European Journal of Operational Research 202 (2010) 538–546

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

A tabu search algorithm for scheduling pharmaceutical packaging operations

Luca Venditti [a], Dario Pacciarelli [a,*], Carlo Meloni [b]

[a] Dipartimento di Informatica e Automazione, Università Degli Studi Roma Tre, Via della Vasca Navale, 79, 00146 Roma, Italy
[b] Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via E. Orabona, 4, 70125 Bari, Italy

ARTICLE INFO

Article history:
Received 19 February 2008
Accepted 22 May 2009
Available online 23 June 2009

Keywords:
Scheduling
Packaging
Tabu search
Pharmaceutical industry

ABSTRACT

This paper addresses a practical scheduling problem arising in the packaging department of a pharmaceutical industrial plant. The problem is modeled as a multi-purpose machine scheduling problem with setup and removal times, release and due dates and additional constraints related to the scarce availability of tools and human operators. The objective functions are minimization of makespan and maximum tardiness in lexicographic order. Representing a solution with a directed graph allows us to devise an effective tabu search algorithm to solve the problem. Computational experiments, carried on real and randomly generated instances, show the effectiveness of this approach.

1. Introduction

In this paper, we address a multi-purpose machine scheduling problem [22] with additional constraints related to the availability of tools and operators, removal and setup times, release times, due dates and deadlines. The objectives are the joint minimization of makespan and maximum tardiness, which are addressed in lexicographic order. The problem is motivated by the practical implementation of a decision support system for scheduling the production orders at the packaging department of a pharmaceutical plant located in Italy. A detailed description of the problem is given in Section 2. Previous attempts to design a computerized tool for production scheduling at this department have had limited success, and similar performance has been observed in practice for many computerized tools for planning and scheduling [11,12]. A common reason for this behavior is that the models adopted by computerized systems suffer from excessive simplification and do not incorporate all the relevant aspects of the shop floor [24]. In fact, most optimization algorithms from the scheduling literature are mainly concerned with the computation of optimal or near-optimal solutions to very simplified problems [16,20]. According to the survey of Reisman et al. [18], out of 170 articles on flowshop scheduling/sequencing published from 1952 to 1994 only 5 were judged to be true applications. On the other hand, in the last years an increasing number of articles focus on realistic scheduling models including more practical constraints than in the past [15,17,20]. However, to the best of our knowledge, no published work addresses exactly the problem studied in this paper, even though many algorithms have been proposed to solve relaxations of this problem, addressing different subsets of constraints. Among the others, we cite the parallel machine scheduling problems with setup and removal times, release and due dates [21], the vehicle routing problem with fixed number of vehicles and time windows [2,3,5] and the resource constrained scheduling problem [4].

In this paper we propose a graph representation of the problem and a tabu search algorithm, described in Sections 3 and 4, respectively. The Tabu Search (TS) metaheuristic [8,9] is a well established iterative algorithm with steepest descent criterion, which accepts non-improving moves to escape from local minima and makes use of a tabu list to restrict the neighborhood to be explored at each step. A move in the tabu list is forbidden and remains in it for a limited number of iterations, called the length of the tabu list, which can be fixed or variable. This mechanism can be overruled when a solution associated with a tabu move satisfies an aspiration criterion. More sophisticated TS schemes have been proposed in the literature [7,14], and this method is certainly among the most successful heuristics for a large number of planning and scheduling problems [13]. In Section 5 we report on our computational experiments, carried on real industrial instances and on randomly generated instances. The comparison with the industrial practice shows the effectiveness of our approach.

2. Description of the problem

The production of pharmaceutical packages is driven by wholesaler orders. A schedule for the next two weeks is produced every

* Corresponding author.
E-mail addresses: venditti@dia.uniroma3.it (L. Venditti), pacciarelli@dia.uniroma3.it, pacciare@dia.uniroma3.it (D. Pacciarelli), meloni@deemail.poliba.it (C. Meloni).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.05.038

week, following a rolling horizon criterion. In the week preceding the packaging, different kinds of tablets are produced in the manufacturing area and stored in sealed bins. Then, in the packaging department, bins are re-opened and processed to produce final products. The packaging department studied in this paper contains three packaging lines working in parallel. Each line can process one lot of identical products at a time and performs all operations from the production of blisters to the final individual specific packages. Therefore, a line acts as a single machine.

A set of production orders, hereinafter called jobs, have to be scheduled on this set of machines. Each job is compatible with a subset of machines and each machine needs two kinds of resources to process a job: a tool, which defines the size of the blister and depends on the job being processed, and a fixed number of operators, which is the same for each production line. Human resources availability is constant within a shift, but it can vary from one shift to another, the night shift being typically less supervised; machines can be unavailable in given periods for preventive maintenance operations. The considered scheduling problem is characterized by different timing constraints. Job release times are determined by the scheduled completion times at the earlier manufacturing departments. Each job has a due date, while a hard deadline is defined for urgent orders, when there is a risk of stock-out at the final customers.

Modeling the practical scheduling problem requires to consider a number of constraints and the incorporation of several details into the model. Tools are shared among families of similar products and available in a limited number of copies, in most cases there is a single copy of each tool. In each machine, sequence-dependent setup and removal times occur before and after the processing of a job, in order to clean and calibrate the machine and to change the tool defining the blister size. The setup/removal is called *minor* when requiring no tool change and *major* in the latter case.

Cleaning operations, mechanical configurations and job processing require a given amount of work for human operators, therefore machines cannot process job nor execute setups or removals without human resources. In order to reduce the risk of cross-contaminations, the plant policy is to assign each operator to a specific machine during the whole shift. Setups and removals cannot be interrupted while the processing of a job on a machine can be interrupted and resumed later on the same machine if the machine becomes unavailable for planned maintenance or if the number of operators in the subsequent shift is not sufficient to process the job. Machine unavailability can be viewed as a special job characterized by a release date, corresponding to its starting time, a processing time, corresponding to its duration, and a deadline corresponding to its finish time. Maintenance operations are compatible with a single machine, while the lack of operators can be moved from a machine to another.

Summarizing the above description, we report the description of this scheduling problem below, using the three-fields classification scheme of Graham et al. [10] and the notation of T'kindt and Billaut [22].

$$OMPM|r_i, d_i, D_i, R_{sd}, S_{sd}, unavail_j|Lex(C_{max}, T_{max})$$

The first field contains the shop environment, in our case *OMPM* is the *Open shop Multi-Purpose Machines*. In fact, the department contains parallel machines but each job has its own set of machines and tools on which it can be processed and has to be assigned to one machine and one tool in its sets. Then the jobs assigned to the same tool or machine must be sequenced. Viewing the jobs assigned to the same tool as operations of an extended job, the sequencing part of the problem is similar to an open shop problem.

The second field contains the constraints of the problem. In our case r_i, d_i, D_i, indicate the presence of release times, due dates and deadlines, respectively. R_{sd} and S_{sd} indicate sequence-dependent removal times and sequence-dependent setup times. Moreover, resources (machines and operators) are not available all the time but only during well defined periods. Notation $unavail_j$ indicate this constraints. Specifically, in our case job processing is resumable, i.e., can be interrupted when the machine or the operators become unavailable and resumed later (but no other job can be processed in between). Setups and removals are not resumable, i.e., they cannot be started if unavailability arise before completion.

The third field contains the optimality criteria of the problem, in our case $Lex(C_{max}, T_{max})$ is the minimization of makespan and maximum tardiness in lexicographic order.

We notice that, in the scheduling literature, even relaxed versions of this problem, such as the vehicle routing problem with release and due dates, are considered particularly difficult NP-hard problems [16]. Moreover, some constraints, such as the resumable unavailability of resources, are not frequently addressed in scheduling literature, at least in combination with others.

3. Graph representation of a solution

In this section, we introduce the notation used throughout the paper and then we formally describe the constraints satisfied by feasible solutions.

Problem data consist of the following. A set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$, each consisting of a single operation, must be scheduled on a set of m machines $\mathcal{M} = \{1, 2, \ldots, m\}$. Each job must be processed entirely on the same machine, which can process at most one job at a time. We denote with $\mathcal{M}_j \subseteq \mathcal{M}$ the set of machines able to process job j. To process job j, a machine must be equipped with a fixed number b of operators and with a tool T_j, chosen in the set of tools $\mathcal{T} = \{1, 2, \ldots, t\}$. We denote with $\mathcal{T}_j \subseteq \mathcal{T}$ the set of tools compatible with job j. Also, let p_j, r_j, d_j and $\tilde{d}_j$ be the processing time, release date, due date and deadline of job j, respectively.

A given *removal time* g_{ij} is required, between two jobs i and j processed consecutively on the same machine, to clean and calibrate the machine and to remove tool T_i if i and j use different tools. If $T_i \neq T_j$ an additional *setup time* f_{ij} is required after g_{ij} to set up the new tool T_j. A similar situation occurs if job i is processed on machine h', job j is processed on machine h and both use consecutively the same tool. Letting $\beta(i)$ the job sequenced after i on machine h' and $\alpha(j)$ the job preceding j on machine h, a removal time $g_{i\beta(i)}$ is required to remove the tool from h' and a setup $f_{\alpha(j)j}$ is required to set up the tool on h. All setup/removal times satisfy the triangular inequality, i.e., $f_{ij} + f_{jk} \geqslant f_{ik}$ and $g_{ij} + g_{jk} \geqslant g_{ik}$.

The number o_i of operators available during shift $i = 1, \ldots, s$ is known in advance, and operators must be assigned to the machines for the entire duration of a shift. Therefore, at most $\lfloor \frac{o_i}{b} \rfloor$ machines can be active during shift i. We model this situation by introducing $m - \lfloor \frac{o_i}{b} \rfloor$ dummy jobs called *unavailabilities*, with release date, deadline and processing time equal to the shift start, completion and duration, respectively. These jobs will be scheduled on the machines with all the other jobs. In some cases, the subset of inactive machines for a shift is partially specified in advance, for example when preventive maintenance operations are planned for some machines during that shift.

We denote with $\mathcal{U} = \{(n+1), \ldots, (n+q)\}$ the set of all unavailabilities, with $q = \sum_{i=1,\ldots,s} (m - \lfloor \frac{o_i}{b} \rfloor b)$, and with $\mathcal{M}_u \subseteq \mathcal{M}$ the set of the machines compatible with unavailability $u \in \mathcal{U}$.

Finding a schedule consists of solving five subproblems. We postpone the feasibility issue to the end of this section.

(i) Assign each job $j \in \mathcal{J}$ to a machine $h \in \mathcal{M}_j$. Let $\mathcal{J}^M(h)$ be the set of jobs assigned to machine $h = 1, \ldots, m$.
(ii) Assign each job $j \in \mathcal{J}$ to a tool $k \in \mathcal{T}_j$. Let $\mathcal{J}^T(k)$ be the set of jobs assigned to tool $k = 1, \ldots, t$.
(iii) Assign each unavailability $u \in \mathcal{U}$ to a machine $h \in \mathcal{M}_u$ such that unavailabilities assigned to the same machine are non-overlapping. Let $\mathcal{U}(h)$ be the set of unavailabilities assigned to machine $h = 1, \ldots, m$, and μ be the m-tuple $\mathcal{U}(1), \mathcal{U}(2), \ldots, \mathcal{U}(m)$.
(iv) Sequence the jobs in $\mathcal{J}^M(h), h = 1, \ldots, m$ and the jobs in $\mathcal{J}^T(k), k = 1, \ldots, t$. Let σ_h be the resulting sequence on machine h and π_k be the resulting sequence on tool k. Let also σ be the m-tuple $\sigma_1, \sigma_2, \ldots, \sigma_m$ and π be the t-tuple $\pi_1, \pi_2, \ldots, \pi_t$.
(v) Define a schedule, i.e., a *starting time* S_j and a *completion time* $C_j \leqslant \tilde{d}_j$ for each job $j \in \mathcal{J}$ such that each machine processes at most one job/unavailability at a time. Let S and C be the vectors of starting/completion time of all jobs, $S = (S_1, S_2, \ldots, S_n), C = (C_1, C_2, \ldots, C_n)$.

Let a *solution* H denote the triple $H = (\mu, \sigma, \pi)$, i.e., the output of subproblems (i)–(iv), and let a *schedule* denote the pair (S, C). For each job $j \in \mathcal{J}^M(h)$ we denote with $\alpha(j)$ the job preceding j in σ_h and with $\beta(j)$ the job succeeding j in σ_h. Similarly, for each job $j \in \mathcal{J}^T(k)$ we denote with $\gamma(j)$ the job preceding j in π_k and with $\delta(j)$ the job succeeding j in π_k.

Let us now focus on the computation of a minimum makespan schedule of a solution H. In order to compute S_j and C_j we assume that the preceding jobs $\alpha(j)$ and $\gamma(j)$ (if exist) have already set $C_{\alpha(j)}$ and $C_{\gamma(j)}$. We schedule non-preemptive activities $g_{\gamma(j)\beta(\gamma(j))}, g_{\alpha(j)j}$ and $f_{\alpha(j)j}$ in the earliest available time periods (see Fig. 1). Then we schedule preemptive job j, possibly splitting it in successive availability periods. Denoting with $(\cdot)$ and $[\cdot]$ open and closed intervals respectively, the earliest available starting time for the removal activity $g_{\gamma(j)\beta(\gamma(j))}$ on machine h' is

$$X_j = \min\left\{\tau : \tau \geqslant C_{\gamma(j)}, (\tau, \tau + g_{\gamma(j)\beta(\gamma(j))}) \cap \bigcup_{i \in \mathcal{U}(h')} [r_i, \tilde{d}_i] = \emptyset\right\}.$$

Similarly, the earliest available starting times for the removal and setup activities $f_{\alpha(j)j}$ and $g_{\alpha(j)j}$ on machine h are

$$Y_j = \min\left\{\tau : \tau \geqslant C_{\alpha(j)}, (\tau, \tau + g_{\alpha(j)j}) \cap \bigcup_{i \in \mathcal{U}(h)} [r_i, \tilde{d}_i] = \emptyset\right\},$$

$$Z_j = \min\left\{\tau : \tau \geqslant X_j + g_{\gamma(j)\beta(\gamma(j))}, \tau \geqslant Y_j + g_{\alpha(j)j}, (\tau, \tau + f_{\alpha(j)j}) \cap \bigcup_{i \in \mathcal{U}(h)} [r_i, \tilde{d}_i] = \emptyset\right\}.$$

If machine h is unavailable at the release time r_j of job j, let $u(j)$ be the last unavailability on machine h starting before r_j, i.e., such that $r_{u(j)} = \max\{r_l \leqslant r_j, l \in \mathcal{U}(h)\}$. Then, job j cannot start before $R_j = \max\{\tilde{d}_{u(j)}; r_j\}$. Therefore, the earliest available starting time of j is:

$$S_j = \max\{R_j; Z_j + f_{\alpha(j)j}\}. \quad (1)$$

The earliest completion time of preemptive job j equals $S_j + p_j$ plus the total time machine h is unavailable for processing between the start and the completion of job j. Let us denote with $u_1, \ldots u_v$ the sequence of unavailabilities in $\mathcal{U}(h)$ starting after S_j, ordered for increasing values of their release times $S_j \leqslant r_{u_1} \leqslant \ldots \leqslant r_{u_{v-1}} \leqslant r_{u_v}$, and let $p_{u_1}, \ldots p_{u_v}$ be their respective processing time. Also, let $u_{v(j)}$ be the last unavailability on machine h before C_j, i.e., $v(j) = \min\left\{l : S_j + p_j + \sum_{i=1}^{l} p_{u_i} \leqslant r_{u_{l+1}}\right\}$. Then, the completion time of job j is:

$$C_j = S_j + p_j + \sum_{i=1}^{v(j)} p_{u_i}. \quad (2)$$

We denote the pair (S, C) computed according to Eqs. (1) and (2) as the schedule associated to solution H. Let us define for job j a modified processing time $\hat{p}_j = C_j - S_j$ and a modified release time $\hat{r}_j$

$$\hat{r}_j = \begin{cases} r_j & \text{if } S_j = \max\{C_{\alpha(j)} + g_{\alpha(j)j}; C_{\gamma(j)} + g_{\gamma(j)\beta(\gamma(j))}\} + f_{\alpha(j)j} \\ S_j & \text{if } S_j > \max\{C_{\alpha(j)} + g_{\alpha(j)j}; C_{\gamma(j)} + g_{\gamma(j)\beta(\gamma(j))}\} + f_{\alpha(j)j} \end{cases} \quad (3)$$

A solution H and the associated schedule can be represented with a graph $\mathcal{G}(H) = (V, E(\sigma) \cup F(\pi) \cup A)$. V is the set of nodes, one for each job $j \in \mathcal{J}$, weighted with $\hat{p}_j$, plus four auxiliary nodes *start*, *end*, *viol_dead*, *viol_due*. $E(\sigma)$ is a set of arcs, one for each pair of consecutive nodes j and $\beta(j)$ processed on the same machine in H and weighted with $g_{j\beta(j)} + f_{j\beta(j)}$. $F(\pi)$ is a set of arcs, one for each pair of consecutive nodes j and $\delta(j)$ processed on the same tool in H and weighted with $g_{\alpha(\delta(j))\delta(j)} + f_{\alpha(\delta(j))\delta(j)}$. Finally, A is a set of additional arcs. For each node $j \in \mathcal{J}$ there is an arc in A from *start* to j, with weight $\hat{r}_j$, an arc $(j, end) \in A$, with weight zero, an arc $(j, viol\_dead) \in A$ if a deadline is defined for j, with weight $-\tilde{d}_j$, and an arc $(j, viol\_due) \in A$ if a due date is defined for j, with weight $-d_j$.

Fig. 2 shows the pictorial representation of a node j and associated arcs in $\mathcal{G}(H)$, where arcs in $E(\sigma) \cup A$ are depicted with solid lines while arcs in $F(\pi)$ are depicted with dashed lines. Note that the graph structure depends on the assignment and sequencing of the jobs on tools and machines, while the arc weights also depend on the assignment of the unavailabilities to the machines.

For a given graph $\mathcal{G}$, we define the *head* $h(j)$ of node j as the length of the longest path in $\mathcal{G}$ from *start* to j (excluding the weight of node j) and the *tail* $q_a(j)$ as the length of the longest path from j to the auxiliary node $a \in \{end, viol\_dead, viol\_due\}$ (including the weight of node j). With this notation, the starting time S_j of job j in a solution is the head of the node associated to job j. Then, $h(end)$ and $\max\{0, h(viol\_due)\}$ are equal to the makespan C_{max} and the maximum tardiness T_{max} of the solution, respectively.

A solution H is feasible if each job is assigned to a compatible machine and to a compatible tool, each unavailability is assigned to a compatible machine, unavailabilities assigned to the same machine are non-overlapping, and $\mathcal{G}(H) = (V, E(\sigma) \cup F(\pi) \cup A)$ is acyclic. A schedule (S, C) is feasible if the associated solution H is feasible and $h(viol\_dead) \leqslant 0$, which implies $h(i) + \hat{p}_j \leqslant \tilde{d}_j$ for each $j \in \mathcal{J}$.

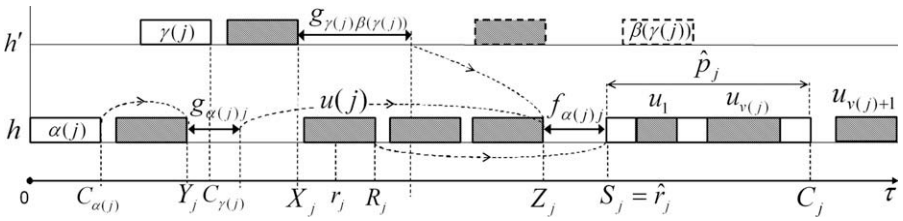

Fig. 1. Gantt chart for machines h and h' and computation of S_j.

4. Tabu search algorithm

In this section, we describe our tabu search algorithm. The *Tabu Search* is a local search based metaheuristic, which makes extensive use of memory for guiding the search. Its basic components are the concepts of move and tabu list, which restrict the set of solutions to be explored. From the incumbent solution, non-tabu moves define a set of solutions, called neighborhood, the best of which is selected as the new incumbent also when is non-improving. The tabu list keeps memory of the last moves performed by the algorithm, and it is used to escape from local optima and to avoid re-evaluating solutions that have been recently visited.

Section 4.1 describes the algorithm used to find an initial solution, not necessarily feasible. Section 4.2 shows the basic moves used by the tabu search algorithm and structural properties of the neighborhood based on these moves. In Section 4.3 we define a restricted version of the neighborhood, which is used by our tabu search algorithm. Finally, in Section 4.4, we describe two procedures to evaluate the quality of a neighbor.

4.1. Greedy algorithm

A fast and simple greedy algorithm is used to obtain an initial solution. The algorithm is designed to reproduce the behavior of human scheduler when building a feasible schedule. In fact, the human schedulers in the plant do not follow any formal procedure to schedule production orders, and the schedules are simply the result of intuition and past experience. However, the schedules produced by hand are quite similar to those produced with the algorithm summarized in the following steps:

1. Compute an estimate workload for each machine as follows:
   (a) Compute the minimum workload W_h for machine $h \in \mathcal{M}$ summing up processing time + minimum setup time + minimum removal time of all jobs that can be executed on machine h only: $W_h = \sum_{j:\mathcal{M}_j=\{h\}}(p_j + \min_i\{f_{ij}\} + \min_i g_{ij}\})$.
   (b) Compute the quantity $Q(i)$, equal to the processing time + minimum setup time + minimum removal time of all jobs that can be executed on machine h and other i machines in $\mathcal{M}$, and add to W_h the quantity $\frac{Q(i)}{i+1}$, for $i = 1,\ldots,m-1$.
2. Assign human operators to machines, proportionally to the values W_h;
3. Partition the time horizon into L time intervals of given length λ, and schedule the jobs in each time interval $[(l-1)\lambda, l\lambda]$, for $l = 1,\ldots,L$, according to the following algorithm:
   (a) Let $\mathcal{J}_l$ be the set of jobs with deadline or due-date smaller than $l\lambda$;
   (b) Group the jobs in $\mathcal{J}_l$ requiring the same tool and sequence the jobs in each group according to the ERD rule (Earliest Release Date first rule);
   (c) Sequence the groups one at a time by assigning a block to the next available machine according the SST rule (Shortest Setup Time first rule) until all groups are scheduled.

4.2. Neighborhood structure

Our tabu search algorithm makes use of two basic moves. The first one is based on the interchange of a pair of adjacent jobs sequenced on the same machine/tool, which is commonly adopted in the tabu search literature on open shop and job shop scheduling problems [14]. The second move is based on removing a job from the sequence of jobs processed on a machine/tool and inserting it in the sequence of another machine/tool. Similar moves are commonly used when dealing with parallel machine scheduling or vehicle routing problems [5,7].

The *interchange* move $\varphi^M(i,j)$ is defined as follows: given a solution H and a pair of consecutive jobs i and j in σ_h of machine h, a new solution H' is obtained from H reversing the precedence order (i,j). A similar move $\varphi^T(i,j)$ deals with rescheduling of consecutive jobs i and j in the sequence π_k of tool k. When two jobs are processed consecutively both on the same machine h and tool k, then a move $\varphi^{MT}(i,j)$ is applied, which exchanges their relative positions both in σ_h and π_k, in order to avoid a cycle of precedence constraints between i and j.

With the *rerouting* move $\theta^M(i,j), i$ can be either a job or an unavailability while j is a job. Moreover, i and j are processed on different machines h and h', respectively, with $h' \in \mathcal{M}_i$. If i is a job, this move consists of removing i from σ_h and inserting it before j in $\sigma_{h'}$. If i is an unavailability, the move consists of removing i from machine h and inserting it in the same machine of j in its given time window. Similarly, move $\theta^T(i,j)$ is applied to jobs i and j processed on different tools k and k', respectively, with $k' \in \mathcal{T}_i$.

4.2.1. Acyclicity property

A solution H is feasible only if the corresponding graph $\mathcal{G}(H)$ is acyclic. Our tabu search algorithm does not perform moves leading to cyclic graphs. To this aim, a cyclicity test is preliminarily checked before each move and the move is removed from the neighborhood if it leads to a cycle. The test can be performed in time $O(n)$ using the Bellman's algorithm, but we next show that this check can be performed in constant time if the conditions of Theorems 4.1 or 4.2 hold.

In the following we call critical path of $\mathcal{G}(H)$ one of the longest paths from *start* to one of the auxiliary nodes and denote H the incumbent solution, H' the solution obtained after a move, $h(i)$ the head of node i in H and $h'(i)$ the head of node i in H'.

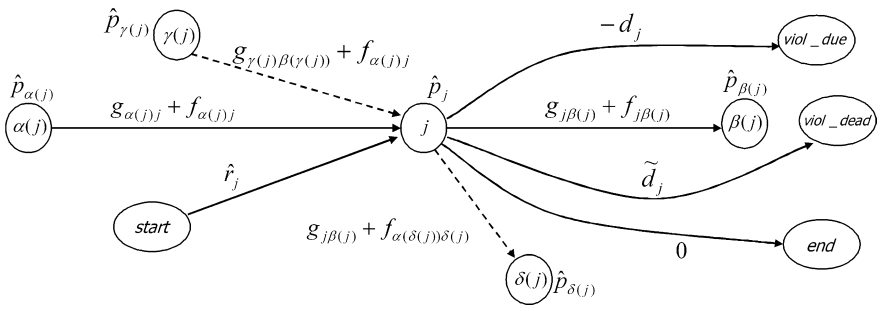

Fig. 2. Node j and weighted sequencing arcs in $\mathcal{G}(H)$.

Theorem 4.1. *Given an arc (i,j) on the critical path of $\mathcal{G}(H)$, the interchange moves $\varphi^M(i,j)$, $\varphi^T(i,j)$, and $\varphi(i,j)^{MT}$ do not create cycles.*

Proof. Let us suppose that there is a cycle in $\mathcal{G}(H')$ after a move $\varphi(i,j)$. Thus, there must be a path in $\mathcal{G}(H)$ from i to j, disjoint from arc (i,j). Let us consider the three moves separately.

- $\varphi^{MT}(i,j)$ move: in this case $j=\beta(i)=\delta(i)$ and there are no other arcs outgoing from i, other than those ingoing in j or in the auxiliary nodes. Therefore, there is no path from i to j, disjoint from arc (i,j) in $\mathcal{G}(H')$, a contradiction.
- $\varphi^{M}(i,j)$ move: in this case $j=\beta(i)$. Let us suppose that a path p from i to j exists in $\mathcal{G}(H)$, besides arc (i,j). This path must include at least arcs $(i,\delta(i))$ and $(\gamma(j),j)$, which are the only arcs outgoing from i and ingoing in j other than (i,j) and the arcs of set A. There are only two possibilities: either $T_i\neq T_j$ or $T_i=T_j$. In both cases, path p includes at least the removal of tool T_i, the setup of tool T_j and the processing time $p_{\delta(i)}>0$. If $T_i=T_j$ the length of p is strictly larger than the length of arc (i,j). If $T_i\neq T_j$ it follows from the triangular inequality that the weight of all setups occurring in p is larger or equal to $g_{ij}+f_{ij}$. Therefore, the weight of p is strictly larger than the weight of the critical arc (i,j), a contradiction.
- $\varphi^{T}(i,j)$ move: in this case $j=\delta(i)$. Let us suppose that a path p from i to j exists in $\mathcal{G}(H)$, besides arc (i,j). This path must include at least arcs $(i,\beta(i))$ and $(\alpha(j),j)$, which are the only arcs outgoing from i and ingoing in j other than (i,j) and the arcs of set A. Also in this case, if follows from the triangular inequality that the length of path p is greater than the weight of the critical arc (i,j), a contradiction.

In conclusion, path p cannot exist if the arc (i,j) is critical, and the thesis follows. □

As far as the rerouting moves are concerned, let us first observe that there can be a cycle in $\mathcal{G}(H')$ after move $\theta^M(i,j)$ if and only if there is a path in $\mathcal{G}(H)$ from j to $\gamma(i)$ or from $\delta(i)$ to $\alpha(j)$. We notice that, if $j=\beta(i)$ the acyclicity of $\mathcal{G}(H)$ implies that of $\mathcal{G}(H')$. Otherwise, a sufficient condition for the acyclicity of $\mathcal{G}(H')$ is given by the following theorem. The proof directly follows from results proved in [6].

Theorem 4.2. *After a move $\theta^M(i,j)$, if $h(j)+\hat{p}_j>h(\gamma(i))$ and $h(\delta(i))+\hat{p}_{\delta(i)}>h(\alpha(j))$ then $\mathcal{G}(H')$ is acyclic.*

Similar conditions hold for $\theta^T(i,j)$. When sufficient conditions do not hold the algorithm explicitly checks the existence of a path from i to j on $\mathcal{G}(H)$, besides (i,j). If such a path exists, the move is removed from the neighborhood.

4.2.2. Connectivity

In this section, we show that the neighborhood defined by moves $\varphi(i,j)$ and $\theta(i,j)$ is connected, i.e., we prove that it is possible to reach a global minimum starting from any feasible solution. Our proof is similar to that reported in [6]. Let A be a resource assignment, i.e., the definition of sets $\mathcal{J}^M(h),\mathcal{U}(h)$ and $\mathcal{J}^T(k)$ for each machine $h\in\mathcal{M}$ and each tool $k\in\mathcal{T}$. Let H_A [respectively, H_A^*] be a generic [an optimal] solution associated to A. H_A [respectively, H_A^*] defines the sequences [the optimal sequences] σ_h and π_k of elements in $\mathcal{J}^M(h)$ and $\mathcal{J}^T(k)$ for each $h\in\mathcal{M}$ and $k\in\mathcal{T}$. We also call $\mathcal{G}(H)$ the graph representation of H and $\mathcal{G}(\overline{H})$ its transitive closure, i.e., the graph obtained from $\mathcal{G}(H)$ including all the redundant arcs. A first result is the following:

Lemma 4.1. *Given a resource assignment A, an optimal sequencing H_A^* and a solution H_A, if H_A is not optimal there is an arc (i,j) such that (i,j) is critical in $\mathcal{G}(H_A)$ and such that $(j,i)\in\mathcal{G}(\overline{H}_A^*)$.*

Proof. Let us suppose that all the critical arcs of $\mathcal{G}(H_A)$ also belong to $\mathcal{G}(\overline{H}_A^*)$ as well. Therefore all the critical paths of $\mathcal{G}(H_A)$ also belong to $\mathcal{G}(\overline{H}_A^*)$, thus implying that H_A^* is not optimal, a contradiction. Hence, there must be at least a critical arc $(i,j)\in\mathcal{G}(H_A)$ that does not belong to $\mathcal{G}(\overline{H}_A^*)$, i.e., such that $(j,i)\in\mathcal{G}(\overline{H}_A^*)$. □

From this lemma the following theorem can be proved.

Theorem 4.3. *Given a resource assignment A and a solution H_A, if H_A is not optimal there is a sequence of interchange moves leading from H_A to an optimal solution H_A^*.*

Proof. The proof directly follows from Lemma 4.1 and Theorem 4.1. Starting from H_A and given an optimal solution H_A^*, Lemma 4.1 guarantees that there exists a pair of consecutive jobs that can be reversed, and Theorem 4.1 guarantees that the resulting solution, say $\widetilde{H}_A$, is feasible. If $\widetilde{H}_A$ is optimal the thesis follows, otherwise the same procedure can be repeated starting from $\widetilde{H}_A$, finally leading to an optimal solution. □

We have shown how an optimal sequencing for a given assignment A can be reached starting from any solution. We next show that an optimal assignment can be reached starting form any assignment. To this aim, we prove the following preliminary result.

Theorem 4.4. *Given a solution H, a machine h, a tool k and a job i, compatible with machine h [with tool k], if $i\notin\mathcal{J}^M(h)$ [if $i\notin\mathcal{J}^T(k)$] there always exists a rerouting move that moves i to $\mathcal{J}^M(h)$ [respectively, to $\mathcal{J}^T(k)$] leading to a feasible solution.*

Proof. Theorem 4.2 provides sufficient conditions for acyclicity after a rerouting move. Therefore, to demonstrate the thesis, it is sufficient to show that, if job i is compatible with machine h [with tool k], and given σ_h [given π_k], there are always in σ_h [in π_k] two consecutive jobs l and j such that i can be inserted between l and j.

We limit ourselves to prove this for move $\theta^M(i,j)$ only, the proof for $\theta^T(i,j)$ being very similar. First observe that every arc in σ_h always satisfies at least one of the two conditions in Theorem 4.2. In fact, if the first condition is not true, then $h(j)+\hat{p}_j\leqslant h(\gamma(i))$. It follows that $h(l)<h(j)+\hat{p}_j\leqslant h(\gamma(i))<h(\delta(i))+\hat{p}_{\delta(i)}$. Similarly, if the second condition is not true, then $h(\delta(i))+\hat{p}_{\delta(i)}\leqslant h(l)$, which implies $h(j)+\hat{p}_j\geqslant h(l)+\hat{p}_l+\hat{p}_j>h(l)\geqslant h(\delta(i))+p_{\delta(i)}>h(\gamma(i))$. In particular the second condition is always satisfied for node *start* and the first for at least an auxiliary node $i\in\{end, viol\_dead, viol\_due\}$. Note also that i cannot satisfy the second condition. Then, in σ_h there must be one last node l such that $h(\delta(i))+\hat{p}_{\delta(i)}>h(l)$ and $h(\delta(i))+\hat{p}_{\delta(i)}\leqslant h(\beta(l))$. Therefore, $j=\beta(l)$ must satisfy the condition $h(j)+\hat{p}_j>h(\gamma(i))$, which concludes the proof. □

Theorem 4.5. *The neighborhood defined by moves φ and θ is connected.*

Proof. Consider an optimal solution H^*. Theorem 4.3 shows that starting from any solution H_A it can be reached an optimal sequencing H_A^* for the given assignment A. If H_A^* is not globally optimal, there must be at least a job i which is assigned to a different resource in H_A^* and H^*. Theorem 4.4 guarantees that there exists a rerouting move that moves i to the same machine as in H^*, thus leading to a new feasible solution H'. If H' is optimal the thesis follows, otherwise the same procedure can be repeated starting from H', finally leading to an optimal solution. □

4.3. Neighborhood exploration

At each step of the algorithm we restrict the interchange move to consecutive jobs (i and $\alpha(i)$ or $\gamma(i)$) on a critical path, i.e.,

a longest path from *start* to one of the other auxiliary nodes. This is a common strategy, e.g., when solving job shop scheduling problems [14]. In fact, as observed, e.g., in [23], resequencing consecutive jobs that are not on the longest path from *start* to *end* cannot improve the makespan. Specifically, we analyze a critical path from *start* to *viol_dead* when $h(viol\_dead) > 0$. Otherwise, we explore the paths from *start* to *end* and *viol_due*.

When dealing with the *rerouting* move $\theta^M(i,j)$, we can move either jobs or unavailabilities. In the former case, we restrict the choice of i to the jobs positioned on a critical path and j to those nodes such that one of the two following conditions is satisfied:

- There is a strictly positive slack time $\Delta_j > 0$ before j, on the machine processing it, during which the machine is available and not busy;
- $j = \delta(i)$, i.e., j is the job using the same tool of i, immediately after it.

When the number of operators in a shift is smaller than b times the number of machines, we also allow to move unavailabilities. We say that an unavailability u is *critical* if one of the two following conditions holds: (1) there is a node j on a critical path of $\mathcal{G}(H)$ and u interrupts the processing of job j; (2) $\hat{r}_j > r_j$ and $\tilde{d}_u$ equals one of the quantities X_j, Y_j, Z_j, R_j defined in Fig. 1. We allow moving an unavailability i with move $\theta^M(i,j)$ only if i is a critical unavailability.

4.3.1. Critical paths and extended critical paths

We define two different critical paths. One is the classical longest path on $\mathcal{G}(H)$ from *start* to a specific auxiliary node, i.e., the path for which the sum of the arc weights is maximum. This path might contain a small number of arcs when its first arc is a modified release time $\hat{r}_j > r_j$. In this case $\hat{r}_j$ is computed according to (3) and can take into account the sequencing of a large number of jobs. Therefore, the starting time of job j might be reduced as well by anticipating the starting time of job $\alpha(j), \gamma(j)$ or by rerouting one of these two jobs since their modification can affect the value of $\hat{r}_j$. In other words, incorporating in the critical path a longest path from *start* to node j different from arc $\hat{r}_j$, would allow a larger possibility to improve upon the incumbent solution. More formally, let us consider an ordinary longest path from *start* to any auxiliary node $a \in \{end, viol\_dead, viol\_due\}$, and let j be the first node after *start* on this path. The extended critical path is recursively defined as the path including all the nodes on the longest path plus the extended critical path from *start* to j. The latter quantity is the empty set if $S_j = R_j$ holds in Eq. (1). If $S_j > R_j$ then, if $Y_j + g_{\alpha(j)j} \geqslant X_j + g_{\gamma(j)\beta(\gamma(j))}$ the extended critical path includes the longest path from *start* to $\alpha(j)$, otherwise it includes the longest path from *start* to $\gamma(j)$. These paths are evaluated iteratively with the same procedure.

4.4. Move evaluation

In the neighborhood of the incumbent solution H we look for a solution H' minimizing the following penalty function.

$$f(H') = b * max\{0, h(viol\_dead)\} + c * h(end) + d * max\{0, h(viol\_due)\}. \quad (4)$$

We developed two alternative algorithms for evaluating $f(H')$ after a move. The first one gives an estimate of $f(H')$ in constant time, while the second provides the exact value of $f(H')$ in linear time. At each step of the tabu search, the neighbor with the smallest value of $f(H')$ is selected as the new incumbent, and the schedule is updated accordingly. Then, according to the Eqs. (1) and (2) of Section 4.4.2, job starting and completion times are updated in $\mathcal{G}(H)$. We next describe the two evaluation algorithms.

4.4.1. Approximate evaluation

The approximate evaluation of $f(H')$ consists of using heads and tails of $\mathcal{G}(H)$ to estimate the length of a longest paths from *start* to an auxiliary node *end*, *viol_dead*, or *viol_due*. We recall the notation $h(i)$ and $q_a(i)$ [respectively, $h'(i)$ and $q'_a(i)$] to define the length of the longest path in $\mathcal{G}(H)$ [respectively, in $\mathcal{G}(H')$] from node start to i and from i to node $a \in \{end, viol\_dead, viol\_due\}$. With respect to move $\varphi(i,j)$, the estimate L'_a of the length of a longest path in $\mathcal{G}(H')$ from *start* to auxiliary node a is computed as an estimate of the longest path passing through i or j:

$$L'_a = max\{h'(i) + q'_a(i), h'(j) + q'_a(j)\}. \quad (5)$$

As for the solutions obtained with the $\varphi^M(i,j)$ move, we have (see Fig. 3):

$$h'(j) = \max\left\{Y_i + g_{\alpha(i)j} + f_{\alpha(i)j}; X_j + g_{\gamma(j)\beta(\gamma(j))} + f_{\alpha(i)j}; R_j\right\}$$
$$h'(i) = \max\left\{h'(j) + p_j + g_{ji} + f_{ji}; X_i + g_{\gamma(i)\beta(\gamma(i))} + f_{ji}; R_i\right\}$$
$$q'_a(i) = \max\left\{p_i + g_{i\beta(j)} + f_{i\beta(j)} + q_a(\beta(j)); p_i + g_{i\beta(j)} + f_{\alpha(\delta(i))\delta(i)} + q_a(\delta(i))\right\}$$
$$q'_a(j) = \max\left\{p_j + g_{ji} + f_{ji} + q'_a(i); p_j + g_{ji} + f_{\alpha(\delta(j))\delta(j)} + q_a(\delta(j))\right\}.$$

These four quantities can be computed in constant time using the information available from $\mathcal{G}(H)$. For moves $\varphi^T(i,j), \varphi^{MT}(i,j)$ there are similar equations. As for moves $\theta^M(i,j)$ [respectively, $\theta^T(i,j)$] we estimate the length of two longest paths: the one passing through the former nodes $\alpha(i)$ and $\beta(i)$ [respectively, $\gamma(i)$ and $\delta(i)$] and the one passing through i in $\mathcal{G}(H')$. Also these quantities can be computed in constant time using the information available from $\mathcal{G}(H)$.

4.4.2. Exact evaluation

We first notice that when changing the sequences σ and π, also the arc weights in $\mathcal{G}(H)$ may change. For this reason, it is necessary re-computing the exact values for the starting/completion times of each job after each move. With our exact evaluation strategy we compute these exact values not only after the application of a move, but also for evaluating the neighborhood. We next show that this computation requires linear time if the nodes of $\mathcal{G}(H')$ are numbered according to a topological order.

Theorem 4.6. *The exact evaluation of function $f(H')$ can be computed in time $O(n+q)$.*

Proof. Let $TO(j)$ be the position of node j in a topological order of the nodes of $\mathcal{G}(H')$. We notice that, for both kinds of moves $\varphi(i,j)$ and $\theta(i,j)$, the starting times of the nodes preceding $\min\{TO(i), TO(j)\}$ remain unchanged when passing from H to H'. If we are moving an unavailability, updating the values R_j for all $j \in \mathcal{J}$ requires linear time if the unavailabilities and the jobs are ordered for increasing values of r_j. Then, we consider the jobs in topological order. For each job j to be processed on machine h, we compute its starting time S_j by first scheduling the removal time $g_{\alpha(j)j}$ on machine h. If the removal time can be accommodated between the completion of job $\alpha(j)$ and the start of the next unavailability in $\mathcal{U}(h)$ this position is accepted, otherwise the value Y_j is computed by scanning the list of unavailabilities in $\mathcal{U}(h)$. Similar computation is necessary to compute X_j and to position the removal time $g_{\gamma(j)\beta(\gamma(j))}$ on the machine processing job $\gamma(j)$ and then the setup time $f_{\alpha(j)j}$. Finally, S_j can be computed as in Eq. (1). Given S_j, the completion time C_j can be computed in linear time according to Eq. (2) by computing the value $v(j)$ and inserting the unavailabilities in $\mathcal{U}(h)$ one at a time for increasing release time, starting from S_j. Note that the whole procedure requires a sequence of elementary steps, each requiring constant time. At each elementary step either: (i) a removal or a setup is positioned in the schedule,

or (ii) the starting/completion time of a job is defined, or (iii) it is decided that some unavailability precedes one of the previous values. This unavailability is not considered again in subsequent computations. Therefore, the overall evaluation requires a total of $O(n+q)$ elementary steps, and the thesis follows. □

5. Computational experiments

In this section we describe the performance of four different versions of our tabu search algorithm, developed by varying the size of the neighborhood and the evaluation strategy. We evaluate either the classical critical path (option A) or the extended one (option B) described in Section 4.3.1. We estimate the penalty function $f(H)$, for each H in the neighborhood of the incumbent, by choosing either the approximate evaluation (option C) or the exact one (option D), as described in Section 4.4. We have, therefore, four versions of the algorithm for the pairs of options A–C, A–D, B–C and B–D.

Each version depends on two parameters, namely the tabu list length λ and the number v of non-improving moves examined before applying a restart strategy. Our restart strategy consists of moving one unavailability from a machine to another, randomly chosen. This kind of move strongly disrupt the schedule since it generates an overload equal to the duration of one shift in one machine and a big slack in the other. For each version of the algorithm, λ and v are calibrated with the procedure CALIBRA, described in [1], for varying λ in the interval $[5,20]$, and v in the interval $[100,1000]$. CALIBRA uses the Taguchi methodology [19] for fractional factorial experimental designs coupled with a local search procedure to estimate the best values of all parameters. The resulting pairs (λ, v) for each version of our tabu search algorithm are: (9,876) for A–C, (16,876) for A–D, (7,876) for B–C and (18,876) for B–D version.

We fixed the values $b = 10^{10}, c = 10^5$ and $d = 1$ in the objective function $f(H)$ for all experiments. This corresponds to giving maximum priority to the respect of deadlines and then considering makespan and tardiness in lexicographical order, since a tardiness of two weeks is penalized less than increasing the makespan by one minute. In Section 5.2 we compare the four versions of our tabu search algorithm and draw several conclusions. All versions of the algorithm were coded in C language and were run on a Pentium® 4, 3.0 GHz with 1024 MB Ram. Two different sets of problem instances, described in the following section, were generated: the first set comes from the industrial practice and consists of two real instances and 24 realistic instances divided into three groups of easy, medium and hard instances. Realistic instances represent a wide range of possibilities that can arise in practice. For the real instances we compare the actual performance attained at the plant when scheduling by hand and by computer. For the realistic instances we compare the performance of our algorithms with that of the greedy algorithm described in Section 4.1, which was designed to perform similarly to the human schedulers of the plant. The second set of instances is randomly generated, in order to evaluate the performance of our algorithm in a more general context.

5.1. Data set description

The two real instances, correspond to the real production plan for several weeks of production during September and October 2006. The first instance concerns with 18 days of production during which 26 production orders are scheduled. The second instance concerns with 16 days of production during which 19 production orders are scheduled. In both instances, there are no urgent orders (i.e., no deadlines). All the jobs are available from the first day and the due dates are fixed equal to the end of the last working day. Operators availability in each week allows to activate three blister lines from Monday to Friday for two consecutive shifts of 7 h in each day, plus a short shift of three hours in which two blisters can be activated from Tuesday to Friday only.

The second set of 24 realistic instances is divided into three groups with 8 instances in each group. The first group contains easy instances. There are no urgent orders, the release dates [the due dates] coincide with the start of the first week [the end of the second week] for all jobs and the total processing times of all jobs (with the exclusion of removal and setup times) is approximately 50% of the department capacity. The second group contains instances of medium level of difficulty. There are no urgent orders but the release dates and the due dates may vary over the two weeks and the workload is slightly larger. The third group contains hard instances, with urgent orders, variable release/due dates and larger workload. Instances of this kind may arise when a disruption occurs in a different plant and its production is redirected to other plants, thus causing a larger workload and a number of urgent orders.

The second data set consists of 120 random instances, obtained by generating 10 instances for each pair (n,m), with the number of jobs n varying in the set $\{20,40,60,100\}$, and the number m of machines varying in $\{2,3,4\}$. Processing times, release dates, due

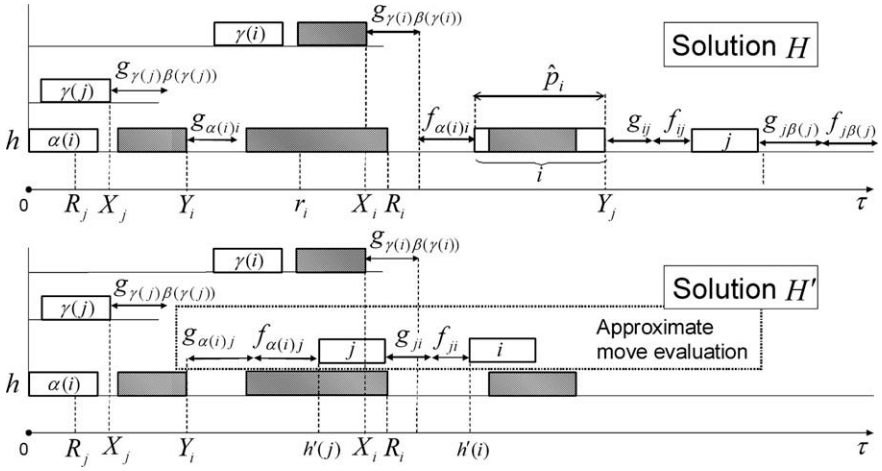

Fig. 3. Approximate evaluation of move $\varphi^M(i,j)$.

dates, deadlines, setups and removal times are also randomly generated, as well as the tool assignment for each job. The total workload for each instance is approximately equal to 90% of the department capacity.

5.2. Computational results

In this section we report on our computational experience on the real, realistic and random instances described in Section 5.1.

5.2.1. Real and realistic instances

In Table 1 we report on the results achieved by the four version of our tabu search. The first two lines refer to the two real instances. In columns 2 and 3 the makespan (in hours) and the maximum tardiness (in hours) for the manual solutions are shown and in the subsequent columns the same values are reported for the four configurations A–C, A–D, B–C and B–D. For these two instances, the greedy algorithm of Section 4.1 achieves in both instances $T_{max} = 0$, While $C_{max} = 393.00$ and $C_{max} = 443.00$, respectively for the first and the second instance. These values are quite similar to those obtained manually. The following tree lines of the table refer to the realistic instances. In each line, the average over the 8 instances is reported. In column"Manual", the performance of the greedy algorithm of Section 4.1 is reported, as a surrogate of the human schedulers. In the following five lines we report the average computation time required and the standard deviation of the makespan C_{max} for each group of instances and each algorithm.

The four versions of the tabu search algorithm clearly outperform the manual schedules, with respect to both makespan and tardiness. Specifically, for what concern the makespan C_{max}, the improvement achieved by the tabu search with respect to the manual/greedy algorithm is always quite larger than the standard deviation. For what concern T_{max}, the greedy algorithm violates the due dates in the 16 medium and hard instances, the tabu search version A–C violates the due dates in only one medium instance, the other three versions do never violate the due dates. For what concern the violation of the deadlines (not reported in table), the greedy algorithm violates the deadlines in the eight hard instances while the four versions of the tabu search do never violate the deadlines.

5.2.2. Random Instances

The second set of instances consists of 120 random instances described in Section 5.1. Table 2 shows the performance of the four configurations of our tabu search algorithm. For each version of the algorithm, columns C_{max} and T_{max} report the average values of makespan and maximum tardiness, respectively, on the 10 instances associated to each pair (n, m). Column σ shows the standard deviation of C_{max} over the 10 instances.

A few comments are in order. Computing the exact value of the objective function (option D) for all neighbors provide much better results than using an approximate evaluation (option C). In particular, combination B–D outperforms the others with respect to both makespan and maximum tardiness.

Using extended paths provides slightly better results than using the classical paths in combination with the exact evaluation. When using the approximate evaluation, better performance are provided by the classical path. In other words, exact evaluation performs better with a larger neighborhood, while approximate

Table 1
Comparison between manual and computerized schedules.

Instance	Manual		A–C		A–D		B–C		B–D	
	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}
1	408.75	0	380	0	377	0	377	0	377	0
2	440.25	0	424.25	0	424	0	424	0	424	0
Easy	255.38	0	249.38	0	240.14	0	249.63	0	238.50	0
Medium	368.31	77.31	256.13	13.56	248.63	0	254.56	0	248.88	0
Difficult	381.13	90.13	267.69	0	266.94	0	270.19	0	264.63	0
	σ	*Time*	σ	*Time*	σ	*Time*	σ	*Time*	σ	*Time*
1	–	> 3600	–	4	–	3	–	13	–	1
2	–	> 3600	–	<1	–	1	–	< 1	–	1
Easy	5.12	<1	6.66	11.63	8.56	10.29	3.70	13.38	8.02	11.75
Medium	21.43	<1	7.85	11.25	7.62	14.38	10.15	16.63	11.21	16.00
Difficult	17.97	<1	9.04	15.00	8.95	21.13	6.37	15.13	7.65	18.88

Table 2
Performance of the tabu search algorithm for varying n and m.

n	m	A–C			A–D			B–C			B–D		
		C_{max}	T_{max}	σ	C_{max}	T_{max}	σ	C_{max}	T_{max}	σ	C_{max}	T_{max}	σ
20	2	252.32	0	17.96	243.43	0	18.87	251.98	0	16.53	243.51	0	19.61
40	2	251.48	0.19	10.23	243.45	0	10.49	251.66	0	10.42	243.26	0	10.46
60	2	247.69	0	11.83	238.38	0	11.81	247.36	0	11.00	238.87	0	10.90
100	2	272.85	28.01	8.53	268.49	0	9.10	273.27	26.94	8.21	268.17	0	9.22
20	3	272.84	0	45.70	256.80	0	39.08	272.11	0	45.56	256.35	0	40.06
40	3	258.94	0	11.46	250.38	0	11.63	261.19	0	11.09	249.97	0.76	10.93
60	3	253.13	0	14.11	246.16	0	14.36	255.19	0	14.30	245.69	0	14.82
100	3	260.08	8.21	9.52	253.38	0	10.01	260.23	1.49	9.82	253.54	0	9.87
20	4	256.77	0	16.39	251.57	0	20.64	260.82	0	29.62	250.75	0	19.20
40	4	255.32	3.32	10.80	245.66	0	11.91	254.69	0	11.13	245.01	0	11.73
60	4	241.70	0	14.89	233.61	1.14	13.97	244.04	0	13.31	233.36	0	14.78
100	4	253.17	0	11.66	247.73	0	9.98	253.31	0	10.10	246.68	0	10.92
Average		256.36	3.31	15.26	248.25	0.10	15.15	257.15	2.37	15.92	247.93	0.06	15.21

evaluation performs best with a smaller neighborhood, the improvements being reached after several restart phases.

It is interesting analyzing the time needed by the algorithm to reach the best solution found within one minute of computation time. With combinations A–C and A–D the best solutions are found after 7.53 and 9.97 s on average, respectively. With combinations B–C and B–D the best solutions are found after 6.83 and 10.03 s on average, respectively. We notice that despite the larger number of solutions to explore in each iteration using the extended path requires approximately the same computation time to find the best solution.

6. Conclusions

In this paper, we studied a practical scheduling problem arising in the packaging department of a pharmaceutical production plant. The problem is formulated as a multi-purpose machine scheduling problem with additional constraints. A tabu search algorithm is able to find good solutions within short computation time, compared to the solutions found by hand and by a greedy algorithm. The makespan reduction is between 3.6% and 7.5% for easy instances and increases to more than 30% for hard instances, which is a remarkable improvement in the productivity of the plant. More important, when analyzed by the plant managers the solutions were considered feasible in practice. The results achieved confirm that scheduling technology is mature to solve complex real problems. To this aim, however, it is important to face the complexity of practical scheduling problems by using detailed scheduling models.

Acknowledgement

This work has been partially funded by the Italian Ministry of Research, Grant number RBIP06BZW8. The authors are also grateful to the anonymous reviewers for their helpful comments.

References

[1] B. Adenso-Díaz, M. Laguna, Fine-tuning of algorithms using fractional experimental designs and local search, Operations Research 54 (2006) 99–114.
[2] O. Braysy, A reactive variable neighborhood search for the vehicle-routing problem with time windows, INFORMS Journal on Computing 15 (4) (2003) 347–368.
[3] O. Braysy, M. Gendreau, Vehicle routing problem with time windows. Part II: Metaheuristics, Transportation Science 39 (1) (2005) 119–139.
[4] P. Brucker, A. Drexl, R. Mohring, K. Neumann, E. Pesch, Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research 112 (1) (1999) 3–41.
[5] W.C. Chiang, R.A. Russell, A reactive tabu search metaheuristic for the vehicle routing problem with time windows, INFORMS Journal on Computing 9 (4) (1997) 417–430.
[6] S. Dauzère-Pérès, W. Roux, J.B. Lasserre, Multi-resource shop scheduling with resource flexibility, European Journal of Operational Research 107 (2) (1998) 289–305.
[7] P.M. Franca, M. Gendreau, G. Laporte, F.M. Muller, A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times, International Journal of Production Economics 43 (1996) 78–89.
[8] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13 (1986) 533–549.
[9] F. Glover, M. Laguna, Tabu Search, Kluwer, 1997.
[10] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic machine scheduling: A survey, Annals of Discrete Mathematics 5 (1979) 287–326.
[11] K.N. McKay, M. Pinedo, S. Webster, Practice-focused research issues for scheduling systems, Production and Operations Management 11 (2002) 249–258.
[12] K.N. McKay, V.C.S. Wiers, Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory, Computers in Industry 50 (2003) 5–14.
[13] T.E. Morton, D.W. Pentico, Heuristic Scheduling Systems, John Wiley & Sons, 1993.
[14] E. Nowicki, C. Smutnicki, An advanced tabu search algorithm for the job shop problem, Journal of Scheduling 8 (2005) 145–159.
[15] D. Pacciarelli, The alternative graph formulation for solving complex factory scheduling problems, International Journal of Production Research 40 (2002) 3641–3653.
[16] M. Pinedo, Scheduling. Theory, algorithms, and systems, Prentice-Hall, 1995.
[17] M. Pinedo, Planning and Scheduling in Manufacturing and Services, Springer, 2005.
[18] A. Reisman, A. Kumar, J. Motwani, Flowshop scheduling/sequencing research: A statistical review of the literature, 1952–1994, IEEE Transactions on Engineering Management 44 (1997) 316–329.
[19] R.K. Roy, A Primer on the Taguchi Method, Van Nostrand Reinhold, New York, 1990.
[20] R. Ruiz, F.S. Şerifoğlu, T. Urlings, Modeling realistic hybrid flexible flowshop scheduling problems, Computers and Operations Research 35 (2008) 1151–1175.
[21] J.M.J. Schutten, R.A.M. Leussink, Parallel machine scheduling with release dates, due dates and family setup times, International Journal of Production Economics 46 (1996) 119–125.
[22] V. T'kindt, J.C. Billaut, Multicriteria Scheduling, Springer, 2006.
[23] P.J.M. van Laarhoven, E.H.L. Aarts, J.K. Lenstra, Job shop scheduling by simulated annealing, Operations Research 40 (1992) 113–125.
[24] G.H. Vieira, J.W. Herrmann, E. Lin, Rescheduling manufacturing systems: A framework of strategies, policies, and methods, Journal of Scheduling 6 (2003) 39–62.