A New Constraint Programming Approach for the Orthogonal Packing Problem

Antoine Jouglet, François Clautiaux, Jacques Carlier and Aziz Moukrim

December 21, 2005

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. Feasibility tests and adjustments based on the resolution of subset-sum problems are suggested. We describe a generalization and improvements of the energetic-reasoning technique for the studied problem. Computational results confirm the efficiency of our method compared to previous methods.

1 Introduction

The two-dimensional orthogonal packing problem ($2OPP$) consists of 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* (C & P) problems, and more precisely to the class of *two-dimensional open-dimension problems* (2ODP) [32]. It is *NP-complete* [19] and is an issue in many packing problems. Caprara and Monaci [9] and Clautiaux *et al.* [13] have shown that this problem can effectively be avoided respectively for $2KP$ and for $2BP$ when good lower bounds are used. However when this problem occurs, it is practically difficult to solve for several instances.

A $2OPP$ instance D is a pair (I, B) in which $I = \{1, \ldots, n\}$ is the set of items i to pack and $B = (W, H)$ is a bin of width W and height H. If there is a solution for this instance, 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 the item i, denoted by (x_i, y_i), corresponds to the coordinates of its bottom-left corner. In the following the notation (w_i, h_i) may be used for item i.

Several exact methods are available in the literature for $2OPP$. Four ways of solving $2OPP$ are used: the first consists in packing items one by one in the bin [20,26,30], the second consists in using constraint programming techniques [29], the third uses a graph-theoretical model [16,17], and the fourth [14] first deals

with a relaxed problem, and then seek for a corresponding solution for $2OPP$. The two last methods [14, 16] are the best known so far. The graph-theoretical model of Fekete and Schepers [16, 17] allows a fast detection of subsets of items that cannot be packed in the bin. The relaxation proposed by Clautiaux *et al.* [14] is interesting, as it seems that for many non-feasible test cases, the relaxed problem is not feasible either.

The relaxation of [14] 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.

The goal of this paper is to propose a method that both uses the relaxation of Clautiaux *et al.* [14] and allows a fast detection of non-feasible sets. Constraint programming has been proved to be efficient for solving many optimization problems, including cumulative scheduling problems, which are close to the orthogonal packing problem. Thus we propose in this paper a new constraint programming method based on the new relaxation. We also enforce the initial program by adding new constraints, and we propose several methods to perform feasibility tests during the tree search. One of these method lies on the concept of two-dimensional energetic reasoning, which generalizes the classical energetic reasoning for the orthogonal packing problem. We also use the fact that previous lower bounds [11] can be used to strengthen the energetic reasoning.

We describe a new model and new techniques to solve the $2OPP$ problem. In Section 3, we propose a new constraint programming model for $2OPP$, and new methods for improving its resolution. We show in Section 4 how solving subset-sum problems can be useful to perform feasibility tests and coordinates' bounds adjustments. It has been shown that energetic reasoning was a powerful tool for scheduling problems. As $2OPP$ (and the relaxation introduced in [14]) can be seen as a special scheduling problem, applying such a method is straightforward. In Section 5, we show that it is possible to improve it by considering two-dimensional windows. A method for increasing the mandatory parts of the items is also described. In Section 6, we report computational experiments testing our new methods against the randomly-generated benchmarks described in [14]. We compare our algorithms to the method proposed by Fekete and Schepers [16] and Clautiaux *et al.* [14].

2 Literature Review

2.1 Exact Methods

Several methods in the literature [12, 20, 26] build a configuration step by step following the *leftmost-downward* strategy. The coordinates considered for an item i are the normal pattern coordinates (see [12]) in which 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 adjacent either to a packed item or to the

bottom edge of the bin. Pisinger *et al.* [29] 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, under, to the left, or the right of j.

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 as only one packing by class is enumerated. A graph $G_d = (V, E_d)$ is associated with each dimension, where $|V| = n$ is the number of items in the bin and $d \in \{w, h\}$ 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 propose a branch&bound method [16] to search for a pair of interval graphs with suitable properties, and then deduce a feasible packing. In comparison with the classical methods this method avoids a large number of redundancies, and it outperforms the method of Martello and Vigo [26].

Clautiaux *et al.* [14] propose two branch&bound methods for $2OPP$. The first algorithm ($LMAO$) is an improvement on the method proposed by Martello and Vigo [26]. Instead of testing the packing of items in each possible coordinate, the algorithm enumerates the packings of items only in the leftmost-downward position. It also tests the possibility of not packing any item in this position.

The second method ($TSBP$) is a two-step branch&bound method based on a new relaxation of the problem (see Figure 1). In the first step (*outer branch&bound method*), all solutions of a relaxed problem are enumerated. For each solution found, a second enumerative method is run (*inner branch&bound method*) to seek a solution for 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, like in [2,6,25,31], but their relaxation is different. All strips pertaining to a given item have to be packed at the same x-coordinate, even if they are not contiguous. So the resulting problem cannot be solved as a $1BP$. Using this relaxation, the problem consists in finding a suitable set of x-coordinates for the items. When a solution is sought for the relaxed problem (*i.e.* a valid list of x-coordinates for the items), the inner branch&bound method searches for a set of y-coordinates to obtain a solution for the initial problem. It appears that for non feasible instances, the relaxed problem rarely has a solution.

Practically speaking, the method of Fekete and Schepers [16] and the $TSBP$ method of Clautiaux *et al.* [14] are the most efficient. 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.

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

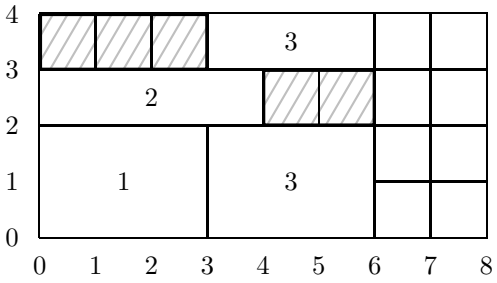

Figure 1: Relaxation of [14]

non-feasible instances of $2OPP$. The more recent polynomial time computable lower bounds are proposed by Carlier *et al.* [11]. The authors use the framework proposed by Fekete and Schepers [18] to compute lower bounds for this problem using so-called dual-feasible functions (DFF). Each pair of DFF is associated with a lower bound. Recently, Caprara *et al.* [8] 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 many cases DFF lead to excellent bounds for $2OPP$.

In a recent paper Clautiaux *et al.* [14] have proposed a feasibility test for a partial solution of $2OPP$. This bound is based on the application of so-called dual-feasible functions, which were defined by Johnson [21] and used several times to obtain lower bounds for bin packing problems [1, 8, 11, 18]. $2OPP$ can be seen as a $2BP$ instance with only one bin, so all bounds for $2BP$ can be used for $2OPP$.

2.2.1 Lower Bounds for $2BP$ [11]

A classical lower bound for $2BP$ and $2OPP$ is the continuous bound L_0. It is equal to the total area of the items divided by the area of one bin. One way of improving this method is to modify the size of the items so that the obtained instance is feasible if the initial instance is feasible. This can be done using *Dual-Feasible Functions* (DFF) [21] following the framework proposed by Fekete and Schepers [18] for *higher-dimensional bin-packing problems*. We use this framework in a previous paper [11], using a discretization of DFF.

Definition 2.1. *A* Discrete DFF f *is a discrete application from* $[0, X]$ *to* $[0, X']$ *(X and X' integers), such that* $x_1 + x_2 + \ldots + x_k \leq X \Rightarrow f(x_1) + f(x_2) + \ldots + f(x_k) \leq f(X) = X'$.

Carlier *et al.* also introduced the concept of *Data-Dependent DFF (DDFF)* [11]. These functions behave like DFF for the specific instance considered. They introduced two families of DFF (f_0^k and f_2^k) [11], similarly to Fekete and

Schepers [18], and a family of DDFF (f_1^k) [11]. A more complete description of the DDFF and the bounds of Carlier *et al.* can be found in [11], or [14] for $2OPP$.

- Let $k = 1, \ldots, \frac{C}{2}$. $f_0^k(x) = 0$ if $x < k$, $f_0^k(x) = x$ if $k \leq x \leq C - k$, and $f_0^k(x) = C$ if $C - k < x \leq C$.
- Let $k = 1, \ldots, \frac{C}{2}$. The data-dependent DFF f_1^k is defined for the value C and a list of integer values $c_1, c_2, \ldots, c_n$ ($I = \{1, \ldots, n\}$). We introduce the set $J = \{i \in I : \frac{1}{2}C \geq c_i \geq k\}$ and for a given integer Y, $M_C(Y, J)$ denotes the maximum number of items c_i such that $i \in J$, which can be packed together in a container of size Y. $f_1^k(x) = 0$ if $x < k$, $f_1^k(x) = 1$ if $k \leq x \leq \frac{1}{2}C$, and $f_1^k(x) = M_C(C - x, J)$ if $\frac{1}{2}C < x$.
- Let $k = 1, \ldots, \frac{C}{2}$. $f_2^k(x) = 2\lfloor \frac{x}{k} \rfloor$ if $x < \frac{1}{2}C$, $f_2^k(x) = \lfloor \frac{C}{k} \rfloor$ if $x = \frac{1}{2}C$, and $f_2^k(x) = 2\lfloor \frac{C}{k} \rfloor - 2\lfloor \frac{C-x}{k} \rfloor$ if $x > \frac{1}{2}C$.

2.2.2 Modification of the Instance [14]

In [14] Clautiaux *et al.* proposed a method to improve the quality of the lower bounds during the enumeration. 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 the items which are packed side by side to create new instances, which are more constrained than the initial instance.

Given a set of packed items 1, they create a set A'_1, which is more constrained than 1. The idea is to maximize the height or the width of the created items to obtain two instances. If several transformed items which are packed side by side are shallow they can be packed one above the other in the new instance. To avoid this situation they operate a second modification on the items. If the packed items have been cut into vertical strips, they add to each new item a height equal to H and the height of the bin is updated to $2H$. So the bounds can take into account the fact that these items cannot be packed one above the other. If the packed items do not fit the width of the bin, a dummy item with size (w^*, H) is created, w^* being the free width to the right of the packed items (see Figure 2). The same operation can be performed with respect to width.

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

Constraint programming is a programming paradigm aimed at solving combinatorial optimization problems that can be modeled as a *Constraint-Satisfaction Problem (CSP)*. The problem is 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

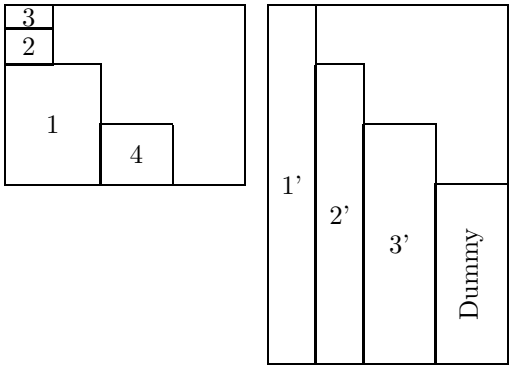

Figure 2: Creating a New Instance

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. If such an assignment exists, it is a solution to the Constraint-Satisfaction Problem. The power of constraint-programming method is largely due to the fact that constraints can be used in an active process called "constraint propagation" in which certain deductions are made to reduce computational effort. It removes values from the domains, deduces new constraints, and detects inconsistencies. In recent years, constraint-programming techniques have been successfully used to model and solve scheduling problems [4]. Thus "constraint-based scheduling" is defined as the discipline that studies how to solve scheduling problems using constraint-programming methods. Combining ideas from Artificial Intelligence and Operations Research, constraint-based scheduling is able to maintain the best of both approaches, *i.e.* , the flexibility of Artificial Intelligence scheduling systems and the efficiency of Operation Research algorithms.

The aim of this section is to propose a branch-and-bound algorithm with constraint-propagation techniques to solve $2OPP$. In subsection 3.1, we describe the $2OPP$ problem as an instance of the Constraint-Satisfaction Problem and describe a classical model. To enforce this model, we propose an original constraint-based scheduling model linked with the first. It allows us to improve the resolution of the problem thanks to effective propagation techniques. Next, we describe in subsection 3.2 our constraint-programing branch-and-bound algorithm and the techniques used to propagate the constraints of our models.

3.1 A constraint-based scheduling model

In constraint programming, $2OPP$ can be classically encoded in terms of variables and constraints and can be then expressed as a CSP.

Two variables X_i and Y_i are associated with each item i. They represent the coordinates of i in the bin. We note $D(X_i) = [x_i^{min}, x_i^{max}]$ 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 domain 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 then 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 associated solution is valid, once the variables are bound (the domain of variables being reduced to only one value) in such a way that all constraints are satisfied.

The principles of Constraint programming have been widely applied in the area of scheduling. Indeed, scheduling problems are usually defined by temporal and capacity constraints, which can be formulated explicitly in an easy way. In aim at improving the efficiency of the constraint-programming model above we propose to link it with a non-preemptive cumulative-scheduling problem associated with two relaxations of the $2OPP$ problem.

Baker [3] 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 $\{R_1, \ldots, R_n\}$. Each activity i has a processing time, requires a certain capacity from a resource R_k and has to be executed in 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 the activity i. Resources have a given capacity that can not 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 execute without interruption from its start time to its end time.

It is easy to see that the relaxed $2OPP$ problem, proposed by Clautiaux *et al.* [14] (see section 1) can be seen as a non-preemptive cumulative scheduling problem. Note that the same observation can be done with the problem obtained by exchanging the roles of the width and the height.

We model the two relaxed problems obtained as a non-preemptive cumulative scheduling problem with two resources in the following way. The two considered resources are called 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 first set $\{A_1^w, \ldots, A_n^w\}$ of activities, where n is the number of items of the considered $2OPP$ instance. Each activity A_i^w has a processing time equal to w_i, requires an amount h_i of the resource R_w, and has to be executed in the time window $[0, W]$. In the same way we define a second set $\{A_1^h, \ldots, A_n^h\}$ of activities. Each activity A_i^h has a processing time equal to h_i, requires an amount w_i of the resource R_h and has to be executed in the time window $[0, H]$.

To model this scheduling problem as a CSP, we introduce two variables $start(A)$ and $end(A)$ for each activity A, representing respectively the start time and the end time of A. For an activity A_i^w, which has to be scheduled on resource R_w, the domain variable of $start(A_i^w)$ is $[0, \ldots, W - w_i]$ and the domain variable of $end(A_i^w)$ is $[w_i, \ldots, W]$. Similarly, for an activity A_i^h, which has to be scheduled on resource R_h, the domain variable of $start(A_i^h)$ is $[0, \ldots, H - h_i]$ and the domain variable of $end(A_i^h)$ is $[h_i, \ldots, H]$. Since activities must execute without interruption from their start time to their end time, with each activity A_i^w, we associate the constraint $[start(A_i^w) + w_i = end(A_i^w)]$ and with each activity A_i^h, we associate the constraint $[start(A_i^h) + h_i = end(A_i^h)]$.

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 in the following way:

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

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

In other words, the sum of resource requirement of those activities A_i^w, for which $start(A_i^w) \leq t < end(A_i^w)$, at time t has to be lower than the resource capacity H of resource R_w and the sum of resource requirement of those activities A_i^h, for which $start(A_i^h) \leq t < end(A_i^h)$, at time t has to be lower than the resource capacity W of resource 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 instancied in such a way that all constraints are satisfied, the corresponding solution is valid.

Note that the domains of the variables are tightened during the branch-and-bound procedure as a result of a combination of decisions and constraints propagation which are described in the next section. Our constraint-based scheduling model is interesting for several reasons. First we express constraints redundant with constraints of the first model in a different way, which can improve the propagation of these constraints. Moreover, it is now possible to use powerful constraint-based scheduling propagation techniques dedicated to the non-preemptive scheduling problems (see for instance [4]) as we will see in next section.

3.2 Solving the problem

Several methods can be used to solve the problem. In our case, one may work equivalently on the X_i and Y_i variables (first constraint-programming model) or on the $start(A_i^w)$ and $start(A_i^h)$ variables (second constraint-based scheduling model). Moreover, one may choose the order in which the variable are instanciated and which value to use. Particularly, several branch-and-bound algorithms

described for the $2OPP$ problem (see Section 1 can be easily used with our constraint-programming models. Several ways have been tested experimentally.

We use the scheduling problem, *i.e.* , on the $start(A_i^w)$ and $start(A_i^h)$ variables. However, it can be easily translated according to the X_i and Y_i variables. We use a schedule-or-postpone method, which works as follows: at each step of the branch-and-bound method, we choose an unscheduled activity and we schedule it as early as possible, as allowed by the previously activities scheduled on the same resource. 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 according to this start time, the activity is postponed until its earliest start time has been removed. This removal can occur as a result of a combination of decisions and constraint propagations. The schedule is built using a depth first strategy.

The best experimental results in average are obtained by working first on the resource R_w and then on the resource R_h: among activities, the non-scheduled activity A_i^w of minimal earliest start time is chosen and scheduled on resource R_w. Once all $start(A_i^w)$ variables are instanciated, the non-scheduled activity A_i^h of minimal earliest start time is chosen and scheduled on resource R_h.

Note that this procedure is near to the two-step branch-and-bound algorithm of Clautiaux *et al.* [14]. Indeed, this algorithm is based on the relaxation of the problem that corresponds to the set $\{A_1^w, \ldots, A_n^w\}$ of activities to be scheduled on resource R_w. In the first step (*outer branch&bound method*), all solutions of the relaxed problem are enumerated. For each solution found, a second enumerative method is run (*inner branch&bound method*) to seek a solution for the initial problem corresponding to the current solution. The first step (*outer branch&bound method*) correspond to the stage in which we choose the activities A_i^w to be scheduled and the second step corresponds to the stage in which we choose the activities A_i^h to be scheduled. Thus, the obtained results confirm that the branch-and-bound of Clautiaux *et al.* is efficient. The main difference with the algorithm of Clautiaux *et al.* is that decisions taken during our branch-and-bound procedure and constraint-propagation techniques 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 tightened or instanciated and inconsistencies can be detected on resource R_h, even in the first stage of our branch-and-bound method. As a result, the search space is reduced more efficiently since waiting for the second stage of the algorithm is not necessary for an inconsistency detection on resource R_h.

Now, we describe the contraint-propagation techniques which allow us to tighten the domains of variables and to detect inconsistencies during the procedure. These methods allow us to reduce efficiently the search space. To propagate the resource constraints, we rely on a data structure called "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. In our case, Time-Table constraint propagation consists in maintaining arc-B-consistency [23] on the resource constraints described in previous section. Several ways can be used to improve the propagation of the constraints. Particularly, several methods relying on jobs' time-windows to propagate the resource constraints are used to update and adjust these sets. 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 more amount of resource than the capacity H of R_w. Hence, either A_i^w precedes A_j^w or A_j^w precedes A_i^h, *i.e.* , the disjunctive constraint holds between these activities. The same reasoning holds for activities $A_1^h, \ldots, A_n^h$ and resource R_h. Arc-B-consistency [23] can be then achieved by the formula

$$[h_i + h_j \leq H] \vee [end(A_i^w) \leq start(A_j^w)] \vee [end(A_j^w) \leq start(A_i^w)].$$

$$[w_i + w_j \leq W] \vee [end(A_i^h) \leq start(A_j^h)] \vee [end(A_j^h) \leq start(A_i^h)].$$

If n activities require the same resource, the constraint can be implemented as $n(n-1)/2$ disjunctive constraints. Finally, we use the edge-finding propagation techniques [4, 10], which are also able to adjust the time-windows of activities according to the resource constraint. Rather than considering only pairs of activities (i, j) to prove that i must precede j or j must precede i, the edge-finding constraint-propagation techniques makes a decision considering the order in which activities are processed on the resource. The goal is to determine that an activity i must be sequenced before (or after) a given set of activities [27, 28]. Two types of conclusions can then be done: new ordering relations ("edges" in the graph representing the possible orderings of jobs) and new time-bounds (earliest and latest start, and end times).

Other new efficient techniques, which allow us to improve the behavior of the branch-and-bound algorithm are described in the next sections.

4 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 resolution of subset-sum problems. This problem is a particular case of the knapsack problem, where the weight of each item is equal to its length. Let C be an integer value, and $L = c_1, c_2, \ldots, c_n$ a list of n integer values less than C. The subset-sum problem is the following:

$$Max \left\{ \sum_{i \in L} c_i \xi_i : \sum_{i \in L} c_i \xi_i \leq C, \xi_i \in \{0, 1\} \right\}$$

It consists in determining the subset of values of L with the larger sum, less than C. The subset-sum algorithm which solves this problem can be used to ensure that the current partial solution may lead to a *dominant* pattern, and also to improve feasibility tests.

4.1 Restricting the Search Space to Dominant Patterns

The *leftmost-downward* rule has been used by the methods based on the packing on items in the bin. Each item is packed in such a way that its left-hand and bottom edges are either adjacent to the bin, or adjacent to another item. We generalize this concept for a model where only intervals are known for the coordinates of the items.

Definition 4.1. *A pattern is said LMD-active if it respects the leftmost-downward rule.*

Similarly, a partial solution in which some variables X_i are not bound is said *LMD-active* if there is a set of ordinates included in the domains of X_i variables which leads to an LMD-active pattern.

If for a given item i of domain $X_i = [x_i^{min}, x_i^{max}]$ there is no item j such that $x_j^* + w_j = x_i^*$ with $x_i^* \in [x_i^{min}, x_i^{max}]$ and $x_j^* \in [x_j^{min}, x_j^{max}]$, then the left-hand edge of this item cannot be packed side-by-side with the bin or another item. The corresponding partial solution is not LMD-active. The property also holds for the height. It also helps reducing intervals for the items. This method can be generalized.

Proposition 4.1. *Let i be an item whose the domain of the X_i variable is not bound. If there is no set of items A^* such that $\sum_{j \in A^*} w_j = x_i^{min}$ then the lower bound of the interval can be increase of one without reducing the set of dominant solutions currently considered.*

Proof. Packing j at abscissa x_i^{min} cannot lead to an LMD-active pattern. Otherwise, there is a set of item A^* whose sum of width is equal to x_i^{min}, which contradicts the initial assumption. □

In Figure 3, the lower bound of the interval can be attained, whereas the upper bound x_i^{max} cannot. The largest value smaller than x_i^{max} which can be attained is x'_{max}. This value is the new upper bound of this interval. The existence of a set A^* can be checked in linear time using a subset-sum problem resolution. Considering several items at the same time could improve on the method, but it would be more time-consuming. Also the property holds for x_i^{max}.

Another type of partial solution cannot lead to a normal pattern. We take into account the fact that items and coordinates are integer. The case we consider occurs when there are not enough items with their right side in a given interval. Let $begin(x)$ be the set of items i such that $x_i^{min} = x_i^{max} = x$ and $end(x)$ the set of items such that $x_i^{min} + w_i \leq x \leq x_i^{max} + w_i$.

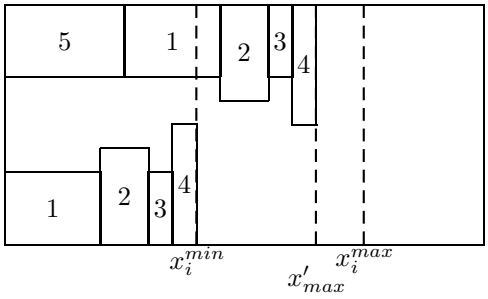

Figure 3: Updating the Intervals

Proposition 4.2. *If for a given x, $\sum_{i \in end(x)} h_i < |begin(x)|$, the current partial solution is not LMD-active.*

Proof. As data are integer, if two items are packed side by side, the height of the contiguous part is at least 1. So if there are not enough strips of height 1 among the items which can be packed side by side with the items packed at position x, the partial solution cannot lead to an LMD-active pattern. □

In Figure 4, the current configuration is LMD-active. For each item whose x-position is x, there is at least one strip pertaining to an item which may have its right-hand edge at position x.

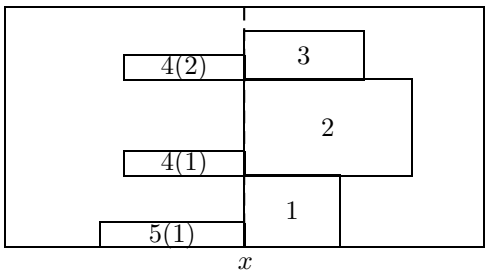

Figure 4: A LMD-Active Partial Solution

This method can be improved by considering only items whose y-coordinates are in a subset of coordinates.

4.2 Using Subset-Sum Algorithms for Performing Feasibility Tests

We now describe a feasibility test based on a subset-sum algorithm. A similar method is used as a preprocessing by Boschetti and Mingozzi [7], and by Fekete and Schepers [16,18] in their exact method. Our purpose is to describe how this method can be effectively used when the constraint model is used.

Consider P a partial solution for a $2OPP$ instance, and x a given x-coordinate. If there is not enough room above the items packed at position x for packing any item, then the area above is lost. This value 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 several items can be packed above a set of packed items, without the possibility of perfectly fitting the free space.

We consider a given interval $[\alpha, \beta]$ of x-coordinates. Let $A1$ be the set of items i whose x-coordinate is fixed such that the interval $[x_i, x_i + w_i]$ intersects $[\alpha, \beta]$. We suppose that the sum of the height of the items is constant on this interval. Let $A2$ be the set of items i of $A \setminus A1$ whose domain for the variable X intersects $[\alpha, \beta]$. We want to determine the minimum area lost above the items of $A1$. For this purpose, we consider the following subset-sum problem: $C = H - \sum_{i \in A1} h_i$, and the list L of values is composed of the heights h_i, $i \in A2$. When this value h^* is computed, the loss $\delta_{[\alpha,\beta]}$ for this interval can be obtained by multiplying h^* by the width of the interval $[\alpha, \beta]$.

The choice of the intervals $[\alpha, \beta]$ is important. We consider intervals where the remaining height is constant. There are at most $n_k + 1$ disjoint maximal intervals, where n_k is the number of packed items. The sum of the values computed for the intervals can be added to the total area of the items. If this value is greater than the total area of the bin, there is no solution including the current configuration. Let K be the set of considered disjoint intervals. If $\sum_{[\alpha,\beta] \in K} \delta_{[\alpha,\beta]} + \sum_{i \in A} w_i h_i > WH$ then a cut is performed.

In the current method a given item i can pertain to the solution of two subset-sum problems, while i could not be packed above the two corresponding sets at the same time. So more cuts can be performed if two or more intervals are considered at the same time. The drawback of this improvement is that the problem to solve becomes much more time consuming, as is no more computable in polynomial time. So we compute the loss for each interval independently.

The same method can be applied to the width when y-coordinates are fixed. The global method provides good results when the area of the items is close the area of the bin.

5 Two-Dimensional Energetic Reasoning

In this section, we describe the energetic reasoning concept, originally developed by Erschler et al. [15, 24] to solve cumulative scheduling problems. We suggest a generalization of energetic reasoning, which allows us to test the feasibility of orthogonal-packing patterns and find new adjustments.

For scheduling problems, deductions made by energetic reasoning are based on activities resource consumptions on some 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 on 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 fixed yet, we then discern the mandatory energy consumption on interval $[\alpha, \beta]$, which is obtained by considering the positions in which the processing of the activity is minimal on $[\alpha, \beta]$. According the balance sheets of energy supplied and consumed on some intervals, the energetic approach aims to develop 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.

We have already seen in Section 3 that the $2OPP$ problem can be modeled as two related cumulative scheduling problems in which the two coordinates of an item correspond to the starting times of activities in the two scheduling problems. Thus, contrary to activities in scheduling problems, the position of an item has to be fixed both on the width and on the height dimensions. We then suggest the following generalization of energetic reasoning. Instead of considering an interval $[\alpha, \beta]$, we consider a rectangular window $[\alpha, \beta, \gamma, \delta]$, $\alpha < \beta$ and $\gamma < \delta$, in which (α, γ), (α, δ), (β, γ) and (β, δ) are the vertices coordinates of the window.

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 most bottom-left, and top-right positions according to the domains of its coordinates variables, in which its consumption is minimal (see Figure 5). 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 a_i in the window $(\alpha, \beta, \gamma, \delta)$ (see Figure 5). 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 a_i in window $[\alpha, \beta, \gamma, \delta]$ is then

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

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

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

5.1 Feasibility tests and bounds adjustments

As in original energetic reasoning, it is obvious that the following proposition holds:

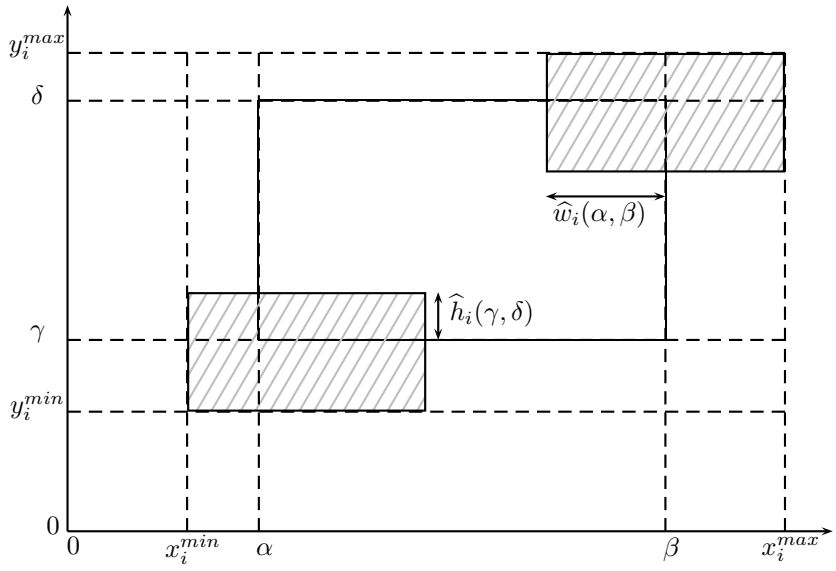

Figure 5: Mandatory part of item i in window $[\alpha, \beta, \gamma, \delta]$ considering the most bottom-left, and top-right positions.

Proposition 5.1. *if there is a feasible packing, then we have* $\forall \alpha, \beta \in [0, W)$, $\forall \gamma, \delta \in [0, H)$, $\widehat{E}(\alpha, \beta, \gamma, \delta) \leq (\beta - \alpha) \times (\delta - \gamma)$.

It means, that for every possible window, energy supplied by the bin has to be at least as large as 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 5.2) in the bin. If there exists a window for which the inequality is not verified, then the considered node cannot lead to a feasible solution and can be consequently pruned.

The values of $\widehat{E}(\alpha, \beta, \gamma, \delta)$ can also be used to adjust domain variables 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 a_i if a_i is packed the most at left according to the domain variable of x_i.
- $w_i^-(\beta) = \max(0, w_i - \max(0, x_i^{max} + w_i - \beta))$, the width of the mandatory part of a_i if a_i is packed the most at right according to the domain variable of x_i.
- $h_i^+(\gamma) = \max(0, h_i - \max(0, \gamma - y_i^{min}))$, the height of the mandatory part of a_i if a_i is packed the most at top according to the domain variable 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 a_i if a_i is packed the most at bottom according to the domain variable of y_i.

Note that we have clearly: $\widehat{w}_i(\alpha, \beta) = \min(\beta - \alpha, w_i^+(\alpha), w_i^-(\beta))$ and $\widehat{h}_i(\gamma, \delta) = \min(\delta - \gamma, h_i^+(\gamma), h_i^-(\delta))$.

Let i be an item and $[\alpha, \beta, \gamma, \delta]$ a window in the bin such that $\beta < x_i^{max} + w_i$. We verify if i can be packed before β, *i.e.* $X_i + w_i \leq \beta$. If i is packed leftmost according to the domain variable of X_i, then $w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta)$ is the mandatory part of i in window $[\alpha, \beta, \gamma, \delta]$. Thus, 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 packed leftmost. 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. In the best case, the height of the mandatory part of i in $[\alpha, \beta, \gamma, \delta]$ is $\min(\delta - \gamma, h_i)$. The lower bound of the domain of X_i, *i.e.* x_i^{min}, can be then adjusted to $\beta - w_i + \frac{1}{\min(\delta - \gamma, h_i)} \times (\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta) - (\beta - \alpha) \times (\delta - \gamma))$. Same reasonings can be applied if we try to pack i after α, under δ or over γ. Consequently the following proposition holds:

Proposition 5.2. *Consider an item i and a window $[\alpha, \beta, \gamma, \delta]$.*

- *If $\beta < x_i^{max} + w_i$ and $\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta) > (\beta - \alpha) \times (\delta - \gamma)$, then $x_i^{min} \geq \beta - w_i + \frac{1}{\min(\delta - \gamma, h_i)} \times (\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + w_i^+(\alpha) \times \widehat{h}_i(\gamma, \delta) - (\beta - \alpha) \times (\delta - \gamma))$.*
- *If $\alpha > x_i^{min}$ and $\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + w_i^-(\beta) \times \widehat{h}_i(\gamma, \delta) > (\beta - \alpha) \times (\delta - \gamma)$, then $x_i^{max} \leq \alpha - \frac{1}{\min(\delta - \gamma, h_i)} \times (\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + w_i^-(\beta) \times \widehat{h}_i(\gamma, \delta) - (\beta - \alpha) \times (\delta - \gamma))$.*
- *If $\delta < y_i^{max} + h_i$ and $\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + h_i^+(\gamma) \times \widehat{w}_i(\alpha, \beta) > (\beta - \alpha) \times (\delta - \gamma)$, then $y_i^{min} \geq \delta - h_i + \frac{1}{\min(\beta - \alpha, w_i)} \times (\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + h_i^+(\gamma) \times \widehat{w}_i(\alpha, \beta) - (\beta - \alpha) \times (\delta - \gamma))$.*
- *If $\gamma > y_i^{min}$ and $\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + h_i^-(\delta) \times \widehat{w}_i(\alpha, \beta) > (\beta - \alpha) \times (\delta - \gamma)$, then $y_i^{max} \leq \gamma - \frac{1}{\min(\beta - \alpha, w_i)} \times (\widehat{E}(\alpha, \beta, \gamma, \delta) - \widehat{E}_i(\alpha, \beta, \gamma, \delta) + h_i^-(\delta) \times \widehat{w}_i(\alpha, \beta) - (\beta - \alpha) \times (\delta - \gamma))$.*

Such adjustments can be done on all relevant windows (see Section 5.2) at each node of the search tree. Algorithm 1 can be then used to perform all feasibility tests and bound adjustments.

5.2 Characterization of relevant and irrelevant intervals

Baptiste *et al.* [5] studied the cumulative scheduling problem and showed that many intervals are not relevant for the energetic reasoning. Using our notation, their result can be stated as follows.

Algorithm 1: Feasibility tests and bound adjustments.

for *all 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**
      - $x_i^{min} = \max(x_i^{min}, \beta - w_i + \left\lceil \frac{w_i^+(\alpha) \times \widehat{h}_i(\gamma,\delta) - S}{\min(\delta - \gamma, h_i)} \right\rceil)$;
    - **if** $S < w_i^-(\beta) \times \widehat{h}_i(\gamma, \delta)$ **then**
      - $x_i^{max} = \min(x_i^{max}, \alpha - \left\lceil \frac{w_i^-(\beta) \times \widehat{h}_i(\gamma,\delta) - S}{\min(\delta - \gamma, h_i)} \right\rceil)$;
    - **if** $S < h_i^+(\gamma) \times \widehat{w}_i(\alpha, \beta)$ **then**
      - $y_i^{min} = \max(y_i^{min}, \delta - h_i + \left\lceil \frac{h_i^+(\gamma) \times \widehat{w}_i(\alpha,\beta) - S}{\min(\beta - \alpha, w_i)} \right\rceil)$;
    - **if** $S < h_i^-(\delta) \times \widehat{w}_i(\alpha, \beta)$ **then**
      - $y_i^{max} = \min(y_i^{max}, \gamma - \left\lceil \frac{h_i^-(\delta) \times \widehat{w}_i(\alpha,\beta) - S}{\min(\beta - \alpha, w_i)} \right\rceil)$;

Proposition 5.3. *Let O_1 and O_2 be two sets.*

1. $O_1 = \{x_i^{min}, i \in I\} \cup \{x_i^{max}, i \in I\} \cup \{x_i^{min} + w_i, i \in I\}$
2. $O_2 = \{x_i^{max} + w_i, i \in I\} \cup \{x_i^{min} + w_i, i \in I\} \cup \{x_i^{max}, i \in I\}$

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

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

Proposition 5.4. *Consider a given partial solution for a $2OPP$ instance and the four following sets of x- and y-coordinates.*

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

If for all $\alpha \in O_1^W$, $\beta \in O_2^W$, $\gamma \in O_1^H$, $\delta \in O_2^H$, $\widehat{E}(\alpha, \beta, \gamma, \delta) > (\beta - \alpha)(\delta - \gamma)$, 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 taken 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 $[x_i^{min}, x_i^{max}]$. Only the values related to the height are modified. So when the proposition of Baptiste *et al.* [5] is applied to this new problem, the same set of relevant x-coordinates is found, whatever the bounds γ and δ. So the values in O_1^W, O_2^W are sufficient.

The 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. □

5.3 Using DDFF in Energetic Reasoning

The item of size $(\widehat{w}_i(\alpha, \beta), \widehat{h}_i(\gamma, \delta))$ is denoted by $\widehat{a}_i(\alpha, \beta, \gamma, \delta)$ and the set of items corresponding to the mandatory parts of the items of A is denoted by $\widehat{A}(\alpha, \beta, \gamma, \delta)$.

Proposition 5.5. *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 $2OPP$ problem with the current domain variables.*

Several methods can be used to show that a given obtained $2OPP$ problem is not feasible, the faster being verifying that the continuous lower bound does not exceed one. This corresponds to the feasibility test described in previous section. More complex lower bounds can be used, like the bounds proposed in [11,14]. If the number of items is small, an exact method can also be applied.

6 Computational Results

In Table 1, we compare the results obtained by our new method (CP) with the method of Fekete and Schepers [16] (OPP) and the method of Clautiaux *et al.* [14] (TSBP). The three methods were run on the benchmarks proposed by Clautiaux *et al.* [14]. The instances are randomly generated for two given parameters: the number of items and the discrepancy between the total area of the items and the area of the bin (see [14]). These instances are available on the web: *http://www.hds.utc.fr/˜fclautia/research.html*. For each method, and each test case, we report the number of nodes enumerated and the CPU Time needed (in seconds). The maximum time allowed for an instance is set to 15 minutes.

7 Conclusion

We propose a new ...

References

[1] Claudio Alves, *Cutting and packing: Problems, models and exact algorithms*, Ph.D. thesis, Universidade do Minho, Portugal, 2005.

[2] A. Amaral and A. N. Letchford, *An improved upper bound for the two-dimensional non-guillotine cutting problem*, submitted to publication (2001).

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

[4] 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.

[5] 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.

[6] M. Boschetti, E. Hadjiconstantinou, and A. Mingozzi, *New upper bounds for the two-dimensional orthogonal non guillotine cutting stock problem*, IMA Journal of Management Mathematics **13** (2002), 95–119.

Instance			Nodes			CPU Time		
ϵ	n	feas.	OPP	TSBP	CP	OPP	TSBP	CP
00	10	N	1	1		0	0	
00	15	N	127	96920		0	2	
00	23	N	-	5968406		-	86	
00	23	N	-	9057985		-	289	
02	17	F	28631	784796		7	12	
02	20	F	-	487230		-	12	
02	22	F	190617	174943		167	4	
02	20	N	1	1		0	1	
03	10	N	1	417		0	0	
03	15	N	13	3707		0	1	
03	16	N	9891	1592400		2	32	
03	17	N	431	313007		0	4	
03	18	F	574	2605815		0	22	
04	15	F	933	3949		0	1	
04	17	F	20270	1942682		13	26	
04	19	F	786057	1075159		560	7	
04	20	F	22796	5876		22	3	
04	15	N	35	42844		0	1	
04	17	N	1	1		0	1	
04	18	N	24593	434824		10	7	
05	15	F	1410	334434		0	3	
05	18	F	262	20245458		0	126	
05	20	F	547708	39387		491	2	
05	15	N	1	1		0	0	
05	17	N	1	993		0	1	
05	15	N	18369	1		2	0	
07	15	F	92	90219		0	1	
07	10	N	17	1758		0	0	
07	15	N	1	1		0	0	
07	15	N	61	651		0	1	
08	15	F	433	22658934		0	117	
08	15	N	1	261		0	1	
10	10	N	5	1		0	0	
10	15	N	77	1		0	0	
10	15	N	7	17603		0	1	
13	10	N	17	1468		0	0	
13	15	N	1	91		0	0	
13	15	N	1	1		0	0	
15	10	N	25	331		0	0	
15	15	N	1	1117		0	0	
20	15	F	325	747		0	1	
20	15	F	36	4355492		0	44	

Table 1: Comparison with previous methods

Instance			Nodes			CPU Time		
ϵ	n	feas.	OPP	TSBP	CP	OPP	TSBP	CP
00	10	N	1	1		0	0	
00	15	N	127	96920		0	2	
00	23	N	-	5968406		-	86	
00	23	N	-	9057985		-	289	
02	17	F	28631	784796		7	12	
02	20	F	-	487230		-	12	
02	22	F	190617	174943		167	4	
02	20	N	1	1		0	1	
03	10	N	1	417		0	0	
03	15	N	13	3707		0	1	
03	16	N	9891	1592400		2	32	
03	17	N	431	313007		0	4	
03	18	F	574	2605815		0	22	
04	15	F	933	3949		0	1	
04	17	F	20270	1942682		13	26	
04	19	F	786057	1075159		560	7	
04	20	F	22796	5876		22	3	
04	15	N	35	42844		0	1	
04	17	N	1	1		0	1	
04	18	N	24593	434824		10	7	
05	15	F	1410	334434		0	3	
05	18	F	262	20245458		0	126	
05	20	F	547708	39387		491	2	
05	15	N	1	1		0	0	
05	17	N	1	993		0	1	
05	15	N	18369	1		2	0	
07	15	F	92	90219		0	1	
07	10	N	17	1758		0	0	
07	15	N	1	1		0	0	
07	15	N	61	651		0	1	
08	15	F	433	22658934		0	117	
08	15	N	1	261		0	1	
10	10	N	5	1		0	0	
10	15	N	77	1		0	0	
10	15	N	7	17603		0	1	
13	10	N	17	1468		0	0	
13	15	N	1	91		0	0	
13	15	N	1	1		0	0	
15	10	N	25	331		0	0	
15	15	N	1	1117		0	0	
20	15	F	325	747		0	1	
20	15	F	36	4355492		0	44	

Table 2: Efficiency of the lower bounds and energetic reasoning

[7] 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.

[8] 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.

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

[10] C. Carlier and E. Pinson, *A practical use of jackson's preemptive schedule for solving the job-shop problem*, Annals of Operations Research **26** (1990), 269–287.

[11] 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).

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

[13] 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).

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

[15] J. Erschler, P. Lopez, and C. Thuriot, *Raisonnement temporel sous contraintes de ressource et problmes 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] D. S. Johnson, *Near optimal bin packing algorithms*, 1973, Dissertation, Massachussetts Institute of Technology, Cambridge, Massachussetts.

[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] O. Lhomme, *Consistency techniques for numeric CSPs*, Thirteenth International Joint Conference on Artificial Intelligence, Chambéry, France (1993).

[24] 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.

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

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

[27] W. Nuijten, *Time and resource constrained scheduling: a constraint satisfaction approach*, Ph.D. thesis, Eindhoven University of Technology, 1994.

[28] W. Nuijten and E.H.L. Aarts, *A computational study of constraint satisfaction for multiple capacitated job-shop scheduling*, European Journal of Operational Research **90** (1996), no. 2, 269–284.

[29] 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.

[30] G. Scheithauer, *Equivalence and dominance for problems of optimal packing of rectangles*, Ricerca Operativa **83** (1998), 3–34.

[31] ———, *LP-based bounds for the container and multi-container loading problem*, International Transactions in Operational Research **6** (1999), 199–213.

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