Memorandum COSOR 92-21

Polynomial-time algorithms for single-machine bicriteria scheduling

J.A. Hoogeveen
S.L. van de Velde

Eindhoven, June 1992
The Netherlands
Polynomial-time algorithms for single-machine bicriteria scheduling

J.A. Hoogeveen
Department of Mathematics and Computing Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

S.L. van de Velde
School of Management Studies, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

We address the problem of scheduling \(n\) independent jobs on a single machine so as to minimize an objective function that is composed of total completion time and a minmax criterion. First, we show that, if the second criterion is maximum cost, then the problem is solvable in \(O(n^3 \min(n, \log n + \log P_{\text{max}}))\) time, where \(P_{\text{max}}\) is the maximum job processing time, for every non-decreasing composite objective function. Second, we show that, if the second criterion is maximum promptness, then the problem is solved in \(O(n^4)\) time for every nondecreasing linear composite objective function if preemption is allowed or if total completion time outweighs maximum promptness.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single-machine scheduling, bicriteria scheduling, composite objective function, Pareto optimal points, extreme points, total completion time, maximum cost, maximum lateness, maximum promptness.
1. Introduction

Scheduling theory was introduced in the 1950's. Since that time, scheduling models have become more and more complex in order to better resemble the underlying practical situations. Although many characterizations of practical problems have been included, the simplification of evaluating a solution with respect to only one criterion has remained common practice. The vast majority of the papers on scheduling deals with problems in which the quality of a solution is measured in terms of a single criterion.

In practice, however, quality is a multidimensional notion. A firm judges a production scheme on the basis of a number of criteria. These are, for example, work-in-process inventories, an indication of the efficiency of the production process, and observance of due dates, an indication of consumer's satisfaction. If only one criterion is taken into account, then the outcome is likely to be unbalanced, no matter what criterion is considered. If everything is set on keeping work-in-process inventories low, then some products are likely to be completed far beyond their due date, whereas, if the main goal is to keep the customers satisfied by observing due dates, then the work-in-process inventories are likely to be large. In order to reach an acceptable compromise, one has to measure the quality of a solution with respect to all important criteria.

We model the firm as a single-machine job shop, which is described as follows. A set of \( n \) independent jobs has to be scheduled on a single machine that is continuously available from time zero onwards and that can process at most one job at a time. Each job \( J_j \) \((j = 1, \ldots, n)\) requires an uninterrupted positive processing time \( p_j \), and has a due date \( d_j \) and a target start time \( s_j \). Without loss of generality, we assume that the processing times, due dates, and target start times are integral. A schedule \( \sigma \) specifies for each job when it is executed with due observance of the machine availability constraints; hence, a schedule defines for each job \( J_j \) its start time \( S_j \) and its completion time \( C_j \). A performance measure or scheduling criterion associates a value \( f(\sigma) \) with each feasible schedule \( \sigma \). The measures we consider in our bicriteria problems involve total completion time \( \Sigma C_j \), maximum lateness \( L_{\text{max}} \), maximum promptness \( P_{\text{max}} \), and maximum cost \( f_{\text{max}} \). Maximum lateness is defined as \( \max_{1 \leq j \leq n} (C_j - d_j) \), maximum promptness is defined as \( \max_{1 \leq j \leq n} (s_j - S_j) \), which reduces to maximum earliness \( E_{\text{max}} \) in case \( s_j = d_j - p_j \), and maximum cost is defined as \( \max_{1 \leq j \leq n} f_j(C_j) \), where each \( f_j \) is an arbitrary regular cost function for \( J_j \); regular means that \( f_j(C_j) \) does not decrease when \( C_j \) is increased. Correspondingly, a performance measure is called regular if it is nondecreasing in the job completion times; total completion time and maximum lateness are of this type. A schedule \( \sigma^* \) is optimal for a given performance measure if \( f(\sigma^*) = \min_{\sigma \in \Omega} f(\sigma) \), where \( \Omega \) denotes the set of feasible schedules. Note that in case of a regular performance measure there is an optimal schedule such that no job can start earlier without affecting the start time of any other job. In that case, a sequence or permutation of the \( n \) jobs defines a unique schedule.

Total completion time \( \Sigma C_j \) is commonly used to measure work-in-process inventories; the parts needed in the processing of the job have to be stored until the job is completed. Maximum lateness measures the observance of due dates; maximum promptness measures the observance of target start times. The maximum cost criterion can be used to make the penalties job-dependent or to penalize large completion times more severely; \( f_j(C_j) = w_j(C_j - d_j) \), for example, resembles the first option, whereas \( f_j(C_j) = (\max(0, C_j - d_j))^2 \) resembles the second option.

The first results in machine scheduling include simple polynomial-time algorithms for the single-criterion problems involving these performance measures. Total completion time is minimized by
sequencing the jobs according to the shortest-processing-time (SPT) rule, that is, in order of nondecreasing $p_i$ (Smith, 1956). Maximum lateness is minimized by sequencing the jobs according to the earliest-due-date (EDD) rule, that is, in order of nondecreasing $d_j$ (Jackson, 1955). Maximum promptness subject to no machine idle time is minimized by sequencing the jobs according to the minimum-target-start-time (MTST) rule, that is, in order of nondecreasing $s_j$; the no-machine-idle-time constraint is necessary to prohibit unbounded solutions. Maximum cost is minimized by Lawler's rule (Lawler, 1973): while there are unassigned jobs, assign the job that has minimum cost when scheduled in the last unassigned position to that position.

In case of multiple criteria, the concept of optimality can be defined in several ways. Basically, there are two methods to do so. The first one is lexicographical minimization; in this case, the objectives are assumed to be subject to a hierarchy, and the objectives are considered sequentially in order of decreasing relevance. The first paper on a scheduling problem of this type is by Heck and Roberts (1972) who minimize total completion time subject to minimum maximum lateness; they present a polynomial-time algorithm based on Smith's algorithm (Smith, 1956) for minimizing total completion time subject to no tardy jobs. Note that in case of hierarchical minimization we do not mind having unbalanced schedules.

The second one is simultaneous minimization. In this method, the performance measures, specified by the functions $f_k (k = 1, \ldots, K)$, are transformed into a single composite objective function $F : \mathbb{R} \to \mathbb{R}$. In this paper, we consider scheduling problems with objective functions of this type. With each schedule $\sigma$ we associate a point $(f_1(\sigma), \ldots, f_K(\sigma))$ in $\mathbb{R}^K$ and a value $F (f_1(\sigma), \ldots, f_K(\sigma))$. In the remainder, the terms schedule and point are used interchangeably. The associated problem, from now on referred to as problem (P), is formulated as

$$\min_{\sigma \in \Omega} F (f_1(\sigma), \ldots, f_K(\sigma)).$$

where $F$ is nondecreasing in each of its arguments. An example is the problem of minimizing total completion time and maximum lateness simultaneously (Van Wassenhove and Gelders, 1980).

A natural question is whether problem (P) is solvable in polynomial time for a given function $F$. It is straightforward that we can solve this problem in polynomial time for any function $F$ that is nondecreasing in its arguments if we can identify all of the so-called Pareto optimal schedules in polynomial time.

**Definition 1.** A schedule $\sigma \in \Omega$ is Pareto optimal with respect to the objective functions $(f_1, \ldots, f_K)$ if there exists no feasible schedule $\pi$ with $f_k(\pi) \leq f_k(\sigma)$ for all $k = 1, \ldots, K$ and $f_k(\pi) < f_k(\sigma)$ for at least one $k, k = 1, \ldots, K$.

Once the Pareto optimal set, that is, the set of all schedules that are Pareto optimal with respect to the functions $(f_1, \ldots, f_K)$, has been determined, problem (P) can be solved for any function $F$ that is nondecreasing in each of its arguments by computing the cost of each Pareto optimal point and taking the minimum. As a consequence, if each Pareto optimal schedule can be found in polynomial time and if the cardinality of the Pareto optimal set is polynomially bounded in the input size, then problem (P) is polynomially solvable.

An interesting special case of (P) is one in which the composite objective function is linear. The problem, referred to as problem $(P_\alpha)$, is then formulated as

$$\min_{\sigma \in \Omega} F_{\alpha}(\sigma) = \min_{\sigma \in \Omega} \sum_{k=1}^K \alpha_k f_k(\sigma).$$

(P$_\alpha$)
where \( \alpha = (\alpha_1, \ldots, \alpha_K) \) is a given real-valued vector of nonnegative weights. In analogy to problem (P), we wish to determine the set of schedules that contains an optimal solution to problem \( (P_\alpha) \) for any weight vector \( \alpha \geq 0 \). We define this set as the set of extreme schedules.

**Definition 2.** A schedule \( \sigma \in \Omega \) is extreme with respect to the objective functions \( (f_1, \ldots, f_K) \) if it corresponds to a vertex of the lower envelope of the Pareto optimal set for \( (f_1, \ldots, f_K) \).

Once the set of extreme schedules with respect to the objective functions \( (f_1, \ldots, f_K) \) has been identified, problem \( (P_\alpha) \) can be solved for any given \( \alpha \geq 0 \) by computing the cost of each extreme point and taking the minimum.

Throughout the paper, we adopt and extend the three-field notation scheme of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) to classify scheduling problems with multiple criteria. For instance, \( 1 || F(\Sigma C_j, L_{\text{max}}) \) denotes the problem of minimizing an arbitrary nondecreasing function of total completion time and maximum lateness on a single machine, whereas \( 1 || \alpha_1 \Sigma C_j + \alpha_2 L_{\text{max}} \) denotes its linear counterpart.

As mentioned before, there are relatively few papers on multicriteria scheduling problems; virtually all of them concern bicriteria problems. For a survey, see Dileepan and Sen (1988); for a survey of polynomial-time algorithms and complexity results, see Hoogeveen (1992). As far as the performance measures \( \Sigma C_j, L_{\text{max}}, P_{\text{max}}, \) and \( f_{\text{max}} \) are concerned the following is known. Hoogeveen (1992) shows that \( 1 || F(P_{\text{max}}, L_{\text{max}}) \) is \( \text{NP}-\text{hard} \) in the strong sense; he also distinguishes a class of problems that are solvable in \( O(n^2 \log n) \) time, including \( 1 || E_{\text{max}} + L_{\text{max}} \) and \( 1 || \max \{ E_{\text{max}}, L_{\text{max}} \} \). For the latter problem, Garey, Tarjan, and Wilfong (1988) give an alternative algorithm running in \( O(n \log(\Sigma P_j)) \) time. In addition, Hoogeveen (1992) presents an \( O(n^3 \log n) \) algorithm to solve \( 1 || F(L_{\text{max}}, f_{\text{max}}) \), and proves that \( 1 || F(P_{\text{max}}, f_{\text{max}}) \) is \( \text{NP}-\text{hard} \) in the strong sense.

In this paper, we consider the bicriteria problems of simultaneously minimizing total completion time \( \Sigma C_j \) and one of the minmax criteria \( L_{\text{max}}, f_{\text{max}}, \) and \( P_{\text{max}} \). For the \( 1 || F(\Sigma C_j, L_{\text{max}}) \) problem, Van Wassenhove and Gelders (1980) and Nelson, Sarin, and Daniels (1986) show that by iterative application of Smith's rule all Pareto optimal points for \( (\Sigma C_j, L_{\text{max}}) \) can be determined. John (1989) extends their algorithm to determine the set of Pareto optimal points for \( (\Sigma C_j, f_{\text{max}}) \). The complexity of all of these algorithms depends on the number of Pareto optimal points. For \( (\Sigma C_j, L_{\text{max}}) \), this number has been subject of a lot of misunderstanding. Lawler, Lenstra, and Rinnooy Kan (1979) claimed that this number is equal to \( n(n-1)/2 + 1 \). Van Wassenhove and Gelders, on the other hand, supposed that the number of Pareto optimal points for \( (\Sigma C_j, L_{\text{max}}) \) is bounded only pseudo-polynomially; hence, they presented their algorithm as being pseudo-polynomial. This inspired Sen and Gupta (1983) to present a branch-and-bound algorithm for \( 1 || \Sigma C_j + L_{\text{max}} \). We prove that the number of Pareto optimal points for \( (\Sigma C_j, f_{\text{max}}) \) is at most equal to \( n(n-1)/2 + 1 \). As a consequence, \( 1 || F(\Sigma C_j, f_{\text{max}}) \) is polynomially solvable. In Section 2, we present an algorithm for \( 1 || F(\Sigma C_j, f_{\text{max}}) \) that runs in \( O(n^3 \min \{ n, \log(\Sigma P_j) \}) \) time; it can be implemented to run in \( O(n^3 \log n) \) time if \( f_{\text{max}} = L_{\text{max}} \).

In Section 3, we consider \( 1 || \text{pmtn} || F(\Sigma C_j, P_{\text{max}}) \); the notation \( \text{pmtn} \) signifies that job splitting is allowed, that is, the execution of a job can be interrupted and resumed later. The main results are that \( 1 || \text{pmtn} || \alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) and, in the case that \( \alpha_1 \geq \alpha_2 \), also \( 1 || \alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) are solvable in \( O(n^4) \) time.
2. Minimizing total completion time and maximum cost

Let $f_j: \mathbb{N} \to \mathbb{R}$ denote a regular cost function for job $J_j$ ($j = 1, \ldots, n$); accordingly, $f_j(C_j)$ denotes the cost incurred by completing job $J_j$ at time $C_j$. In addition, let $f_{\max} = \max_j f_j(C_j)$. We show that $1 \mid F(\Sigma C_j, f_{\max})$ is solvable in $O(n^{1.5} \ln n + \log p_{\max})$ time, with $p_{\max} = \max_j p_j$, for any function $F$ that is nondecreasing in both $\Sigma C_j$ and $f_{\max}$. Note that $1 \mid F(\Sigma C_j, L_{\max})$ corresponds to a special case of $1 \mid F(\Sigma C_j, f_{\max})$.

The first result on this problem is due to Emmons (1975), who considered the hierarchical problem of minimizing $\Sigma C_j$ subject to the constraint that $f_{\max}$ is minimal; this problem is denoted as $1 \mid f_{\max} \mid \Sigma C_j$, where $f^*$ denotes the solution value of the outcome of $1 \mid f_{\max}$. Once $f^*$ has been determined by Lawler's algorithm, Emmons's algorithm requires $O(n^2)$ time to minimize total completion time subject to minimum maximum cost. Observe, however, that an upper bound on $f_j(C_j)$ induces a deadline $d_j$ on the completion time of $J_j$. Each deadline can be determined in $O(\log p_{\max})$ time by binary search over the $O(\Sigma p_j)$ possible completion times. Furthermore, $d_j$ is computed in constant time if $f_j$ has an inverse. Once the deadlines have been computed, the problem in the second phase is to minimize total completion time subject to deadlines, denoted as $1 \mid d_j \mid \Sigma C_j$, which requires only $O(n \log n)$ time (Smith, 1956).

We state the algorithm based on deadline determination for $1 \mid f_{\max} \leq f \mid \Sigma C_j$, where $f$ is some upper bound on the cost of the schedule.

Algorithm I (Smith, 1956)
Step 1. Compute for each job $J_j$ the deadline $d_j$ induced by $f_j(C_j) \leq f$.
Step 2. $T \leftarrow \Sigma p_j$.
Step 3. Determine $U \leftarrow \{J_j \in J \mid d_j \geq T\}$ as the set containing the jobs that may be completed at time $T$.
Step 4. Determine $J_j$ such that $p_j = \max \{p_j \mid J_j \in U\}$; in case of ties, $J_j$ is chosen to be the job with smallest cost when completed at time $T$.
Step 5. $J \leftarrow J \setminus \{J_j\}$; $T \leftarrow T - p_j$.
Step 6. If $T > 0$, then go to Step 3.

Theorem 1. Algorithm I determines a Pareto optimal point with respect to $\Sigma C_j$ and $f_{\max}$.

Proof. It suffices to show that the algorithm generates a schedule $\sigma$ that solves the problems $1 \mid f_{\max} \leq f \mid \Sigma C_j$ and $1 \mid \Sigma C_j \leq \Sigma C_j(\sigma) \mid f_{\max}$ simultaneously. Evidently, $\sigma$ solves $1 \mid f_{\max} \leq f \mid \Sigma C_j$. Assume that not $\sigma$, but $\pi$ is optimal for $1 \mid \Sigma C_j \leq \Sigma C_j(\sigma) \mid f_{\max}$. This implies that $f_{\max}(\pi) < f_{\max}(\sigma) \leq f$; hence, $\pi$ is also feasible for $1 \mid f_{\max} \leq f \mid \Sigma C_j$. Therefore, we have $\Sigma C_j(\pi) = \Sigma C_j(\sigma)$. Compare the two schedules, starting at the end. Suppose that the first difference occurs at the $k$th position, which is occupied by jobs $J_i$ and $J_j$ in $\sigma$ and $\pi$, respectively. Since $f_{\max}(\pi) < f$ and because of the choice of job $J_i$ in the algorithm, we have $p_i \geq p_j$. If $p_i > p_j$, then $\pi$ cannot be optimal, as the schedule that is obtained by interchanging $J_i$ and $J_j$ in $\pi$ is feasible with respect to the constraint $f_{\max} \leq f$ and has smaller total completion time. Hence, it must be that $p_i = p_j$ and, because of the choice of job $J_i$ in the algorithm, $f_i(C_i(\pi)) < f_i(C_i(\sigma))$. This implies, however, that the jobs $J_i$ and $J_j$ can be interchanged in $\pi$ without affecting the cost of the schedule. Repetition of this argument shows that $\pi$ can be transformed into $\sigma$ without affecting the cost, thereby contradicting the assumption that $f_{\max}(\pi) < f_{\max}(\sigma)$. Therefore, $\sigma$ also solves $1 \mid \Sigma C_j \leq \Sigma C_j(\sigma) \mid f_{\max}$; hence, $\sigma$ is Pareto optimal for $\Sigma C_j$ and $f_{\max}$. \qed
It is obvious that the maximum cost of each Pareto optimal schedule ranges from $f^*$ to $f_{max}(SPT)$, where $SPT$ denotes the schedule obtained by settling ties in the $SPT$-order to minimize maximum cost. The next algorithm, which is similar to Van Wassenhove and Gelders's algorithm for $1\mid |F \Sigma C_j, L_{max}\rangle$, exploits this property for finding the Pareto optimal set.

**Algorithm II**

Step 1. Compute $f^*$ and $f_{max}(SPT)$; let $k \leftarrow 1$.

Step 2. Solve $1 \mid f_{max} < f_{max}(SPT) \mid \Sigma C_j$; this produces the first Pareto optimal schedule, denoted as $\sigma^{(1)}$, and the first Pareto optimal point, denoted as $(\Sigma C_j(\sigma^{(1)}), f_{max}(\sigma^{(1)}))$.

Step 3. $k \leftarrow k + 1$. Solve $1 \mid f_{max} < f_{max}^{(k-1)} \mid \Sigma C_j$; this produces the $k$th Pareto optimal schedule, denoted as $\sigma^{(k)}$, and the $k$th Pareto optimal point, denoted as $(\Sigma C_j(\sigma^{(k)}), f_{max}(\sigma^{(k)}))$.

Step 4. If $f_{max}(\sigma^{(k)}) > f^*$, then go to Step 3.

A crucial issue is the number of Pareto optimal points generated by Algorithm II. In the remainder of this section, we prove that there are $O(n^2)$ such schedules, thereby establishing the polynomial nature of the algorithm.

We define the indicator function $\delta_{ij}(\sigma)$ as

$$
\delta_{ij}(\sigma) = \begin{cases} 
1 & \text{if } S_j(\sigma) < S_j(\sigma) \text{ and } p_i > p_j, \\
0 & \text{otherwise},
\end{cases}
$$

and $\Delta(\sigma) = \sum_{i,j} \delta_{ij}(\sigma)$. Note that $\delta_{ij}(\sigma) = 1$ implies that the interchange of the jobs $J_i$ and $J_j$ will decrease total completion time. In that respect, $\delta_{ij}(\sigma) = 1$ signals a positive interchange. Observe that $\Delta(SPT) = 0$ and $\Delta(\sigma) \leq n(n-1)/2$ for any $\sigma \in \Omega$. In addition, we define a neutral interchange with respect to $\sigma$ as the interchange of two jobs $J_i$ and $J_j$ with $p_i = p_j$.

**Lemma 1.** If schedule $\pi$ can be obtained from schedule $\sigma$ through a positive interchange, then $\Delta(\pi) < \Delta(\sigma)$.

**Proof.** Suppose that $J_i$ and $J_j$, with $p_i > p_j$, are the jobs that have been interchanged. The interchange affects only the jobs scheduled between $J_i$ and $J_j$. Let $J_i$ be an arbitrary job that is scheduled between $J_i$ and $J_j$ in $\sigma$. Then it is easy to verify that $\delta_i(\sigma) + \delta_{ij}(\sigma) \geq \delta_{ij}(\sigma) + \delta_i(\pi)$. □

**Theorem 2.** Consider two arbitrary Pareto optimal schedules $\sigma$ and $\pi$. If $\Sigma C_j(\sigma) < \Sigma C_j(\pi)$, then $\Delta(\sigma) < \Delta(\pi)$.

**Proof.** We show that schedule $\sigma$ can be obtained from schedule $\pi$ by using positive and neutral interchanges only. Compare the two schedules, starting at the end. Suppose that the first difference between the schedules occurs at the $k$th position: $J_i$ occupies the $k$th position in $\sigma$, whereas job $J_j$ occupies the $k$th position in $\pi$. Because of the choice of $J_i$ and $J_j$ in Algorithm I, we have $p_i \geq p_j$; the interchange of $J_i$ and $J_j$ in $\pi$ is therefore positive or neutral. We proceed in this way until we reach schedule $\sigma$. As $\Sigma C_j(\sigma) < \Sigma C_j(\pi)$, at least one of the interchanges must have been positive, and application of Lemma 1 yields the desired result. □

**Theorem 3.** The number of Pareto optimal schedules is bounded by $n(n-1)/2 + 1$, and this bound is tight.
Proof. The first part follows immediately from Theorem 9. For the second part, consider the following instance of $1 | F(\Sigma C_j, L_{\text{max}})$: there are $n$ jobs with processing times $p_j = n - 2 + j$ and due dates $d_j = \Sigma_{i=j}^n p_i + n - j$, for $j = 1, \ldots, n$. Straightforward computations show that this example generates $n(n-1)/2+1$ Pareto optimal schedules.

Corollary 1. The $1 | F(\Sigma C_j, f_{\text{max}})$ problem is solvable in $O(n^2 \min\{n, \log n + \log p_{\text{max}}\})$ time.

Proof. Emmons's algorithm requires $O(n^2)$ time to solve $1 | f_{\text{max}} \leq f | \Sigma C_j$. An alternative is to determine the induced deadlines, which requires $O(\log(\Sigma p_j))$ time, and to apply Smith's algorithm subsequently. There are $O(n^2)$ of such problems to be solved.

Corollary 2. The $1 | F(\Sigma C_j, L_{\text{max}})$ problem is solvable in $O(n^3 \log n)$ time.

4. Minimizing total completion time and maximum promptness

In this section, we analyze the problem of minimizing total completion time and maximum promptness simultaneously. First, we make the additional assumption that machine idle time is forbidden, implying that all jobs are to be scheduled in the interval $[0, \Sigma p_j]$; the insight gained from analyzing this special case is used to deal with the general problem. In the three-field notation scheme, the no-machine-idle-time constraint will be denoted by the acronym $nmit$ in the second field.

Due to this constraint, we have for each Pareto optimal schedule $\sigma$ that $P_{\text{max}}(\sigma)$ ranges from $P^*$, defined as the solution value of $1 | nmit | P_{\text{max}}$, to $P_{\text{max}}(\text{SPT})$, and that $\Sigma C_j(\sigma)$ ranges from $\Sigma C_j^*(\text{MTST})$, where ties in the SPT and MTST schedule are settled in order to minimize maximum promptness and completion time, respectively. Observe that an upper bound $P$ on $P_{\text{max}}$ induces for each job $J_j$ a release time $r_j = \max\{0, s_j - P\}$. The associated value of $\Sigma C_j$ can then be computed by solving $1 | r_j, nmit | \Sigma C_j$. Lenstra, Rinnooy Kan, and Brucker (1977), however, show this problem to be $\mathcal{NP}$-hard in the strong sense (Garey and Johnson, 1979).

Therefore, we make the additional assumption that preemption of jobs is allowed, that is, the execution of jobs may be interrupted and resumed later. This is a crucial relaxation, since the relaxed problem, denoted by $1 | \text{pmtn}, r_j | \Sigma C_j$ problem is solvable in $O(n \log n)$ time by Baker's algorithm (Baker, 1974): always keep the machine assigned to the available job with minimum remaining processing time. Note that this algorithm always generates a schedule without machine idle time if $P \geq P^*$.

The introduction of preemption has also a less convenient effect. Any value of $P_{\text{max}}$ in the range $[P^*, P_{\text{max}}(\text{SPT})]$ is now attainable, and therefore corresponds to a Pareto optimal point. Since $P_{\text{max}}(\text{SPT}) - P^* \leq \Sigma p_j$, the number of Pareto optimal schedules is only pseudo-polynomially bounded.

Corollary 3. The $1 | \text{pmtn}, nmit | F(\Sigma C_j, P_{\text{max}})$ problem is solvable in $O(n \Sigma p_j)$ time.

Proof. A decrease of $P$ does not affect the order of the release dates; hence, we have to sort the release dates only once.

As to the complexity of $1 | nmit, \text{pmtn} | F(\Sigma C_j, P_{\text{max}})$, note that we can obtain a series of $2^n$ consecutive Pareto optimal points by multiplying the processing times by $2^n$. As the problem of minimizing an arbitrary function $F(x, y)$ that is nondecreasing in both arguments over $2^n$ consecutive integral...
values is \( \mathcal{N}^P \)-hard in the strong sense (Schrijver; see Hoogeveen, 1992), we have that
\[ 1 \mid \text{numt, ptn} \mid F (\Sigma C_j, P_{\text{max}}) \]
is \( \mathcal{N}^P \)-hard in the ordinary sense (but not in the strong sense, as the processing times are exponential).

It follows immediately from the above reasoning that
\[ 1 \mid \text{pmtn} \mid F (\Sigma C_j, P_{\text{max}}) \]
is \( \mathcal{N}^P \)-hard in the strong sense, as we do not have to multiply the processing times with \( 2^n \) to obtain \( 2^n \) consecutive Pareto optimal points.

In the remainder of this section, we restrict ourselves to linear objective functions \( \alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \). To solve the linear variant, we only have to determine the set of extreme points. We start again with the assumption that machine idle time is not allowed; hence, we only have to consider \( P_{\text{max}} \) values in the interval \([P^*, P_{\text{max}}(\text{SPT})]\).

Let \( \sigma(P) \) denote the schedule obtained by Baker's algorithm for
\[ 1 \mid \text{pmtn}, P_{\text{max}} \leq P \mid \Sigma C_j \]; \( \sigma(P) \) corresponds to \((P, \Sigma C_j(\sigma(P)))\). We say that a complete interchange has occurred in \( \sigma(P) \) if there are two jobs \( J_i \) and \( J_j \) such that \( J_i \) is started before \( J_j \) in \( \sigma(P - 1) \), whereas \( J_j \) is started before \( J_i \) in \( \sigma(P) \).

Lemma 2. An upper bound \( P \) on \( P_{\text{max}} \) can only correspond to an extreme point for \((\Sigma C_j, P_{\text{max}})\) if a complete interchange has occurred in \( \sigma(P) \).

Proof. Consider an arbitrary extreme point \((P, \Sigma C_j(\sigma(P)))\). Define \( \Delta = \Sigma C_j(\sigma(P - 1)) - \Sigma C_j(\sigma(P)) \). As \((P, \Sigma C_j(\sigma(P)))\) is extreme, we must have \( \Sigma C_j(\sigma(P)) - \Sigma C_j(\sigma(P + 1)) < \Delta \). It is easy to see that this can only be the case if a complete interchange has taken place in \( \sigma(P) \). □

Obviously, the next step to determine the extreme set is to select the candidate values \( P \); these should be such that a complete interchange takes place in \( \sigma(P) \). Given a pair of jobs \( J_i \) and \( J_j \) with \( p_i > p_j \) and \( J_i \) started before \( J_j \) in \( \sigma(P) \), the increase necessary to enable a complete interchange of \( J_i \) and \( J_j \) is equal to the difference between the release date for \( J_j \) that follows from the constraint \( P_{\text{max}} \leq P \) and the start time of \( J_i \) in \( \sigma(P) \). However, if \( J_i \) is executed between the start and completion time of a preemptive job \( J_k \), then an increase of \( P \) will first result in a shift of \( J_i \) and \( J_j \) to the left; the complete interchange of \( J_i \) and \( J_j \) cannot take place before a complete interchange has taken place between \( J_k \) and both \( J_i \) and \( J_j \).

These observations are used in Algorithm III that, given an upper bound value \( P \) and the corresponding schedule \( \sigma(P) \), computes the smallest value \( P > P \) that possibly corresponds to an extreme point. The variable \( a_j \) \((j = 1, \ldots, n)\) signifies the increase of \( P \) necessary to let a complete interchange involving \( J_j \) take place.

Algorithm III

Step 1. Let \( T \leftarrow 0 \) and \( a_j \leftarrow \infty \) for \( j = 1, \ldots, n \).

Step 2. Let \( J_j \) be the job that starts at time \( T \). Consider the following two cases:

(a) \( J_j \) is a preempted job. Then \( a_j \) is equal to the length of this portion of \( J_j \). Let \( J_l \) be the first job that starts after time \( C_j(\sigma(P)) \) with \( p_l \geq a_j \). Set \( T \leftarrow S_l(\sigma(P)) \).

(b) \( J_j \) is not a preempted job. Then \( a_j \leftarrow \min\{s_j - P - S_j(\sigma(P)) \mid J_i \in J \} \), where \( J \) denotes the set of jobs for which \( s_j - P > S_j(\sigma(P)) \) and \( p_j > p_j \). Set \( T \leftarrow C_j(\sigma(P)) \).

Step 3. If \( T < \Sigma p_j \), then go to Step 2.

Step 4. Put \( P \leftarrow \min_j \{a_j \} + P \).
Theorem 4. All values $P$ that may correspond to an extreme point $(P, \sum C_j(\sigma(P)))$ are generated by the iterative application of Algorithm III.

Proof. Suppose that $P$, although corresponding to an extreme point, was not determined by iteratively applying Algorithm III. This implies that there is a value $P_1 < P$ such that Algorithm III determines a value $P_2 > P$ when initialized with $P_1$. Hence, we have the situation that Algorithm III did not notice the complete interchange of two jobs $J_i$ and $J_j$, which implies that the start time of $J_i$ in $\sigma(P)$ was not considered in Step 2. This, however, conflicts with the earlier observation that the interchange of $J_i$ and $J_j$ has to wait until $J_k$ has been interchanged with both $J_i$ and $J_j$. □

We prove that the number of values $P$ of $P_{\text{max}}$ generated through Algorithm III is polynomially bounded, thereby establishing that $1|\text{pmtn},n|\text{nmit}|\sigma_1\Sigma C_j+\sigma_2 P_{\text{max}}$ is polynomially solvable. We define for a given schedule $\sigma$ the indicator function $\delta_{ij}(\sigma)$ as

$$\delta_{ij}(\sigma) = \begin{cases} 1 & \text{if } C_i(\sigma) \leq S_j(\sigma) \text{ and } p_i > p_j, \\ 0 & \text{otherwise.} \end{cases}$$

We further define $\Delta_j(\sigma)$ as the sum of the number of preemptions in $J_j$ and $\Sigma_{i=1}^n \delta_{ij}; \Delta(\sigma) = \Sigma_{i,j} \delta_{ij}(\sigma)$.

Theorem 5. Let $P_1$ and $P_2$ be two $P_{\text{max}}$ values that are generated through Algorithm III, with $P_1 > P_2$. Then $\Delta(\sigma(P_1)) < \Delta(\sigma(P_2))$.

Proof. We start by showing that $\Delta(\sigma(P_1)) \leq \Delta(\sigma(P_2))$. Suppose to the contrary that $\Delta(\sigma(P_1)) > \Delta(\sigma(P_2))$. Then there must exist a job $J_j$ for which $\Delta_j(\sigma(P_1)) > \Delta_j(\sigma(P_2))$. There are two possibilities for an increase of $\Delta_j$.

First, the number of preemptions of $J_j$ in $\sigma(P_1)$ may be greater than in $\sigma(P_2)$. An extra preemption of $J_j$ can only occur when some job $J_k$ with $p_k < p_j$ is started after $C_j$ in $\sigma(P_2)$ but before $C_j$ in $\sigma(P_1)$. We then have, however, that $\delta_{jk}(\sigma(P_1)) = 0$ and $\delta_{jk}(\sigma(P_2)) = 1$. This implies that an extra preemption of $J_j$ decreases some $\Delta_k$ by the same amount; hence, an extra preemption does not increase $\Delta$.

Second, we may have $\delta_{ij}(\sigma(P_1)) = 1$, whereas $\delta_{ij}(\sigma(P_2)) = 0$. This implies that there exists some job $J_i$ with $p_i > p_j$ that is completed before $J_j$ is started in $\sigma(P_1)$ but not in $\sigma(P_2)$. As $P_1 > P_2$, this can only occur if there exists a job $J_k$ that is completed before $J_i$ in $\sigma(P_2)$ but after $J_i$ in $\sigma(P_1)$. Hence, $\Delta_j$ is then increased by 1, but $\Delta_k$ is decreased by at least 1, implying that $\Delta$ does not increase. The same argument holds if there are some jobs scheduled between $J_i$ and $J_j$ in $\sigma(P_1)$. Note that the decrease of $\Delta_k$ is always greater than the increase of $\Delta_j$, unless $J_k$ is preempted at the start of $J_i$ in $\sigma(P_1)$.

As $P_1$ has been determined by Algorithm III such that either a preemption is removed or an interchange has been completed, we have that $\Delta(\sigma(P_1)) < \Delta(\sigma(P_2))$. □

Corollary 4. If preemption is allowed, then the number of extreme schedules with respect to $P_{\text{max}}$ and $\Sigma C_j$ is bounded by $n(n-1)/2+1$.

Proof. It is easy to show that $\Delta(\sigma)$ is at most equal to $n(n-1)/2$ for every schedule $\sigma$. Therefore, Theorem 5 yields the desired result. □
Although it is easy to construct an instance such that Algorithm III determines \( n(n-1)/2+1 \) different \( P_{\text{max}} \) values, it is yet an open question whether this bound is tight for the number of extreme points.

**Corollary 5.** The \( 1|\text{pmtn, nmit}|\alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) problem is solvable in \( O(n^4) \) time.

**Theorem 6.** If \( \alpha_1 = \alpha_2 \), then there exists a nonpreemptive schedule that is optimal for \( 1|\text{pmtn, nmit}|\alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \). If \( \alpha_1 > \alpha_2 \), then any optimal schedule for \( 1|\text{pmtn, nmit}|\alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) is nonpreemptive.

**Proof.** Suppose that the optimal schedule contains a preempted job. Start at time 0 and find the first preempted job \( J_i \) immediately scheduled before some nonpreempted job \( J_j \). Consider the schedule obtained by interchanging job \( J_j \) and this portion of job \( J_i \). If the length of the portion of job \( J_i \) is \( \Delta \), then \( P_j \) is increased by \( \Delta \), while \( C_j \) is decreased by \( \Delta \). As \( \alpha_1 = \alpha_2 \), the interchange does not increase the objective value. The argument can be repeated until a nonpreemptive schedule remains. In case \( \alpha_1 > \alpha_2 \), then such an interchange would decrease the objective value contradicting the optimality of the obtained schedule.

We now drop the no-machine-idle-time constraint. As the insertion of idle time does not decrease \( \alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) if total completion time outweighs maximum promptness, we have the following corollary.

**Corollary 6.** If \( \alpha_1 \geq \alpha_2 \), then \( 1|\alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) is solvable in \( O(n^4) \) time.

If \( \alpha_1 < \alpha_2 \), then the insertion of idle time can decrease the value of the objective function. Consider the schedules \( \sigma(P) \) and \( \sigma(P+1) \), with \( P < P^* \). The idle time inserted between the jobs displays the same behavior as a preemptive job that is completed last: if \( P \) is increased by one unit then all jobs that have idle time between their start time and time 0 are shifted one unit to the left. Hence, given the \( P_{\text{max}} \) value \( \bar{P} \) of the first extreme point we can determine the set of extreme points by adding an extra job \( J_0 \) to the instance with \( p_0 = P^* - \bar{P} + P_{\text{max}} + 1 \) and \( s_0 = 0 \). The value \( P \) depends on the ratio \( q = \alpha_2/\alpha_1 \). If \( q > n \), then the insertion of idle time always decreases the value of the objective function and the optimal solution is unbounded. If \( q \leq n \), then the insertion of idle time decreases the value of the objective function as long as there are no more than \( \lceil q-1 \rceil \) jobs that have idle time between their start time and time 0. The corresponding value of the upper bound on \( P_{\text{max}} \) is easily determined.

As the number of extreme points is at most equal to \( n(n+1)/2+1 \), and as each \( P_{\text{max}} \) value that corresponds to an extreme point is determined by iterative application of Algorithm III, the \( 1|\text{pmtn}|\alpha_1 \Sigma C_j + \alpha_2 P_{\text{max}} \) problem is solved in \( O(n^4) \) time.

**Acknowledgement**

The authors like to thank Jan Karel Lenstra for his helpful comments.
References


<table>
<thead>
<tr>
<th>Number</th>
<th>Month</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>92-01</td>
<td>January</td>
<td>F.W. Steutel</td>
<td>On the addition of log-convex functions and sequences</td>
</tr>
<tr>
<td>92-02</td>
<td>January</td>
<td>P. v.d. Laan</td>
<td>Selection constants for Uniform populations</td>
</tr>
<tr>
<td>92-04</td>
<td>February</td>
<td>H.J.C. Huijberts, H. Nijmeijer</td>
<td>Strong dynamic input-output decoupling: from linearity to nonlinearity</td>
</tr>
<tr>
<td>92-05</td>
<td>March</td>
<td>S.J.L. v. Eijndhoven, J.M. Soethoudt</td>
<td>Introduction to a behavioral approach of continuous-time systems</td>
</tr>
<tr>
<td>92-06</td>
<td>April</td>
<td>P.J. Zwietering, E.H.L. Aarts, J. Wessels</td>
<td>The minimal number of layers of a perceptron that sorts</td>
</tr>
<tr>
<td>92-07</td>
<td>April</td>
<td>F.P.A. Coolen</td>
<td>Maximum Imprecision Related to Intervals of Measures and Bayesian Inference with Conjugate Imprecise Prior Densities</td>
</tr>
<tr>
<td>92-08</td>
<td>May</td>
<td>I.J.B.F. Adan, J. Wessels, W.H.M. Zijm</td>
<td>A Note on “The effect of varying routing probability in two parallel queues with dynamic routing under a threshold-type scheduling”</td>
</tr>
<tr>
<td>92-10</td>
<td>May</td>
<td>P. v.d. Laan</td>
<td>Subset Selection: Robustness and Imprecise Selection</td>
</tr>
<tr>
<td>92-11</td>
<td>May</td>
<td>R.J.M. Vaessens, E.H.L. Aarts, J.K. Lenstra</td>
<td>A Local Search Template (Extended Abstract)</td>
</tr>
<tr>
<td>92-12</td>
<td>May</td>
<td>F.P.A. Coolen</td>
<td>Elicitation of Expert Knowledge and Assessment of Imprecise Prior Densities for Lifetime Distributions</td>
</tr>
<tr>
<td>92-13</td>
<td>May</td>
<td>M.A. Peters, A.A. Stoorvogel</td>
<td>Mixed $H_2/H_{\infty}$ Control in a Stochastic Framework</td>
</tr>
<tr>
<td>Number</td>
<td>Month</td>
<td>Author</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>92-14</td>
<td>June</td>
<td>P.J. Zwietering, E.H.L. Aarts, J. Wessels</td>
<td>The construction of minimal multi-layered perceptrons: a case study for sorting</td>
</tr>
<tr>
<td>92-15</td>
<td>June</td>
<td>P. van der Laan</td>
<td>Experiments: Design, Parametric and Nonparametric Analysis, and Selection</td>
</tr>
<tr>
<td>92-16</td>
<td>June</td>
<td>J.J.A.M. Brands, F.W. Steutel, R.J.G. Wilms</td>
<td>On the number of maxima in a discrete sample</td>
</tr>
<tr>
<td>92-17</td>
<td>June</td>
<td>S.J.L. v. Eijndhoven, J.M. Soethoudt</td>
<td>Introduction to a behavioral approach of continuous-time systems part II</td>
</tr>
<tr>
<td>92-18</td>
<td>June</td>
<td>J.A. Hoogeveen, H. Oosterhout, S.L. van der Velde</td>
<td>New lower and upper bounds for scheduling around a small common due date</td>
</tr>
<tr>
<td>92-19</td>
<td>June</td>
<td>F.P.A. Coolen</td>
<td>On Bernoulli Experiments with Imprecise Prior Probabilities</td>
</tr>
<tr>
<td>92-20</td>
<td>June</td>
<td>J.A. Hoogeveen, S.L. van de Velde</td>
<td>Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing</td>
</tr>
<tr>
<td>92-21</td>
<td>June</td>
<td>J.A. Hoogeveen, S.L. van de Velde</td>
<td>Polynomial-time algorithms for single-machine bicriteria scheduling</td>
</tr>
</tbody>
</table>