Worst-Case Evaluation of Flexible Solutions in Disjunctive Scheduling Problems

Mohamed Ali Aloulou, Christian Artigues

Worst-case evaluation of flexible solutions in disjunctive scheduling problems

Mohamed Ali Aloulou[1] and Christian Artigues[2,3]

[1] LAMSADE – Université Paris Dauphine
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France
aloulou@lamsade.dauphine.fr,
WWW home page: http://www.lamsade.dauphine.fr/∼aloulou/
[2] LIA – Université d'Avignon
339 chemin meinajariés, Agroparc, BP 1228, 84911 Avignon Cedex 9, France
christian.artigues@univ-avignon.fr
[3] CRT – Université de Montréal
C.P. 6128, succursale Centre-ville Montréal, QC H3C 3J7, Canada
artigues@crt.umontreal.ca

Abstract. In this paper, we consider the problem of evaluating the worst case performance of flexible solutions in non-preemptive disjunctive scheduling. A flexible solution represents a set of semi-active schedules and is characterized by a partial order on each machine. A flexible solution can be used on-line to absorb the impact of some data disturbances related for example to job arrival, tool availability and machine breakdowns. Providing a flexible solution is useful in practice only if it can be assorted with an evaluation of the complete schedules that can be obtained by extension. For this purpose, we suggest to use only the best case and the worst case performance. The best case performance is an ideal performance that can be achieved only if the on-line conditions allow to implement the best schedule among the set of schedules characterized by the flexible solution. In contrast, the worst case performance indicates how poorly the flexible solution may perform. These performances can be obtained by solving corresponding minimization and maximization problems. We focus here on maximization problems when a regular min-max objective function is considered. In this case, the worse objective function value can be determined by computing the worse completion time of each operation separately. We show that this problem can be solved by finding an elementary longest path in the disjunctive graph representing the problem with additional constraints. In the special case of the flow-shop problem with release dates and additional precedence constraints, we give a polynomial algorithm that determines the worst case performance of a flexible solution.

1 Introduction

We consider a general non preemptive disjunctive problem in which a set of operations has to be scheduled on a set of machines, each operation requiring a

single machine during its execution and each machine being able to process only one operation simultaneously. The operations are linked by simple precedence constraints that do not necessarily form chains. Such a model encompasses the standard flow-shop and job-shop models.

An important issue in scheduling concerns the support provided to the end-user(s) on line schedule execution after the off line scheduling phase which consists in providing an optimal or suboptimal schedule.

In disjunctive scheduling, as soon as a regular minmax objective function is considered, the support for on line scheduling often lies in providing for each machine the mandatory sequence of operations, and for each operation an earliest and a latest start time yielding operation slacks. The sequences and the time windows are such that scheduling the operations in the predetermined order and inside their time windows is feasible and keeps the objective function in a range of acceptable values. Such a flexibility provided to the end-user is referred to as temporal flexibility. This paper addresses the problem of providing more flexibility than the classical temporal one in disjunctive scheduling problems where the objective is to minimize a regular minmax objective function. As already considered in previous studies [2,3,5,6,7,12], this can be achieved by defining only a partial order of the operations on each machine, leaving to the end-user the possibility to make the remaining sequencing decisions. This is the principle of the groups of permutable operations model that has been studied by several authors [3,5,6,7,12]. The group model sets restrictions on the proposed partial orders that we relax in this paper. Indeed we represent the partial orders through general precedence constraints between operations of the same machine, that have not to be distinguished from the structural precedence constraints.

Providing a partial solution through partial orders is useful in practice only if it can be assorted with an evaluation of the complete solutions that can be obtained by extension. More precisely, given a reasonable decision policy followed by the decision maker, the following questions have to be answered. Do there remain decisions (following the decision policy) leading to a feasible schedule ? What is the best and the worse objective function value reachable by the remaining set of decisions ? Answering these questions provides a performance guarantee if the given on line policy is followed.

In this paper, we focus on the worst case performance evaluation. We consider disjunctive problems with a minmax objective function. We assume that the decision maker follows a semi-active schedule policy to extend the proposed partial orders. A schedule is called semi-active if the operations cannot be shifted to start earlier without changing the operation sequences or violating precedence constraints or release dates [4]. In this case, the worse objective function value can be determined by computing the worst-case completion time of each operation separately. We show that the problem of computing the worse completion time of an operation in all feasible semi-active schedules can be done by finding an elementary longest path in the disjunctive graph representing the problem with additional constraints. This gives a general framework integrating previous studies [1,3,5,6,7,12].

In the special case of the flow-shop problem with release dates and additional precedence constraints, we give a polynomial algorithm that computes the maximal completion times of all operations in all feasible semi-active schedules. These results generalize some results previously established for the single machine version of this problem [1].

In section 2, we present the problem in more details. Then, we discuss the related work in section 3. In section 4, the longest path formulation of the problem is given. In section 5, we present the polynomial algorithm for the flow-shop case. Finally, we provide a conclusion in section 6.

2 Problem setting

We consider the following disjunctive scheduling problem. There is a set $N = \{1, \ldots, n\}$ of operations to be scheduled on m machines. m_i, p_i and r_i denote the machine, processing time and release date of operation i, respectively. The release date is the earliest time when the operation processing can start. Each operation is associated with a non-decreasing cost function $f_i(C_i)$ of its completion time C_i. We introduce two dummy operations 0 and $n+1$ such that $p_0 = p_{n+1} = 0$. This problem is represented by a disjunctive graph $\mathcal{G} = (V, C, D)$ [11]. V is the set of vertices corresponding to operations $i \in N$ and the two dummy operations 0 and $n+1$. C is the set of conjunctive arcs representing the precedence constraints between the operations. Each conjunctive arc (i, j) is valuated by p_i. D is a set of pairs of disjunctive arcs $\{(i, j), (j, i)\}$ for each pair of operations $i, j \in N$ requiring the same machine for their execution ($m_i = m_j$). We have $D = \{\{(i, j), (j, i)\} | i \neq j \text{ and } m_i = m_j\}$. Arc (i, j) represents the decision to sequence i before j, whereas arc (j, i) represents the decision to sequence j before i on the machine. In the remaining a pair of disjunctive arcs $\{(i, j), (j, i)\}$ is called a disjunction and denoted by e_{ij} or e_{ji}.

Let $\mathcal{D}$ denote the set of all disjunctive arcs, i.e. $\mathcal{D} = \{(i, j) | e_{ij} \in D\}$. A selection π is a (possibly empty) set of arcs such that $\pi \subseteq \mathcal{D}$ and $|\pi \cap e_{ij}| \leq 1$, for all $e_{ij} \in D$. Let $D(\pi) = \{e_{ij} \in D | e_{ij} \cap \pi = \emptyset\}$. A selection is complete if $D(\pi) = \emptyset$, otherwise it is partial. The disjunctive graph issued from a complete or partial selection π is denoted by $\mathcal{G}(\pi) = (V, C \cup \pi, D(\pi))$. Given a set of arcs E, let $G(E)$ denote graph $(V, C \cup E)$. A complete selection π is feasible if the graph $G(\pi) = (V, C \cup \pi)$ is acyclic. The completion time $C_i(\pi)$ of any operation $i \in N$ in the semi-active schedule derived from the complete feasible selection π is equal to the length of the longest path in $G(\pi)$ from 0 to i plus p_i. Let Π denote the set of feasible complete selections. The objective of the classical scheduling problem is to find a complete feasible selection $\pi \in \Pi$ such that a *regular* minmax objective function $F(C_1(\pi), \ldots, C_n(\pi)) = \max_{i=1,\ldots,n} f_i(C_i(\pi))$ is minimized.

Here, we assume the decision maker makes on line the remaining sequencing decisions on each machine following a semi-active policy until obtaining a complete selection π. A feasible semi-active schedule can be obtained by a list scheduling algorithm as soon as C is acyclic : sort the operations in a non de-

creasing order of their level in $G = (V, C)$ then sequence as soon as possible on its machine each operation according to this order. The problem tackled in this paper is the following problem (SP) : Given a disjunctive graph $\mathcal{G} = (V, C, D)$, what is the worst case objective function value of the feasible semi-active schedules, i.e compute $\max_{\pi \in \Pi} F(C_1(\pi), \ldots, C_n(\pi))$?

To illustrate this problem, consider the following 2-machine and 4-job flow-shop problem. This gives 8 operations and the flow-shop context sets $m_1 = m_3 = m_5 = m_7 = 1$ and $m_2 = m_4 = m_6 = m_8 = 2$. Furthermore, we have $p_1 = 1$, $p_2 = 6$, $p_3 = 2$, $p_4 = 5$, $p_5 = 4$, $p_6 = 6$, $p_7 = 6$ and $p_8 = 1$. All release dates are equal to 0 except for $r_5 = 2$. All objective functions are the completion times of the activities. Let us consider additional precedence constraints $\{(1,3),(1,5),(1,7),(3,7),(2,4),(2,6),(6,8)\}$. We obtain the disjunctive graph $\mathcal{G}$ displayed in Figure 1.

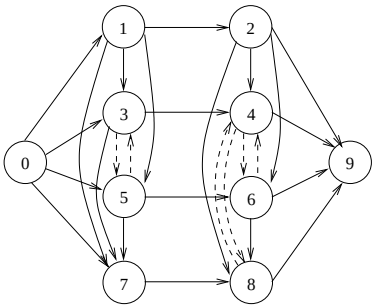

Fig. 1. The disjunctive graph for a partial selection

The precedence constraints restrict the possible sequences to $(1, 3, 5, 7)$ and $(1, 5, 3, 7)$ on machine 1 and $(2, 4, 6, 8)$, $(2, 6, 4, 8)$ and $(2, 6, 8, 4)$ on machine 2. We obtain the 6 schedules displayed in Figure 2. Note that such restriction cannot be modeled by the group of permutable operation representation used in [1,5,6,7,12]. The solution of problem SP is 20 which is the worst-case makespan value of the 6 semi-active schedules. Note that the optimal makespan of the flow-shop problem is 19.

3 Literature review

Problem SP can be viewed as a maximization problem of an objective function that is naturally minimized. Several works have aleady been proposed in this domain. In particular, Aloulou, Kovalyov and Portmann [1] adapt the traditional three-field notation $\alpha|\beta|\gamma$ to this class of problems.

They denote this family of considered maximization problems as $\alpha(sa)|\beta|(f \rightarrow \max)$. The first field α provides the shop environment. Here $\alpha \in$

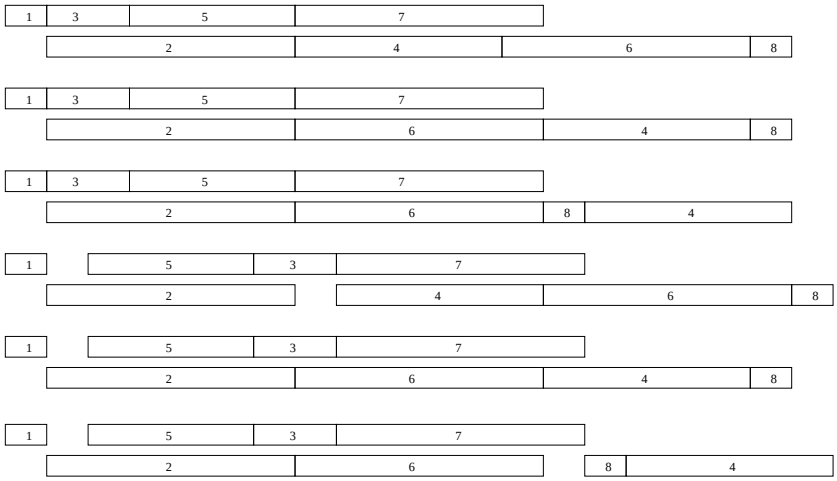

Fig. 2. Sequences and semi-active schedules compatible with the partial selection

$\{1, F, J\}$ for respectively single machine (1), flow shop (F) and job shop (J) problems. *sa* indicates that we search for the worst schedule among all semi-active schedules, which correspond to the considered online scheduling policy. The second field gives additional constraints on operations. The third field contains information about the criterion to maximize.

To the best of our knowledge, the first related results we are aware are due to Posner [10]. Posner studied reducibility among single machine weighted completion time scheduling problems including minimization as well as maximization problems. In these problems, the jobs may have release dates and deadlines but there are no precedence constraints between the jobs. Besides, inserting idle times between the jobs is allowed.

Aloulou *et al* [1] studied several maximization versions in a single machine environment. They examined problems $1(sa)|\beta|(\gamma \rightarrow \max)$, where $\beta \subseteq \{r_i, prec\}$ and $\gamma \in \{f_{\max}, C_{\max}, L_{\max}, T_{\max}, \sum(w_i)C_i, \sum(w_i)U_i, \sum(w_i)T_i\}$. They showed that these problems are at least as easy as their minimization counterparts, except for problems $1(sa)||(\sum w_i T_i \rightarrow \max)$ and $1(sa)|r_i|(\sum w_i T_i \rightarrow \max)$, which are still open. In particular, problems $1(sa)|r_i, prec|(L_{\max} \rightarrow \max)$ and $1(sa)|r_i, prec|(T_{\max} \rightarrow \max)$ can be solved in $O(n^3)$ times while the minimization counterparts are strongly NP-hard, even if $prec = \emptyset$ [9].

This work is closely related to the work of the present paper. Indeed, problem (SP) can be denoted, in the Aloulou *et al* notation, as $\alpha(sa)|r_i, prec|(f_{\max} \rightarrow \max)$. Besides, we propose in section 5 an algorithm solving the flow-shop problem $F(sa)|r_i, prec_k|(f_{\max} \rightarrow \max)$, generalizing the algorithm of Aloulou *et al* for problem $1(sa)|r_i, prec|(f_{\max} \rightarrow \max)$ [1]. Here $prec_k$ denotes precedence constraints appearing only between operations scheduled on the same machine, besides the classical flow-shop precedence constraints.

Another class of related work in the context of flexibility generation for online scheduling is linked to the concept of groups of permutable operations [5,6], also called ordered group assignment [3,12]. A group of permutable operations is a restriction of the sequential flexibility considered here in such a way that each operation is assigned to a group and, there is a complete order between the groups of operations performed on the same machine. There are no precedence constraints between the operations of the same group. Heuristics have been designed to generate groups of permutable operations for general disjunctive problems [5,6] and multiobjective methods have been designed to find a compromise between flexibility and performance in the two-machine flowshop [7]. Artigues *et al* [3] propose a polynomial algorithm to perform the exact worst-case evaluation of an ordered group assignment. This method is based on longest path computations in a so-called worst-case graph, derived from the considered ordered group assignment. This method solves problem SP for general disjunctive problems (e.g. job-shop) where the disjunctions appear only between operations of the same group (inside each group the graph of disjunctions e_{ij} is a clique) and when the precedence constraints are defined between operations of different groups.

As an illustration, the selection π proposed for the flow shop example yielding the 6 feasible schedules of Figure 2 with a worst-case makespan of 20 cannot be represented by groups of permutable operations.

4 A longest path formulation of the problem

Let $\hat{C}_i$ denote the worst case completion time of operation i, i.e. $\hat{C}_i = \max_{\pi \in \Pi} C_i(\pi)$. Computing $\hat{C}_i$, for each $i \in N$ solves problem SP since the objective function is a minmax function of non decreasing functions of the completion times. Recall that $\mathcal{D}$ is the set of all disjunctive arcs. Let us now consider the following Constrained Longest Path problem associated to operation i ($CLP(i)$).

Definition 1. *Given a disjunctive graph $\mathcal{G} = (V, C, D)$ and an operation i, problem $CLP(i)$ consists in computing the longest elementary path $L^*(0, i)$ from 0 to i in $G(\mathcal{D}) = (V, C \cup \mathcal{D})$ such that $G(L^*) = (V, C \cup L^*(0, i))$ is acyclic.*

We have the following result.

Theorem 1. *The worst case completion time $\hat{C}_i$ is equal to the length of path $L^*(0, i)$ solution of problem $CLP(i)$.*

Proof. We first show that (a) $\hat{C}_i$ is the length of an elementary path l from 0 to i in $G(\mathcal{D})$ and that $G(l)$ is acyclic. Let $\pi \in \Pi$ such that we have $\hat{C}_i = C_i(\pi)$. π is the complete selection such that $\hat{C}_i$ is the length of a longest path l from 0 to i in $G(\pi)$. Since $G(\pi)$ is acyclic, l is elementary and since $l \subseteq \pi$, $G(l)$ is also acyclic. Since $\pi \subset \mathcal{D}$, l is also an elementary path in $G(\mathcal{D}) = (V, C \cup \mathcal{D})$.

Let us show that (b) any elementary path L from 0 to i in $G(\mathcal{D})$ verifying $G(L)$ is acyclic is such that there exists a feasible complete selection $\pi \in \Pi$

verifying $C \cup L \subseteq C \cup \pi$. Suppose that L includes only conjunctive arcs. Then $L \subseteq C$ and (b) is verified. Suppose now that L includes also disjunctive arcs. Since L is elementary, we have $|L \cap e_{ij}| \leq 1$ for each disjunction e_{ij}. Hence $L \setminus L \cap C$ is a partial selection. Furthermore, since $G(L) = G(V, C \cup L)$ is acyclic $C \cup L$ defines a new acyclic precedence constraints graph and the disjunctive problem defined by $(V, C \cup L, D(L))$ is feasible. Hence $L \setminus L \cap C$ is included in a feasible compete selection.

From (a) and (b) it follows that, $\hat{C}_i$ is the length of $CLP(i)$-solution. □

Note that in the general case, problem $CLP(i)$ may be not easy to solve since it admits as a particular case the search for the longest elementary path in a graph with positive length cycles. This problem is known to be NP-hard for general graphs [8]. In Figure 3 we illustrate the problem and the necessity of the no-cycling condition in definition 1 for a job-shop with 2 machines, 3 jobs and no release dates. Operations 1,4 and 5 are assigned to the first machine and operations 2,3 and 6 are assigned to the second machine. Structural precendence constraints are $(1, 2)$, $(3, 4)$ and $(5, 6)$. $(3, 2)$ and $(1, 4)$ are additional precedence constraints. The longest elementary path from 0 to 7 is displayed in bold. Such a path is infeasible since it induces a cycle with precedence constraints $(3, 4)$.

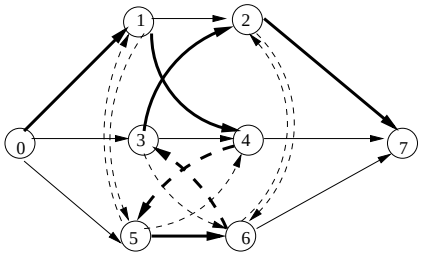

Fig. 3. A job-shop example and an infeasible elementary longest path

The following proposition shows on the opposite that the problem is simplified in a flow-shop context, since the no cycling condition is not necessary and in next Section we show that the problem is polynomially solvable.

Proposition 1. *Given a disjunctive graph $\mathcal{G} = (V, C, D)$ of a flow-shop with additional precedence constraints and an operation i, if $L(0, i)$ is an elementary path from 0 to i in $G(\mathcal{D}) = (V, C \cup \mathcal{D})$ then $G(L) = (V, C \cup L)$ is acyclic.*

Proof. Due to the flow-shop structure, the strong connected component of $G(\mathcal{D})$ include only operations assigned to the same machine. Hence any elementary cycle of $G(\mathcal{D})$ involves only operations assigned to the same machine. Let L denote an elementary path in $G(\mathcal{D})$. By definition L is acyclic and no cycle can be created by adding structural precedence constraints to L. □

5 A polynomial algorithm for the flow-shop case

In this section, we consider the flow-shop problem with operation release dates and additional precedence constraints appearing only between operations scheduled on the same machine, as in the example presented is section 2. In this case, any sequence of operations compatible with the precedence constraints of machine k yields a feasible complete selection. For each operation j, let j^- denote its job predecessor. We assume that if j is the first operation of its job, then j^- is a dummy operation denoted j^0. Let Γ_j^- (resp. Γ_j^+) denote the set of operations that must be scheduled before (resp. after) j on machine m_j. Let I_j denote the set of operations of machine m_j that are not linked to j with any precedence constraint. We have

$$\Gamma_j^- = \{i \neq j | m_i = m_j \text{ and there is a path from } i \text{ to } j \text{ in } G = (V, C)\} \quad (1)$$

$$\Gamma_j^+ = \{i \neq j | m_i = m_j \text{ and there is a path from } j \text{ to } i \text{ in } G = (V, C)\} \quad (2)$$

$$I_j = \{i \neq j | m_i = m_j, i \notin \Gamma_j^- \text{ and } j \notin \Gamma_i^-\} \quad (3)$$

Let us define $\hat{C}_{j^0} = r_j$. We have the following result.

Lemma 1. *The worst case completion time of any operation j is given by*

$$\hat{C}_j = p_j + \max \begin{cases} r_j, & (a) \\ \hat{C}_{j^-}, & (b) \\ \max\limits_{i \in I_j \cup \Gamma_j^-} \{\max(r_i, \hat{C}_{i^-}) + \sum\limits_{x \in I_j \cup \Gamma_j^- \setminus \Gamma_i^-} p_x\} & (c) \end{cases}$$

Proof. A recursion argument is used on machine k.

Consider machine 1, we have $\hat{C}_{j^-} = \hat{C}_{j^0} = r_j$, hence terms (a) and (b) are redundant. Denote by S the schedule in which $C_j(S) = \hat{C}_j$ and the block B of operations consecutive on machine 1, ending with j, is such that there is no idle time between any two consecutive operations in B and B is of maximal size. B always exists since we have at least $j \in B$. If $B = \{j\}$, then the starting time of j, S_j, is such that $S_j = r_j$ and (a) is verified.

If $|B| > 1$, then we have $S_j \geq r_j$. Let i be the first operation of block B. i is not a machine successor of j and $i \in I_j \cup \Gamma_j^-$. Similarly, by definition all operations inside B, except j itself, belong to $I_j \cup \Gamma_j^- \setminus \Gamma_i^-$ (they cannot be machine predecessors of i). Let us now consider the operations scheduled before i on machine 1. Let x denote the operation scheduled at the largest position before i such that $x \notin \Gamma_i^-$. This operation could be inserted right after i increasing the completion time of j which contradicts the maximality of $C_j(S)$. Hence all operations scheduled before i are in Γ_i^-. This implies that all operations of

$I_i \cup \Gamma_j^- \setminus \Gamma_i^-$ are scheduled after i. Conversely, suppose that x is the operation scheduled at the smallest position after j such that x is not a successor of j, i.e. $x \notin \Gamma_j^+$. This operation could be inserted right before j, increasing the start time of j which contradicts the maximality of $C_j(S)$. Hence all operations scheduled after j are successors of j. It follows that if $S_j > r_i$ then

$$\hat{C}_j \leq \max_{i \in I_j \cup \Gamma_j^-} \{r_i + \sum_{x \in I_j \cup \Gamma_j^- \setminus \Gamma_i^-} p_x\}. \tag{4}$$

Can we have $i \in I_j \cup \Gamma_j^-$ such that $\hat{C}_j < r_i + \sum_{x \in I_j \cup \Gamma_j^- \setminus \Gamma_i^-} p_x$?

Suppose that i is such an operation. It is possible to build a feasible schedule in which all the operations before i are machine predecessors of i and the operations after j are only machine successors of j. This can be made by scheduling the operations of Γ_i^- in an order compatible with the precedence constraints within this set, then the operations of $I_i \cup \Gamma_j^+$ in an order compatible with the precedence constraints within this set, then operation j, then the operations of Γ_j^+ in an order compatible with the precedence constraints within this set. The operations on machine 2 can be scheduled in any order compatible with the precedence constraints of machine 2, and so on. In such a schedule S, we have $C_j \geq r_i + \sum_{x \in I_j \cup \Gamma_j^- \setminus \Gamma_i^-} p_x$. Hence (5.3) is verified to equality and the recusion holds for machine 1.

Suppose now the recusion holds for machine $k-1$. We can prove the recursion is then verified on machine k with simular arguments.

We first consider the schedule S in which $C_j(S) = \hat{C}_j$ and the block B of operations consecutive on machine 1, ending with j, is such that there is no idle time between any two consecutive operations in B and B is of maximal size. If $|B| = 1$ then we have either $C_j(S) = r_j + p_j$ or C_j is set by C_{j^-}. To maximize this value we have $C_j = \hat{C}_{j^-} + p_j$.

If $|B| > 1$ we can also state that an operation $i \in I_j \cup \Gamma_j^-$ starts the block with $S_i = r_i$ or $S_i = C_{i^-}$. With similar arguments as for machine 1, we prove that all operations of the set $I_j \cup \Gamma_j^- \setminus \Gamma_i^-$ are in the block. Furthermore if $S_i > r_i$ then we have $S_i = C_{i^-} = \hat{C}_{i^-}$ to have $\hat{C}_j$ maximal. Last we can show that for any operation $i \in I_j \cup \Gamma_j^-$, we can build a feasible schedule S in which all the operations before i are machine predecessors of i and the operations after j are machine successors of j and $S_i = \max(r_i, \hat{C}_{i^-})$. This achieves the recursion. □

Let $\nu = n/m$ be the number of jobs. Due to lemma 1, we have the following result.

Theorem 2. *Problem* $F(sa)|r_i, prec_k|(f_{\max} \to \max)$ *can be solved in* $O(m\nu^2)$ *times if each function* f_i *is computable in* $O(1)$ *time.*

Proof. Once sets Γ_j^- and I_j are built for each operation j, all worst-case completion times can be computed trivially via the proposed recursion by dynamic programming in $O(m\nu^2)$ where $\nu = n/m$. □

In the illustrative example of section 2, the worst case completion times are given (in the order of their computation) by $\hat{C}_1 = r_1 + p_1 = 1$ (a), $\hat{C}_3 = r_5 + p_5 + p_3 = 8$ (c), $\hat{C}_5 = r_1 + p_1 + p_3 + p_5 = 7$ (c), $\hat{C}_7 = r_5 + p_5 + p_3 + p_7 = 14$ (c), $\hat{C}_2 = \hat{C}_1 + p_2 = 7$ (b), $\hat{C}_4 = \hat{C}_7 + p_8 + p_4 = 20$ (c), $\hat{C}_6 = \hat{C}_3 + p_4 + p_6 = 19$ (c), $\hat{C}_8 = \hat{C}_3 + p_4 + p_6 + p_8 = 20$ (c).

6 Conclusion

In this paper, we proposed a longest path formulation of the problem of evaluating the worst case performance of flexible solutions in disjunctive scheduling with minmax regular objective function. A flexible solution is defined by an operation partial order on each machine. We proved that this problem is polynomial in the special case of the flow-shop problem with release dates and additional precedence constraints between operations scheduled on the same machine.

We should now focus on extension of this approach to more general problems. Unfortunately, extending this approach to the job shop is not trivial. A way to solve it is to study the complexity of the problem of finding the constrained longest elementary path in the disjunctive graph of the job-shop problem.

References

1. M. Aloulou, M. Kovalyov, and M.C. Portmann. Maximization in single machine scheduling. *Annals of Operations Research*, 129:21–32, 2004.
2. M.A. Aloulou and M.-C. Portmann. An efficient proactive-reactive scheduling approach to hedge against shop floor disturbance. In G. Kendall, E.K. Burke, S. Petrovic, and M. Gendreau, editors, *Multidisciplinary Scheduling: Theory and Applications 1st International Conference, MISTA '03 Nottingham, UK, 13-15 August 2003. Selected Papers*. Elsevier, 2005. to appear.
3. C. Artigues, J.C. Billaut, and C. Esswein. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2):314–328, 2005.
4. K. R. Baker. *Introduction to sequencing and scheduling*. Wiley, 1974.
5. J.C. Billaut and F. Roubellat. A new method for workshop real time scheduling. *International Journal of Production Research*, 34(6):1555–1579, 1996.
6. J. Erschler and F. Roubellat. An approach for real time scheduling for activities with time and resource constraints. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*. Elsevier, 1989.
7. C. Esswein, J.C. Billaut, and V. Strusevich. Two-machine shop scheduling: Compromise between flexibility and makespan value. *European Journal of Operational Research*, 167(3):796–809, 2005.
8. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.
9. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, 1977.
10. M. E. Posner. Reducibility among wighted completion time scheduling problems. *Annals of Operations Research*, pages 91–101, 1990.

11. B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraintes disjonctives, 1964. D.S. vol. 9, SEMA, Paris, France.
12. S.D. Wu, E.S. Byeon, and R.H. Storer. A graph-theoretic decomposition of the job-shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124, 1999.