta - \alpha$. The subset-sum problem to consider is now $C_3 = H - H_\alpha - \sum_{i \in I'_{[\alpha,\beta)}} h_i$ and $L_3 = \{h_i / i \in I_{[\alpha,\beta)} \setminus I'_{[\alpha,\beta)}\}$. The computed loss can be greater than the one obtained by considering the original subset-sum problem. Note that if $X_i^{max} < X_i^{min} + w_i$ then $\forall \alpha, \beta \in [X_i^{max}, X_i^{min} + w_i)$ such that $\alpha < \beta$ we have $\widehat{w}_i(\alpha, \beta) = \beta - \alpha$. Let Z_3 be the set of intervals such that $Z_3 = Z_2 \cup \{X_i^{min} + w_i, X_i^{max} / i \in I_{[\alpha,\beta)}\}$. We effectively improve the deductions of the technique SS_2 by considering intervals related to Z_3 instead of Z_2. It allows us to make full use of the width of the mandatory parts of items in some intervals. Again, note that $|\rho(Z_3)|$ is in $O(n)$. From now on we shall refer to the technique considering intervals related to Z_3 as technique SS_3.

6 Experimental Results

In this section we provide experimental results in order to show the efficiency of the techniques described in this paper. Moreover, we compare our results with the best previous ones in the literature. Our results were obtained using a Pentium M 1.8 GHz running Windows XP. In order to be able to compare our results to previous relevant experimental studies, methods were run on the benchmarks proposed by Clautiaux *et al.* [13]. The instances are randomly generated with two given parameters: the number of items and the discrepancy between the total area of the items and the area of the bin (see [13]). These instances are available on the web: *http://www.hds.utc.fr/˜fclautia/research.html*. In each table, an instance "D, X, M" will mean an instance of M items with a discrepancy of D percent. If the instance is feasible then X equals F, else X equals N. For different methods and for each instance we report the number of nodes generated by the method (columns "nodes") and the amount of CPU time in seconds required by the method (columns "cpu"). The maximum time allowed for an instance is set to 15 minutes. Note that "_" means that the instance has not been proved to be feasible or non-feasible by the method within the allowed time limit.

6.1 Techniques SS_1, SS_2, SS_3

In Table 1 we show the efficiency of the techniques SS_1, SS_2 and SS_3 which allow the search tree to be pruned using the solution of subset-sum problems (see Section 5). We report the results obtained by the branch-and-bound procedure in the absence of any technique, and then with each of the techniques. We can see that each technique dramatically reduces the search space and the computation time. We remark that while SS_2 generally requires more time than SS_1, it also produces a greater reduction of the search space. The technique SS_3 would appear to be the most helpful. As expected, the methods provide better results when the area of the items is close to the area of the bin.

			SS1		SS2		SS3	
instance	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
00, N, 23	3675002	159.29	121929	8.15	87000	9.79	50095	5.40
00, N, 23	1496331	69.35	11143	0.81	5394	0.73	3208	0.43
05, N, 15	181195	8.74	124619	8.57	110923	13.61	107432	13.19
05, N, 15	75172	3.18	66578	4.15	64156	7.13	61408	6.88
05, F, 20	27515	0.98	27515	0.98	27515	0.98	27515	0.96
04, F, 20	59795	3.02	11454	0.99	9441	1.14	8336	1.05
13, N, 15	14209	0.51	14144	0.78	13690	1.24	13274	1.16
10, N, 15	22683	1.47	18084	1.58	17062	2.50	13786	1.96
03, N, 16	14897	0.89	10279	0.92	9277	1.36	8008	1.12
20, F, 15	7229	0.16	7229	0.33	7229	0.54	6863	0.54
04, N, 15	10181	0.67	9247	0.81	8960	1.23	8002	1.09
03, N, 15	3562	0.27	3491	0.36	3441	0.59	3085	0.53
10, N, 15	2622	0.17	2622	0.23	2622	0.37	2437	0.32
07, N, 15	2149	0.14	2149	0.18	2149	0.32	2133	0.31
05, N, 17	1673	0.09	1669	0.11	1669	0.22	1649	0.22
08, N, 15	1420	0.05	1420	0.07	1420	0.13	1420	0.12
07, F, 15	1584	0.09	1374	0.11	1363	0.14	1358	0.14
03, F, 18	1111	0.08	1104	0.09	1101	0.11	1060	0.11
04, N, 18	1032	0.04	811	0.03	803	0.06	697	0.05
02, F, 20	2245	0.10	275	0.02	242	0.04	236	0.03
04, F, 17	474	0.01	474	0.03	474	0.03	474	0.01
13, N, 10	343	0.01	327	0.01	318	0.04	316	0.03
04, F, 15	287	0.01	285	0.02	285	0.03	285	0.03
03, N, 17	260	0.02	231	0.03	223	0.05	163	0.03
00, N, 15	610	0.05	171	0.03	100	0.02	96	0.02
20, F, 15	96	0.01	96	0.01	96	0	96	0
05, F, 15	49	0	42	0.01	40	0.02	36	0.01
02, F, 17	31	0.01	31	0.01	31	0	29	0
02, F, 22	31	0.01	31	0	31	0.02	31	0.01
04, F, 19	29	0	29	0	29	0	29	0.01
05, F, 18	25	0.01	25	0.01	25	0.02	25	0
08, F, 15	21	0	21	0.01	21	0.01	21	0
04, N, 17	3	0.01	3	0	3	0	3	0.01
13, N, 15	1	0	1	0	1	0	1	0
15, N, 15	1	0	1	0.01	1	0.01	1	0.01
00, N, 10	1	0.01	1	0.01	1	0.01	1	0
02, N, 20	1	0.01	1	0.01	1	0	1	0.01
03, N, 10	1	0.01	1	0.01	1	0	1	0
07, N, 10	1	0	1	0	1	0.01	1	0
07, N, 15	1	0	1	0	1	0.01	1	0
10, N, 10	1	0.01	1	0	1	0	1	0
mean	136680	6.09	10705	0.72	9199	1.04	7893	0.87

Table 1: Tests SS_1, SS_2, SS_3

6.2 Studying the efficiency of each technique

Table 2 shows the efficiency of our techniques. We report the results obtained using the constraint based scheduling model (see Section 3.2), with different combinations of techniques used inside the branch-and-bound procedure. "LB" refers to the use of lower bounds (see Section 2.2), "ER" to the use of energetic reasoning (see Section 4.1), "ER(LB)" to the use of DDFF in energetic reasoning (see Section 4.3), and finally "SS" refers to the use of the SS_3 technique to prune the search tree, relying on the solution of subset-sum problems (see Section 5). Because of the branching scheme of our method (in which we have chosen to bound the X_i variables before the Y_i variables), two-dimensional energetic reasoning is no more efficient than classical energetic reasoning, which can be seen as a special case of two-dimensional energetic reasoning in which only the windows $[\alpha, \beta, 0, H)$ are considered. This is why "ER" denotes classical energetic reasoning as well as two-dimensional energetic reasoning. Note that if we were to use another branching scheme, such as [11, 16, 17, 20, 24, 25], two-dimensional energetic reasoning should be more efficient. However, not surprisingly, two-dimensional energetic reasoning requires significantly more CPU time than classical energetic reasoning. In the table we can see that each improving technique is able to reduce the search space and the computation time.

			LB				LB				LB						LB				LB				LB	
					ER		ER		ER(LB)		ER(LB)						ER		ER		ER(LB)		ER(LB)	
													SS		SS		SS		SS		SS		SS	
instance	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
00, N, 23	3675002	159.3	307510	51	88446	5.98	73437	15	71987	36	62220	41.58	50095	5.40	46664	12.31	42207	5.65	39503	11	34842	21	32926	25
00, N, 23	1496331	69.35	45105	8.03	7134	0.48	5560	1.17	4828	1.64	3887	1.91	3208	0.43	3025	0.87	3283	0.49	2808	0.87	1731	0.67	1731	0.94
05, N, 15	181195	8.743	55913	9.13	82787	6.08	29097	5.83	8349	4.25	7532	4.78	107432	13.19	39859	9.74	63661	9.59	26742	7.58	8293	5.13	7506	5.52
05, N, 15	75172	3.175	26315	4.32	49065	3.17	22400	4.16	3220	1.10	2732	1.33	61408	6.88	25082	6.00	44812	6.25	21854	5.72	3207	1.39	2732	1.57
05, F, 20	27515	0.981	27515	0.97	27513	0.97	27513	0.96	27513	0.96	27513	0.96	27515	0.96	27515	0.96	27513	0.96	27513	0.96	27513	0.96	27513	0.98
04, F, 20	59795	3.024	3668	0.65	9002	0.79	1008	0.23	771	0.32	745	0.38	8336	1.05	1163	0.31	5580	0.89	882	0.25	675	0.33	657	0.37
13, N, 15	14209	0.511	7317	0.90	11897	0.65	6374	0.94	2085	0.63	2085	0.82	13274	1.16	6927	1.22	11566	1.24	6260	1.27	2081	0.78	2081	0.96
10, N, 15	22683	1.472	9065	1.54	11161	1.10	6875	1.40	1814	1.05	1748	1.19	13786	1.96	8333	2.02	8884	1.60	6681	1.87	1801	1.23	1743	1.34
03, N, 16	14897	0.891	8846	1.92	5183	0.57	4745	1.32	4186	2.92	4184	3.62	8008	1.12	6405	2.00	4743	0.92	4395	1.60	3905	3.04	3905	3.70
20, F, 15	7229	0.16	463	0.07	7229	0.24	463	0.06	27	0.01	27	0.01	6863	0.54	452	0.09	6863	0.60	452	0.09	27	0.01	27	0.01
04, N, 15	10181	0.671	8059	1.70	4945	0.52	4863	1.24	3405	1.92	3373	2.44	8002	1.09	6804	1.98	4508	0.81	4441	1.49	3316	2.15	3286	2.66
03, N, 15	3562	0.27	2974	0.66	3084	0.29	2674	0.65	2502	1.44	2259	1.59	3085	0.53	2728	0.88	2863	0.56	2573	0.89	2440	1.63	2217	1.78
10, N, 15	2622	0.17	2406	0.47	2622	0.21	2406	0.49	1932	1.04	1869	1.27	2437	0.32	2284	0.60	2437	0.36	2284	0.63	1891	1.17	1844	1.38
07, N, 15	2149	0.141	2051	0.39	2143	0.17	2045	0.42	1974	0.84	1968	1.08	2133	0.31	2048	0.53	2128	0.33	2043	0.56	1970	0.99	1966	1.22
05, N, 17	1673	0.09	1451	0.21	1661	0.10	1449	0.24	87	0.05	87	0.06	1649	0.22	1447	0.32	1637	0.25	1445	0.36	87	0.06	87	0.07
08, N, 15	1420	0.05	1084	0.14	1420	0.08	1084	0.16	1	0.01	1	0	1420	0.12	1084	0.19	1420	0.15	1084	0.22	1	0	1	0.01
07, F, 15	1584	0.09	738	0.12	970	0.08	505	0.09	301	0.15	301	0.14	1358	0.14	701	0.15	921	0.12	496	0.13	301	0.18	301	0.17
03, F, 18	1111	0.08	1050	0.14	1056	0.09	1038	0.15	862	0.18	855	0.22	1060	0.11	1008	0.16	1015	0.10	997	0.17	845	0.19	838	0.22
04, N, 18	1032	0.04	317	0.04	812	0.05	299	0.05	448	0.07	180	0.05	697	0.05	279	0.06	579	0.05	265	0.05	315	0.07	149	0.04
02, F, 20	2245	0.1	38	0	275	0.03	34	0.01	34	0	34	0.02	236	0.03	30	0.01	165	0.02	30	0	30	0.01	30	0.01
04, F, 17	474	0.01	474	0.03	474	0.03	474	0.02	474	0.02	474	0.03	474	0.01	474	0.01	474	0.02	474	0.02	474	0.03	474	0.03
13, N, 10	343	0.01	192	0.03	334	0.02	192	0.04	132	0.04	132	0.05	316	0.03	187	0.03	308	0.03	187	0.04	132	0.05	132	0.06
04, F, 15	287	0.01	285	0.04	286	0.02	285	0.05	285	0.09	284	0.12	285	0.03	285	0.07	285	0.03	285	0.06	284	0.11	284	0.14
03, N, 17	260	0.021	256	0.05	212	0.02	212	0.06	212	0.09	212	0.12	163	0.03	163	0.06	163	0.04	163	0.05	163	0.08	163	0.11
00, N, 15	610	0.05	300	0.08	102	0.01	89	0.03	95	0.05	82	0.06	96	0.02	92	0.04	89	0.02	86	0.04	83	0.06	80	0.06
20, F, 15	96	0.01	96	0.01	96	0	96	0.01	96	0	96	0.01	96	0	96	0.02	96	0.01	96	0.02	96	0.01	96	0.01
05, F, 15	49	0	23	0	36	0.01	23	0.01	36	0.02	23	0.02	36	0.01	23	0	33	0.01	23	0.01	33	0.01	23	0.01
02, F, 17	31	0.01	31	0.02	31	0.01	31	0.02	31	0.01	31	0.02	29	0	29	0.01	29	0.02	29	0.01	29	0.02	29	0.02
02, F, 22	31	0.01	31	0	31	0	31	0	31	0.01	31	0.01	31	0.01	31	0.01	31	0.01	31	0	31	0.02	31	0.01
04, F, 19	29	0	29	0.01	29	0.01	29	0.01	29	0.02	29	0.02	29	0.01	29	0.01	29	0.01	29	0.01	29	0.01	29	0.02
05, F, 18	25	0.01	25	0.01	25	0.01	25	0	25	0	25	0.01	25	0	25	0.02	25	0.01	25	0.01	25	0.01	25	0.02
08, F, 15	21	0	21	0.01	21	0	21	0.01	21	0.01	21	0	21	0	21	0	21	0	21	0.01	21	0.02	21	0.01
04, N, 17	3	0.01	3	0	3	0	3	0	3	0	3	0	3	0.01	3	0	3	0.01	3	0	3	0.01	3	0
13, N, 15	1	0	1	0.01	1	0	1	0	1	0.01	1	0	1	0	1	0	1	0	1	0.01	1	0	1	0
15, N, 15	1	0	1	0.01	1	0.01	1	0	1	0.01	1	0	1	0.01	1	0.01	1	0	1	0	1	0	1	0.01
00, N, 10	1	0.01	1	0	1	0.01	1	0	1	0.01	1	0.01	1	0	1	0	1	0.01	1	0	1	0	1	0
02, N, 20	1	0.01	1	0	1	0.01	1	0.01	1	0	1	0.01	1	0.01	1	0	1	0	1	0.01	1	0	1	0.01
03, N, 10	1	0.01	1	0	1	0.01	1	0.01	1	0	1	0	1	0	1	0	1	0.01	1	0	1	0.01	1	0
07, N, 10	1	0	1	0.01	1	0.01	1	0.01	1	0	1	0	1	0	1	0.01	1	0	1	0	1	0	1	0
07, N, 15	1	0	1	0	1	0.01	1	0.01	1	0	1	0	1	0	1	0.01	1	0.01	1	0	1	0.01	1	0.01
10, N, 10	1	0.01	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0.01	1	0.01	1	0
mean	136680	6.09	12528	2.02	7807	0.53	4766	0.85	3361	1.33	3050	1.56	7893	0.87	4520	0.99	5814	0.76	3761	0.89	2358	1.02	2267	1.19

Table 2: Efficiency of techniques

In terms of search space, using all techniques provides the best results. Using DDFF in energetic reasoning generally requires more time. However, this technique provides the best results both for instances “05, N, 15” and instances “10, N, 15”, “05, N, 17”, “07, F, 15”. Surprisingly, it allows us to directly prove instance “08, N, 15” as non-feasible. The techniques are not efficient for an instance such as “05, F, 20”, which is one of the rare instances for which there exist solutions of the first part of the branch-and-bound (all X_i variables are instantiated) with no complete solution for the second part of the branch-and-bound (all Y_i variables have to be instantiated). The best compromise between the reduction of the search space and the time required seems to be the method used with energetic reasoning and SS_3, although the other techniques may be very helpful for some instances (see, for example, “05, N, 15”).

6.3 Comparison with previous methods in the literature

In Table 3 we compare the best previous results in the literature with our method. We report the results obtained by the method of Fekete and Schepers [16] (“OPP”), the method of Clautiaux *et al.* [13] (“TSBP”) and the method of Beldiceanu and Carlsson [4] (“SWEEP”). The results for OPP and TSBP have been provided by the authors and were obtained respectively using a Pentium IV 3.2 GHz and a Pentium IV 2.6 GHz. The results for SWEEP have been obtained by a program provided by Beldiceanu and Carlsson, which runs in SICStus Prolog [21]. The latter have been run on the same computer as the one used for our method. We then report the results obtained by our method with the best combination of improving techniques (in terms of time consumed or in terms of search space). Moreover, at the root node of our methods, we use the preprocessing methods and the lower bounds used by Clautiaux *et al.* [13]. As we can see, the new method generally outperforms the previous result in the literature. However, some instances (see, for example, “04, N, 15” or “05, N, 17”) continue to be solved better by the method of Fekete and Schepers [16]. Our experiments also confirm the effectiveness of the sweep algorithm of Beldiceanu and Carlsson [4]. Indeed, used alone in the “basic” model, this pruning technique is able to solve most of the instances in competitive time. For some feasible benchmarks it provides the best results. Nevertheless, the method we propose in this paper, using improving techniques, dramatically reduces the search space in comparison with all previous methods, and the new method used in the absence of improving techniques is almost competitive with the previous methods. All instances can be solved in less than seven seconds with the new method, unlike the previous results, where some instances cannot be solved within 15 minutes with OPP or with SWEEP, or within 4 minutes for TSBP.

7 Conclusion

We describe a constraint-based scheduling model for the two-dimensional orthogonal packing problem. Using this model we provide a branch-and-bound

	previous methods						**constraint based scheduling model**									
	OPP		TSBP		SWEEP										LB	
									ER				ER		ER(LB)	
											SS		SS		SS	
instance	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
00, N, 23	-	-	9057985	289	-	-	3675002	160	88446	6.75	50095	6.09	42207	6.46	32926	26.50
00, N, 23	-	-	5968406	86	-	-	1496331	70	7134	1.20	3208	1.14	3283	1.21	1731	1.59
05, N, 15	1	0	1	0	-	-	1	0.00	1	0.01	1	0.01	1	0.01	1	0.01
05, N, 15	18369	2	1	0	111815	17.03	1	0.20	1	0.20	1	0.20	1	0.20	1	0.18
05, F, 20	547708	491	39387	2	78	0.00	27515	1.65	27513	1.63	27515	1.64	27513	1.65	27513	1.63
04, F, 20	22796	22	5876	3	238252	11.96	59795	3.68	9002	1.40	8336	1.71	5580	1.53	657	0.96
13, N, 15	1	0	1	0	201	0.08	1	0.00	1	0.01	1	0.00	1	0.00	1	0.00
10, N, 15	77	0	1	0	-	-	1	0.07	1	0.06	1	0.06	1	0.06	1	0.07
03, N, 16	9891	2	1592400	32	308003	62.90	14897	1.34	5183	1.02	8008	1.60	4743	1.38	3905	4.17
20, F, 15	325	0	747	1	71	0.01	7229	0.52	7229	0.60	6863	0.88	6863	0.94	27	0.37
04, N, 15	35	0	42844	1	107351	11.04	10181	0.95	4945	0.79	8002	1.37	4508	1.12	3286	3.02
03, N, 15	13	0	3707	1	5164	1.16	3562	0.69	3084	0.72	3085	0.94	2863	0.98	2217	2.25
10, N, 15	7	0	17603	1	47106	6.85	2622	0.42	2622	0.46	2437	0.57	2437	0.61	1844	1.66
07, N, 15	61	0	651	1	52477	10.39	2149	0.56	2143	0.59	2133	0.71	2128	0.74	1966	1.67
05, N, 17	1	0	993	1	175651	11.23	1673	0.35	1661	0.37	1649	0.47	1637	0.51	87	0.36
08, N, 15	1	0	261	1	9263	2.11	1420	0.29	1420	0.33	1420	0.36	1420	0.40	1	0.26
07, F, 15	92	0	90219	1	10216	1.88	1584	0.35	970	0.35	1358	0.40	921	0.39	301	0.43
03, F, 18	574	0	2605815	22	3205	0.33	1111	0.56	1056	0.55	1060	0.57	1015	0.56	838	0.70
04, N, 18	24593	10	434824	7	3529193	329.21	1032	0.31	812	0.31	697	0.31	579	0.31	149	0.32
02, F, 20	-	-	487230	12	951640	42.25	2245	0.93	275	0.84	236	0.84	165	0.85	30	0.82
04, F, 17	20270	13	1942682	26	66	0.00	474	0.53	474	0.54	474	0.55	474	0.54	474	0.57
13, N, 10	17	0	1468	0	3818	0.51	343	0.15	334	0.16	316	0.17	308	0.17	132	0.20
04, F, 15	933	0	3949	1	7679	1.29	287	0.43	286	0.43	285	0.44	285	0.44	284	0.58
03, N, 17	431	0	313007	4	7163	0.89	260	0.40	212	0.40	163	0.39	163	0.41	163	0.49
00, N, 15	127	0	96920	2	813351	136.59	610	0.46	102	0.44	96	0.44	89	0.44	80	0.52
20, F, 15	36	0	4355492	44	58	0.00	96	0.24	96	0.24	96	0.24	96	0.25	96	0.26
05, F, 15	1410	0	334434	3	3496	0.27	49	0.30	36	0.30	36	0.30	33	0.29	23	0.32
02, F, 17	28631	7	784796	12	3706	0.97	31	0.52	31	0.52	29	0.49	29	0.49	29	0.53
02, F, 22	190617	167	174943	4	4110	0.38	31	0.82	31	0.79	31	0.81	31	0.81	31	0.83
04, F, 19	786057	560	1075159	7	700	0.08	29	0.78	29	0.77	29	0.77	29	0.77	29	0.75
05, F, 18	262	0	20245458	126	149	0.02	25	0.50	25	0.51	25	0.50	25	0.51	25	0.52
08, F, 15	433	0	22658934	117	68	0.00	21	0.38	21	0.38	21	0.38	21	0.38	21	0.36
04, N, 17	1	0	1	1	18973	2.31	1	0.23	1	0.23	1	0.23	1	0.23	1	0.23
13, N, 15	1	0	91	0	102913	9.34	1	0.07	1	0.07	1	0.08	1	0.07	1	0.07
15, N, 15	1	0	1117	0	99905	6.78	1	0.17	1	0.17	1	0.15	1	0.16	1	0.16
00, N, 10	1	0	1	0	329	0.04	1	0.00	1	0.01	1	0.01	1	0.00	1	0.01
02, N, 20	1	0	1	1	15331	2.00	1	0.32	1	0.32	1	0.31	1	0.31	1	0.31
03, N, 10	1	0	417	0	7	0.00	1	0.09	1	0.09	1	0.09	1	0.09	1	0.08
07, N, 10	17	0	1758	0	819	0.19	1	0.15	1	0.17	1	0.14	1	0.15	1	0.15
07, N, 15	1	0	1	0	263	0.06	1	0.01	1	0.00	1	0.01	1	0.01	1	0.01
10, N, 10	5	0	1	0	135	0.02	1	0.05	1	0.05	1	0	1	0.05	1	0.05
mean	43521	33.53	1764380	19.73	179262.8	18.11	129527	6.09	4029	0.61	3115	0.65	2670	0.65	1924	1.32

Table 3: Comparison with previous methods

algorithm and several techniques improving the behavior of the method. Experimental results have shown the efficiency of these techniques. Nevertheless, the branching scheme used means that two-dimensional energetic reasoning is no more effective than classical energetic reasoning, and requires significantly more computational time. Whereas some instances are still better solved by the method of Fekete and Schepers [16] or by the method of Beldiceanu and Carlsson [4], the obtained procedure generally outperforms previous results in the literature.

Acknowledgements

The authors would like to thank Nicolas Beldiceanu and Mats Carlsson for providing their method using "sweep" algorithm.

References

[1] K.R. Baker, *Introduction to sequencing and scheduling*, John Wiley and Sons, 1974.

[2] Ph. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-based scheduling, applying constraint programming to scheduling problems*, International Series in Operations Research and Management Science, vol. 39, Kluwer, 2001.

[3] Ph. Baptiste, C. Le Pape, and W. Nuijten, *Satisfiability tests and time bound adjustments for cumulative scheduling problems*, Annals of Operational Research **92** (1999), 3305–3333.

[4] N. Beldiceanu and M. Carlsson, *Sweep as a generic pruning technique applied to the non-overlapping rectangles constraints*, Principles and Practice of Constraint Programming (CP'2001), Lecture Notes in Computer Science **2239** (2001), 377–391.

[5] N. Beldiceanu, M. Carlsson, and S. Thiel, *Sweep synchronization as a global propagation mechanism*, Computers and Operations Research **33** (2006), no. 10, 2835–2851.

[6] M. Boschetti and A. Mingozzi, *The two-dimensional finite bin packing problem. part I: New lower bounds for the oriented case*, 4OR **1** (2003), 27–42.

[7] A. Caprara, M. Locatelli, and M. Monaci, *Bilinear packing by bilinear programming*, Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005 (Michael Jünger and Volker Kaibel, eds.), Lecture Notes in Computer Science, vol. 3509, Springer, June 8-10 2005, pp. 377–391.

[8] A. Caprara and M. Monaci, *On the two-dimensional knapsack problem*, Operations Research Letters **32** (2004), 5–14.

[9] J. Carlier, F. Clautiaux, and A. Moukrim, *New reduction procedures and lower bounds for the two-dimensional bin-packing problem with fixed orientation*, Computers and Operations Research **to appear** (2005).

[10] J. Carlier and E. Néron, *Computing redundant resources for cumulative scheduling problems*, Accepted in European Journal of Operational Research (2005).

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

[12] F. Clautiaux, J. Carlier, and A. Moukrim, *A new exact method for the two-dimensional bin-packing problem with fixed orientation*, submitted to Operations Research Letters (2005).

[13] ______, *Exact method for the two-dimensional orthogonal packing problem*, European Journal of Operational Research **accepted** (2006).

[14] F. Clautiaux, A. Moukrim, and J. Carlier, *New data-dependent dual-feasible functions and lower bounds for a two-dimensional bin-packing problem*, submitted to Discrete Optimization (2006).

[15] J. Erschler, P. Lopez, and C. Thuriot, *Raisonnement temporel sous contraintes de ressource et problèmes d'ordonnancement*, Revue d'Intelligence Artificielle **5** (1991), no. 3, 7–32.

[16] S. Fekete and J. Schepers, *An exact algorithm for higher-dimensional orthogonal packing.*, Revised for Operations Research (2003).

[17] ______, *A combinatorial characterization of higher-dimensional orthogonal packing*, Mathematics of Operations Research **29** (2004), 353–368.

[18] ______, *A general framework for bounds for higher-dimensional orthogonal packing problems*, Mathematical Methods of Operations Research **60** (2004), 311–329.

[19] M. R. Garey and D. S. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*, Freeman, New York, 1979.

[20] E. Hadjiconstantinou and N. Christofides, *An exact algorithm for general, orthogonal, two-dimensional knapsack problem*, European Journal of Operational Research **83** (1995), 39–56.

[21] Intelligent Systems Laboratory, *Sicstus prolog users manual*, 2006.

[22] C. Le Pape, *Implementation of resource constraints in ILOG SCHEDULE: A library for the development of constraint-based scheduling systems*, Intelligent Systems Engineering **3** (1994), no. 2, 55–66.

[23] P. Lopez, J. Erschler, and P. Esquirol, *Ordonnancement de tâches sous contraintes : une approche énergétique*, RAIRO Automatique, Productique, Informatique Industrielle **26** (1992), no. 6, 453–481.

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

[25] D. Pisinger and M. Sigurd, *On using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem*, Tech. report, Department of Computer Science, University of Copenhagen, 2003.

[26] G. Wäscher, H. Haussner, and H. Schumann, *An improved typology of cutting and packing problems*, European Journal of Operational Research **to appear** (2006).