RAIRO Operations Research
RAIRO Oper. Res. **41** (2007) 1–18
DOI: 10.1051/ro:20070012

EVALUATING FLEXIBLE SOLUTIONS IN SINGLE MACHINE SCHEDULING VIA OBJECTIVE FUNCTION MAXIMIZATION: THE STUDY OF COMPUTATIONAL COMPLEXITY

Mohamed Ali Aloulou[1], Mikhail Y. Kovalyov[2] and Marie-Claude Portmann[3]

Abstract. We study a deterministic problem of evaluating the worst case performance of flexible solutions in the single machine scheduling. A flexible solution is a set of schedules following a given structure determined by a partial order of jobs and a type of the schedules. In this paper, the schedules of active and non-delay type are considered. A flexible solution can be used on-line to absorb the impact of data disturbances related to, for example, job arrival, tool availability or machine breakdowns. The performance of a flexible solution includes the best case and the worst case performances. The best case performance is an ideal performance that can be achieved only if the on-line conditions allow to implement the best schedule of the set of schedules characterizing the flexible solution. In contrast, the worst case performance indicates how poorly the flexible solution may perform when following the given structure in the on-line circumstances. The best-case and the worst-case performances are usually evaluated by the minimum and maximum values of the considered objective function, respectively. We present algorithmic and computational complexity results for some maximization scheduling problems. In these problems, the jobs to be scheduled have different release dates and precedence constraints may be given on the set of jobs.

Keywords. Scheduling, single machine, schedule flexibility, maximization problems, active and non-delay schedules.

Received January 1st, 2006. Accepted June 15, 2006.

[1] LAMSADE – Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France; aloulou@lamsade.dauphine.fr

[2] Faculty of Economics, Belarusian State University, and United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus; koval@newman.bas-net.by

[3] MACSI team of INRIA-Lorraine and LORIA-INPL, École des Mines de Nancy, Parc de Saurupt, 54042 Nancy Cedex, France; portmann@loria.fr

1. Motivation

In real-life scheduling applications, it is often the case that processing environment can change while implementing a schedule because of unexpected machine breakdowns, processing time uncertainty, changes in the job arrivals, etc. In this situation, a predictive (off-line) schedule that does not take into account the presence of perturbations rapidly becomes impracticable and finally yields quite poor performances, see for example Artigues, Roubellat and Billaut [5], Artigues, Billaut and Esswein [4], Daniels and Kouvelis [9] and Mehta and Uzsoy [14]. Herroleen and Leus [12] and Wu, Byeon and Storer [18] noticed that introducing flexibility in the solutions computed off-line allows to increase robustness of a scheduling system.

On the other hand, a predictive schedule should not be too much flexible because it is used as a basis for planning external activities such as raw material procurement, preventive maintenance and delivery of orders to the customers. Consequently, if an on-line schedule, which is one of the realizations of a flexible predictive schedule, deviates considerably from the initial schedule, then it may delay an execution of the related external activities and induce some extra costs for early procurement of raw materials or late delivery of finished products, see Aloulou and Portmann [3]. Therefore, a realistic off-line schedule should be sufficiently flexible, follow a structure determined by the technological constraints, and coordinate with other participants of the considered manufacturing system and/or supply chain.

The results of this paper are related to the proactive-reactive scheduling approach developed by Aloulou and Portmann [1,3]. Consider a scheduling problem to minimize an objective function associated with completion times of the jobs to be processed in a manufacturing system. In this paper, we concentrate on a single machine system. Aloulou and Portmann defined a flexible off-line solution as a partial order (precedence relations) on the set of jobs. They suggested to select a flexible solution that provides a sufficiently detailed sketch of the schedule to serve as a basis for planning the related external activities, and that remains enough flexible to allow changes in job processing when the on-line conditions force to make them.

The approach of Aloulou and Portmann has two phases: an off-line phase (proactive algorithm) and an on-line phase (reactive algorithm), see Figure 1. In the off-line phase, the set of all partial orders (flexible solutions) is explored by a genetic algorithm that proposes a selection of partial orders to a decision maker. The decision maker chooses an appropriate partial order for the on-line execution.

A partial order implicitly represents several schedules with the same precedence constraints between the jobs. The genetic algorithm and the decision maker use two main criteria to evaluate a particular partial order. First criterion is flexibility, which is characterized by the number of distinct schedules associated with the partial order. Second criterion is performance, which is characterized by the objective function values of schedules associated with the partial order. Since the number of schedules associated with a given partial order can be sufficiently large,

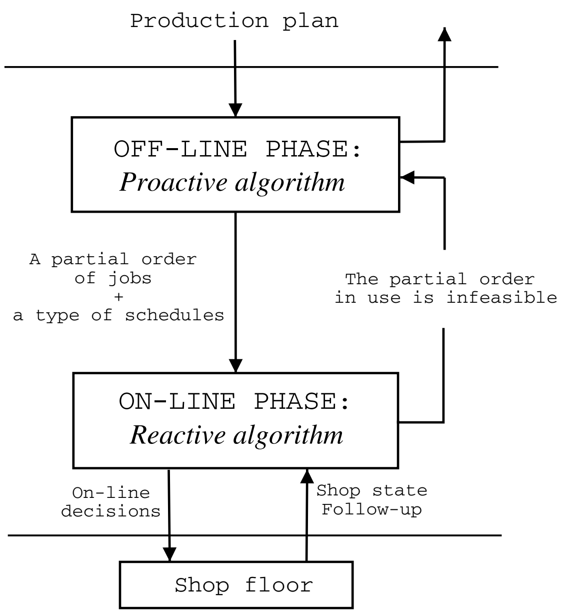
Production plan
OFF-LINE PHASE: Proactive algorithm
A partial order of jobs + a type of schedules
The partial order in use is infeasible
ON-LINE PHASE: Reactive algorithm
On-line decisions
Shop state Follow-up
Shop floor

FIGURE 1. The proposed proactive-reactive approach.

it is natural to consider only their most important representatives. Aloulou and Portmann suggested to restrict the set of the considered schedules to those often used in practice: active, semi-active and non-delay schedules, see Baker [6] and the following section for definitions. Furthermore, they suggested to use only the best and the worst case performances to evaluate the performance of a flexible schedule. The best and the worst case performances are lower and upper bounds, respectively, on the value of the objective function calculated for schedules following the partial order. They can be obtained by solving corresponding minimization and maximization problems.

During the on-line phase, whenever a job has to be selected for processing on a machine, a decision maker is provided with several alternatives that respect the chosen partial order. He/she can make a choice that satisfies other preferences or non-modeled constraints. In the presence of disruptions, it is often the case that one of the schedules associated with the used flexible solution remains feasible and there is no need to calculate a new solution.

As we already mentioned, one of the reasonable approaches to evaluate the performance of a flexible solution (partial order) is to solve two deterministic optimization problems: a minimization problem for the best case performance and a maximization problem for the worst case performance. Notice that there is no need to calculate corresponding schedules.

Minimization problems have been abundantly considered in the scheduling literature. In this paper, we focus on new maximization scheduling problems never considered in the past. Observe that if the jobs are allowed to start arbitrarily late, then the maximum values of many traditional objective functions can be arbitrarily large. In this case, they cannot be used to evaluate the worst case performance of any solution. Therefore, the set of feasible schedules must be restricted so that the jobs do not start too late. We propose in this paper to restrict the schedules to be active or non-delay. A flexible solution is now characterized by a structure defined by a partial job order and a type of the schedules.

In the following section, we formulate the problems to be studied, review related publications and briefly describe the results of this paper.

2. Problem formulation and literature review

We study the following single machine scheduling problem. There is a set $N = \{1, \dots, n\}$ of jobs to be scheduled for processing on a single machine. Each job $j \in N$ has a processing time $p_j > 0$, a release date $r_j \geq 0$, and it may have a due date $d_j \geq 0$, a weight $w_j > 0$, and a non-decreasing function $f_j(t)$ associated with it. The release date is the earliest time when the job processing can start, the due date is an ideal time for its completion, the weight indicates a relative importance of the job and the function $f_j(t)$ represents the cost that has to be paid when the job completes at time t. All data are assumed to be integers. Functions $f_j(t)$ are assumed to be integer valued.

Precedence constraints may be given on the set of jobs. If job i precedes job j, then job j can start its processing only after job i is completed. A schedule is specified by the sequence of the jobs and their starting times. No preemption of job processing is allowed.

A schedule is called *active* if a job cannot be shifted to start earlier without increasing completion time of another job or violating precedence constraints or release dates. A schedule is called *non-delay* if the machine does not stand idle at any time when there is a job available for processing at this time. If it is given that only active schedules are considered, then any such schedule is completely characterized by the corresponding job sequence. The same statement is valid for non-delay schedules, see Baker [6].

Given a schedule, the job completion times C_j, $j = 1, \dots, n$, can be easily determined. The objective is to find an active or non-delay schedule such that a *regular* function $F(C_1, \dots, C_n)$ non-decreasing in the job completion times is maximized.

Adapting the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems proposed by Graham *et al.* [11], we denote the above problem as $1(\theta)|\beta|(\gamma \rightarrow \max)$, where $\theta \in \{a, nd\}$ indicates the type of schedules to be considered (active and non-delay, respectively), $\beta \subseteq \{prec, r_j\}$ and $\gamma \in \{F, f_{\max}, f_\Sigma, C_{\max}, L_{\max}, T_{\max}, \sum(w_j)C_j, \sum(w_j)U_j, \sum(w_j)T_j\}$. If the descriptor r_j is present in the second field, then the job release dates are assumed to be arbitrary. Otherwise, they are all equal to zero. If the descriptor *prec* is present in the second field, then the precedence constraints are assumed to be represented by an arbitrary acyclic graph.

If *prec* is not present, then the jobs are assumed to be independent and the corresponding graph contains no arcs. The third field contains an information about the criterion, which is to maximize one of the following regular functions:

- arbitrary regular function $F = F(C_1, \dots, C_n)$;
- maximum cost function $f_{\max} = \max\{f_j(C_j)\}$;
- total cost function $f_\Sigma = \sum f_j(C_j)$;
- maximum completion time (makespan) $C_{\max} = \max\{C_j\}$;
- maximum lateness $L_{\max} = \max\{L_j\}$, $L_j = C_j - d_j$;
- maximum tardiness $T_{\max} = \max\{T_j\}$, $T_j = \max\{0, C_j - d_j\}$;
- total (weighted) completion time $\sum(w_j)C_j$;
- (weighted) number of late jobs $\sum(w_j)U_j$; where $U_j = 0$, if $C_j \leq d_j$ and $U_j = 1$ if $C_j > d_j$;
- total (weighted) tardiness $\sum(w_j)T_j$.

Observe that our objective is to find a worst active or non-delay schedule with respect to the traditional scheduling criteria. To the best of our knowledge, the only relevant results are due to Posner [15] and our earlier paper [2]. 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, an insertion of idle times between the jobs is allowed.

In [2], we studied maximization problems where semi-active schedules were considered. For a semi-active schedule, a job cannot be shifted to start earlier without changing the job sequence or violating the feasibility. This specificity makes the results of [2] applicable for the problems formulated above only if the release dates are all equal to zero. For instance, the problem of maximizing $f_{\max}$ is solvable in $O(n^2)$ time, and the problems of maximizing $\sum w_jC_j$ and $\sum w_jU_j$ are equivalent to their minimization counterparts with the inverse precedence constraints. Therefore, in the sequel, we assume that the release dates are not equal.

Restricting the schedules to be semi-active means that a decision maker can choose any job to execute on the machine once it follows the given structure. The machine can stand idle and wait until the chosen job is ready for execution. In the case of active schedules, an algorithm presents to a decision maker a set of different actions that keeps the schedule active. Consideration of active schedules instead of semi-active schedules allows to decrease considerably objective function values. However, flexibility, in terms of the number of the relevant schedules, decreases in this case. It is also interesting to consider non-delay schedules. Indeed, a non-delay schedule is easy to implement and the scheduling strategy is more understandable by a decision maker who is generally used to keep the machine busy if there is an available job.

The remainder of this paper is organized as follows. In Section 3, we consider the case of active schedules. We prove that the problem $1(a)|r_j|(C_{\max} \to \max)$ is NP-hard even if there are only three distinct release dates and that the problem $1(a)|r_j|(\sum w_jC_j \to \max)$ is NP-hard in the strong sense. It is NP-hard if there are two distinct release dates. In Section 4, we consider the case of non-delay schedules.

We show that the problem $1(nd)|prec, r_j|(C_{\max} \to \max)$ is solvable in $O(n^2)$ time unlike the same problem for active schedules. We further demonstrate that the problem $1(nd)|prec, r_j|(f_{\max} \to \max)$ can be solved in $O(n^4)$ time. Finally, we prove that the problem $1(nd)|prec, r_j|(F \to \max)$ for $F \in \{\sum w_jC_j, \sum w_jU_j\}$ reduces to the corresponding minimization problems without job release dates but with job deadlines. The paper concludes with a comparative exposition of the complexity of the considered maximization problems, maximization problems where only semi-active schedules are considered, and minimization counterparts of all these maximization problems.

Our results mainly concern computational complexity of the considered problems. It is a necessary first step in an investigation of any real-life problem to be solved on a computer. There is a comprehensive bibliography on computational complexity of minimization scheduling problems, see the up-to-date information maintained by Brucker and Knust [8]. However, there are only the results that we already mentioned for the maximization scheduling problems.

3. Active schedules

In this section, a search for an optimal schedule is restricted to active schedules. We prove that maximizing $C_{\max}$ is NP-hard and maximizing $\sum w_jC_j$ is strongly NP-hard even if there are no precedence constraints. Note that these problems are polynomially solvable for semi-active schedules, see Aloulou, Kovalyov and Portmann [2]. In our NP-hardness proofs, we construct reductions from the following problems.

- Partition: Given $m+1$ positive integers $a_1, \dots, a_m$ and A such that $\sum_{j=1}^{m} a_j = 2A$, is there a set $X \subset M = \{1, \dots, m\}$ such that $\sum_{j \in X} a_j = A$?
- Equal Cardinality Partition: Given $2m+1$ positive integers $a_1, \dots, a_{2m}$ and A such that $\sum_{j=1}^{2m} a_j = 2A$, is there a set $X \subset M = \{1, \dots, 2m\}$ such that $|X| = m$ and $\sum_{j \in X} a_j = A$?
- 3-Partition: Given $3m+1$ positive integers $a_1, \dots, a_{3m}$ and A such that $A/4 < a_j < A/2$, $j = 1, \dots, 3m$, and $\sum_{j=1}^{3m} a_j = mA$, is there a partition of the set $\{1, \dots, 3m\}$ into m subsets $X_1, \dots, X_m$, for which $\sum_{j \in X_l} a_j = A$, $l = 1, \dots, m$?

Problems Partition and Equal Cardinality Partition are NP-complete and problem 3-Partition is NP-complete in the strong sense (Garey and Johnson [10]).

Theorem 1. *The problem* $1(a)|r_j|(C_{\max} \to \max)$ *is NP-hard even if there are only three distinct release dates.*

Proof. We use a polynomial transformation from Equal Cardinality Partition. Given an instance of this problem, construct the following instance of the problem $1(a)|r_j|(C_{\max} \to \max)$.

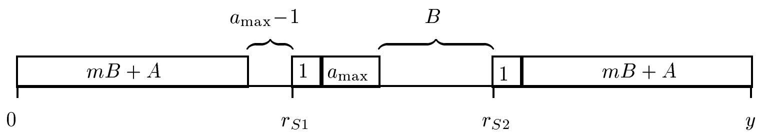

FIGURE 2. Schedule with $C_{\max} = y$.

Calculate $a_{\max} = \max_{j\in M} a_j$ and $B = A + a_{\max}$.
There are $2m + 3$ jobs. Among them there are

- $2m$ *partition* jobs j with zero release dates and processing times $p_j = B + a_j$, $j = 1, \ldots, 2m$;
- one $a_{\max}$-job with zero release date and processing time $a_{\max}$; and
- two jobs $S1$ and $S2$ with unit processing times and release dates $r_{S1} = (m+1)B - 1 = mB + A + a_{\max} - 1$ and $r_{S2} = r_{S1} + 1 + a_{\max} + B$.

We show that EQUAL CARDINALITY PARTITION has a solution if and only if there exists a solution to the constructed instance of the problem $1(a)|r_j|(C_{\max}\to\max)$ such that $C_{\max} \geq y := r_{S2} + 1 + mB + A$.

"Only if". Assume that set X is a solution to EQUAL CARDINALITY PARTITION. Construct a schedule in which jobs $S1$ and $S2$ start at their release dates, $a_{\max}$-job starts just after $S1$, partition jobs of the set X are scheduled before job $S1$ and partition jobs of the set $M\backslash X$ are scheduled after job $S2$. A diagram of such a schedule is given in Figure 2.

Observe that the constructed schedule is active because the length of the first idle interval is $a_{\max} - 1$ which is less than the minimum processing time of all available jobs scheduled after it, and the length of the second idle interval is B which is less than the minimum processing time of all available jobs scheduled after it. For the constructed schedule, we have $C_{\max} = y$.

"If". Let there exist an active schedule with value $C_{\max} \geq y$. For such a schedule, denote the set of partition jobs completed before r_{S1} as X and denote $A_X = \sum_{j\in X} a_j$.

We first assume that at most $m-2$ partition jobs are completed before r_{S1}. In this case, the total processing time of these jobs is equal to $G \leq (m-2)B + A_X$ and the length of an idle interval before r_{S1} is equal to

$$r_{S1} - G \geq 3B - A_X - 1 = 2A - A_X - 1 + A + 3a_{\max} \geq B + 2a_{\max},$$

which is not less than the total processing time of any partition job and the $a_{\max}$-job. We obtained a contradiction because the schedule is active.

We now assume that exactly $m-1$ partition jobs are completed before r_{S1}, *i.e.*, $|X| = m-1$. Their total processing time is equal to $G' \leq (m-1)B + A_X$ and the length of the idle interval before r_{S1} is equal to

$$r_{S1} - G' \geq B + A - A_X - 1 + a_{\max} = 2A - A_X - 1 + 2a_{\max} > a_{\max}.$$

Since the schedule is active and mth partition job cannot appear before r_{S1} by assumption, the latter inequality implies that the $a_{\max}$-job must be processed before r_{S1}. Then the length of the idle interval before r_{S1} is at least $B+A-A_X-1$. Denote $a_{\min}^{M\backslash X} = \min_{j\in M\backslash X} a_j$. The statement that no partition job can fit into the above idle interval implies that $B + a_{\min}^{M\backslash X} > B + A - A_X - 1$, from where we obtain

$$a_{\min}^{M\backslash X} \geq A - A_X. \tag{1}$$

Since the $a_{\max}$-job is processed before r_{S1} and $r_{S2} - r_{S1} = B + 1 + a_{\max}$, job $S1$ and some partition job from the set $M\backslash X$ must be processed before r_{S2}. Then the number of the partition jobs scheduled after r_{S2} is equal to m and their total processing time is equal to

$$D \leq mB + 2A - \left(A_X + a_{\min}^{M\backslash X}\right).$$

From $C_{\max} \geq y$, we obtain $D \geq mB + A$. Therefore, $a_{\min}^{M\backslash X} \leq A - A_X$. This inequality and (1) imply $A_X + a_{\min}^{M\backslash X} = A$, *i.e.*, $X \cup \{j^0\}$ where j^0 is the job from $M\backslash X$ with processing time $B + a_{\min}^{M\backslash X}$, is a solution of Equal Cardinality Partition.

Observe that no more than m partition jobs can be completed by r_{S1} because $r_{S1} = (m+1)B - 1$ is less than the total processing time of any $m+1$ partition jobs. Furthermore, at least m partition jobs can be scheduled after r_{S2} because otherwise

$$C_{\max} \leq r_{S2} + 1 + a_{\max} + (m-1)B + 2A - 1 < r_{S2} + 1 + mB + A = y.$$

Therefore, it remains to consider the case where exactly m partition jobs are scheduled before r_{S1} and exactly m such jobs are scheduled after r_{S2}. In this case, job $S1$ and the $a_{\max}$-job must be scheduled between r_{S1} and r_{S2}. Otherwise, a partition job will be scheduled there. Then the length of the idle interval before r_{S1} should not exceed $a_{\max} - 1$. This statement is equivalent to $A_X \geq A$. On the other hand, $C_{\max} \geq y$ is equivalent to $A_X \leq A$.

Hence, in the remaining case we obtain that $|X| = m$ and $A_X = A$, *i.e.*, X is a solution of Equal Cardinality Partition. $\square$

Corollary 1. *The problem* $1(a)|r_j|(\gamma \to \max)$ *is NP-hard for* $\gamma \in \{f_{\max}, f_{\Sigma}, L_{\max}, T_{\max}, \sum(w_j)U_j, \sum(w_j)T_j\}$.

Computational complexity of the problem $1(a)|r_j|(C_{\max} \to \max)$ remains open with respect to the strong NP-hardness and for the case where there are only two distinct release dates.

Theorem 2. *The problem* $1(a)|r_j|(\sum w_j C_j \to \max)$ *is NP-hard in the strong sense.*

Proof. A pseudo-polynomial transformation from the strongly NP-complete problem 3-Partition is used.

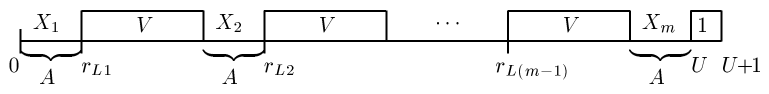

FIGURE 3. Schedule with $\sum w_jC_j \geq y$.

Given an instance of 3-PARTITION, we construct the following instance of the problem $1(a)|r_j|(\sum w_jC_j \to \max)$. Calculate $a_{\max} = \max_{1\leq j\leq 3m}\{a_j\}$, $V = 3ma_{\max}\ (mA + m - 1) + 1$ and $U = (m - 1)V + mA$. There are $4m$ jobs. Among them there are

- $3m$ *partition* jobs j with parameters $r_j = 0$, $p_j = w_j = a_j$, $j = 1, \dots, 3m$;
- $m-1$ *long* jobs Ll with parameters $r_{Ll} = (l-1)V + lA$, $p_{Ll} = V$, $w_{Ll} = 1$, $l = 1, \dots, m-1$;
- one *heavy* job H with parameters $r_H = 0$, $p_H = 1$, $w_H = W$, $W := (r_{L(m-1)} + U)(m - 1 + mA)$.

We show that 3-PARTITION has a solution if and only if there exists a solution to the constructed instance of the problem $1(a)|r_j|(\sum w_jC_j \to \max)$ such that

$$\sum w_jC_j \geq y := V(A+1)m(m-1)/2 + (U+1)W.$$

"Only if". Assume that $X_1, \dots, X_m$ is a solution to 3-PARTITION. Construct a schedule in which long jobs start at their release dates r_{Ll}, $l = 1, \dots, m-1$, heavy job H starts at time U and partition jobs are scheduled before job H in the remaining time intervals. Jobs of the set X_1 are sequenced before long job $L1$, jobs of the set X_2 are sequenced between jobs $L1$ and $L2$, and so on. A diagram of such a schedule is given in Figure 3.

Observe that, in the constructed schedule, there is no idle time between the jobs. For such a schedule, it follows from the definition of the objective function $\sum w_jC_j$ that *contribution of any job to this objective function is equal to its processing time multiplied by the total weight of jobs scheduled after it plus its own weight.* Below we use this statement for calculating contributions of the jobs to the objective function.

For the constructed schedule, we have $\sum w_jC_j = F_1 + (U+1)W$, where F_1 denotes $\sum w_jC_j$ value for the schedule without heavy job H and $(U+1)W$ is the contribution of job H to the original objective function. Furthermore, $F_1 = F_L + F_P$, where F_L and F_P are contributions of long and partition jobs, respectively, to the F_1 value.

Denote $A_l = \sum_{j\in X_l} a_j$, $l = 1, \dots, m$. We have

$$\begin{aligned} F_L &= p_{L1}(m - 1 + A_2 + \cdots + A_m) + p_{L2}(m - 2 + A_3 + \cdots + A_m) \\ &\quad + \cdots + p_{L(m-1)}(1 + A_m) = V(A+1)m(m-1)/2 \end{aligned} \tag{2}$$

and $0 \leq F_P \leq 3ma_{\max}(mA + m - 1) = V - 1$. Therefore, for the constructed schedule, we have

$$\sum w_j C_j \geq V(A+1)m(m-1)/2 + (U+1)W = y.$$

"If". Assume that there exists an active schedule with value $\sum w_j C_j \geq y$. Observe that the heavy job H cannot be completed after time $U+1$ because then there will be an idle time before it and the schedule will not be active (H can be removed to start earlier without increasing completion times of the other jobs).

Assume that job H completes at time $C_H \leq U$. In this case,

$$\sum w_j C_j < UW + (r_{L(m-1)} + \textit{total processing time of long and partition jobs}) \\ \times (\textit{total weight of long and partition jobs}) = W(U+1) < y.$$

Therefore, $C_H = U+1$ and there is no idle time before job H.

Since long jobs have equal processing times and weights, assume without loss of generality that they are scheduled in the order $L1, L2, \ldots, L(m-1)$. Denote by $X_1, X_2, \ldots, X_m$ the sets of partition jobs scheduled before job $L1$, between jobs $L1$ and $L2$, and so on, respectively. Structure of such a schedule is presented in Figure 3. At this point, however, it is not required that long jobs start exactly at their release dates.

Similar to part "only if" and keeping the same notations, we have $\sum w_j C_j = F_1 + (U+1)W$, $F_1 = F_L + F_P$ and $F_P \leq V - 1$. From $\sum w_j C_j \geq y$, we obtain

$$F_L + F_P \geq V(A+1)m(m-1)/2. \tag{3}$$

Recall that $A_l = \sum_{j \in X_l} a_j$. Since release dates of long jobs are observed, we know that

$$A_m \leq A,\ A_{m-1} + A_m \leq 2A,\ \ldots,\ A_2 + A_3 + \cdots + A_m \leq (m-1)A.$$

Notice that value F_L is maximized if long jobs are scheduled as early as possible, *i.e.*, when they start at their release dates. In this case, we have equation (2) satisfied. Assume that $A_m \leq A - 1$. In this case, because of the no idle time assumption, long job $L(m-1)$ cannot start earlier than at time $r_{L(m-1)} + 1$, and therefore, $F_L \leq V(A+1)m(m-1)/2 - V$. We obtain

$$F_L + F_P \leq V(A+1)m(m-1)/2 - V + V - 1 < V(A+1)m(m-1)/2,$$

which contradicts (3). Therefore, $A_m = A$.

By continuing in a similar way, we can show that $A_{m-1} = A$, $A_{m-2} = A, \ldots$, $A_2 = A$. Then, since $\sum_{l=1}^{m} A_l = mA$, we get $A_1 = A$. Thus, there exists a solution to 3-PARTITION, as required. □

With the use of the problem PARTITION, the above proof can easily be adapted for the following theorem.

Theorem 3. *The problem* $1(a)|r_j|(\sum w_jC_j \to \max)$ *is NP-hard if there are two distinct release dates.*

The proof of Theorem 3 becomes evident if we consider m partition jobs, one long and one heavy job, and will look for two sets $X_1 \subseteq M$ and $X_2 = M \backslash X_1$ of partition jobs such that X_1 is a solution of PARTITION.

Corollary 2. *The problem* $1(a)|r_j|(\sum w_jT_j \to \max)$ *is strongly NP-hard. It is NP-hard if there are two distinct release dates.*

Computational complexities of the problems $1(a)|r_j|(\sum C_j \to \max)$ and $1(a)|r_j|(\sum U_j \to \max)$ remain unknown.

4. Non-delay schedules

In this section, a search for an optimal schedule is restricted to non-delay schedules. We first state that maximizing $\sum w_jC_j$ is strongly NP-hard as it is in the case of active schedules. Then we suggest a procedure that constructs a non-delay schedule. An analysis of this procedure shows that all feasible non-delay schedules have the same $C_{\max}$ value, which is optimal for the minimization problem $1|prec, r_j|C_{\max}$. Therefore, the problem $1(nd)|prec, r_j|(C_{\max} \to \max)$ is polynomially solvable unlike the same problem for active schedules. We further prove that the problem $1(nd)|prec, r_j|(f_{\max} \to \max)$ can be solved in $O(n^4)$ time. We also demonstrate that the problems $1(nd)|prec, r_j|(F \to \max)$ for $F \in \{\sum w_jC_j, \sum w_jU_j\}$ reduce to the corresponding minimization problems without job release dates but with job deadlines.

Theorem 4. *The problem* $1(nd)|r_j|(\sum w_jC_j \to \max)$ *is NP-hard in the strong sense. It is NP-hard in the ordinary sense if there are two distinct release dates.*

Proof. The proof of Theorem 2 for the case of active schedules can be used because the schedules considered in this proof are non-delay ones. $\square$

Corollary 3. *The problem* $1(nd)|r_j|(\sum w_jT_j \to \max)$ *is strongly NP-hard. It is NP-hard if there are two distinct release dates.*

It is easy to see that any feasible non-delay schedule can be represented as a collection of disjoint subsequences of jobs such that jobs of the same subsequence are processed without idle times between them and the first job of a subsequence starts at its release date. The first subsequence starts at the minimum release date of the jobs having no *predecessors* with respect to *prec*. The structure of such a schedule is shown in Figure 4.

Given set X of jobs, denote by X^+ the set of jobs from X that have no predecessors with respect to *prec*. The following procedure, denoted as ND, constructs a feasible non-delay schedule. In this procedure, k is the number of the created sets N_l (in which the jobs are scheduled jointly), $\pi^{(l)} = (\pi_1^{(l)}, \pi_2^{(l)}, \ldots, \pi_{|N_l|}^{(l)})$ is the processing sequence of the jobs in the set N_l, and $r^{(l)}$ is the starting time for the sequence $\pi^{(l)}$, $l = 1, \ldots, k$. Recall that $N = \{1, \ldots, n\}$.

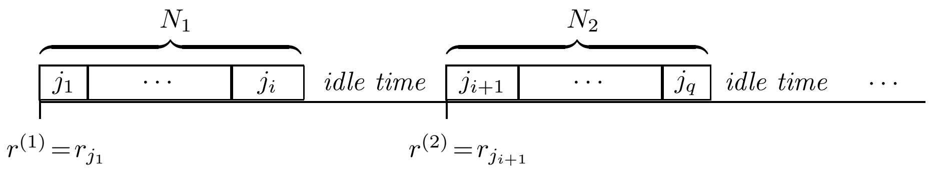

FIGURE 4. Structure of a non-delay schedule.

Procedure ND

Step 1. Set $l = 0$, $X = N$, $i = 1$ and $Z = 0$. Re-number the jobs in the non-decreasing order of their release dates such that $r_1 \leq \cdots \leq r_n$.
Step 2. Determine set X^+. Calculate $r_{\min} = \min\{r_j | j \in X^+\}$. If $Z < r_{\min}$, then select a job $j^0 \in \{j | r_j = r_{\min}, j \in X^+\}$. Set $i = 1$, re-set $l := l + 1$, calculate $r^{(l)} = r_{\min}$, $Z = r^{(l)} + p_{j^0}$ and $\pi_1^{(l)} = j^0$. Job j^0 is scheduled to start at its release date.
Otherwise, if $Z \geq r_{\min}$, then select a job $j^0 \in \{j | j \in X^+, r_j \leq Z\}$. Re-set $i := i + 1$, calculate $Z = Z + p_{j^0}$ and $\pi_i^{(l)} = j^0$. Job j^0 is scheduled immediately after the previous job in $\pi^{(l)}$ without an idle time between them.
In either case, re-set $X := X \backslash \{j^0\}$.
If $X \neq \phi$, then repeat Step 2. Otherwise, calculate $k = l$ and stop.

Theorem 5. *Procedure ND constructs a feasible non-delay schedule and runs in $O(n^2)$ time.*

Proof. It is easy to see that procedure ND constructs a feasible non-delay schedule because as soon as the machine becomes idle at a time Z, the procedure selects for processing the first available job, which is an arbitrary job j whose predecessors have already been scheduled and $r_j \leq Z$, or if there is no such job, then an arbitrary job with minimum release date among the jobs whose predecessors have been scheduled. In fact, procedure ND is an algorithmic definition of a non-delay schedule for the considered problem.

Let us determine the time complexity of procedure ND. Sequencing the jobs in the non-decreasing order of their release dates requires $O(n \log n)$ time. In Step 2, computation of X^+, the set of jobs that have no predecessors, and selection the job j^0 requires $O(n)$ time. Step 2 is repeated at most n times. This gives the overall time complexity of $O(n^2)$. $\square$

We observe that all feasible non-delay schedules can be constructed by procedure ND if ties in this procedure are settled in all possible ways. A tie in procedure ND appears if several jobs satisfy the selection conditions of Step 2. An analysis of procedure ND shows that all such jobs are assigned to the same sequence $\pi^{(l)}$. We deduce that all schedules constructed by procedure ND, *i.e.*, all feasible non-delay schedules, can differ only by the order of jobs in the sequences $\pi^{(l)}$, $l = 1, \ldots, k$. This observation implies the following corollaries.

Corollary 4. *A collection of the release dates $r^{(l)}$ and the sets N_l, $l = 1, \ldots, k$, is the same for each feasible non-delay schedule.*

Corollary 5. *All feasible non-delay schedules have the same $C_{\max}$ value.*

We now observe that procedure ND is, in fact, an optimal algorithm for the classical minimization problem $1|prec, r_j|C_{\max}$, see for example, Tanaev *et al.* [17]. Therefore, the statement of the previous corollary can be strengthened as follows.

Corollary 6. *All feasible non-delay schedules have the same $C_{\max}$ value, which is equal to the minimum $C_{\max}$ value for the problem $1|prec, r_j|C_{\max}$.*

We call the sequence of jobs in the non-decreasing order of their release dates the *Earliest Release Date (ERD)* sequence.

Observe that if $prec = \phi$, then procedure ND can be modified to assign jobs to the sequences $\pi^{(1)}, \pi^{(2)}, \ldots$ in the non-decreasing order of their release dates. In this case, it will actually partition the ERD sequence into subsequences and will run in $O(n)$ time, provided that the ERD sequence is given. This observation together with Corollary 6 imply the following corollary.

Corollary 7. *The problem $1(nd)|prec, r_j|(C_{\max} \to \max)$ can be solved in $O(n^2)$ time and the problem $1(nd)|r_j|(C_{\max} \to \max)$ can be solved in $O(n \log n)$ time.*

One more useful corollary can be formulated.

Corollary 8. *Given $r^{(l)}$ and N_l, $l = 1, \ldots, k$, the problem $1(nd)|prec, r_j|(\Phi \to \max)$, where Φ is an arbitrary (not necessarily regular) function, reduces to k problems $1(noidle)|prec, r_j|(\Phi \to \max)$ of scheduling jobs of the set N_l in the interval $[r^{(l)}, r^{(l)} + \sum_{j \in N_l} p_j]$ to maximize Φ, $l = 1, \ldots, k$. Notation "noidle" is used to specify that no idle time between the jobs is allowed.*

Corollary 8 allows to decompose the problem $1(nd)|prec, r_j|(\Phi \to \max)$ into several subproblems of smaller dimension. However, it is not clear if this decomposition provides an easy solution to the original problem. Below we show that in some cases it does.

Consider the problem $1(nd)|prec, r_j|(f_{\max} \to \max)$ and prove the following result.

Theorem 6. *The problem $1(noidle)|prec, r_j|(f_{\max} \to \max)$ and, hence, the problem $1(nd)|prec, r_j|(f_{\max} \to \max)$ can be solved in $O(n^4)$ time if each function f_j is computable in $O(1)$ time.*

Proof. The following modification of procedure ND, denoted as MND, can be used to construct a feasible schedule for the problem $1(noidle)|prec, r_j|(f_{\max} \to \max)$, if it exists. Let $(k_1, \ldots, k_n)$ be the ERD sequence of jobs.

Procedure MND

Step 1. Set $i = 0$ and $X = N$. Denote a sequence of jobs to be constructed as $S = (S_1, \ldots, S_n)$.

Step 2. Determine set X^+ and find job index $j^0 = \min\{j | k_j \in X^+\}$; Re-set $i := i + 1$ and calculate $S_i = k_{j^0}$. Re-set $X := X \backslash \{k_{j^0}\}$. If $X \neq \phi$, then repeat Step 2. Otherwise, go to Step 3.

Step 3. Construct a non-delay schedule corresponding to the job sequence S. It is the required schedule, if it has no idle time between the jobs. Otherwise, such a schedule does not exist. Stop.

Observe that the job sequence created in Step 2 of procedure MND does not depend on the particular values of r_j, $j = 1, \dots, n$, but on their relative values. In other words, procedure MND will produce the same job sequence for $r_1, \dots, r_n$ and $r'_1, \dots, r'_n$ if $r_1 \leq \dots \leq r_n$ and $r'_1 \leq \dots \leq r'_n$.

Let f^* be an optimal objective function value for the problem $1(noidle)|\, prec, r_j|$ $(f_{\max} \to \max)$. A feasible schedule is optimal for this problem if and only if there exists job j^* such that $f_{j^*}(C_{j^*}) = f^*$, or equivalently

$$C_{j^*} \geq R_{j^*}(f^*), \tag{4}$$

where $R_{j^*}(f^*)$ is such a value that $f_{j^*}(R_{j^*}(f^*) - 1) < f^*$ and $f_{j^*}(R_{j^*}(f^*)) = f^*$.

Inequality (4) implies that job j^* does not start earlier than at time $R_{j^*}(f^*) - p_{j^*}$. Given value f^* and job j^* satisfying (4), introduce new release dates: $r'_{j^*} = \max\{r_{j^*}, R_{j^*}(f^*) - p_{j^*}\}$ and $r'_i = r_i$ for $i \neq j^*$. The problem $1(noidle)|\, prec, r_j|$ $(f_{\max} \to \max)$ reduces to constructing a job schedule feasible with respect to the release dates r'_j, $j = 1, \dots, n$, precedence constraints *prec*, and such that $C_{\max} = \sum_{i=1}^n p_i$, $j = 1, \dots, n$. Procedure MND can be used to construct such a schedule.

Assume without loss of generality that $r_1 \leq \dots \leq r_n$ and $r'_{i_1} \leq \dots \leq r'_{i_n}$. Observe that in the sequence $(i_1, \dots, i_n)$, only relative position of job j^* may be changed. Furthermore, it can only be shifted to the right comparing with its relative position in the sequence $(1, \dots, n)$, *i.e.*, this sequence is of the form $(1, \dots, j^* - 1, j^* + 1, \dots, i - 1, j^*, i + 1, \dots, n)$. Clearly, there are $n - j^* + 1$ such sequences. Consider such a sequence $\sigma^{(i)}$ with job j^* in the position i. Apply procedure MND assuming that $\sigma^{(i)}$ is the ERD job sequence. Let the job sequence $S^{(i)}$ be found in Step 2 of this procedure.

Given j^*, we can construct all $n - j^* + 1$ sequences $S^{(i)}$, $i = j^*, \dots, n$. Among them, we can choose sequences corresponding to the schedules feasible with respect to the original release dates and having no idle time between the jobs. The sequence with the largest $f_{\max}$ value among these sequences is an optimal solution to the problem $1(noidle)|prec, r_j|(f_{\max} \to \max)$.

An optimal solution to the problem $1(noidle)|prec, r_j|(f_{\max} \to \max)$ can be found by applying the approach described above for each job $j^* = 1, \dots, n$.

Given a job sequence, feasibility with respect to the release dates and the no idle time condition can be checked in $O(n)$ time. The corresponding $f_{\max}$ value can be computed in $O(n)$ time if each function f_j is computable in $O(1)$ time. Since we have at most $n(n+1)/2$ sequences to consider and procedure MND runs in $O(n^2)$ time, the proposed algorithm requires $O(n^4)$ time. □

Since procedure MND runs in $O(n)$ time if there are no precedence constraints, we obtain the following corollary.

Corollary 9. *The problem* $1(nd)|r_j|(f_{\max} \to \max)$ *can be solved in* $O(n^3)$ *time.*

Denote by $prec^R$ precedence constraints that are reversed with respect to *prec*. Introduce notation $\bar{d}_j$ for job deadlines. If descriptor $\bar{d}_j$ is present in the second field of the problem notation then in the corresponding problem, the job deadlines $\bar{d}_j$, $j = 1, \ldots, n$, are given, which should not be violated. Set $P = \sum_{1 \le j \le n} p_j$.

Theorem 7. *The problem* $1(noidle)|prec, r_j|(F \to \max)$ *is equivalent to the minimization problem* $1|prec^R, \bar{d}_j|F$ *for* $F \in \{\sum w_j C_j, \sum w_j U_j\}$.

Proof. Let π be an arbitrary job sequence feasible with respect to r_j, $j = 1, \ldots, n$, and *prec*. To simplify notation, assume that $\pi = (1, 2, \ldots, n)$. Consider the reversed sequence $\pi^R = (n, n-1, \ldots, 1)$. It is clear that π^R is feasible with respect to $prec^R$.

Denote by $C_j(\sigma)$ the completion time of job j according to the job sequence σ. We have

$$C_j(\pi) = P - C_j(\pi^R) + p_j, \quad j = 1, \ldots, n.$$

From the above equations, we obtain that sequence π is feasible with respect to r_j, $j = 1, \ldots, n$, if and only if sequence π^R is feasible with respect to the deadlines $\bar{d}_j = P - r_j$, $j = 1, \ldots, n$.

We have

$$\sum w_j C_j(\pi) = \sum w_j (P - C_j\left(\pi^R\right) + p_j) = \sum w_j (P + p_j) - \sum w_j C_j\left(\pi^R\right).$$

These equations show that π is optimal for $1(noidle)|prec, r_j|$ $(\sum w_j C_j \to \max)$ if and only if π^R is optimal for the corresponding minimization problem $1|prec^R, \bar{d}_j|\sum w_j C_j$.

Introduce due dates $d_j^R = P + p_j - d_j - 1$, $j = 1, \ldots, n$. Let job j be on-time with respect to the sequence π and the due date $d_j : C_j(\pi) \le d_j$. Then $C_j(\pi^R) = P - C_j(\pi) + p_j \ge P - d_j + p_j > d_j^R$, *i.e.*, job j is late with respect to the sequence π^R and the due date d_j^R.

Let job j be late with respect to the sequence π and the due date $d_j : C_j(\pi) > d_j$. Then $C_j(\pi^R) < P - d_j + p_j$ and, since P, p_j and d_j are integer, $C_j(\pi^R) \le P - d_j + p_j - 1 = d_j^R$, *i.e.*, job j is on-time with respect to the sequence π^R and the due date d_j^R.

Denote by X (X_R) the set of jobs which are late with respect to the job sequence π and due dates d_j, $j = 1, \ldots, n$ (the job sequence π^R and due dates d_j^R, $j = 1, \ldots, n$). We have $\sum_{j \in X} w_j = \sum w_j - \sum_{j \in X_R} w_j$.

The latter equation shows that π is optimal for $1(noidle)|prec, r_j|(\sum w_j U_j \to \max)$ if and only if π^R is optimal for $1|prec^R, \bar{d}_j|\sum w_j U_j$ with due dates d_j^R, $j = 1, \ldots, n$. $\square$

It is known that the minimization problem $1|\bar{d}_j|\sum C_j$ is solvable in $O(n \log n)$ time, see Smith [16]. Therefore, the problem $1(nd)|r_j|(\sum C_j \to \max)$ is solvable in $O(n \log n)$ time. However, the problem $1|\bar{d}_j|\sum U_j$ is NP-hard, see Lawler [13], and consequently, the maximization problem $1(nd)|r_j|(\sum U_j \to \max)$ is also NP-hard.

TABLE 1. Complexity of the studied problems.

Problem	Complexity	Complexity of the minimization counterpart
$1(sa)\|prec, r_j\|(C_{max} \to \max)$	$O(n^2)$	strongly NP-hard
$1(sa)\|prec, r_j\|(f_{max} \to \max)$	$O(n^3)$	strongly NP-hard
$1(sa)\|r_j\|(f_{max} \to \max)$	$O(n)$	strongly NP-hard
$1(sa)\|r_j\|(\sum w_jC_j \to \max)$	$O(n \log n)$	strongly NP-hard
$1(sa)\|r_j\|(\sum U_j \to \max)$	$O(n^2 \log n)$	strongly NP-hard
$1(sa)\|r_j\|(\sum w_jU_j \to \max)$	NP-hard	strongly NP-hard
$1(a)\|r_j\|(C_{\max} \to \max)$	NP-hard	$O(n \log n)$
$1(a)\|r_j\|(\sum w_jC_j \to \max)$	strongly NP-hard	strongly NP-hard
$1(nd)\|prec, r_j\|(C_{max} \to \max)$	$O(n^2)$	strongly NP-hard
$1(nd)\|prec, r_j\|(f_{max} \to \max)$	$O(n^4)$	strongly NP-hard
$1(nd)\|r_j\|(f_{\max} \to \max)$	$O(n^3)$	strongly NP-hard
$1(nd)\|r_j\|(\sum C_j \to \max)$	$O(n \log n)$	strongly NP-hard
$1(nd)\|r_j\|(\sum w_jC_j \to \max)$	strongly NP-hard	strongly NP-hard
$1(nd)\|r_j\|(\sum U_j \to \max)$	NP-hard	strongly NP-hard
$1(nd)\|prec, r_j\|(\sum C_j \to \max)$	strongly NP-hard	strongly NP-hard
$1(nd)\|prec, r_j\|(\sum U_j \to \max)$	strongly NP-hard	strongly NP-hard

TABLE 2. Open problems.

Problem	minimal/maximal open
$1(sa)\|r_j\|(\sum w_jT_j \to \max)$	maximal open
$1(a)\|r_j\|(C_{\max} \to \max)$ with 2 distinct release dates	minimal open
$1(a)\|r_j\|(\sum C_j \to \max)$	minimal open
$1(a)\|r_j\|(\sum w_jC_j \to \max)$ with 2 distinct release dates	maximal open
$1(a)\|r_j\|(\sum U_j \to \max)$	minimal open

5. CONCLUSION

In this paper, the problem of evaluating the worst case performance of a flexible solution has been studied. A flexible solution is completely characterized by a schedule type and a partial order of the jobs. Its worst case performance is characterized by the maximal value of the objective function defined on the set of schedules associated with the flexible solution.

As a first investigation step, we considered single machine scheduling problems where the jobs have different release dates and precedence relations are given on the set of jobs. The objective is to find an active or non-delay schedule that maximizes a regular objective function. Results obtained in this paper and our earlier paper [2] are summarized in Table 1. Notation *sa* indicates that only semi-active schedules are considered.

Table 2 enumerates the problems that remain open and indicates if they are minimal or maximal open.

Minimal open problems are those for which the complexity status is not known, but all their easier cases are polynomially solvable. Maximal open problems are those for which the complexity status is not known, but all their harder cases are NP-hard, see Brucker [7].

In our earlier paper [2], we observed that maximization problems for semi-active schedules are at least as easy as their minimization counterparts. It is also the case for non-delay schedules. However, it seems that maximization problems for active schedules are at least as difficult as their minimization counterparts. Indeed, the problem of minimizing the makespan is very simple but the corresponding maximization problem is NP-hard. However, we cannot make a general conclusion because, for example, the problem $1(a)|r_j|(\sum C_j \rightarrow \max)$ is open and the minimization counterpart is strongly NP-hard.

From a practical point of view, the obtained complexity results can suggest that, in the evaluation phase of the proactive-reactive scheduling approach of Aloulou and Portmann [1, 3], semi-active or non-delay schedules are more preferable than active schedules to be considered because their worst-case performances can be evaluated easier. Furthermore, these schedules are easier to implement on-line.

For future investigations, it is interesting to consider other processing systems such as flow and job shops. At present, we are working on a generalization of the proactive-reactive approach [1, 3] and the results of this paper for a flow shop system.

Acknowledgements. We would like to thank a referee who pointed out several places in the original submission, which needed to be corrected. The research of M.Y. Kovalyov was supported in part by the International Science and Technology Center under grant number B-986.

References

[1] M.A. Aloulou, On the reactive scheduling design using flexible predictive schedules, in *Proceedings of IEEE SMC'2002*, 6 pages in CD–ROM, Hammamet, October 2002.

[2] M.A. Aloulou, M.Y. Kovalyov and M.C. Portmann, Maximization problems in single machine scheduling. *Ann. Oper. Res.* **129** (2004) 21–35.

[3] M.A. Aloulou and M.C. Portmann, An efficient proactive reactive approach to hedge against shop flow disruptions, in *Multidisciplinary Scheduling: Thoery and Applications*, edited by G. Kendall, E. Burke, S. Petrovic and M. Gendreau, Springer (2005) 223–246.

[4] C. Artigues, J.-C. Billaut and C. Esswein, Maximization of solution flexibility for robust shop scheduling. *Eur. J. Oper. Res.* **165** (2005) 314–328.

[5] C. Artigues, F. Roubellat and J.-C. Billaut, Characterization of a set of schedules in a resource constrained multi-project scheduling problem with multiple modes. *Inter. J. Industrial Eng. Applications Practice* **6** (1999) 112–122.

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

[7] P. Brucker, *Scheduling algorithms.* Springer-Verlag (1998).

[8] P. Brucker and S. Knust, Complexity results for scheduling problems. `http://www.mathematik.uni-osnabrueck.de/research/OR/class/` (2003).

[9] R.L. Daniels and P. Kouvelis, Robust scheduling to hedge against processing time uncertainty in single stage production. *Manage. Sci.* **41** (1995) 363–376.

[10] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, edited by W.H. Freeman (1979).

[11] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic machine scheduling: a survey. *Ann. Discrete Math.* **5** (1979) 287–326.

[12] W. Herroelen and R. Leus, Project scheduling under uncertainty: survey and research potentials. *Eur. J. Oper. Res.* **165** (2005) 289–306.

[13] E.L. Lawler, *Scheduling a single machine to minimize the number of late jobs.* Technical report, Computer Science Division, University of California, Berkeley, USA (1983).

[14] S.V. Mehta and R. Uzsoy, Predictable scheduling of a single machine subject to breakdowns. *Inter. J. Compu. Integrated Manufacturing* **12** (1999) 15–38.

[15] M.E. Posner, Reducibility among weighted completion time scheduling problems. *Ann. Oper. Res.* (1990) 91–101.

[16] W.E. Smith, Various optimizers for single-stage production. *Naval Research Logistics Quarterly* **3** (1956) 59–66.

[17] V.S. Tanaev, V.S. Gordon and Y.M. Shafransky, *Scheduling Theory. Single-Stage Systems.* Kluwer Academic Publishers (1994).

[18] S.D. Wu, E.S. Byeon and R.H. Storer, A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Oper. Res.* **47** (1999) 113–124.