Job-shop scheduling models with set-up times

Marco Ballicu, Alessandro Giua, Carla Seatzu

Dip. Ingegneria Elettrica ed Elettronica, Università di Cagliari,
Piazza d'Armi, 09123 Cagliari, Italy.
marcobal@tiscalinet.it, {giua,seatzu}@diee.unica.it

Abstract— **In this paper we consider the classical representation of job-shop scheduling problems in terms of disjunctive graphs. We derive a mixed integer-linear programming model that keeps track of the immediate precedence relation between operations. Finally, we show how this framework can be used to solve job-shop scheduling problems with sequence-dependent set-up times. Two cases are considered: the set-up operation required before processing a job on a machine may start as soon as the machine is available, or only if both job and machine are available.**

I. Introduction

A classical representation of a job-shop scheduling problem takes the form of a graph [2], [3] where each node represents an operation and the constraints among operations are represented by two types of arcs. Precedence constraints between operations that belong to the same job are represented by a path of *conjunctive* arcs that specifies the order in which the different operations that compose the job should be executed. Mutual exclusion constraints between operations that should be executed on the same machine are represented by a set of *disjunctive* arcs. The solution of a scheduling problem requires to find, among all sets of disjunctive arcs that link all operations belonging to the same machine, an acyclic directed path (that represents the order in which the machine processes the different operations) so as to optimize a given performance index.

A common model [2] used to describe a job-shop scheduling problem with N operations is a mixed integer-linear program (MILP) in the unknowns $x_{ij} \in \{0,1\}$ where $i,j = 1,\ldots,N$. The variable x_{ij} takes the value 1 if operation j is processed after operation i and takes the value 0 if operation i is processed after operation j. Note that *after* does not necessarily mean *immediately after*, in the sense that if operations 1, 2 and 3 are processed in the order $(1,2,3)$, then $x_{12} = x_{13} = x_{23} = 1$, i.e., operation 1 is processed before operation 3 albeit not immediately so.

The fact that we do not keep the information on the immediate precedence relation between operations does not create any problem when we solve scheduling problems in which the set-up times are not considered. When the set-up times between operations associated to different jobs are specified, this model may need to be changed. Assume, in fact, that the length of the set-up depends on the job just completed *and* on the one about to be started: we say that the set-up times are *sequence-dependent* [3]. In this case it is necessary to keep the information on the immediate precedence relation between operations.

In [2] it was shown an MILP that solves the *single machine* scheduling problem $1/s_{ij}/\gamma$ with sequence-dependent set-ups. In [2] it was taken $\gamma = \sum_i w_i C_i$, i.e., the objective was that of minimizing the total weighted completion time, but the same approach can be easily applied to a wider class of objective functions. The new MILP contained unknowns $x_{ij} \in \{0,1\}$: variable x_{ij} takes the value 1 if and only if operation j is processed *immediately* after operation i.

In this paper we derive a different MILP that can be applied to more general job-shop scheduling problems $J/s_{ij}/\gamma$ with sequence dependent set-ups and *an arbitrary number of machines*. The formulation we propose in this paper may handle two different cases. In the first case we assume that the set-up needs to be performed only on the machine, and may start as soon as the machine has completed the precedent operation. In the second case, we assume that the set-up can only be performed if *both* machine and job are available. In this second case falls the class of set-up problems that can also be described with the software Lekin presented in [3], based on heuristic algorithms.

Finally, we compare the results of our MILP approach with the results obtained with the software Lekin, applying both programs to the same example in the service area. In particular, we found out that for some performance indices the approach in terms of integer programming provides a better solution with respect to the heuristic approach. This paper is based on the work presented in [1].

II. The job shop scheduling problem

Scheduling problems involve a set of jobs and a set of processors. Each job consists of a set of operations. For each operation we know the set of machines able to perform it and its processing time. Note that the processing time is known because lots, unlike in lot–sizing problems, are already sized. Operations are related by precedence constraints [2]. Classical machine scheduling models rely on a set of restrictive assumptions that are briefly summarized in the following:

- single parts and batches of parts are always treated as a single job;
- preemption is not allowed;
- job cancellation is not allowed;
- processing times are independent of the schedule;
- work-in-process is allowed: this means that jobs may wait in a queue until the next machine required for processing is free;
- machines are able to process one job at a time;
- each job visits all machines at most once;
- machines are always available.

In scheduling problems each job is characterized by a sequence of operations. For each operation we exactly know the machine in which it will be executed and its process time. The main parameters identifying a job are the following [2]:

• C_i: the *completion time* of the i–th job, i.e., the time at which the last operation of the i–th job is completed;
• d_i: the *due date*, denoting the time at which the i–th job should be completed. Note that a due date represents a soft constraint, in the sense that it can be violated at a certain price, whereas a *deadline* is a hard constraint;
• r_i: the *release time*, i.e., the time at which the job is released and we may start processing it;
• $F_i = C_i - r_i$: the *flow time* that is usually considered synonymous of lead time since it is the time the job spends on the shop floor.

Different classification schemes of scheduling problems have been proposed in the literature [2], [5], [6]. The most common one is due to Graham [6] and is based on a three field coding. The classical objective functions are built by considering the following elementary functions [2]:
• *flow time*: $F_i = C_i - r_i$;
• *lateness*: $L_i = C_i - d_i$;
• *tardiness*: $T_i = max\{L_i, 0\}$;
• *earliness*: $E_i = max\{-L_i, 0\}$.
The most commonly used objective functions are either of the 'minsum' or the 'minmax' type. They are mostly built by combining elementary functions $\gamma_i(C_i)$ of the completion time of the i–th job. The most significant minsum objective functions are:
• $\sum_i C_i$: *total completion time*;
• $\sum_i T_i$: *total tardiness*;
• $\sum_i w_i C_i$: *total weighted completion time*;
• $\sum_i w_i T_i$: *total weighted tardiness*.
The most significant minmax objective functions are:
• $L_{\max} = \max_i L_i$: *maximum lateness*;
• $C_{\max} = \max_i C_i$: *makespan*.
Note that makespan is related to machine utilization and minimizing makespan implies minimizing machine idle time [2].

III. The disjunctive graph

In this section we briefly recall how the problem of minimizing the makespan in a job shop can be represented by a disjunctive graph. For more details we address to [3].

Consider a job shop problem with n jobs and m machines. Each job has to be processed by a set of machines in a given order, and there is no recirculation. The processing of job j on machine i is referred to as operation (i, j), and its duration is $p_{i,j}$.

Consider a directed graph G with a set of nodes A and two sets of arcs P and D. The nodes A correspond to all the operations (i, j) that must be performed on the n jobs. The so-called *conjunctive* (solid) arcs P represent the routes of the jobs. If arc $(i, j) \rightarrow (k, j)$ is part of P, then job j has to be processed on machine i before being processed on machine k; that is operation (i, j) precedes operation (k, j). Two operations that belong to two different jobs and that have to be processed on the same machine are connected to one another by two so-called *disjunctive* (broken) arcs going in opposite directions. The disjuntive arcs D form m cliques of double arcs, one clique for each machine [1]. All operations (nodes) on the same clique have to be done on the same machine. All arcs outputting from a node, conjunctive as well as disjunctive, have associated a weight representing the processing time of the operation that is represented by that node. In addition, there is a source S and a sink U, which are dummy nodes. The source node S has in output n conjunctive arcs going to the first operations of the n jobs, and the sink node U has in input n conjunctive arcs coming from all the final operations. The arcs emanating from the source have length 0. We denote this graph by $G = (A, P, D)$. An example is reported in figure 1.a [3] where for simplicity of notation, cliques have been denoted as double arrows, and many arc weights are not shown.

Note that in many cases a simpler representation of the previous graph is adopted, where each node is denoted by only one integer number instead of a couple. The equivalent representation of the graph in figure 1.a is reported in figure 1.b. We may also observe that only one subscript is required to denote the duration of each operation. Thus, in general, p_j denotes the processing time of the j-th operation represented by the node j.

A feasible schedule corresponds to a *selection* of one disjunctive arc from each pair such that the resulting directed graph is acyclic. That the graph is acyclic implies that the selection of arcs within a clique must be acyclic. Such a selection determines the sequence in which the operations have to be performed on that machine [3].

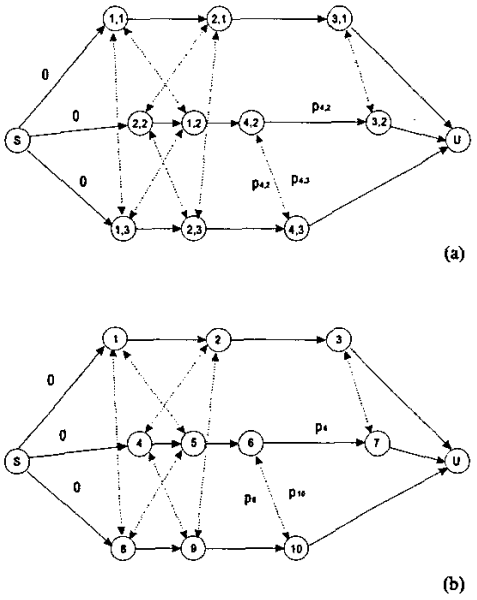

Fig. 1. Disjunctive graphs.

IV. A mathematical model with no set-up times

In this section we discuss in detail a common model used to describe a job-shop scheduling problem with N operations that is based on the disjunctive graph [2]. Using the same notation as above, $A = \{S, 1, \cdots, N, U\}$

[1] *Clique* is a term in graph theory that refers to a graph in which any two nodes are connected to one another; in this case each connection within a clique is a pair of disjunctive arcs [3].

denotes the set of nodes of the disjunctive graph, while P is the set of conjunctive arcs. We also define $A' = \{1, \cdots, N\}$ as the set of nodes that actually correspond to the operations. Moreover, we call D' the set of couples of nodes relative to the operations of different jobs processed by the same machine, i.e., $D' = \{(i,j) \mid \exists$ a disjunctive arc from i to j with $i < j\}$. Finally, A_S and A_U are the set of successors and predecessors of S and U respectively. We assume that our goal is that of minimizing the makespan.

This problem may be easily written an a mixed integer-linear program (MILP) [2]:

$$\begin{cases} \min C_{\max} \\ s.t. \\ (a) \quad C_j \geq C_i + p_j & \forall (i,j) \in P \\ (b) \quad \begin{cases} C_j \geq C_i + p_j - M(1 - x_{ij}) \\ C_i \geq C_j + p_i - M x_{ij} \end{cases} & \forall (i,j) \in D' \\ (c) \quad C_i \geq p_i + r_i & \forall i \in A_S \\ (d) \quad C_{\max} \geq C_i & \forall i \in A_U \\ (e) \quad x_{ij} \in \{0,1\} & \forall (i,j) \in D'. \end{cases} \tag{1}$$

The known terms are: the process times p_i for all $i \in A'$, the release times r_i for all $i \in A_S$, an arbitrarily large constant M that should be greater than the maximum allowable makespan.

The unknowns are: the completion time C_i for all $i \in A'$, i.e., the time at which the i-th operation is completed; the makespan $C_{\max}$, and the binary variables $x_{ij} \in \{0,1\}$ for $i,j = 1, \cdots, N$. The variable x_{ij} takes the value 1 if operation j is processed after operation i on some machine and takes the value 0 if operation i is processed after operation j. Note that *after* does not necessarily means *immediately after*, in the sense that if operations 1, 2 and 3 are processed in the order $(1,2,3)$, then $x_{12} = x_{13} = x_{23} = 1$.

Constraint (a) refers to operations belonging to the same job, and imposes that operations should be executed according to a pre-specified ordering.

Equations (b) imply that if two operations require the same machine, they cannot overlap, i.e., one must be scheduled before the other. In formulae, we must enforce a disjunctive constraint:

$$\text{either} \quad C_j \geq C_i + p_j$$

$$\text{or} \quad C_i \geq C_j + p_i.$$

We enforce the disjunction of these constraints by introducing a suitable large constant M and requiring

$$C_j - p_j \geq C_i - M x_{ji} = C_i - M(1 - x_{ij})$$

$$C_i - p_i \geq C_j - M x_{ij}$$

where M should be an upper bound on the schedule makespan. If $x_{ij} = 1$ and $x_{ji} = 0$, i.e., operation i precedes operation j, the second constraint is redundant and the first one is enforced; the contrary happens if $x_{ij} = 0$ and $x_{ji} = 1$.

Equation (c) is relative to the first operation of each job and imposes that the completion time should be greater or equal to the sum of the release time and the processing time of that operation.

Equation (d) refers to the last operation of each job.

So as to better clarify the above problem formulation, let us consider the disjunctive graph in figure 1.b. Let us focus our attention on the first job. The set of constraints relative to that job are:

$$\begin{array}{ll} (a) & C_2 \geq C_1 + p_2; \;\; C_3 \geq C_2 + p_3 \\ (c) & C_1 \geq p_1 + r_1 \\ (d) & C_{\max} \geq C_3. \end{array}$$

Similar constraints should also be written for the other two jobs.

Let us now consider the first machine. Constraints (b) should be written for each disjunctive arc connecting operations 1, 5 and 8:

$$\text{Arc } 1 \longleftrightarrow 5 \begin{cases} C_5 \geq C_1 + p_5 - M(1 - x_{15}) \\ C_1 \geq C_5 + p_1 - M x_{15} \end{cases}$$

$$\text{Arc } 1 \longleftrightarrow 8 \begin{cases} C_8 \geq C_1 + p_8 - M(1 - x_{18}) \\ C_1 \geq C_8 + p_1 - M x_{18} \end{cases}$$

$$\text{Arc } 5 \longleftrightarrow 8 \begin{cases} C_8 \geq C_5 + p_8 - M(1 - x_{58}) \\ C_5 \geq C_8 + p_5 - M x_{58} \end{cases}$$

Similar constraints hold for the other machines.

The advantage of this mathematical formulation is that it can be easily solved with an appropriate software tool for integer programming problems, such as Lindo.

Let us finally observe that also the solution of the scheduling problem may be efficiently represented with a graph. As an example, if we consider again the above scheduling problem, one admissible solution for the first machine may be represented as shown in figures 2.a-b. Both these figures represents the same solution in which machine 1 first processes operation 1, then operation 5 and finally operation 8. In figure 2.a. each arc represents an *immediate* precedence, while in figure 2.b. the additional arc $1 \rightarrow 8$ represents a *non-immediate* precedence. On the contrary, an unfeasible solution is shown in figure 2.c. In fact, in such a case the set of equations (b) would be:

$$\begin{array}{l} C_5 \geq C_1 + p_5 \\ C_8 \geq C_5 + p_8 \\ C_1 \geq C_8 + p_1 \end{array}$$

that has no solution being all processing times different from zero.

To conclude, we also observe that even if all machines admit acyclic solutions, this does not imply a priori that the graph relative to the whole scheduling problem is acyclic as well. As an example, let us consider the graph in figure 2.d. In this case there is no cycle associated to the single machines, but the graph contains the cycle $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$. Nevertheless, we may easily verify that a spurious solution of this kind may never occur due to equations (a) and (b) that force the following constraints:

$$\begin{array}{ll} (a) & C_3 \geq C_1 + p_3; \;\; C_4 \geq C_2 + p_4 \\ (b) & C_2 \geq C_3 + p_2; \;\; C_1 \geq C_4 + p_1. \end{array}$$

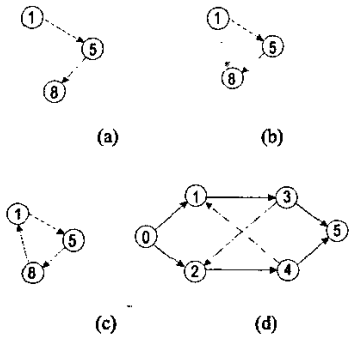

Fig. 2. Disjunctive graphs.

V. A MATHEMATICAL MODEL WITH SET-UP TIMES

Machines often have to be reconfigured or cleaned between jobs. This process is known as a changeover or *set-up*. If the length of the set-up depends on the job just completed and on the one about to be started, then the set-up times are *sequence-dependent* [3].

For example, paint operations often require changeover. Every time a new color is used, the painting devices must be cleaned. The cleanup time often depends on the color just used as well as the colour about to be used. In practice the best sequence is to go from light to dark colors because the cleanup process is easier.

Besides taking valuable machine time, set-ups also involve costs in the form of labor, waste of raw material, and so on. For example, machines in the process and chemical industries are not stopped when going from one grade of material to another. Instead, a certain amount of the material produced at the start of a new run is usually off-quality and is discarded or recirculated [3].

In this section we present the original contribution of this work. In particular, we derive a new model that can be applied to different job-shop problems with set-up times. In fact, the new model enables us to take into account the exact order of operations, and this additional information is an essential requirement in many scheduling problems with set-ups. On the contrary, this information was not considered in the previous model. In fact, in that case, whenever the binary variable $x_{ij} = 1$ we know that operation i precedes operation j, but we do not know if i comes *immediately* before j or if some other operation is performed among them. As an example, in figure 2.b an arc connecting nodes 1 and 8 has been drawn denoting that operation 1 precedes operation 8. However, we can immediately observe that, given the other two arcs, it is redundant and can be omitted (see figure 2.a).

As an intermediate result we prove the following lemma.

Lemma 1: Given a positive integer $n \in \mathbb{N}^+$ consider the constraint set

$$\begin{cases} \sum_{i=1}^{n} \sum_{j=1;\ j\neq i}^{n} x_{ij} = n-1 \\ \sum_{j=1;\ j\neq i}^{n} x_{ij} \leq 1 \qquad \forall i = 1,\dots,n \\ \sum_{i=1;\ i\neq j}^{n} x_{ij} \leq 1 \qquad \forall j = 1,\dots,n \\ x_{ij} \in \{0,1\} \end{cases} \tag{2}$$

Let $X^* = \{0,1\}^{n\times n}$ be a matrix whose elements x^*_{ij} (for $i \neq j$) are feasible solutions of (2) while $x^*_{ii} = 0$ (for $i = 1,\dots,n$). The directed graph $G = (V, B)$ with set of nodes $V = \{1,\dots,n\}$ and set of directed arcs $B = \{(i,j) \mid x^*_{ij} = 1\}$ is such that $\text{card}(B) = n-1$ and consists of one directed acyclic path of length between 0 and $n-1$ plus zero or more elementary directed cycles.

Proof: To prove the above statement, we may consider, without loss of generality, a graph with four nodes. In this case, all solutions reported in figure 3 are feasible. We want to prove that these solutions are the only ones, apart from those that can be obtained by simply renaming the nodes. But this immediately follows from the consideration that the statement of the lemma would be violated if any of the situations reported in figure 4 occur. The first case (a) would violate the second condition stating that at most one arc may exit from each node. The second case (b) is in contrast with the third condition, that states that at most one arc can enter a node. The third case (c) that contains $K > 1$ acyclic paths violates the first condition since the number of arcs is in this case $n - K < n - 1$. □

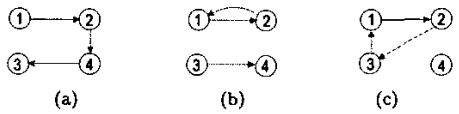

Fig. 3. Feasible solutions.

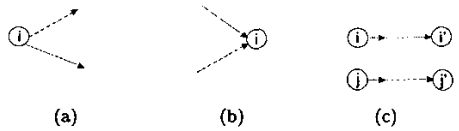

Fig. 4. Unfeasible solutions.

Now, let us introduce a new mathematical model that also takes into account the set-up times.

Let $A = \{S, 1, \cdots, N, U\}$ be the set of nodes of the disjunctive graph representative of a scheduling problem, and $A' = \{1, \cdots, N\}$ be the set of nodes obtained from the previous one by simply removing the dummy nodes. As in the previous section, A_S and A_U are the set of successors nodes of S and the set of predecessors nodes of U, respectively. Let P be the set of conjunctive

arcs connecting nodes relative to operations in the same job. Moreover, let $\{\Delta_1, \cdots, \Delta_m\}$ be a partition of A' in m classes, where m is the number of machines. Thus, by definition, $(\cup_{q=1}^{m} \Delta_q) \cup \{S, U\} = A$ and $\Delta_q \cap \Delta_{q'} = \emptyset$ if $q \neq q'$. Finally, let $D_q = \{(i,j) \in \Delta_q \times \Delta_q \,|\, i \neq j\}$ and $D' = \cup_{q=1}^{k} D_q$.

Now, assume that the performance index we want to minimize is the makespan (but it is easy to extend this approach to a wider class of objective functions). Then we can write the following MILP.

$$
\begin{cases}
\min C_{\max} & \\
s.t. & \\
(a)\ C_j \geq C_l + p_j & \forall (l,j) \in P \\
(b)\ C_j \geq p_j + r_j & \forall j \in A_S \\
(c)\ C_j \geq C_i + p_j + s_{ij} - M(1 - x_{ij}) & \forall (i,j) \in D' \\
(d)\ C_j \geq C_l + p_j + s_{ij} - M(1 - x_{ij}) & \begin{cases} \forall (i,j) \in D' \\ \forall (l,j) \in P \end{cases} \\
(e)\ \sum_{(i,j) \in D_q} x_{ij} = |\Delta_q| - 1 & \forall q \in \{1, \cdots, m\} \\
(f)\ \sum_{j:(i,j) \in D'} x_{ij} \leq 1 & \forall i \in A' \\
(g)\ \sum_{i:(i,j) \in D'} x_{ij} \leq 1 & \forall j \in A' \\
(h)\ C_{\max} \geq C_i & \forall i \in A_U \\
(i)\ x_{ij} \in \{0,1\} &
\end{cases}
\tag{3}
$$

The known terms are: the processing times p_j for all $j \in A'$, the release times r_j for all $j \in A_S$, the set-up times s_{ij} for all couples $(i,j) \in D'$, the constant M, and the number of nodes in each disjunctive class Δ_q.

The unknown variables are: the completion times C_j for all $j \in A'$, the makespan, and the binary variables x_{ij}, for $i, j = 1, \cdots, N$.

Note that if we also want to take into account both the tardiness and the weighted tardiness, we simply have to add constraints (d), (e) and (g) of the previous problem (1) with no set-up times.

Now, we will informally discuss in detail the set of constraints.

We can immediately observe the equivalence between constraints (a) and (b) of problem (3) with constraints (a) and (c) of problem (1).

In equation (c) we have also included the set-up times that occur whenever job i is processed before job j in a certain machine: s_{ij} denotes the time interval required by the machine before starting the processing of job j, after the processing of i is completed. If the set-up needs to be performed only on the machine, and may start as soon as the machine has completed the precedent operation constraints (c) captures all constraints imposed by the set-up.

On the contrary, if the set-up can only be performed whenever *both* machine and job are available we also need to add constraint (d) that says that the set-up from i to j cannot start before the completion of the operation l belonging to the same job of j and that is the immediate predecessor of operation j.

Equation (e) establishes that the number of arcs of the sub-graph relative to the generic machine q is equal to the number of nodes minus one.

Equations (f) and (g) make sure that the number of input and output arcs of each node is at most equal to one.

Finally, equation (i) states that x_{ij} are binary variables. In particular, if we compute $x_{ij} = 1$ then operation j is processed immediately after operation i.

Proposition 1: The MILP (3) solve the problem $J/s_{ij}/C_{\max}$ when the set-up may start only if both machine and job are available. By removing constraints (d) we obtain a solution for the case in which the set-up may start as soon as the machine is available.

Proof: Follows from the previous discussion, and the fact that equations (e), (f) and (g) are the conditions of lemma 1 written for each machine. The introduction of these equations in the set of constraints, insures that for each machine the resulting graph can only assume one of the structures shown in figure 4. Moreover, if all processing time are non zero there may be no cycles in the graph that describes the solution (as in figure 2.a) thus solutions like those reported in figure 3.b and c are not feasible. □

VI. A NUMERICAL EXAMPLE

In this section we compare the results obtained with the approach discussed in the previous section with those obtained using the software Lekin. We consider a working area consisting of four buildings (see figure 5) that have to be restored.

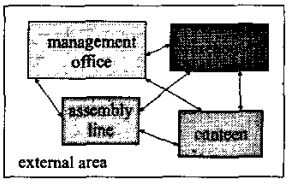

Fig. 5. Working area.

Each sub-area may be considered as a different job, i.e., we define: $J1$ (management office), $J2$ (warehouse), $J3$ (assembly line), $J4$ (canteen), and $J5$ (external area).

Each machine may be considered as a team involved in a specific operation: $M1$ (cleaning team), $M2$ (informatic team, e.g., LAN net installation), $M3$ (telephony team), $M4$ (restoring team), $M5$ (equipment transport and assembly team).

We assume that each job can be processed by only one machine at a time and each machine can process only one job at a time. Moreover, we assume that for each job the first operation is $M1$, i.e., each area should be cleaned before any other operation is performed on it. Finally, the equipment transport and assembly work cannot be performed before the restoring is finished, i.e., $M4$ must always follow $M5$.

Numerical data are summarized as follows. Table 6 shows the order in which the operations ($M1$ to $M5$) should be processed on different jobs, and the corresponding processing times. Also due dates are reported, while release times are set to 0.

Set-up times are shown in figure 7 and denote the time a certain team requires to move its equipment from a sub-area, where an operation has been just finished, to another one where the next operation should be performed. We assume that each set-up operation can only be executed if both the machine and the job are available. Note that in the first line of each table (referred to as the zero line) we have reported the time a machine

	(Machine, processing time)				due date
J1	(M1,15)	(M3,20)	(M2,18)	(M4,35) (M5,10)	100
J2	(M1,16)	(M4,10)	(M3,5)	(M2,8)	100
J3	(M1,22)	(M2,14)	(M4,22)	(M5,36)	100
J4	(M1,17)	(M4,8)	(M5,12)	(M3,14)	100
J5	(M1,28)	(M3,21)	(M5,18)	(M2,15)	100

Fig. 6. Processing times and due dates.

requires before starting to process a given job in the case that this job is the first one to be executed.

Many scheduling problems have been considered with different objective functions. Each one required the solution of a MILP of the form (3) where the set of constraints (that are not reported here for brevity's sake) always keeps the same. The software Lindo has been used to solve them. Numerical results are summarized in the first table of figure 8 where the most significant parameters are given depending on the objective functions. Computational times are also reported.

Finally, the same job-shop problems have also been solved using the software Lekin and different heuristic methods have been adopted. In particular, we have used algorithms of three types: general algorithms with a shifting bottleneck nature ($GR/C_{\max}$, $GR/T_{\max}$, $GR/\sum C$, $GR/\sum T$), algorithms with two objective functions ($SB/T_{\max}$), and algorithms based on priority rules (EDD, $FCFS$, LPT, SPT). More details on them can be found in [3]. Numerical results are reported in the second table in figure 8 where the same characteristic parameters are computed for all the objective functions.

Comparing the results obtained with the two approaches, we can immediately observe that when our goal is that of minimizing either the total tardiness or the total completion time, the approach based on linear programming provides a better solution. On the contrary, when our goal is that of minimizing either the makespan or the maximum tardiness, some heuristic algorithms also provide the optimal solution. Finally, we may observe that in every case the computational time required by heuristic approaches is significantly less than that required for the solution of the MILP.

VII. Conclusions

In this paper we have derived an original model of job-shop scheduling problems. In particular, we have shown how it is possible to write down an mixed integer-linear program with unknown binary variables that keep into account the exact order in which operations are performed. The main advantage of the proposed approach is that it can be used to solve job-shop scheduling problems with set-up times.

References

[1] M. Ballicu, *Job shop scheduling: a comparison between exact and heuristic methods*, Laurea Thesis, DIEE, University of Cagliari, Italy, 2001.

[2] P. Brandimarte, A. Villa, *Advanced models for manufacturing systems management*, CRC Press, 1995.

M1 : Cleaning Team

	J1	J2	J3	J4	J5
0	1	1	1	1	1
J1		8	4	10	1
J2	8		6	3	1
J3	4	6		5	5
J4	10	3	5		1
J5	1	1	1	1	

M2 : Informatic Team

	J1	J2	J3	J5
0	1	1	1	1
J1		4	2	1
J2	4		3	1
J3	2	3		1
J5	1	1	1	

M4 : Restoring

	J1	J2	J3	J4
0	1	1	1	1
J1		7	3	8
J2	7		5	2
J3	3	5		4
J4	8	2	4	

M3 : Telephony Team

	J1	J2	J4	J5
0	1	1	1	1
J1		4	5	1
J2	8		3	1
J4	5	1		1
J5	1	1	1	

M5 : Eq. transport Team

	J1	J3	J4	J5
0	1	1	1	1
J1		5	12	1
J3	5		6	1
J4	12	6		1
J5	1	1	1	

Fig. 7. Set-up times.

objective function	C_{max}	$\Sigma_i C_i$	$\Sigma_i T_i$	T_{max}	Time(sec)
$\Sigma_i T_i$	188	670	170	88	4380
C_{max}	164	706	206	64	3120
$\Sigma_i C_i$	201	612	194	101	4500
T_{max}	164	742	247	64	1800

risolutive method	C_{max}	T_{max}	ΣC_i	ΣT_i	Time
GR/C_{max}	164	64	702	204	1
GR/T_{max}	164	64	702	204	1
GR/ΣC	204	104	623	205	1
GR/ΣT	190	90	675	191	1
SB/T_{max}	190	90	675	191	1
EDD	211	111	778	283	1
FCFS	217	117	759	287	1
LPT	179	79	783	283	1
SPT	204	104	623	205	1

Fig. 8. The results of different job-shop scheduling problems obtained by solving an MILP (first table) and using the software Lekin (second table).

[3] M. Pinedo, X. Chao, *Operations scheduling with applications in manufacturing and services*, Irwin McGraw-Hill, 1998.

[4] E. Demirkol, S. Mehta, R. Uzsoy, "Benchmarks for shop scheduling problems," *European Journal of Operational Research*, Vol. 109, No. 1, pp. 137-141, August 1988.

[5] S. French, *Sequencing and scheduling: an introduction to the mathematics of the job shop*, Ellis Horwood, Chichester, UK, 1982.

[6] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, *Optimization and approximation in deterministic sequencing: a survey, Annals of Discrete Mathematics 5*, pp. 287–326, 1979.