Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Hoogeveen

Minimizing maximum earliness and maximum lateness on a single machine

Department of Operations Research, Statistics, and System Theory Report BS-R9001 February

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Minimizing maximum earliness and maximum lateness on a single machine

J.A. Hoogeveen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam
The Netherlands

A set of n jobs has to be scheduled on a single machine which can handle only one job at a time. Each job requires a given positive uninterrupted processing time p_i and has a given due date d_i. The problem is to find a schedule that minimizes a function of maximum earliness and maximum lateness that is nondecreasing in both arguments. We present $O(n^2 \log n)$ algorithms for the variant in which idle time is not allowed and for the special case in which the objective function is linear. We prove that the problem is $\mathcal{NP}$-hard if neither of these restrictions is imposed.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single machine scheduling, bicriteria scheduling, Pareto optimal points

1. Introduction

Suppose n independent jobs have to be scheduled on a single machine, which can handle only one job at a time. The machine is assumed to be continuously available from time 0 onwards. Job J_i $(i = 1, \ldots, n)$ requires a given positive uninterrupted processing time p_i and should ideally be completed at a given due date d_i. A *schedule* defines for each job J_i its completion time C_i such that the jobs do not overlap in their execution. Given a schedule S, the earliness and the lateness of job J_i are defined by $E_i = d_i - C_i$ and $L_i = C_i - d_i$, respectively. Correspondingly, the maximum earliness and maximum lateness are defined by $E_{\max} = \max_{1 \leqslant i \leqslant n} E_i$ and $L_{\max} = \max_{1 \leqslant i \leqslant n} L_i$, respectively. We consider the problem of finding a schedule S that minimizes the scheduling cost $F(S) = F(E_{\max}, L_{\max})$, where $F(\cdot, \cdot)$ is a given function that is nondecreasing in both arguments. We will also consider a variant of the problem in which idle time is not allowed. Extending standard notation, the first problem will be denoted by $1 \mid\mid F(E_{\max}, L_{\max})$ and the second by $1 \mid nmit \mid F(E_{\max}, L_{\max})$, where *nmit* denotes that machine idle time is forbidden.

Although the first bicriteria scheduling problem was already solved by Smith [1956], only a few bicriteria scheduling problems have been investigated since then. Most of these problems are concerned with minimizing a hierarchical type of objective function: the secondary criterion has to be minimized subject to the constraint that the primary criterion has to be minimal. Examples are minimizing the sum of completion time subject to minimal maximum lateness [Smith, 1956] and minimizing maximum lateness subject to minimum number of late jobs [Shantikumar, 1983]. Only a few of the papers on bicriteria scheduling consider simultaneous

Report BS-R9001
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

optimization, in which the criteria are transformed into a single composite objective function. An example hereof is minimizing the number of late jobs and maximum lateness simultaneously [Nelson et al., 1986]. Most contributions in the area of bicriteria scheduling concerns branch and bound algorithms. A notable exception is provided by Hoogeveen and Van de Velde [1990], who present an $O(n^2 \min\{n^2, \log \Sigma p_i\})$ time algorithm to solve $1||F(\gamma_{\max}, \Sigma p_i)$, where $\gamma_{\max} = \max\{\gamma_i \mid i = 1, \ldots, n\}$, such that all γ_i are nondecreasing in C_i. Furthermore, they present an $O(n^4)$ time algorithm to solve $1||\alpha E_{\max} + \Sigma p_i$, with $\alpha \leqslant 1$.

The organization of this paper is as follows. In Section 2 we repeat some basic theory. We also derive a dominance rule, which can be applied to both variants of the problem. In Section 3 we consider the case in which idle time is not allowed. In Section 4 we drop this constraint and analyze the general problem, which we prove to be $\mathcal{NP}$-hard in Section 5.

2. Basic concepts

The $1||F(E_{\max}, L_{\max})$ problem originates from $1||L_{\max}$ and $1|nmit|E_{\max}$, where the 'no idle time' constraint is added to $1||E_{\max}$ in order to avoid unbounded solutions. Both problems are solvable in $O(n \log n)$ time.

Earliest Due Date (EDD) Rule [Jackson, 1955]. *$L_{\max}$ is minimized by sequencing the jobs in order of nondecreasing due dates d_i.*

Minimum Slack Time (MST) Rule. *If no idle time is allowed, $E_{\max}$ is minimized by sequencing the jobs in order of nondecreasing values of $d_i - p_i$.*

The *MST* rule forms a generalization of the *EDD* rule (see Theorem 3). If both orderings coincide, then the corresponding job sequence solves both versions of $1||F(E_{\max}, L_{\max})$. However, in general both orderings will differ and it is unlikely that a single sequence minimizes both $E_{\max}$ and $L_{\max}$. Hence, in order to solve $1||F(E_{\max}, L_{\max})$ with or without idle time, we see no other way than to determine the set of feasible schedules that correspond to a Pareto optimal point with respect to the scheduling criteria $E_{\max}$ and $L_{\max}$.

Definition 1. A feasible schedule S is *Pareto optimal* with respect to the objective functions $f_1, \cdots, f_K$ if there is no feasible schedule $\bar{S}$, $\bar{S} \neq S$, such that $f_k(\bar{S}) \leqslant f_k(S)$ for all $k = 1, \ldots, K$, and $f_k(\bar{S}) < f_k(S)$ for at least one $k \in \{1, \ldots, K\}$.

Theorem 1. *Consider the composite objective function F with $F(S) = F(f_1(S), \ldots, f_K(S))$, where F is nondecreasing in all performance criteria f_k. There is a Pareto optimal schedule with respect to $f_1, \cdots, f_K$ that minimizes the function F.* □

It follows immediately from Theorem 1 that, if the number of Pareto optimal points is polynomially bounded in n and if all these points can be determined in polynomial time, then the function F can be minimized in polynomial time.

In order to find the set of Pareto optimal points, we solve the problem of minimizing $L_{\max}$ subject to $E_{\max} \leqslant E$, where E corresponds to the $E_{\max}$ value of a Pareto optimal point. This approach will be used in Section 3 and Section 4. In the remainder of this section, we present a

dominance rule that applies to both $1 \mid E_{\max} \leqslant E$, *nmit* $\mid L_{\max}$ and $1 \mid E_{\max} \leqslant E \mid L_{\max}$. We need a preliminary lemma.

LEMMA 1. *Consider an arbitrary schedule S. Let J_i and J_j be two jobs, where J_i is scheduled before J_j in S. If $E_j(S) > E_i(S)$, or equivalently, $L_i(S) > L_j(S)$, then $d_j - p_j > d_i$.*

PROOF. $E_j(S) > E_i(S)$ implies $d_j - C_j(S) > d_i - C_i(S)$. As J_i is scheduled before J_j, $C_j(S) \geqslant C_i(S) + p_j$. The combination of these two inequalities yields $d_j - p_j > d_i$. □

DOMINANCE RULE. *Let J_i and J_j be two arbitrary jobs. If there exist both an MST sequence and an EDD sequence in which J_i precedes J_j, so both $d_i - p_i \leqslant d_j - p_j$ and $d_i \leqslant d_j$, where at least one of the inequalities is strict, then there exists an optimal schedule for both $1 \mid E_{\max} \leqslant E$, nmit $\mid L_{\max}$ and $1 \mid E_{\max} \leqslant E \mid L_{\max}$ in which J_i precedes J_j.*

PROOF. The proof consists of showing that an optimal schedule S that does not satisfy the dominance rule can be transformed by applying interchanges (not necessarily adjacent) into a feasible schedule $\bar{S}$ that is optimal and that satisfies the dominance rule.

Consider an optimal schedule S which contains two jobs J_i and J_j that satisfy the conditions of the dominance rule, while J_j precedes J_i in S. Let J_i and J_j be chosen such that the jobs scheduled between J_i and J_j in S satisfy the order of the dominance rule. This implies for every job J_l, scheduled between J_j and J_i in S, that $d_j - p_j < d_l$ and $d_l - p_l < d_i$.

Consider the schedule $\bar{S}$, obtained by interchanging J_i and J_j. In order to prove that $\bar{S}$ is also an optimal feasible schedule, it suffices to prove the following two claims.

(1) $\bar{S}$ is feasible with respect to the release dates induced by the constraint $E_{\max} \leqslant E$.

(2) The lateness in $\bar{S}$ of J_j and of the jobs scheduled between J_i and J_j in S does not exceed $L_i(S) \leqslant L_{\max}(S)$.

Proof of (1). Suppose there is a job J_l in $\bar{S}$ scheduled between J_i and J_j, with $E_l(\bar{S}) > E$. As $E_i(\bar{S}) \leqslant E$, the earliness of J_l is larger than the earliness of J_i, while J_i precedes J_l. Application of Lemma 1 yields $d_l - p_l > d_i$, contradicting the assumption.

Proof of (2). Suppose there is a job J_l in $\bar{S}$ scheduled between J_i and J_j, with $L_l(\bar{S}) > L_i(S)$. As $d_i \leqslant d_j$, we have $L_i(S) = C_i(S) - d_i \geqslant C_i(\bar{S}) - d_j = C_j(\bar{S}) - d_j = L_j(\bar{S})$, so the lateness of J_l is larger than the lateness of J_j in schedule $\bar{S}$ while J_l precedes J_j. Application of Lemma 1 yields $d_j - p_j > d_l$, contradicting the assumption.

Because $d_i \leqslant d_j$ and $d_i - p_i \leqslant d_j - p_j$, the claims (1) and (2) also hold for the jobs J_i and J_j. The interchange argument can be repeated until a schedule is obtained that satisfies the dominance rule. This schedule is also feasible and optimal. □

3. PARETO OPTIMAL SCHEDULES IF NO IDLE TIME IS ALLOWED

3.1. *A polynomial algorithm for* $1 \mid E_{\max} \leqslant E$, *nmit* $\mid L_{\max}$

The constraint $E_{\max} \leqslant E$ implies that $d_i - C_i \leqslant E$ for every job J_i. This is equivalent to $S_i \geqslant d_i - p_i - E$, where S_i denotes the starting time of J_i $(i = 1, \ldots, n)$. Hence, $1 \mid E_{\max} \leqslant E$, *nmit* $\mid L_{\max}$ is identical to $1 \mid r_j = d_j - p_j - E$, *nmit* $\mid L_{\max}$. Although $1 \mid r_j$, *nmit* $\mid L_{\max}$ with general release dates is $\mathcal{NP}$-hard in the strong sense [Lenstra, Rinnooy

Kan and Brucker, 1977], we are able to present a polynomial algorithm for the special case that the release dates r_i are induced by the constraint $E_{\max} \leqslant E$ for an arbitrary value E of the upper bound on $E_{\max}$. The algorithm is based on the observation that, if two jobs J_i and J_j with $d_i < d_j$ can both be scheduled in the kth position of a partial schedule in which the first $k-1$ jobs have been fixed, then there exists an extension of this partial schedule that is optimal for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$ in which J_i precedes J_j.

ALGORITHM A.

Step 0. $T := 0$; $r_j := d_j - p_j - E$ for $j = 1, \ldots, n$; $k := 1$; $U := \{J_1, \ldots, J_n\}$; $V := \varnothing$.
{Initialization: T denotes the starting time of the job in the kth position}

Step 1. For each job $J_j \in U$: if $r_j \leqslant T$ then $V := V \cup \{J_j\}$ and $U := U \setminus \{J_j\}$.
{U denotes the set of unscheduled jobs that are not allowed to start at time T, V denotes the set of unscheduled jobs that are allowed to start at time T}

Step 2. If V is empty, then stop. Otherwise, determine the job with the smallest due date in the set V. If there are ties, then choose the job with the smallest release date. If there are still ties, then choose the job with the smallest index. Suppose job J_i is chosen. This job is scheduled in the kth position.

Step 3. $T := T + p_i$; $k := k + 1$; $V := V \setminus \{J_i\}$.

Step 4. If there are unscheduled jobs left, then go to 1.

THEOREM 2. *Algorithm A yields an optimal schedule for* $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$.

PROOF. Suppose that Algorithm A yields a schedule S which is not optimal. Let $\bar{S}$ be an optimal schedule that satisfies the dominance rule.

Compare S and $\bar{S}$, starting at the front. Suppose the first difference occurs in the kth position; let J_i be scheduled in the kth position in S and let J_j be scheduled in the kth position in $\bar{S}$. Because of the construction of the schedule S, $d_i \leqslant d_j$.

Let $\hat{S}$ be the schedule that results when J_i and J_j are interchanged in $\bar{S}$. It now suffices to prove the following claims in order to prove that S is an optimal schedule that is feasible with respect to the release dates induced by the constraint $E_{\max} \leqslant E$.

(1) $\hat{S}$ is feasible with respect to the release dates induced by the constraint $E_{\max} \leqslant E$.

(2) $\hat{S}$ is also optimal.

(3) $\hat{S}$ can be transformed into a new schedule $\bar{S}$ that is optimal, feasible with respect to the release dates and equal to S with respect to the first k positions, while $\bar{S}$ satisfies the dominance rule.

Proof of (1). Analogous to the proof of claim (1) in the dominance rule.

Proof of (2). Consider an arbitrary job J_l, scheduled between J_j and J_i in $\bar{S}$. J_j precedes J_l in $\bar{S}$, so because of the dominance rule, $d_j - p_j < d_l$. Furthermore, $d_i \leqslant d_j$ because of the choice of J_i in S. Suppose $L_l(\hat{S}) > L_i(\bar{S})$. As $d_i \leqslant d_j$, $L_i(\bar{S}) \geqslant L_j(\hat{S})$. Furthermore, J_l precedes J_j in $\hat{S}$, so application of Lemma 1 yields $d_j - p_j > d_l$, contradicting the assumption that $\bar{S}$ satisfies the dominance rule. This implies for each J_l, scheduled between J_j and J_i, that the lateness of J_l in $\hat{S}$ does not exceed the value of $L_{\max}(\bar{S})$. As $L_j(\hat{S}) \leqslant L_i(\bar{S})$, this completes the proof of (2).

Proof of (3). It is possible that there exists a job J_l, scheduled between J_j and J_i in $\bar{S}$, with $d_l > d_j$ and $d_i - p_i > d_l - p_l > d_j - p_j$, implying that $\hat{S}$, obtained from $\bar{S}$ by interchanging J_j and J_i, does not satisfy the dominance rule. However, it follows immediately from the proof of the dominance rule that $\hat{S}$ can then be adjusted to a new schedule $\bar{S}$ that is also feasible and optimal, that satisfies the dominance rule, and in which the first k jobs are the same as in S.

The interchange argument can be repeated until the schedule S and the newly obtained schedule $\bar{S}$ are the same. This proves that S, obtained by the application of Algorithm A, is an optimal schedule that is feasible with respect to the release dates induced by the constraint $E_{\max} \leqslant E$, and that satisfies the dominance rule. □

THEOREM 3. *Algorithm A solves* $1 \mid L_{\max} \leqslant L, nmit \mid E_{\max}$ *to optimality.*

PROOF. Consider an arbitrary instance $V_1 = \{d_{1,1}, p_{1,1}, \ldots, d_{n,1}, p_{n,1}, L\}$ of $1 \mid L_{\max} \leqslant L, nmit \mid E_{\max}$. Now construct the following instance $V_2 = \{d_{1,2}, p_{1,2}, \ldots, d_{n,2}, p_{n,2}, E\}$ for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$:

$$d_{i,2} = \sum_{j=1}^{n} p_j + p_i - d_{i,1} \text{ for } i = 1, \ldots, n,$$

$$p_{i,2} = p_{i,1} \text{ for } i = 1, \ldots, n,$$

$$E = L.$$

Suppose Algorithm A produces a schedule S as an optimal schedule for V_2. An optimal schedule $\bar{S}$ for V_1 is then obtained by reversing S, since

$$E_i(\bar{S}) = d_{i,1} - C_i(\bar{S}) = \sum_{j=1}^{n} p_j + p_i - d_{i,2} - C_i(\bar{S}) = C_i(S) - d_{i,2} = L_i(S).$$

This implies that $L_{\max}(S) \leqslant L$ if and only if $E_{\max}(\bar{S}) \leqslant E$, and S is optimal and feasible if and only if $\bar{S}$ is optimal and feasible. □

3.2. *Determination of Pareto optimal points with respect to* $E_{\max}$ *and* $L_{\max}$

We now present an algorithm to determine all values E of $E_{\max}$ that may correspond to a Pareto optimal point. Once such a value E is known, the corresponding value of $L_{\max}$ can be determined in $O(n \log n)$ time by solving the corresponding $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$ problem by Algorithm A. Furthermore we prove that the number of Pareto optimal points with respect to $E_{\max}$ and $L_{\max}$ is no more than n, and that at most n values E of $E_{\max}$ have to be considered to determine all Pareto optimal points. We first prove the following lemma.

LEMMA 2. *Consider an arbitrary job* J_k *in* S, *where* S *is the schedule constructed by Algorithm A to solve* $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$. *There are no two jobs* J_i *and* J_j, *scheduled before* J_k, *with a due date larger than* d_k.

PROOF. Suppose the opposite and assume that there are two such jobs J_i and J_j. Without loss of generality, let J_i be scheduled before J_j. Because of the construction of the schedule, job J_k may not be chosen by Algorithm A at time $C_i(S)$, so $d_k - p_k - E > C_i(S) \geqslant d_i - E$, implying $d_k - p_k > d_i$, so $d_k > d_i$, contradicting the assumption. □

Given an optimal schedule S for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$, obtained by Algorithm A, where E is an arbitrary value of the upper bound on $E_{\max}$, it is possible to decrease $L_{\max}$ only by interchanging two jobs that are not scheduled in *EDD* order. We prove that S can be partitioned into blocks that have the property that an interchange, necessary to decrease $L_{\max}$, can only take place within such a block. Partition the schedule S into blocks according to the following algorithm.

PARTITIONING ALGORITHM.

Step 0. Start at the beginning of the schedule.

Step 1. Let J_i be the first job in a block. Compare the due date of job J_i with the due date of its successors, until a job is found that does not have a smaller due date. Let this be J_k. The block contains job J_i and all its successors up to J_k.

Step 2. Proceed until the schedule has been completely partitioned into blocks.

PROPOSITION 1. *Let S be an optimal schedule for the $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$ problem obtained by Algorithm A, where E is an arbitrary upper bound on $E_{\max}$, with $E \geqslant E^{MST}$, where E^{MST} is equal to $E_{\max}(MST)$. Partition S into blocks, according to the Partitioning algorithm. Any block B has the following properties.*

(1) *If job J_i is the first job in B, then all jobs J_j in S with smaller due date, scheduled after J_i, also belong to block B.*

(2) *The jobs in B are scheduled in the following order: the job with the largest due date is scheduled first, the other jobs are scheduled in EDD order.*

PROOF. (1) Suppose that there exists a job J_j with $d_j < d_i$, scheduled after J_i, that does not belong to B. According to the Partitioning algorithm, there must exist a job J_l, scheduled between J_i and J_j, with $d_l > d_i > d_j$. But then, both J_i and J_l have larger due date and are scheduled before J_j, contradicting Lemma 2. This contradiction proves Property 1.

(2) Property 2 follows immediately from Lemma 2 and the Partitioning algorithm. □

THEOREM 4. *Let E_1 and E_2 be two arbitrary values of the upper bound on $E_{\max}$, with $E_1 \leqslant E_2$. Let S_1 and S_2 be the optimal schedules obtained by applying Algorithm A to $1 \mid E_{\max} \leqslant E_1, nmit \mid L_{\max}$ and $1 \mid E_{\max} \leqslant E_2, nmit \mid L_{\max}$, respectively. Partition S_1 into blocks according to the Partitioning algorithm. Let B be an arbitrary block of S_1, let T_1 and T_2 be the starting and completion time of block B in S_1, respectively. Then the jobs belonging to B are processed in S_2 during the interval $[T_1, T_2]$.*

PROOF. Consider the first block B of the schedule S_1. Let J_j be an arbitrary job that does not belong to B. As $E_2 \geqslant E_1$, it is feasible to schedule all jobs of B in the same position as in S_1. As J_j does not belong to B, its due date is at least as large as the due date of the first job in B and therefore Algorithm A will not choose J_j until all jobs that belong to B have been scheduled. This argument can be repeated for the second block in S_1 and so on, until only the last block remains. □

THEOREM 5. *Let S be an optimal schedule for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$ obtained by Algorithm A, where E is an arbitrary upper bound on $E_{\max}$. Let B be a block that contains a job J_i with $L_i = L_{\max}(S)$. In order to decrease $L_{\max}(S)$, it is necessary to increase E such that another job within block B can be scheduled first.*

PROOF. $L_{\max}(S)$ can only be decreased by decreasing the completion time of job J_i. Application of Theorem 4 implies that a decrease of the completion time of job J_i has to be achieved by changing its position within the block B. As all jobs in B, except the job in the first position, are scheduled in *EDD* order, another job has to be scheduled in the first position of B in order to change the scheduling order in B. □

Theorem 4 and 5 provide the basis for the algorithm we present for determining all Pareto optimal points. First we prove that the number of Pareto optimal points is no more than n.

THEOREM 6. *Let S_1 and S_2 be two schedules obtained by Algorithm A, which both correspond to a Pareto optimal point, (E_1, L_1) and (E_2, L_2), respectively. Suppose $E_1 < E_2$. Partition S_1 and S_2 into blocks by applying the Partitioning algorithm. The number of blocks in which S_1 has been partitioned is smaller than the number of blocks in which S_2 has been partitioned.*

PROOF. Application of Theorem 4 yields that S_2 has been partitioned in at least as many blocks as S_1. Furthermore, Theorem 5 implies that, in order to achieve a lower value of $L_{\max}$, at least one of the blocks in which S_1 has been partitioned, must have been split in at least two blocks in S_2. □

COROLLARY 6.1. *The number of Pareto optimal points is bounded by n, and this bound is tight.*

PROOF. The number of blocks is at most equal to the number of jobs. From Theorem 6 it follows immediately that there are no two different Pareto optimal points, such that the corresponding schedules, obtained by Algorithm A, are partitioned by the Partitioning algorithm, into the same number of blocks. This implies that the number of Pareto optimal points is bounded by n.

The following example shows that the bound is tight:

$$p_i = 1 \quad \text{for } i = 1, \ldots, n-1,$$

$$d_1 = 2,\ d_{i+1} = d_i + i + 2 \text{ for } i = 1, \ldots, n-2,\ d_n = p_n = d_{n-1} + 1. \quad \square$$

From Theorem 5 it follows immediately that a new Pareto optimal point can only be obtained by increasing the value E of the upper bound on $E_{\max}$ such that for every block B that contains a job J_i with $L_i = L_{\max}$ another job can be scheduled in the first position in B. This observation forms the basis for an algorithm that, given an optimal schedule S for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$, determines the next $E_{\max}$ value that corresponds to a Pareto optimal point.

ALGORITHM NEXT E

Step 0. Partition S into blocks, according to the Partitioning algorithm.

Step 1. Determine for each block B that contains a job J_i with $L_i(S) = L_{\max}(S)$ the new value of E that is necessary to schedule another job in B in the first position. If J_i is the only job in B, then $L_{\max}$ cannot be decreased. Stop.

Step 2. Choose the maximum of the new values found at Step 1. This maximum is the new value E.

The combination of all properties derived above leads to Algorithm B that determines all Pareto optimal points in $O(n^2 \log n)$ time. As the proof of this algorithm should be trivial by now, we omit it.

ALGORITHM B

Step 0. Solve $1 \mid\mid L_{\max}$; this yields L^{EDD} and solve $1 \mid nmit \mid E_{\max}$; this yields E^{MST}.

Step 1. Solve $1 \mid E_{\max} \leqslant E^{MST}, nmit \mid L_{\max}$ by applying Algorithm A; this leads to the first Pareto optimal point.

Step 2. If L^{EDD} has been reached, then go to 4. Otherwise, determine the next value E of the upper bound on $E_{\max}$ that corresponds to a Pareto optimal point by applying Algorithm Next E.

Step 3. Solve $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$; this leads to a new Pareto optimal point, go to 2.

Step 4. All Pareto optimal points have been determined. The $1 \mid nmit \mid F(E_{\max}, L_{\max})$ problem can be solved by computing the value of $F(E_{\max}, L_{\max})$ for all of these points.

4. PARETO OPTIMAL SCHEDULES IF IDLE TIME IS ALLOWED

We prove that the $1 \mid E_{\max} \leqslant E \mid L_{\max}$ problem for a given value E can be solved in $O(n^2 \log n)$ time. We show that the trade-off curve of $E_{\max}$ and $L_{\max}$, defined as the curve that connects all points (E,L) where L is the outcome of $1 \mid E_{\max} \leqslant E \mid L_{\max}$, is piecewise linear with gradient alternately -1 and 0. As every point on the trade-off curve with negative gradient is not dominated, the number of Pareto optimal points with respect to $E_{\max}$ and $L_{\max}$ is no longer polynomially bounded by n. Hence, we do not know how to solve $1 \mid\mid F(E_{\max}, L_{\max})$ in polynomial time, unless we add the constraint that $F(E_{\max}, L_{\max})$ is linear.

Consider the $1 \mid E_{\max} \leqslant E \mid L_{\max}$ problem. As $L_{\max}$ is a regular performance measure, we may restrict our attention to *active* schedules. An active schedule is a schedule in which no job can start earlier without increasing the completion time of at least one other job.

The possibility of inserting idle time in a schedule has an important consequence. Consider a partial schedule without idle time in which the first $k-1$ jobs have been fixed. Instead of scheduling the job selected by Algorithm A in the kth position, it now may be advantageous to wait until another job with a smaller due date becomes available. So, if we stick to a schedule without idle time, it may be advantageous to increase the upper bound E on the earliness of the job that is to be scheduled in the kth position. The increase of the upper bound E should be considered for every position separately. Let $J_{[i]}$ denote the job in the ith position in the schedule and let K_i denote the upper bound on the earliness of job $J_{[i]}$. We analyze the $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$ problem instead of the $1 \mid E_{\max} \leqslant E \mid L_{\max}$ problem. As the sequence

without idle time has to be made feasible with respect to $E_{\max} \leqslant E$ by inserting idle time, there is no use in considering vectors $K = (K_1, \ldots, K_n)$ that do not satisfy $K_i \leqslant K_{i+1}$ for $i = 1, \ldots, n-1$.

Let K be a vector of upper bounds. Let S be the corresponding sequence, S is obtained by solving $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$. Let $S(E)$ be the corresponding active schedule that is feasible with respect to the constraint $E_{\max} \leqslant E$. We will solve $1 \mid E_{\max} \leqslant E \mid L_{\max}$ for every value E by determining the set of vectors $\{K^1, \ldots, K^m\}$ such that for every value E one of the corresponding schedules $S^1(E), \ldots, S^m(E)$ solves $1 \mid E_{\max} \leqslant E \mid L_{\max}$. The vectors $K^1, \ldots, K^m$ have to be minimal, so there is no vector $\bar{K}$ smaller than K^j such that the schedule S^j is feasible with respect to $E_{[i]} \leqslant \bar{K}_i$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Therefore, $K_j^k = \max\{E_{[i]}(S^j) \mid i = 1, \ldots, k\}$ for $j = 1, \ldots, m$ and $k = 1, \ldots, n$.

The $1 \mid E_{[i]} \leqslant K^i, nmit \mid L_{\max}$ problem is a straightforward generalization of the $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$ problem. Therefore, an optimal sequence for the $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$ problem has almost the same properties and can be determined in the same fashion as an optimal sequence for $1 \mid E_{\max} \leqslant E, nmit \mid L_{\max}$.

LEMMA 3. *Given a vector* $K = (K_1, \ldots, K_n)$ *with* $K_i \leqslant K_{i+1}$ *for* $i = 1, \ldots, n-1$, *there exists an optimal schedule* S *for* $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$ *that satisfies the dominance rule.*

PROOF. As $K_i \leqslant K_{i+1}$ for $i = 1, \ldots, n-1$, the proof of Lemma 3 is analogous to the proof of the dominance rule. □

We now adjust Algorithm A such that it can be applied to solve $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$.

ALGORITHM AA.

Step 0. $T := 0$; $k := 1$; $U := \{J_1, \ldots, J_n\}$; $V := \varnothing$.

{Initialization: T denotes the starting time of the job in the kth position}

Step 1. For each job $J_j \in U$: if $d_j - p_j - K_k \leqslant T$ then $V := V \cup \{J_j\}$ and $U := U \setminus \{J_j\}$.

{U denotes the set of unscheduled jobs that are not allowed to start at time T, V denotes the set of unscheduled jobs that are allowed to start at time T}

Step 2. If V is empty, then stop. Otherwise, determine the job with the smallest due date in the set V. If there are ties, then choose the job with the smallest value of $d_j - p_j$. If there are still ties, then choose the job with the smallest index. Suppose job J_i is chosen. This job is scheduled in the kth position.

Step 3. $T := T + p_i$; $k := k + 1$; $V := V \setminus \{J_i\}$.

Step 4. If there are unscheduled jobs left, then go to 1.

THEOREM 7. *Algorithm AA yields an optimal schedule for* $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$.

PROOF. The proof of Theorem 7 is analogous to the proof of Theorem 2. □

We now come to the point of how to determine the set of upper bound vectors $\{K^1, \ldots, K^m\}$. We will follow the same approach as in Section 3.2.

Consider a vector K from the set $\{K^1, \ldots, K^m\}$. Let k and l be such that $K_{k-1} < K = \cdots = K_l < K_{l+1}$. The set of positions $\{j, \ldots, k\}$ is called a *group of positions*, the set of jobs $\{J_{[k]}, \ldots, J_{[l]}\}$ is called a *group* of jobs. Note that the corresponding schedule $S(E)$ contains no idle time within a group of jobs. Define the maximum lateness within the group G as $L(G)_{\max} = \max\{L_{[i]}(S) \mid J_{[i]} \in G\}$ and define $K(G)$ as the K-value of a position within G.

PROPOSITION 2. *Let S^j be the sequence obtained by applying Algorithm AA to $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$, where $K^j \in \{K^1, \ldots, K^m\}$. Partition S_j in groups, let G_1 and G_2 be two arbitrary groups of jobs, where G_1 is completed before G_2. Let $J_{[i]}$ and $J_{[k]}$ be two arbitrary jobs in G_1 and G_2, respectively. Then $d_{[i]} < d_{[k]}$.*

PROOF. Let $J_{[l]}$ be the first job in G_2; therefore $J_{[l]}$ must start at time $d_{[l]} - p_{[l]} - K^j_{[l]}$. The starting time of $J_{[l]}$ is larger than or equal to the completion time of $J_{[i]}$, which is at least equal to $d_{[i]} - K^j_{[i]} > d_{[i]} - K^j_l$. Therefore, $d_{[l]} - p_{[l]} > d_{[i]}$, and as S^j satisfies the dominance rule stated in Lemma 3, $d_{[j]} > d_{[l]} - p_{[l]} > d_{[i]}$. □

Proposition 2 enables us to state Theorem 8, the analogue of Theorem 4.

THEOREM 8. *Let K^j and K^k be two arbitrary vectors from the set $\{K^1, \ldots, K^m\}$. Let K^k be the larger of the two. Let S^j and S^k be two optimal sequences obtained by applying Algorithm AA to $1 \mid E_{[i]} \leqslant K^j_i, nmit \mid L_{\max}$ and $1 \mid E_{[i]} \leqslant K^k_i, nmit \mid L_{\max}$, respectively. Partition S^j and S^k in groups. Let G_1 and G_2 be two arbitrary groups of jobs in S^j, where G_1 is completed before G_2 in S^j. Let $J_{[a]}$ and $J_{[b]}$ be two arbitrary jobs in G_1 and G_2, respectively. Then $J_{[a]}$ precedes $J_{[b]}$ in S^k.*

PROOF. The proof follows from Proposition 2 and the way the jobs are chosen in Algorithm AA. □

From Theorem 8 it follows immediately that, if we start with a vector K for the upper bound on $E_{[i]}$, then the only way to decrease $L(G)_{\max}$ is to increase the value of the upper bound for the whole group G or for a part of the group. This observation provides the basis for the following algorithm.

ALGORITHM NEXT K.

Step 0. Let K be a given vector of upper bounds on $E_{[i]}$. Let S be the sequence obtained by applying Algorithm AA to $1 \mid E_{[i]} \leqslant K_i, nmit \mid L_{\max}$.

Step 1. Let G be the first group in the schedule that attains $\max\{L(G)_{\max} + K(G)\}$. Partition this group of jobs into blocks, by applying the Partitioning algorithm.

Step 2. Determine the set of blocks $\mathfrak{B}$ in G that contain a job J_i with $L_i(S) = L(G)_{\max}$.

Step 3. Determine for each block B in $\mathfrak{B}$ how much the upper bound $K(G)$ has to be increased in order to let another job within B be scheduled in the first position in B. Let this value be equal to $K(B)$. If B consists of a single job, then $K(B) = \infty$ and hence, $L(G)_{\max}$ cannot be decreased. Stop.

Step 4. The next vector of upper bounds K can be computed from the old upper bound vector as follows. Let the first block in $\mathfrak{B}$ start in the $(k+1)$th position. The first k elements stay the same. Now consider the remaining positions in G, suppose this are the positions $k+1, \ldots, l$. The new upper bound value K_{i+1} becomes equal to K_i, unless a block $B \in \mathfrak{B}$, with $K(B) > K_i$ starts at position $i+1$. The elements of the new upper bound vector

K, corresponding to positions after G become equal to the maximum of K_l and their old value.

Because every group of jobs G has the same K-value for every job J_i in G, the correctness of this algorithm follows from Theorem 5.

Note that K_i^j cannot be smaller than K_i^{j+1} $(i = 1, \ldots, n; j = 1, \ldots, m-1)$, otherwise $L_{\max}(S^{j+1})$ will be larger than $L_{\max}(S^j)$.

We are now able to formulate an algorithm to determine all vectors $K^1, \ldots, K^m$ and the corresponding sequences $S^1, \ldots, S^m$. Define the vector K^{MST} such that $K_i^{MST} = \max\{E_{[j]}(MST) \mid j = 1, \ldots, i\}$ for $i = 1, \ldots, n$.

Algorithm BB.

Step 0. Determine the vector K^{MST}; $l := 1$; $K^l := K^{MST}$.

Step 1. Solve $1 \mid E_{[i]} \leqslant K_i^l, nmit \mid L_{\max}$ by applying Algorithm AA, this yields sequence S^l.

Step 2. $l := l + 1$. Compute the next vector K^l by applying Algorithm Next K. If $K < \infty$, then go to 1.

Step 3. All vectors $K \in \{K^1, \ldots, K^m\}$ have been determined.

The set of vectors $\{K^1, \ldots, K^m\}$ can be screened to remove all vectors that lead to dominated sequences. A sequence S^{j+1} is dominated by sequence S^j if $L_{\max}(S^j) \leqslant L_{\max}(S^{j+1})$.

THEOREM 9. *Let K^j and K^k be two arbitrary vectors from the set $\{K^1, \ldots, K^m\}$. Let K^k be the larger of the two. Let S^j and S^k be the optimal sequences obtained by applying Algorithm AA to $1 \mid E_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ and $1 \mid E_{[i]} \leqslant K_i^k, nmit \mid L_{\max}$, respectively. Partition S^j and S^k into blocks according to the Partitioning algorithm. Then S^k is partitioned into more blocks than S^j so, as the number of blocks is bounded by n, Algorithm BB computes at most n schedules.*

PROOF. The proof is analogous to the proof of Theorem 6 and Corollary 6.1. □

Let $S^j(E)$ be the active schedule obtained by inserting idle time in S^j to make $S^j(E)$ feasible with respect to the constraint $E_{\max} \leqslant E$, for $j = 1, \ldots, m$. Let $J_{[i]}$ be an arbitrary job in $S^j(E)$. We now have that $L_{[i]}(S^j(E)) = L_{[i]}(S^j) + \max\{0, K_i^j - E\}$, so the trade-off curve of $L_{\max}(S^j(E))$ and E forms a continous, piecewise linear step-function with gradient alternately -1 and 0, for $j = 1, \ldots, m$. Furthermore, the number of breakpoints in every trade-off curve is no more than n. These trade-off curves can be combined to one trade-off curve by choosing for each value of E the minimum value of $L_{\max}(S^j(E))$ for $j = 1, \ldots, m$. The number of breakpoints in this trade-off curve is no more than $mn \leqslant n^2$. As $1 \mid E_{\max} \leqslant E \mid L_{\max}$ is solved by one of the schedules $S^1(E), \ldots, S^m(E)$ for every value of E, the trade-off curve derived above is exactly equal to the trade-off curve of $L_{\max}$ and E, where $L_{\max}$ is equal to the outcome of the $1 \mid E_{\max} \leqslant E \mid L_{\max}$ problem.

5. Solving the $1||F(E_{\max}, L_{\max})$ problem is $\mathcal{NP}$-hard

In this section we prove that $1||F(E_{\max}, L_{\max})$ is $\mathcal{NP}$-hard in the strong sense. First, we need to prove that the problem of minimizing an arbitrary function $f(x)$, where x belongs to an arbitrary set P of integers, is $\mathcal{NP}$-hard, by showing that the corresponding decision problem is $\mathcal{NP}$-complete. The reduction is from the Hamiltonian circuit problem [Schrijver, 1989].

Hamiltonian circuit problem [Garey and Johnson, 1979]: Given a graph $G = (V,H)$, does G contain a Hamiltonian circuit?

We start by describing a reduction from the Hamiltonian circuit problem to the problem of minimizing an arbitrary function $f(x)$, where x belongs to an arbitrary set P of integers. Let $G = (V,H)$ be an arbitrary graph and let the edges in H be numbered $1, \ldots, |H|$, where $|H|$ denotes the cardinality of H. Define $P = \{0, \ldots, 2^{|H|}\}$. Every integer $x \in P$ can be described by $|H|$ zeros and ones, by using binary encoding. Further, define for every $x \in P$ a subset $P_x \subset H$ in the following way: the ith edge in H belongs to P_x if and only if the ith digit in the binary representation of x is equal to 1. Now we define the following function $f(x)$.

$$f(x) = \begin{cases} 0 & \text{if } |P_x| = n \text{ and if } P_x \text{ is Hamiltonian,} \\ 1 & \text{otherwise.} \end{cases}$$

Clearly, the value of $f(x)$ can be established in every point x in polynomial time.

Theorem 10. *Given a set of integers S and a nonnegative integer y, the problem of deciding whether there exists an integer $x \in S$ with $f(x) \leqslant y$ is $\mathcal{NP}$-complete.*

Proof. The decision problem is clearly in $\mathcal{NP}$. For any given instance of the Hamiltonian circuit problem, we construct a set of integers S and a function $f(x)$ as described above, and set $y = 0$. This reduction requires polynomial time. The decision problem will be answered affirmatively if and only if the graph G contains a Hamiltonian circuit. □

Theorem 11. *The $1||F(E_{\max}, L_{\max})$ problem is $\mathcal{NP}$-hard in the strong sense.*

Proof. The trade-off curve is piecewise linear with gradient -1 and 0 alternately. This implies that the number of Pareto optimal points with respect to the criteria $E_{\max}$ and $L_{\max}$ is unbounded, as every value $E \in \mathbb{Z}$, with E not larger than the E-value of the first breakpoint, corresponds to a Pareto optimal point. Therefore, we are able to select $2^{|H|}$ consecutive Pareto optimal points (with H equal to the edge set of an arbitrary graph) and therefore, we can carry out the reduction from the Hamiltonian Circuit problem, with $F(E_{\max}, L_{\max}) = f(E - c)$, where c is such that $0 \leqslant E - c < 2^{|H|}$ for every selected E-value. □

Note that if we impose the restriction that $E_{\max}$ has to be nonnegative, then only a pseudo-polynomial number of Pareto optimal points remains, so solving $1||F(E_{\max}, L_{\max})$ takes at most pseudo-polynomial time. Suppose that a polynomial algorithm exists for this restricted problem. In that case, given a graph $G = (V,H)$, all processing times can be multiplied by a factor $O(2^{|H|})$, after which we can select $2^{|H|}$ consecutive Pareto optimal points (the idle time

can still be changed by one unit at a time) and we can carry out the reduction as described above. Therefore, even in case $E_{\max}$ is bounded from below there is no polynomial algorithm for $1||F(E_{\max}, L_{\max})$, unless $\mathcal{P}=\mathcal{NP}$. Further, note that the reduction from the Hamiltonian circuit problem is not polynomial anymore, when $E_{\max}$ is assumed to be nonnegative and therefore, we cannot conclude that this special case of $1||F(E_{\max}, L_{\max})$ is $\mathcal{NP}$-hard in the strong sense.

However, if we consider linear objective functions $F(E_{\max}, L_{\max}) = \alpha_1 E_{\max} + \alpha_2 L_{\max}$, with $\alpha_1, \alpha_2 \geqslant 0$, then the situation is much brighter. If $\alpha_1 > \alpha_2$, then the optimum cost value will be equal to $-\infty$, otherwise an optimum is found in one of the breakpoints. As every breakpoint can be found in constant time, $1||\alpha_1 E_{\max} + \alpha_2 L_{\max}$ can be solved in $O(n^2 \log n)$ time, the time needed to determine the schedules $S^1, \ldots, S^m$.

ACKNOWLEDGEMENT

The author would like to thank Jan Karel Lenstra for his helpful comments.

REFERENCES

M.R. GAREY, D.S. JOHNSON (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

J.A. HOOGEVEEN, S.L. VAN DE VELDE (1990). *Some results on single machine multi criteria scheduling*, Working Paper, Centre for Mathematics and Computer Science, Amsterdam.

J.R. JACKSON (1955). *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, University of California, Los Angeles.

J.K. LENSTRA, A.H.G. RINNOOY KAN, P. BRUCKER (1977). Complexity of machine scheduling problems. *Ann. Discrete Math. 1*, 343-362.

R.T. NELSON, R.K. SARIN, R.L. DANIELS (1986). Scheduling with multiple performance measures: the one-machine case. *Management Science 32*, 464-479.

A. SCHRIJVER (1989). Private communication.

J.G. SHANTHIKUMAR (1983). Scheduling n jobs on one machine to minimize the maximum tardiness with minimum number tardy. *Computers and Operations Research 10*, 255-266.

W.E. SMITH (1956). Various Optimizers for single-stage production. *Naval Research Logistics Quarterly 1*, 59-66.