Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A single machine scheduling problem with two-dimensional vector packing constraints

Jean-Charles Billaut[a], Federico Della Croce[b,c,*], Andrea Grosso[d]

[a] Université François-Rabelais Tours, CNRS LI EA 6300, OC ERL-CNRS 6305 64 avenue Jean Portalis, 37200 Tours, France
[b] D.A.I., Politecnico di Torino, Italy
[c] CNR, IEIIT, Torino, Italy
[d] D.I., Universitá di Torino, Italy

ARTICLE INFO

Article history:
Received 19 May 2014
Accepted 23 November 2014
Available online 9 December 2014

Keywords:
Scheduling
Two-dimensional vector packing
Recovering beam search
Matheuristics

ABSTRACT

We consider a scheduling problem where jobs consume a perishable resource stored in vials. It leads to a new scheduling problem, with two-dimensional jobs, one dimension for the duration and one dimension for the consumption. Jobs have to be finished before a given due date, and the objective is to schedule the jobs on a single machine so that the maximum lateness does not exceed a given threshold and the number of vials required for processing all the jobs is minimized. We propose a two-step approach embedding a Recovering Beam Search algorithm to get a good-quality initial solution reachable in short time and a more time consuming matheuristic algorithm. Computational experiments are performed on the benchmark instances available for the two-dimensional vector packing problem integrated with additional due dates to take into account the maximum lateness constraints. The computational results show very good performances of the proposed approach that remains effective also on the original two-dimensional vector packing instances without due dates where 7 new bounds are obtained.

1. Introduction

Scheduling problems are well studied in the literature since the 1950s and several books and recent review papers show the wide variety of problems considered today (Brucker, 2007; Hartmann & Briskorn, 2010; Pinedo, 2012; Ruiz & Vázquez-Rodriguez, 2010). In the same way, bin packing and vector packing problems have received a great attention in the literature and some survey papers are dedicated to these problems (De Carvalho, 2002; Lodi, Martello, & Vigo, 2002). In this paper, we introduce a new category of scheduling problems where vector packing constraints have to be considered together with the scheduling problem. These packing constraints are known in the literature as two-dimensional vector packing constraints (the problem may also be called two-constraint bin packing problem). We can refer to (Alves, de Carvalho, Clautiaux, & Rietz, 2014; Caprara & Toth, 2001; Garey, Graham, Johnson, & Yao, 1976; Spieksma, 1994) for resolution methods of this problem with m dimensions.

The origin of the problem comes from the production of chemotherapy drugs for cancer treatment by intravenous injection. In (Mazier, Billaut, & Tournamille, 2010), the authors describe this particular production environment and present a resolution method for a static or dynamic environment. In this production environment, the jobs to perform are called "*preparations*" and the raw materials are called *monoclonal antibodies* ("*products*" in the following). The monoclonal antibodies bind to specific cancer cells and induce an immunological response against the target cell. These products can be stored in vials for a long time before use, under specific storage temperatures and conditions. For some antibodies, freezing at −20 degree Celsius or −80 degree Celsius in small aliquots is the optimal storage condition. However, once a vial is opened or once the active agent has been mixed with a solute, it must be used before a given time limit, in order to keep intact the properties of the anticancer active agents. The maximum delay of use after opening depends on the agent and may vary between several hours to several days. In the meantime, the product has to be stored in a freezer or in a fridge for temperature and darkness reasons. If the time limit of use is exceeded, the monoclonal antibodies have lost their properties and they have to be destroyed according to a specific process. In other words, the time period between the starting time of the first preparation using a product in a vial and the completion time of the last preparation assigned to the same vial cannot be greater than the life time of the product. Furthermore, without loss of generality we assume that the total consumption associated to one vial cannot be greater than the total volume of the vial and we also assume

* Corresponding author. Tel.: +39 011 5647059; fax: +39 011 5647099.
E-mail addresses: jean-charles.billaut@univ-tours.fr (J. Billaut), federico.dellacroce@polito.it (F. Della Croce), grosso@di.unito.it (A. Grosso).

http://dx.doi.org/10.1016/j.ejor.2014.11.036
0377-2217/© 2014 Elsevier B.V. All rights reserved.

that two different vials cannot be assigned to the same preparation. We have here the two dimensions of the vector packing problem.

The cost of these products is very important. In Tours, the UBCO production center (described in Mazier et al., 2010) produces around 150 preparations per day. A preparation has an average cost of 400 euros, but it can reach 15,000 euros in certain cases. It is clear that the saving of these products may have an economic impact, absolutely not negligible. In the problem that we consider, one objective function is related to the waste of products, which we want to minimize. Notice that in such a context, the deadline for the use of the raw material becomes a variable of the problem, which is directly related to the production scheduling decisions: each time the life duration or the capacity of the vial is exceeded, a new vial is opened. Another version of this problem has been studied in Billaut (2011) and Billaut, Esquirol, and Tournamille (2011).

Furthermore, because each preparation has to be delivered to a patient for a given due date, another objective of the problem is to minimize the maximum lateness related to these due dates.

The same type of problem arises in concrete production because once prepared, the concrete has to be used within a given amount of time. Similarly, in food production, after the food is out of the freezer, it has to be used within a given time limit. Another possible application of this study arises for the resources management in the cloud environment (Padhy & Patra, 2013). For cloud providers, the problem is to allocate resources dynamically in the form of virtual machines to end users. Each task needs a virtual machine, i.e. a given quantity of RAM and of CPU. The tasks assigned to a resource cannot require more RAM and more CPU than the available quantity. Our problem, while different, presents also some similarities with parallel batch scheduling problems. But to the best of our knowledge, the problem that we consider in this paper has never been considered in the literature before.

The paper is organized as follows. In Section 2, the notations are introduced and the problem is formally defined. An MILP model is given. In Section 3, a recovering beam search and a matheuristic algorithm are proposed. These methods are tested on two dimensional packing benchmark instances, with due dates considerations and without due dates, for a comparison with methods dedicated to this problem. The results presented in Section 4 show that the proposed methods have very good performances, even without due date considerations. In this latter case, 7 new bounds are obtained with respect to the available literature. Section 5 presents the conclusions and some future research directions.

2. Problem statement and notations

We consider a simplified version of the problem of chemotherapy drugs production, assuming that there is only one machine and only one type of product (raw material). We have a set of n jobs to schedule on a single machine. To each job $J_j \in \{1, \dots, n\}$ is associated a processing time p_j, a consumption b_j and a due date d_j. Without loss of generality, the jobs are supposed to be numbered in EDD order, i.e. $d_1 \le d_2 \le \dots \le d_n$. The life duration of the product after opening is equal to T and the volume of one vial is equal to V. We assume without loss of generality that $p_j < T$ and $b_j < V, \forall j, 1 \le j \le n$. The number of vials is not limited but supposed to be bounded by n. We denote by C_j the completion time of J_j, L_j the lateness defined by $L_j = C_j - d_j$ and the maximum lateness is defined by $L_{\max} = \max_{1 \le j \le n} L_j$. We assume that the maximum lateness is bounded by a given value Q.

$$L_{\max} \le Q \tag{1}$$

We also assume that the remaining quantity of product in the last opened vial is lost at the end of the time horizon. Therefore, minimizing the quantity of lost product is equivalent to minimizing the number of vials that are opened. That is the reason why the problem is a mix between a scheduling problem and a two dimensional

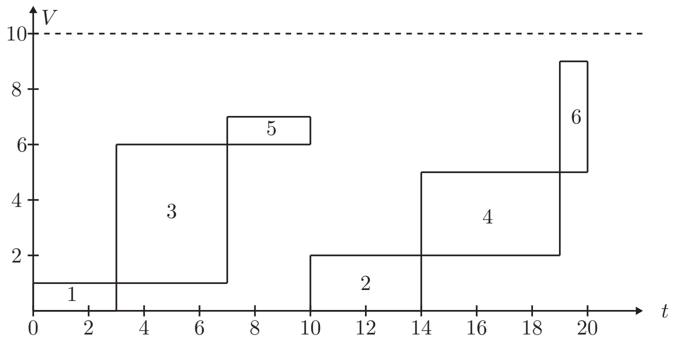

Fig. 1. Two-dimensional Gantt chart representation of schedule $(J_1, J_3, J_5, J_2, J_4, J_6)$.

vector packing problem. Without due dates (or with extremely large due dates), the problem is exactly a two dimensional vector packing problem and thus clearly strongly NP-hard. With a big value of T and a big value of V, the problem is the classical single machine problem with the $L_{\max}$ minimization. In the following, we call a *bin*, the set of jobs performed with the same vial. It is well known from the literature (Spieksma, 1994) that a trivial lower bound on the number of bins is the following:

$$\mathrm{LB} = \max\left(\left\lceil \sum_{j=1}^{n} p_j/T \right\rceil, \left\lceil \sum_{j=1}^{n} b_j/V \right\rceil\right). \tag{2}$$

We illustrate the problem by a numerical example.

2.1. Example

We consider a set of six jobs, $T = 10$ and $V = 10$.

j	1	2	3	4	5	6
p_j	3	4	4	5	3	1
b_j	1	2	5	3	1	4
d_j	7	9	11	13	14	16

Schedule $(J_1, J_3, J_5, J_2, J_4, J_6)$ is represented in Fig. 1. In this two dimensional Gantt chart, a job J_j is represented by a rectangle with the duration p_j on the x-axis and the consumption b_j on the y-axis. Jobs of the same bin are connected by the south-west corner of the rectangle. The first job of a bin is put on the x-axis. In Fig. 1, one can see that job J_1 is the first job of the bin composed by jobs $\{J_1, J_3, J_5\}$ and job J_2 is the first job of the bin composed by jobs $\{J_2, J_4, J_6\}$. Job J_2 cannot be included in the first bin because the duration of this bin would exceed T. The maximum lateness of this sequence is equal to $L_{\max} = \max(-4, 5, -4, 6, -4, 4) = 6$ and this schedule requires two bins. Notice that schedule $(J_1, J_2, J_3, J_4, J_5, J_6)$ is optimal for the $L_{\max}$, but requires three bins (see Fig. 2).

2.2. Dominance condition

Proposition 1. *In a bin, the jobs are scheduled in EDD order.*

This condition is clear because the order in a bin does not modify the number of bins that are used, and tends to improve the $L_{\max}$ value.

2.3. MILP formulation

We now propose an MILP formulation of the problem where the aim is to minimize the number of bins, assuming that the $L_{\max}$ value is bounded.

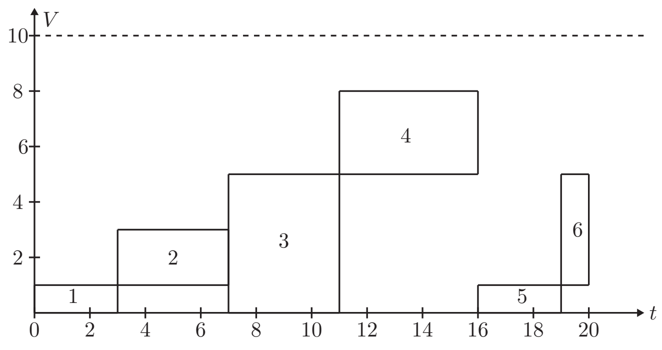

Fig. 2. Two-dimensional Gantt chart representation of schedule $(J_1, J_2, J_3, J_4, J_5, J_6)$.

We denote by $u_k \in \{0, 1\}$ a boolean variable equal to 1 if bin k is used, and 0 otherwise and by $x_{j,k} \in \{0, 1\}$ a boolean variable equal to 1 if job J_j is assigned to bin k, and 0 otherwise. The problem can be formalized by a binary linear program as follows.

$$\text{minimize} \sum_{k=1}^{n} u_k \tag{3}$$

subject to

$$\sum_{k=1}^{n} x_{j,k} = 1, \forall j \in \{1, \dots, n\} \tag{4}$$

$$\sum_{j=1}^{n} p_j x_{j,k} \le T u_k, \forall k \in \{1, \dots, n\} \tag{5}$$

$$\sum_{j=1}^{n} b_j x_{j,k} \le V u_k, \forall k \in \{1, \dots, n\} \tag{6}$$

$$\sum_{h=1}^{k-1}\sum_{i=1}^{n} p_i x_{i,h} + \sum_{i=1}^{j} p_i x_{i,k} \le d_j + Q + M(1 - x_{j,k}),$$
$$\forall j \in \{1, \dots, n\}, \forall k \in \{1, \dots, n\} \tag{7}$$

$$u_{k+1} \le u_k, \forall k \in \{1, \dots, n\} \tag{8}$$

Constraints (4) ensure that each job is performed by using one vial (is assigned to one bin). Constraints (5) and (6) correspond to the temporal and capacity limits. Constraints (7) suppose that job J_j is in bin k and correspond to the bound on the $L_{\max}$: $\sum_{h=1}^{k-1}\sum_{i=1}^{n} p_i x_{i,h}$ is the completion time of the $k-1$th bin, $\sum_{i=1}^{j} p_i x_{i,k}$ is the completion time of job J_j in its bin (remember that the jobs are numbered in EDD order). Constraints (8) ensure that the bins are used in their index increasing order. This model contains $n(n+1)$ binary variables and $n^2 + 4n$ constraints.

3. Resolution methods

Two original resolution methods are proposed for solving the problem. The first one is a recovering beam search algorithm and the second one is a matheuristic algorithm.

3.1. Recovering beam search algorithm

The *Beam Search* algorithm is a truncated branch-and-bound method where a subset of w nodes at each level are selected for branching. w is called the *beam width*. This method was first proposed in Ow and Morton (1988). For the selection of nodes, each node is evaluated by a combination of a lower bound (LB) and an upper bound (UB), generally a weighted sum $V = (1 - \alpha)\,\text{LB} + \alpha\text{UB}$. Because the selected nodes are not necessarily the best at a given level of the tree, among the set of possible nodes of a pure branch-and-bound algorithm, a recovering phase is applied in the *Recovering Beam Search* (RBS) algorithm . The aim of this phase is to recover from wrong decisions jumping to a better node at the same level of the search tree. For a detailed description of RBS we refer to Della Croce, Ghirardi, and Tadei (2004). Fig. 3 illustrates the exploration of the tree in RBS with a beam width of 2.

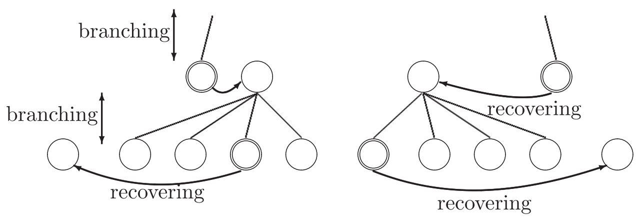

Fig. 3. RBS exploration.

The RBS method has already been successfully used in the literature for solving scheduling problems (Della Croce et al., 2004; Dong, Huang, & Chen, 2009; Esteve, Aubijoux, Chartier, & T'kindt, 2006; Ghirardi & Potts, 2005; Rakrouki, Ladhari, & T'Kindt, 2012; Valente, 2010). In order to apply the RBS approach, it is necessary to specify its main components, namely: branching scheme, lower bound, upper bound and recovering step. In this problem, a node of the search tree is defined by a partial sequence $S(\sigma)$ of the set σ of scheduled jobs, a set $\overline{\sigma}$ of unscheduled jobs, a lower bound LB(σ), and an upper bound UB(σ).

At the root node, the initial sequence of unscheduled jobs is determined as follows. Starting from the EDD sequence, the following steepest descent algorithm is used to reduce the number of bins, without violating the constraint on the $L_{\max}$. In a first phase, the job in position $k = n$ is swapped with the smallest job among its predecessors (the job J_j with minimum $p_j \times b_j$ value), if it is feasible, and then the process iterates with the job in position $k - 1$. In a second phase, the job in position $k = n$ is inserted in a bin scheduled before, that can accept it without violating the bounds and without violating the feasibility constraint. And the process iterates with the job in position $k - 1$.

The branching scheme is the typical n-ary branching: the sequence is constructed by adding one job at a time starting from position 1 and the search tree is such that a node at level k indicates which is the job placed in position k. For the lower bound computation, we denote by $Bin(S)$ a function that computes in $O(n)$ time the number of bins used by a partial sequence S and the sum of jobs processing times and of jobs consumptions in the last bin (respectively called $RestP(S)$ and $RestB(S)$). Algorithm 1 provides the lower bound value computed at a generic node of the search tree.

Algorithm 1 LB(σ).

```
$lb = Bin(S(\sigma)) - 1$
$sp = RestP(S(\sigma)) + \sum_{j\in\overline{\sigma}} p_j$
$sb = RestB(S(\sigma)) + \sum_{j\in\overline{\sigma}} b_j$
$lb = lb + \max(\lceil sp/T \rceil, \lceil sb/V \rceil)$
return $lb$
```

Basically the lower bound is the trivial lower bound expressed by Eq. (2) updated in order to take into account the set of bins already filled by the partial sequence.

For the upper bound computation, Algorithm 2 provides the upper bound value computed at a generic node of the search tree, where a partial sequence $S(\sigma)$ has already been defined. Let $S_{\text{father}}(\overline{\sigma})$ denote the sequence of the jobs in $\overline{\sigma}$ obtained by keeping for these jobs the same order they had in the upper bound of the father node in the search tree. Also, let $S(\sigma)//S_{\text{father}}(\overline{\sigma})$ denote the concatenation of subsequences $S(\sigma)$ and $S_{\text{father}}(\overline{\sigma})$.

Algorithm 2 UB(σ).

```
$S'(\sigma) = S(\sigma)//S_{\text{father}}(\overline{\sigma})$
If $S'(\sigma)$ is unfeasible (constraint on $L_{\max}$ violated), then $ub = \infty$
Else $ub = Bin(S'(\sigma))$
Return($ub$)
```

The rationale here is to keep for the jobs in $\overline{\sigma}$ the same order they had in the previous branch of the search tree (in other words, the order of the unscheduled jobs is the same as in the root node at lower levels of the tree). If the sequence provided by the upper bound computation is unfeasible, then, UB(σ) $= \infty$ and correspondingly the related search tree node is discarded. The evaluation of a node is given by $V(\sigma) = (1-\alpha)\text{LB}(\sigma) + \alpha\text{UB}(\sigma)$. Preliminary testing indicated that best results were obtained with $\alpha = 0.2$. Typically, there is a huge number of solutions with the same value of upper and lower bounds. In order to make a difference between solutions having the same value at a level of the tree, a second-level evaluation has been used. Let us define the "*surface of a bin*", denoted by Surf(k), as follows:

$$\text{Surf}(k) = \sum_{J_j \in B_k} p_j b_j$$

Whenever two sequences have the same value of $V(\sigma)$, we break ties by selecting the sequence with the smallest surface of the last bin.

The recovering phase is composed by two types of neighborhood called SWAP and EBSR (extraction and backward shift reinsertion) (Della Croce et al., 2004). Consider two jobs J_i and J_j in $S(\sigma) = \pi_1 J_i \pi_2 J_j \pi_3$, with J_i before J_j. SWAP generates sequence $\pi_1 J_j \pi_2 J_i \pi_3$ and EBSR generates sequence $\pi_1 J_j J_i \pi_2 \pi_3$. More precisely, considering a node S with a sequence σ of scheduled jobs and a sequence $\overline{\sigma}$ of unscheduled jobs, for any job J_i in σ, and for any job J_j in $\overline{\sigma}$, the SWAP operator is tested, and the best feasible neighbor improving UB(σ) is kept. Then, after this phase, the EBSR operator is applied for any job J_i in σ and for any job J_j in $\overline{\sigma}$ and again, the best feasible neighbor improving UB(σ) is kept. No further attempt is made after applying EBSR operator.

3.2. Matheuristic algorithm

Matheuristics constitute a combination of exact methods and metaheuristics that exploit the strength of both approaches within a "hybrid" procedure. A distinguishing feature is the exploitation of nontrivial mathematical programming tools as part of the solution process. We refer to Della Croce, Grosso, and Salassa (2013) for a general description of matheuristics.

Here, we propose a *Variable Partitioning Local Search* (VPLS) procedure which can be seen as a matheuristic neighborhood search approach based on a partial re-optimization of a variables partitioning. This approach can be also considered as a generalized $k-opt$ neighborhood search procedure. The VPLS framework can be seen as a local search approach for MIPs, specially suited for 0–1 variables, using a generalization of the k-exchange neighborhood.

Consider a general MIP formulation

$$\min c^{\mathrm{T}}X \text{ s.t. } AX \le b,\ X \in \{0,1\}$$

where $X^t = (x_1, x_2, \ldots, x_n)$ is a vector of n variables of the problem and $\overline{X}^t = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$ is a feasible solution to the MIP. If this is the case, it is always possible to define a subset S of a defined size of variables indices $\{1, 2, \ldots, n\}$. The neighborhood $N_S(\overline{X})$ consists of all solutions of the MIP where the jth variable is equal to the value of the jth variable in $\overline{X}$ for all $j \notin S$, namely $N_S(\overline{X}) = \{X \mid x_j = \overline{x_j},\ \forall j \notin S\}$. The resulting neighborhoods $N_S(\overline{X})$ can then be searched for an improving solution using a MIP-solver both optimally or approximately. The main idea stems in representing the MIP as a permutation problem where variables belonging to the current solution are partitioned into two sets. A first set of variables with indexes in S is then reoptimized by means of an MILP solver generating a permutated assignment, while variables in the second set $\{x_j, \forall j \notin S\}$ keep the same assignment as in the current solution.

For the considered problem, the incumbent solution returned by the RBS algorithm induces correspondingly a sequence of the bins where we assume that γ bins are used. The neighborhood exploration works as follows. Starting with the first bin, consider the rth bin in the sequence along with bins $r+1, r+2, \ldots, r+H-1$ with item set $S = S_r \cup S_{r+1} \cup \cdots \cup S_{r+H-1}$. Solve the problem of rescheduling the items in S so that $w(S_{r+H-1}) = \sum_{j \in S_{r+H-1}} w_j$ is minimized, where we use $w_j = \max\{b_j, p_j\}$ (this performance measure has been selected experimentally).

The rationale is to empty as much as possible bin $r+H-1$. If a (sub)sequence with $w(S_{r+H-1}) = 0$ is obtained, then one bin is saved, γ is reduced by one unit and the process can restart with $r = 1$. Alternatively, the new subsequence is kept anyway as the space of bin S_{r+H-1} has been optimized and will be used in the next iterate. The approach is then iterated for $r = 1, 2, \ldots, \gamma - H + 1$. Whenever $r = \gamma - H + 1$ is reached, the process restarts with $r = 1$ until a time limit is exceeded. The problem of rescheduling the items in S is done by solving by means of an ILP solver the following ILP model adapted from the one above in order to take into account the fact that bins $1, \ldots, r-1$ and $r+H, \ldots, \gamma$ are not rescheduled and that the objective is to minimize the weight of the items assigned to bin $r+H-1$.

$$\text{minimize } z \tag{9}$$

subject to

$$\sum_{k=1}^{\gamma} x_{j,k} = 1, \quad \forall j \in \{1, \ldots, n\} \tag{10}$$

$$\sum_{j=1}^{n} p_j x_{j,k} \le T, \quad \forall k \in \{1, \ldots, \gamma\} \tag{11}$$

$$\sum_{j=1}^{n} b_j x_{j,k} \le V, \quad \forall k \in \{1, \ldots, \gamma\} \tag{12}$$

$$\sum_{j=1}^{n} w_j x_{j,\overline{k}} \le z \quad (\text{with } \overline{k} = r+H-1) \tag{13}$$

$$x_{j,k} = \overline{x}_{j,k}, \quad \forall j \in \{1, \ldots, n\}, \quad \forall k \in \{(1, \ldots, r-1) \cup (r+H, \ldots, \gamma)\} \tag{14}$$

$$\sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h} + \sum_{i=1}^{j} p_i x_{i,k} \le d_j + Q + M(1 - x_{j,k}), \quad \forall j \in \{1, \ldots, n\}, \quad \forall k \in \{1, \ldots, \gamma\} \tag{15}$$

with $w_j = \max\{p_j, b_j\}, \forall j, 1 \le j \le n$.

Constraints (10) ensure that each job is performed by using one of the γ vials and correspond to constraints (4) of the previous model. Constraints (11) and (12) indicate the temporal and capacity limits of the γ vials and correspond to constraints (4) and (5) of the previous model. Constraints (13) link to the objective function z temporal and capacity consumption of vial $r+H-1$. Constraints (14) indicate that the assignment of items to bins $1, \ldots, r-1$ and $r+H, \ldots, \gamma$ is kept as it is in the incumbent solution. Finally, Eq. (15) matches constraints (7) of the previous model.

Fig. 4 displays the matheuristic neighborhood search approach proposed.

A pseudo-code of this procedure is depicted below.

4. Computational experiments

As there are no known instances for this problem, we present computational experiments on the set instances from the literature

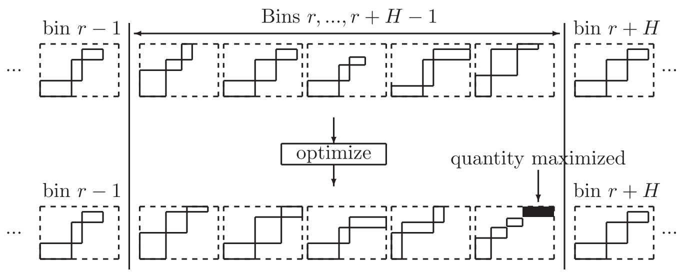

Fig. 4. Illustration of the matheuristic algorithm.

Algorithm 3 Search $(H, \gamma, \bar{x}, \bar{z})$.

```
repeat
    Set $r := 1$;
    Set $improved[k] :=$ true for each $k = 1, \ldots, \gamma$;
    while $r \le n - H + 1$ and $z^* > 0$ do
        Solve model (9)–(14) with optimum $z^*, x^*$;
        if $z^* = 0$ then
            $\bar{x} := x^*, \bar{z} := z^*$;
            $\gamma := \gamma - 1$;
        end if
        Set $improved[r + H - 1] =$ true iff $z^* < \sum_{j=1}^{n} w_j \bar{x}_{j,r+H-1}$;
        Set $r := r + 1 \pmod{\gamma + H - 1}$;
    end while
until not $\left(\bigvee_{k=1}^{\gamma} \text{improved}[k]\right)$ or ⟨time limit expired⟩;
```

for the two dimensional vector packing problem to which we added correspondingly the jobs due dates. The tests have been performed on a PC Intel Dual Core i5, CPU 1.7 Gigahertz and 4 Gigabytes RAM. The RBS method has been coded in C language and CPLEX v12.5 has been used for solving the MILP models.

The instances introduced by Caprara and Toth (2001) are grouped in 5 main classes, each class containing 10 instances for each value of n (30 instances per class in total with $n \in \{50, 100, 200\}$), which makes a total of 150 instances. These instances are available at:

www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm.

For each instance, the value in the first column is associated to the processing time and the value in the second column is associated to the vial consumption. Due dates have been randomly generated. We denote by P the sum of processing times: $P = \sum_{j=1}^{n} p_j$. For each job J_j, a due date is randomly generated in $[\alpha P(1-\beta), \alpha P(1+\beta)]$ where α and β are two coefficients belonging to interval [0, 1]. For the generation of the due dates, α and β have been fixed to 0.5.

For each data set, the EDD sequence has been computed for finding $L^*_{\max}$, the optimal value of the maximum lateness. For each instance, the bound Q is defined by $Q = \eta L^*_{\max}$ with a given coefficient $\eta \in \{1, 1000\}$. The value of $\eta = 1$ imposes that the solution is optimal regarding the $L_{\max}$. On the other side, the large value for η is equivalent to neglect the constraint on the $L_{\max}$ and the problem becomes equivalent to the original two dimensional vector packing problem. We denoted by RBS the results of the recovering beam search algorithm standalone and by "RBS-MH" the two-phase approach applying first RBS and then the matheuristic algorithm using the RBS solution as initial solution. For the matheuristic algorithm, we set $H = 4$ after some preliminary experiments.

Table 1 reports the computational results for $\eta = 1$. In this case, the solution has to be optimal for the maximum lateness. For each class and for each value of n, columns LB* and UB* of the table denote the best known lower and upper bounds respectively available for the two dimensional vector packing problem with no constraint on $L_{\max}$. These bounds were taken from Monaci and Toth (2006). Column "EDD" and column "Desc" indicate the sum of the number of bins

Table 1
Results of the algorithms for the two dimensional vector packing instances with due dates ($\eta = 1$).

$\eta = 1$						RBS		RBS-MH	
Class	n	LB*	UB*	EDD	Desc	UB	CPU(s)	UB	Ttb(s)
1	50	135	135	158	155	144	0.9	136	1.3
	100	255	260	313	311	287	25.8	260	10.8
	200	503	510	639	634	571	120	511	98.0
6	50	214	215	289	274	242	0.8	221	2.2
	100	405	410	599	587	483	11.6	426	40.5
	200	803	811	1218	1207	988	120	821	437.8
7	50	196	197	251	249	219	0.9	203	1.2
	100	398	405	529	526	444	17.6	412	5.0
	200	799	802	1076	1073	927	120	808	170.9
9	50	144	145	178	176	160	0.97	147	1.2
	100	257	267	331	328	297	18.8	268	15.7
	200	503	513	651	647	582	120	514	235.4
10	51	170	170	230	226	207	1.1	185	5.6
	99	330	330	442	425	396	16.9	350	20.9
	201	670	670	888	867	840	120	680	277.7

obtained for the 10 instances by EDD and by the descent after EDD, respectively. Also, column UB indicates the sum of the number of bins obtained for the 10 instances by the related method, column CPU(s) for RBS indicates the average computation time and column Ttb(s) for RBS-MH indicates the average time for obtaining the best solution (time-to-best). Note that for RBS a time limit of 120 seconds has been imposed while for the matheuristic algorithm a time limit of 1800 has been imposed, so that globally RBS-MH runs at most for 1920 seconds. Enlarging such limit does not seem to improve substantially the solution quality. Note also that for each iteration of the matheuristic algorithm a time limit of 60 seconds is imposed.

From the results, if we compare the values of the proposed approaches to the best feasible solutions (indicated by the entry UB*) known for the two dimensional vector packing problem, we notice that the solution quality of the combined RBS-MH method is particularly good on classes 1, 9, still well performing on class 7 and with partial degradation on classes 6, 10. Also, the time-to-best is typically significantly inferior to the 600 seconds of CPU time. Notice, however, that the strong constraint on $L_{\max}$ does not necessarily guarantee that the UB* values are reachable. The performances of RBS standalone are weaker, but require limited time. Notice also, that additional tests not reported here indicate that the performances of RBS are much better than the standard list scheduling EDD (earliest due date) sequence.

4.1. Instances without due dates

We consider in this subsection the case $\eta = 1000$. In this case, due dates are not considered and hence the original vector packing instances are solved. Correspondingly, constraint (15) becomes redundant and the bins selected for rescheduling are no more required to be consecutive. Hence, the following simpler ILP model can be considered in the matheuristic.

$$\text{minimize } z \tag{16}$$

subject to (10), (11), (12), (14) and

$$\sum_{j=1}^{n} w_j x_{j,\bar{k}} \le z \quad \text{for a chosen } \bar{k} \in I \tag{17}$$

Further, a more general matheuristic approach could be devised as no sequencing constraint is present anymore. The approach works as follows. First, H disjoint subsets of bins with H bins in each subset are selected and optimized in order to empty as much as possible one bin in each subset. Then the H disjoint emptied bins are re-optimized always trying to empty as much as possible one of them. The approach is iterated until a time limit expires. A pseudo-code of this approach is depicted below.

Algorithm 4 Search-BP ($H, \gamma, \bar{x}, \bar{z}$).

```
repeat
    $N = \{1, \dots, \gamma\}$;
    Randomly select $H$ disjoint index sets $I_1, \dots, I_H \subset N$ with $|I_1| = \dots = I_H = H$;
    for each $I_r, r = 1, \dots, H$ do
        Solve model (16)–(17) for $I_r$, with optimum $z^*, x^*$;
    end for
    Let $I' = \{k: k = \arg\min\{\sum_{j=1}^{n} w_j x^*_{j,k},\ k \in I_r,\ r = 1, \dots, H\}\}$;
    Solve model (16)–(17) for $I'$, with optimum $z', x'$;
    if $z' = 0$ then
        Set $\gamma := \gamma - 1$;
    end if
until ⟨time limit expired⟩;
```

Table 2
Results of the algorithms for the two dimensional vector packing instances without due dates ($\eta = 1000$).

$\eta = 1000$				RBS		RBS-MH	
Class	n	LB*	UB*	UB (*opt*)	CPU(s)	UB	Ttb(s)
1	50	135	135	139	3.7	135 (10)	0.1
	100	255	260	263	98.4	258 (7)	14.3
	200	503	510	527	120	505 (8)	190.9
6	50	214	215	219	3.0	215 (9)	0.5
	100	405	410	421	30.6	410 (5)	48.1
	200	803	811	847	120	819 (0)	233.2
7	50	196	197	201	3.8	198 (8)	1.1
	100	398	405	406	23.7	405 (3)	0.1
	200	799	802	810	120	803 (6)	134.5
9	50	144	145	146	2.7	145 (9)	0.1
	100	257	267	274	48.7	267 (0)	0.5
	200	503	513	533	120	513 (0)	7.3
10	51	170	170	187	4.2	180 (0)	0.4
	99	330	330	351	28.4	340 (0)	2.5
	201	670	670	713	120	680 (0)	30.7

Table 2 provides the same entries of Table 1 for the original two dimensional vector packing problem that is the case with no constraints on the due dates. In column "UB (*opt*)" we indicate in parentheses the number of times the method can certify an optimal value (equality with the lower bound).

The results show that RBS-MH is competitive with the state of the art procedures being superior on class 1, comparable on classes 7, 9 and inferior on classes 6, 10. We point out, in particular, that as far as class 1 is concerned, two new upper bounds were established for $n = 100$ and five new upper bounds were established for $n = 200$. The corresponding solutions are available at

http://www.di.unito.it/~grosso/newbests2CBP.txt.

Also, the CPU time required by RBS-MH is on the average well below 400 seconds. We point out that it is hard to get a full comparison to the recent work of Alves et al. (2014) where most of the results with $n = 200$ are missing and detailed solution values of each instance are not available. However, it appears from the instances approached by both algorithms that Alves et al. (2014) reaches better results (71 optimal solutions vs 43 over 80 instances) at the expense though of a much larger computational effort (on class 9 with $n = 50$, for instance, approximately 400 vs 1.2 seconds).

5. Conclusions

We considered the problem of scheduling activities which consume perishable raw materials that induce a deadline on their use once started, where the deadline is a part of the decision process. The problem has been modeled as a single machine scheduling problem with additional duration and consumption constraints. The problem can also be seen as a two dimensional vector packing problem combined to the single machine problem with the requirement that the maximum lateness of the resulting single machine sequence does not exceed a given threshold. A two-phase procedure has been proposed where in the first phase a fast Recovering Beam Search approach has been applied and in the second phase a more time consuming matheuristic procedure using the RBS solution as initial solution has been proposed. The two procedures have been tested showing very good performances within reasonable time on benchmark two dimensional vector packing instances suitably integrated to handle the maximum lateness requirement. The proposed procedures have also been tested on the original two dimensional vector packing instances showing to be competitive with the state of the art procedures and finding 8 new best bounds. As a future research direction, it would be worthy to search for improved lower bounds taking into account the presence of the due dates and to study the corresponding problems for different values of α and β. Another research direction could be to adapt the proposed approach to the original problem discussed in Mazier et al. (2010), where several parallel machines (pharmacy technicians) and several types of cytotoxic drugs have to be taken into account. Another research direction could be to consider that more than one vial can be assigned to the same preparation. It would lead to another model, presenting less similarities with the two dimensional vector packing problem, but with undoubtedly potential applications.

References

Alves, C., de Carvalho, J. V., Clautiaux, F., & Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, *233*, 43–63.

Billaut, J.-C. (2011). New scheduling problems with perishable raw materials constraints. In *16th IEEE international conference on emerging technologies and factory automation*. Toulouse, France. September 5–9

Billaut, J.-C., Esquirol, P., & Tournamille, J.-F. (2011). Scheduling chemotherapy preparations with perishable raw material constraints. In *Informs healthcare*. Montreal, Canada. June 20–22

Brucker, P. (2007). *Scheduling algorithms* (5th ed.). New York, Berlin, Heidelberg: Springer.

Caprara, A., & Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, *111*, 231–262.

De Carvalho, J. M. V. (2002). Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, *141*(2), 253–273.

Della Croce, F., Ghirardi, M., & Tadei, R. (2004). Recovering beam search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, *10*, 10–89.

Della Croce, F., Grosso, A., & Salassa, F. (2013). Matheuristics: embedding MILP solvers into heuristic algorithms for combinatorial optimization problems. In P. Siarry (Ed.), *Heuristics: theory and applications* (pp. 31–52). Nova Science Publishers.

Dong, X., Huang, H., & Chen, P. (2009). An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research*, *36*, 1664–1669.

Esteve, B., Aubijoux, C., Chartier, A., & T'kindt, V. (2006). A recovering beam search algorithm for the single machine just-in-time scheduling problem. *European Journal of Operational Research*, *172*(3), 798–813.

Garey, M., Graham, R., Johnson, D., & Yao, A. (1976). Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory Series A*, *21*, 257–298.

Ghirardi, M., & Potts, C. N. (2005). Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach. *European Journal of Operational Research*, *165*(2), 457–467.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207*(1), 1–14.

Lodi, A., Martello, S., & Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, *123*(1–3), 379–396.

Mazier, A., Billaut, J.-C., & Tournamille, J.-F. (2010). Scheduling preparation of doses for a chemotherapy service. *Annals of Operations Research*, *178*(1), 145–154.

Monaci, M., & Toth, P. (2006). A set-covering-based heuristic approach for bin-packing problems. *INFORMS Journal on Computing*, *18*(1), 71–85.

Ow, P. S., & Morton, T. E. (1988). Filtered beam search in scheduling. *International Journal of Production Research*, *26*, 297–307.

Padhy, R. P., & Patra, M. R. (2013). Service support aware resource allocation policy for enterprise cloud-based systems. *International Journal of Cloud Computing and Services Science*, *2*(4), 296–312.

Pinedo, M. (2012). *Scheduling. Theory, algorithms and systems* (4th ed.). Springer.

Rakrouki, M. A., Ladhari, T., & T'Kindt, V. (2012). Coupling genetic local search and recovering beam search algorithms for minimizing the total completion time in the single machine scheduling problem subject to release dates. *Computers and Operations Research*, *39*(6), 1257–1264.

Ruiz, R., & Vázquez-Rodriguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1–18.

Spieksma, F. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers and Operations Research*, *21*, 19–25.

Valente, J. M. S. (2010). Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society*, *61*(4), 620–631.