Scheduling with target start times

J.A. Hoogeveen
Department of Mathematics and Computing Science
Eindhoven University of Technology
E-mail: slam@win.tue.nl

S.L. van de Velde
Department of Mechanical Engineering
University of Twente
E-mail: s.l.vandevelde@wb.utwente.nl

Abstract

We address the single-machine problem of scheduling n independent jobs subject to target start times. Target start times are essentially release times that may be violated at a certain cost. The goal is to minimize an objective function that is composed of total completion time and maximum promptness, which measures the observance of these target start times. We show that in case of a linear objective function the problem is solvable in $O(n^4)$ time if preemption is allowed or if total completion time outweighs maximum promptness.

1 Introduction

A production company has to deal with the traditional conflict between internal and external efficiency of the production. *Internal efficiency* is the efficient use of the scarce resources. It results in a cost reduction and hence in possibly more competitive prices or higher profits. *External efficiency* is achieved by meeting the conditions superimposed by external relations. Clients, for instance, insist on product quality, short delivery times, and in-time delivery, among other things. Compromising product quality is playing with fire, but many a company tries to get away with late deliveries. After all, a good due-date performance may be achieved only in case of putting work out, overwork, frequent setups, or high setup costs. Unfortunately, many companies do not realize that a better planning may accomplish the same. This type of external efficiency, between the company and its clients, is actually *downstream*; it is the extent by which the company successfully copes with the requirements on the *demand* side.

We also distinguish *upstream* external efficiency. This is the extent by which the company successfully copes with the conditions on the *supply* side. A company, for instance, negotiates on the prices and delivery times of raw material. In order to achieve a higher internal efficiency, but especially a better due date performance, it may be worthwhile to pay a higher price to get the raw material sooner.

There exist several single-machine scheduling models of the trade-off between internal and downstream external efficiency. Van Wassenhove and Gelders (1980), for instance, consider a model for making the trade-off between work-in-process inventories and due date performance; see also Hoogeveen and Van de Velde (1995). Schutten, Van de Velde, and Zijm (1996) consider a batching problem for balancing out utilizing machine capacity against due date performance. Single-machine problems seem to be oversimplified models, but the study of these models makes sense, if we think of a company as a single-machine shop, or if there is a single bottleneck. What is more, single-machine models serve as building-blocks for solving complex scheduling problems.

In this paper, we study a single-machine scheduling model for striking a rational balance between internal and upstream external efficiency. Our model specification is 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 processing during a positive time p_j and has a target start time s_j. Without loss of generality, we assume that the processing times and target start times are integral. A *schedule* σ specifies for each job when it is executed while observing the machine availability constraints; hence, a schedule σ defines for each job J_j its start time $S_j(\sigma)$ and its completion time $C_j(\sigma)$. The *promptness* $P_j(\sigma)$ of job J_j is defined as $P_j(\sigma) = s_j - S_j(\sigma)$, and the maximum promptness is defined as $P_{\max}(\sigma) = \max_{1 \leq j \leq n} P_j(\sigma)$. We note that the maximum promptness $P_{\max}(\sigma)$ equals the *maximum earliness* $E_{\max}(\sigma) = \max_{1 \leq j \leq n}(d_j - C_j(\sigma))$ if each J_j has a due date d_j for which $s_j = d_j - p_j$ and if interruption of job processing is not allowed.

The problem we consider is to schedule the jobs so as to minimize total completion time $\sum_{j=1}^n C_j$ and maximum promptness $P_{\max}$ simultaneously. Total completion time $\sum_{j=1}^n C_j$ is a measure of the work-in-process inventories as well as the average leadtime. Hence, it is a performance measure for internal efficiency as well as downstream external efficiency.

Maximum promptness measures the observance of target start times. If it is positive, then it signals an inefficiency: at least one job is scheduled to start before its target start time. Generally, this is possible only if we are willing to pay a penalty. In case the target start times are derived from the delivery times of raw material, then this penalty is actually the price of a speedier delivery. In case the target start times are derived from the completion times of the parts in the preceding production stage, then this penalty may be an overwork bonus to expedite the production. If the maximum promptness is negative, then it signals a slack, which implies that we may increase the deadlines that are used in the preceding production stage.

It is important to realize that the target start times are actually *release times* that may be violated at a certain cost. In this sense, our problem comes close to the well-studied single-machine problem of minimizing total completion time subject to release times; see for instance Lenstra, Rinnooy Kan, and Brucker (1977) and Ahmadi and Bagchi (1990).

We now give a formal specification of our objective function. We associate with each schedule σ a point $(\sum_{j=1}^n C_j(\sigma), P_{\max}(\sigma))$ in $\Re^2$ and a value $F(\sum_{j=1}^n C_j(\sigma), P_{\max}(\sigma))$. The function $F : \Omega \rightarrow \Re$, where Ω denotes the set of all feasible schedules, is a given composite objective function that is nondecreasing in either of its arguments; this implies that for any two schedules σ and π with $\sum_{j=1}^n C_j(\sigma) \leq \sum_{j=1}^n C_j(\pi)$ and $P_{\max}(\sigma) \leq P_{\max}(\pi)$ we have that $F(\sum_{j=1}^n C_j(\sigma), P_{\max}(\sigma)) \leq F(\sum_{j=1}^n C_j(\pi), P_{\max}(\pi))$. Our problem is then formulated as

$$\min\{F(\sum_{j=1}^{n} C_j(\sigma), P_{\max}(\sigma)) \mid \sigma \in \Omega\}.$$

Extending the three-field notation scheme of Graham, Lawler, Lenstra, and Rinnooy Kan (1979), we denote this problem by $1||F(\sum_{j=1}^n C_j, P_{\max})$. The special case in which the function F is linear is denoted by $1||\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$, where $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.

In comparison to single-criterion problems, there are few papers on multicriteria scheduling problems. We refer to Dileepan and Sen (1988) and Hoogeveen (1992) for an overview of problems, polynomial algorithms, and complexity results.

This paper is organized as follows. In Section 2, we make some general observations and outline a generic strategy for solving the $1||F(\sum_{j=1}^n C_j, P_{\max})$ problem. We also point out that $1||F(\sum_{j=1}^n C_j, P_{\max})$ as well as its preemptive version $1|pmtn|F(\sum_{j=1}^n C_j, P_{\max})$, in which jobs may be interrupted and resumed later on, are $\mathcal{NP}$-hard in the strong sense. In Section 3, we consider the linear variant $1|pmtn|\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$. Our main results are that $1|pmtn|\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$ and, in the case that $\alpha_1 \geq \alpha_2$, also $1||\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$ are solvable in $O(n^4)$ time.

2 General observations

The fundamental question is whether the $1||F(\sum_{j=1}^n C_j, P_{\max})$ problem is solvable in polynomial time for any given function F that is nondecreasing in its arguments. The first observation we make is that this is so, if we can identify all 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 $(\sum_{j=1}^n C_j, P_{\max})$ if there exists no feasible schedule π with either $\sum_{j=1}^n C_j(\pi) \leq \sum_{j=1}^n C_j(\sigma)$ and $P_{\max}(\pi) < P_{\max}(\sigma)$, or $\sum_{j=1}^n C_j(\pi) < \sum_{j=1}^n C_j(\sigma)$ and $P_{\max}(\pi) \leq P_{\max}(\sigma)$.*

Once the *Pareto optimal set*, that is, the set of all schedules that are Pareto optimal with respect to the functions $(\sum_{j=1}^n C_j, P_{\max})$, has been determined, the $1||F(\sum_{j=1}^n C_j, P_{\max})$ problem can be solved for any function F by computing the cost of each Pareto optimal point and taking the minimum. Hence, if each Pareto optimal schedule can be found in polynomial time and the number of Pareto optimal schedules is polynomially bounded, then the problem is solvable in polynomial time.

We start with analyzing the two single-criterion problems that are embedded within $1||F(\sum_{j=1}^n C_j, P_{\max})$, that is, $1||P_{\max}$ and $1||\sum_{j=1}^n C_j$. The $1||P_{\max}$ problem is clearly meaningless, since we can improve upon each solution by inserting extra idle time at the beginning of the schedule. Hence, we impose the restriction that machine idle time before the processing of any job is prohibited, that is, all jobs are to be scheduled in the interval $[0, \sum_{j=1}^n p_j]$. It is easily checked that in case of a given overall deadline $D > \sum_{j=1}^n p_j$ the optimal schedule is obtained by inserting $D - \sum_{j=1}^n p_j$ units of idle time before the start of the first job. In the three-field notation scheme, the no machine idle time constraint is denoted by the acronym *nmit* in the second field. The $1|nmit|P_{\max}$ problem is solved by sequencing the jobs in order of *non-decreasing target start times s_j*. The $1||\sum_{j=1}^n C_j$ problem is solved by sequencing the jobs in order of non-decreasing processing times p_j (Smith, 1956). Let now $MTST$ be an optimal schedule for the $1|nmit|P_{\max}$ problem in which ties are settled to minimize total completion time; $MTST$ is the abbreviation of minimum target start time. In addition, let SPT be an optimal schedule for the $1||\sum_{j=1}^n C_j$ problem, in which ties are settled to minimize maximum promptness; SPT is the abbreviation of shortest processing time. It then follows that $P_{\max}^* \leq P_{\max}(\sigma) \leq P_{\max}(SPT)$ and $\sum_{j=1}^n C_j^* \leq \sum_{j=1}^n C_j(\sigma) \leq \sum_{j=1}^n C_j(MTST)$ for any Pareto optimal schedule σ, where $P_{\max}$ and $\sum_{j=1}^n C_j^*$ denote the outcome of the respective single-criterion problems.

Consider any Pareto optimal schedule σ; let $(P_{\max}(\sigma), \sum_{j=1}^n C_j(\sigma))$ be the corresponding Pareto optimal point. By definition, σ solves the problems $1|P_{\max} \leq P_{\max}(\sigma)|\sum_{j=1}^n C_j$ and

$1|\sum_{j=1}^n C_j \leq \sum_{j=1}^n C_j(\sigma)|P_{\max}$; the notation $P_{\max} \leq P_{\max}(\sigma)$ in the second field means that we impose $P_{\max} \leq P_{\max}(\sigma)$ as an extra constraint that each feasible schedule has to satisfy. Hence, if we know some $P_{\max}$ value P that may correspond to a Pareto optimal point, then we can determine the corresponding schedule σ and $\sum_{j=1}^n C_j$ value by solving $1|P_{\max} \leq P|\sum_{j=1}^n C_j$. Since any given value $P_{\max}$ induces for each job J_j a release date $r_j = \max\{0, s_j - P_{\max}\}$, we have to solve a problem of the form $1|r_j|\sum_{j=1}^n C_j$. A generic strategy for solving the bicriteria problem is then to solve this type of problem for all $P_{\max}$ values that may correspond to a Pareto optimal point and evaluate the function F for all the resulting combinations $(P_{\max}, \sum_{j=1}^n C_j)$. Lenstra, Rinnooy Kan, and Brucker (1977), however, show that the $1|r_j|\sum_{j=1}^n C_j$ problem is $\mathcal{NP}$-hard in the strong sense.

We therefore make the additional assumption that preemption of jobs is allowed, that is, the execution of any job may be interrupted and resumed later on. This assumption implies a crucial relaxation of the original problem; it has both positive and negative aspects. To start with the positive part: we can apply the generic approach now, since the $1|pmtn, r_j|\sum_{j=1}^n C_j$ 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 $P_{\max} \geq P_{\max}^*$. The disadvantage is that we lose the equivalence that existed between the maximum promptness criterion and the maximum earliness criterion in case $s_j = d_j - p_j$. This is so, since a given value $E_{\max}$ induces an earliest completion time for each job, not a release date.

Another crucial issue with respect to the applicability of the generic approach concerns the number of Pareto optimal points. Unfortunately, this number can grow arbitrarily large in general, since each value $P_{\max} \leq P_{\max}(SPT)$ corresponds to a Pareto optimal point, as we are allowed to preempt at any point in time, not just at the integral points. Seemingly, this is another disadvantage of allowing preemption, but this problem complicates the nonpreemptive version as well, since idle time can be inserted in any amount. The above implies that we can obtain a series of 2^n consecutive Pareto optimal points with $P_{\max}$ values that are multiples of 2^{-n}. Using the result by Schrijver (see Hoogeveen, 1996) that the problem of minimizing an arbitrary function $F(x, y)$ that is nondecreasing in both arguments over 2^n consecutive integral y values is $\mathcal{NP}$-hard in the strong sense, we conclude that $1|pmtn|F(\sum_{j=1}^n C_j, P_{\max})$ and $1||F(\sum_{j=1}^n C_j, P_{\max})$ are $\mathcal{NP}$-hard in the strong sense.

3 The linear variant $1|pmtn|\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$

To deal with this infinite number of Pareto optimal points, we assume from now on that the composite objective function is linear; we can then limit ourselves to the subset of the set of Pareto optimal schedules that contains an optimal solution to the $1|pmtn|\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$ problem for *any* $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. We define this set as the set of *extreme* schedules.

Definition 2 *A schedule $\sigma \in \Omega$ is extreme with respect to $(\sum_{j=1}^n C_j, P_{\max})$ if it corresponds to a vertex of the lower envelope of the Pareto optimal set for $(\sum_{j=1}^n C_j, P_{\max})$.*

If the extreme set can be found in polynomial time and if its cardinality is polynomially bounded, then the $1||\alpha_1 \sum_{j=1}^n C_j + \alpha_2 P_{\max}$ problem is solved in polynomial time by computing the cost of each extreme point and taking the minimum.

We start by analyzing the special case in which machine idle time before the processing of any job is prohibited; we later show that any instance of the general problem can be dealt with by reformulating it as an instance of the problem with no machine idle time allowed.

3.1 No machine idle time allowed

Recall that if machine idle time is not allowed, then all jobs are processed in the interval $[0, \sum_{j=1}^{n} p_j]$. Hence, we only have to consider $P_{\max}$ values in the interval $[P^*_{\max}, P_{\max}(SPT)]$, and for each $P_{\max}$ value P in this interval, Baker's algorithm provides an optimal schedule for the corresponding $1|r_j, pmtn|\sum_{j=1}^{n} C_j$ problem that does not contain idle time; let $\sigma(P)$ denote this schedule and let $(P, \sum_{j=1}^{n} C_j(\sigma(P)))$ denote the point in $\Re^2$ corresponding to it.

The problem is of course to distinguish between an extreme schedule and an ordinary Pareto optimal schedule. By definition, the schedule $\sigma(P_{\max})$ is extreme if increasing $P_{\max}$ by some $\epsilon > 0$ yields a smaller decrease in $\sum_{j=1}^{n} C_j$ than a decrease of $P_{\max}$ by the same amount ϵ would cost.

To illustrate the impact of an increase of $P_{\max}$, consider the following two-job example with $p_1 = 10$, $p_2 = 5$, $s_1 = 0$, and $s_2 = 10$. We have that $P^*_{\max} = 0$ and the corresponding $\sum_{j=1}^{n} C_j$ value amounts to 25. If we increase $P_{\max}$, nothing happens until it becomes advantageous to preempt job 1; this is the case for $P_{\max} = 5$. Then, until $P_{\max} = 10$, we gain ϵ on $\sum_{j=1}^{n} C_j$ by increasing $P_{\max}$ by ϵ; the value $P_{\max} = 10$ allows the SPT schedule.

From this example, we conclude that a schedule can only be extreme if a *complete interchange* has occurred in $\sigma(P)$, where an interchange is defined to be a complete interchange if there are two jobs J_i and J_j such that J_i is started before J_j in $\sigma(P - \epsilon)$, whereas J_j is started before J_i in $\sigma(P)$.

Lemma 1 *If $P > P^*_{\max}$, then the point $(P, \sum_{j=1}^{n} C_j(\sigma(P)))$ can be extreme only if a complete interchange has occurred in $\sigma(P)$.* □

The next step is to determine the $P_{\max}$ values P such that their corresponding points $(P, \sum_{j=1}^{n} C_j(\sigma(P)))$ satisfy this necessary condition. Given a pair of jobs J_i and J_j with $p_i > p_j$ and J_i started before J_j in $\sigma(P)$, we have to increase the upper bound on $P_{\max}$ such that J_j can start at time $S_i(\sigma(P))$. This will lead to a complete interchange of J_i and J_j in $\sigma(P^1)$, unless J_i itself is started at an earlier time in the schedule $\sigma(P^1)$, where $P^1 = s_j - S_i(\sigma(P))$ is the value of the upper bound on $P_{\max}$ that makes J_j available at time $S_i(\sigma(P))$. It is not possible to determine beforehand whether J_i gets started earlier when the upper bound on $P_{\max}$ is increased from P to P^1 J_i, except for one situation: J_i is executed between the start and completion time of a preemptive job J_k. In that case, increasing the upper bound on $P_{\max}$ will first lead to a uniform shift forward of J_i and J_j at the expense of J_k; 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.

Algorithm I exploits these observations to generate each point $(P, \sum_{j=1}^{n} C_j(\sigma(P_{\max}))$ for which a complete interchange in $\sigma(P)$ may take place. The variable a_j $(j = 1, \ldots, n)$ signifies the increase of the current $P_{\max}$ value necessary to let a complete interchange involving J_j and some successor take place.

Algorithm I

Step 0. Let $P = P^*_{\max}$.

Step 1. Let $T \leftarrow 0$ and $a_j \leftarrow \infty$ for $j = 1, \ldots, n$; determine $\sigma(P)$ through Baker's rule.

Step 2. Let J_k be the job that starts at time T in $\sigma(P)$. Consider the following two cases:
(a) J_k is a preempted job. Then a_k is equal to the length of this portion of J_k. Set $T \leftarrow C_k(\sigma(P))$.
(b) J_k is not a preempted job. Then $a_k \leftarrow \min\{s_j - P - S_k(\sigma(P)) \mid J_j \in \mathcal{V}\}$, where $\mathcal{V}$ denotes

the set of jobs J_j for which $s_j - P > S_k(\sigma(P))$ and $p_j > p_k$. Set $T \leftarrow C_k(\sigma(P))$.

Step 3. If $T < \sum_{j=1}^{n} p_j$, then go to Step 2.

Step 4. Put $P \leftarrow \min_{1 \leq j \leq n} a_j + P$.

Step 5. If $P = P_{\max}(SPT)$, then stop; otherwise go to Step 1.

Theorem 1 *Algorithm I generates all $P_{\max}$ values P for which a complete interchange has taken place in the corresponding schedule $\sigma(P)$.*

Proof. Suppose that a complete interchange of the jobs J_i and J_j with $p_i > p_j$ took place in the schedule $\sigma(P)$, where P was not detected by Algorithm I. Hence, $S_i(\sigma(P_{\max}))$ must have been ignored in Step 2, which could have happened only in Step 2(a): J_i is started between the start and completion time of some preempted job J_k. 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. □

As remarked before, the algorithm may generate too many $P_{\max}$ values P: in some of the schedules $\sigma(P)$ not a complete interchange has taken place. This is due to Step 2b. There we implicitly assumed that the part of the schedule before J_k, which was defined as the job to be interchanged, would remain scheduled before J_k, that is, that J_k itself would not be started earlier. This is not necessarily the case, however, since an increase of the upper bound on $P_{\max}$ may cause J_k to move forward at the expense of some job J_l with $p_l > p_k$, where the increase of the upper bound is not large enough to allow a complete interchange; J_k will preempt J_l then. Nevertheless, we now prove that the number of values $P_{\max}$ generated by Algorithm I is polynomially bounded, thereby establishing that $1|pmtn, nmit|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ is polynomially solvable. We define for a given schedule σ the indicator function $\delta_{ij}(\sigma)$ as

$$\delta_{ij}(\sigma) = \begin{cases} 2, & \text{if } C_i(\sigma) \leq S_j(\sigma) \text{ and } p_i > p_j, \\ 0, & \text{otherwise.} \end{cases}$$

We further define $\Delta_j(\sigma)$ as $\sum_{i=1}^{n} \delta_{ij}(\sigma)$ plus the number of preemptions of J_j, and we let $\Delta(\sigma) = \sum_{j=1}^{n} \Delta_j(\sigma)$.

Theorem 2 *Let P^1 be the $P_{\max}$ value that is found by Algorithm I when applied to $\sigma(P)$, where P is any $P_{\max}$ value determined by Algorithm I. We then have that $\Delta(\sigma(P^1)) < \Delta(\sigma(P))$.*

Proof. As explained above, one of the following three things has happened in $\sigma(P^1)$ in comparison to $\sigma(P)$:

(i) a preemption has been removed (Step 2a);

(ii) two jobs not in SPT-order have been interchanged (successful Step 2b);

(iii) a new preemption has been created (unsuccessful Step 2b).

All three cases have a negative effect on the value of Δ, as is easily checked (in the third case we do create an extra preemption (effect $+1$), but this pair of jobs is no longer in the wrong order (effect -2)). Hence, we only have to show that there are no moves possible that have an overall positive effect on the value of Δ. The candidates for such a move are a switch of two jobs from SPT order to LPT order and the addition of an extra preemption. We first investigate the effect of the 'wrong' switch.

Suppose that there are two jobs J_i and J_j with $p_i > p_j$ such that J_i succeeds J_j in $\sigma(P)$, whereas the order is reversed in $\sigma(P^1)$. Since Baker's algorithm prefers J_j to J_i if both

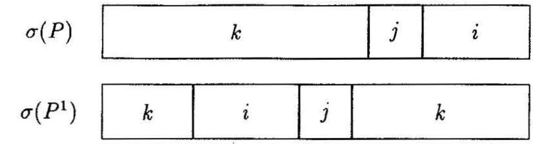

Figure 1: 'WRONG' SWITCH

jobs are available, J_i starts earlier in $\sigma(P^1)$ than J_j in $\sigma(P)$, which means that the execution of (a part of) some job J_k is postponed until J_i is completed. See Figure 1 for an illustration.

It is easily checked that we have $\Delta(\sigma(P)) = 4$ and $\Delta(\sigma(P^1)) = 3$. All we have to do is to show is that the situation depicted in Figure 1 is worst possible for this configuration. It is sufficient to prove that J_j is available at time $C_i(\sigma(P^1))$, that is, $s_j - P^1 \leq C_i(\sigma(P^1)) = s_i - P^1 + p_i$; if so, Baker's algorithm will prefer it to J_k, since the remainder of J_k has length at least equal to p_i. Hence, we have to show that $s_j \leq s_i + p_i$. As J_i did not preempt J_k in $\sigma(P)$, we must have $s_i - P + p_i \geq C_k(\sigma(P)) \geq s_j - P$, where the last inequality follows from the availability of J_j at time $C_k(\sigma(P))$. Since the smaller job is available as soon as the larger job involved in the wrong switch is completed, the increase of δ_{ij} is compensated for by the decrease of δ_{ki}. Moreover, job J_k cannot trigger a set of nested wrong switches, where we mean with a set of nested wrong switches that $\sigma(P)$ and $\sigma(P^1)$ contain the subschedules J_k, J_j, J_i, J_h and J_h, J_i, J_j, J_k with $p_j < p_i < p_h < p_k$.

Now consider the situation that the number of preemptions of a job J_k increases. Hence, there must be a job J_i with $p_i < p_k$ that succeeds J_k in $\sigma(P)$ but not in $\sigma(P^1)$, which move decreases the Δ function by one. □

Corollary 1 *If preemption is allowed, then the number of extreme schedules with respect to* $(P_{\max}, \sum_{j=1}^{n} C_j)$ *is bounded by* $n(n-1)+1$.

Proof. We have that $\Delta(\sigma) \leq n(n-1)$ for any schedule σ. Application of Theorem 2 yields the desired result. □

It is easy to construct an instance for which Algorithm I determines $O(n^2)$ different $P_{\max}$ values. We have not found an example with $O(n^2)$ extreme points yet.

Corollary 2 *The* $1|pmtn, nmit|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ *problem is solvable in* $O(n^4)$ *time.* □

Theorem 3 *If* $\alpha_1 = \alpha_2$, *then there exists a nonpreemptive optimal schedule for the* $1|pmtn, nmit|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ *problem. If* $\alpha_1 > \alpha_2$, *then any optimal schedule for the* $1|pmtn, nmit|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ *problem 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 ϵ, then P_j is increased by ϵ, while C_j is decreased by ϵ. 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 $\alpha_1 > \alpha_2$ such an interchange would decrease the objective value, contradicting the optimality of the initial schedule. □

3.2 The general case

We now drop the no machine idle time constraint. Obviously, if total completion time outweighs maximum promptness, then the insertion of machine idle time before the processing of any job makes no sense. Hence, we have the following.

Corollary 3 *If $\alpha_1 \geq \alpha_2$, then $1||\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ is solvable in $O(n^4)$ time.* □

If $\alpha_1 < \alpha_2$, then the insertion of idle time may decrease the value of the objective function. We now show that we can solve the $1|pmtn|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ problem by using Algorithm I, which was initially designed for solving $1|nmit, pmtn|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$.

Suppose that α_1 and α_2 are given. Define $q = \alpha_2/\alpha_1$. If $q > n$, then it is always advantageous to decrease $P_{\max}$, which implies that the execution of the first job will be delayed for ever and ever. To prevent unbounded solutions, we therefore assume that $q \leq n$. A straightforward computation then shows that in any optimal schedule at least $\lfloor n - q + 1 \rfloor$ jobs are scheduled before the first incidence of idle time. The smallest value $P_{\max}(q)$ for maximum promptness that leads to such a schedule is readily obtained. Moreover, no optimal schedule with $P_{\max} \geq P^*_{\max}$ contains idle time. Therefore, we need to consider the case $P_{\max}(q) \leq P_{\max} \leq P^*_{\max}$ only.

Consider any instance $\mathcal{I}$ of $1|pmtn|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$; let $\sigma(P_{\max})$ denote any optimal schedule for $\mathcal{I}$ of $1|r_j, pmtn|\sum C_j$ for any $P_{\max}$ with $P_{\max}(q) \leq P_{\max} \leq P^*_{\max}$ and $r_j = \max\{0, s_j - P_{\max}\}$.

We create a very large job J_0 that is available from time 0 onwards to saturate $\sigma(P_{\max})$ by filling in J_0 in the periods of idle time. In fact, J_0 is so large that Baker's rule prefers each job in $\mathcal{I}$ to it; the choices $s_0 = P_{\max}(q)$ and $p_0 = P^*_{\max} - P_{\max}(q) + \max_{1 \leq j \leq n} p_j + 1$ ensure such a saturation for any $P_{\max}(q) \leq P_{\max} \leq P^*_{\max}$. Let $\mathcal{I}'$ denote the instance $\mathcal{I}$ to which J_0 is added. Due to the choice of p_0 and s_0, we have that no optimal schedule for the instance I' of $1|nmit, pmtn|\alpha_1 \sum_{j=1}^{n} + \alpha_2 P_{\max}$ contains machine idle time, and moreover, that by simply removing J_0 and leaving the rest of the schedule intact we obtain an optimal schedule for the original instance $\mathcal{I}$ of $1|pmtn|\alpha_1 \sum_{j=1}^{n} + \alpha_2 P_{\max}$. After all, we have that $C_0 = \sum_{j=0}^{n} p_j$ and that $P_0 < P_{\max}$ for any value of $P_{\max}$. Hence, instead of solving $1|pmtn|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ for $\mathcal{I}$, we solve $1|nmit, pmtn|\alpha_1 \sum_{j=0}^{n} C_j + \alpha_2 P_{\max}$ for $\mathcal{I}'$. This approach provides us with the extreme points for $(\sum_{j=1}^{n} C_j, P_{\max})$ with $P_{\max}(q) \leq P_{\max} \leq P^*_{\max}$. If q is unknown, then we obtain all bounded extreme points by running the above procedure with $q = n$; this choice of q corresponds to the smallest value $P_{\max}(q)$ that may correspond to a bounded extreme point.

As the number of extreme points is at most equal to $n(n+1)+1$ (we have $n+1$ jobs now), and as each $P_{\max}$ value that corresponds to an extreme point is determined by Algorithm I, the $1|pmtn|\alpha_1 \sum_{j=1}^{n} C_j + \alpha_2 P_{\max}$ problem is solved in $O(n^4)$ time.

Finally, we wish to mention two important special cases of our problem. These are the case that promptness is assumed to be nonnegative, that is, $P_j = \max\{s_j - S_j, 0\}$, and the case that there is a given externally determined upper bound on $P_{\max}$. Either case can be dealt with by simply adjusting the objective function, and our algorithm can be used to solve the problem after the boundary points have been determined.

References

[1] R. Ahmadi, U. Bagchi (1990). Lower bounds for single-machine scheduling problems. *Naval Res. Log. Quart.* 37, 967-979.

[2] K.R. BAKER (1974). *Introduction to Sequencing and Scheduling*, Wiley, New York.

[3] P. DILEEPAN AND T. SEN (1988). Bicriterion static scheduling research for a single machine. *Omega 16*, 53-59.

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

[5] R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, AND A.H.G. RINNOOY KAN (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 5*, 287-326.

[6] J.A. HOOGEVEEN (1992). *Single-machine bicriteria scheduling*, PhD Thesis, CWI, Amsterdam.

[7] J.A. HOOGEVEEN (1996). Minimizing maximum promptness and maximum lateness on a single machine. *Mathematics of Operations Research 21*, 100-114.

[8] J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1995). Minimizing total completion and maximum cost simultaneously is solvable in polynomial time, *Operations Research Letters 17*, 205-208.

[9] J.K. LENSTRA, A.H.G. RINNOOY KAN, AND P. BRUCKER (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343-362.

[10] M. SCHUTTEN, S.L. VAN DE VELDE, AND W.H.M. ZIJM (1996). Single-machine scheduling with release dates, due date, and family setup times, *Management Science* 42, 1165-1174.

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

[12] L.N. VAN WASSENHOVE AND F. GELDERS (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research 4*, 42-48.