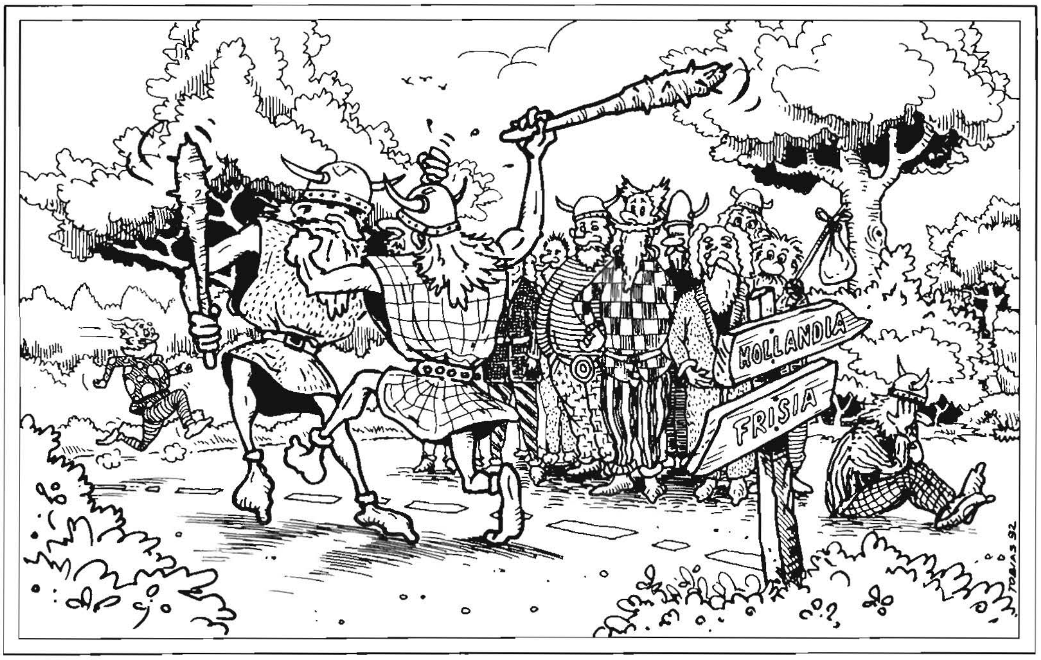
Single-machine bicriteria scheduling

DOI:
10.6100/IR367531

Document status and date:
Published: 01/01/1992

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Single-Machine Bicriteria Scheduling



Han Hoogeveen

Single-Machine Bicriteria Scheduling

Single-Machine Bicriteria Scheduling

PROEFSCHRIFT
ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof. dr. J.H. van Lint, voor een commissie aangewezen door het College van Dekanen in het openbaar te verdedigen op vrijdag 7 februari 1992 te 16.00 uur
door

JOHANNES ADZER HOOGEVEEN

geboren te Oss

1992
CWI, Amsterdam

Dit proefschrift is goedgekeurd door
de promotor:

Prof. dr. J. K. Lenstra

Foar ús Heit en ús Mem

Acknowledgements

The process of writing this thesis has been completed successfully thanks to numerous people. Especially, I would like to thank Steef van de Velde, who helped me to get acquainted with scientific research and who conducted joint research with me. I also would like to thank Gerard Kindervater, Bart Veltman, and Peter de Waal for their willingness to always help me in my permanent struggle with the computer, and Tobias Baanders, who designed the cover. Furthermore, I would like to thank the Dutch Organization for Scientific Research (NWO) for their financial support that made this PhD-research possible.

I am truly grateful to my supervisor Jan Karel Lenstra, who spent a lot of time on teaching me how to write proper articles in Standard American.

Han Hoogeveen

Table of contents

Machine scheduling and multicriteria optimization: an introduction

1. Motivation

Scheduling theory was introduced in the 1950's. Since that time, the scheduling models that have been addressed by researchers 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, for instance, judges a production scheme on the basis of a number of criteria, for example, work-in-process inventories and observance of due dates. 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, while, 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 on all important criteria.

An important drawback of considering such problems lies in the difficulty of defining an appropriate notion of optimality and, given such a notion, finding an optimal solution. Obviously, the situation becomes more complicated when more criteria are involved, unless the criteria are not in conflict with each other; roughly speaking, two criteria are not in conflict if a solution that performs well on one criterion is likely to perform well on the other criterion. If the criteria are conflicting, then the different solutions have to be weighed against each other. To that end, various options exist. The first one is to specify an upper bound on the value of the most important criterion; a solution is then selected that performs well on the other criteria while satisfying the bound. The second option is to aggregate the criteria into a single objective function; a solution is chosen that is optimal for this objective function. The third option is based upon an interactive version of decision making: an analyst determines a candidate solution and presents it to a decison maker, who either decides to accept it or tells the analyst on which criterion the score should be improved. Unfortunately, the determination of n candidate solutions usually takes more time than solving n times one of the basic single-criterion problems; sometimes, it is not even possible to guarantee that *one* reasonable candidate solution is found in a reasonable amount of time. It is of great importance to know beforehand what the consequences are of taking extra criteria into account. If it is difficult to find a good set of candidate solutions, then one might prefer to look for a solution of a somewhat lesser quality that is more easily obtained.

An important issue concerns the question what constitutes a representative set of candidate solutions. An obvious choice is the set of all *nondominated* solutions. A solution is said to be nondominated if it outperforms any other solution on at least one criterion. If the number of nondominated solutions is large, then an analyst may impose extra restrictions upon the set of candidate solutions; for example, he can impose an upper bound on the value of a criterion. We analyze this kind of strategies in the next section, in which the problem setting is introduced.

2. Problem setting

The setting of our problems is as follows. A set of n independent jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The machine is assumed to be continuously available from time 0 onwards. Job J_j $(j = 1, \ldots, n)$ requires processing during a given uninterrupted time p_j; to each job are assigned a given weight w_j, denoting its relative importance, and a given due date d_j, at which J_j should be delivered. It is assumed that all values p_j, w_j, and d_j are positive and integral.

A *schedule* σ defines for each job J_j a completion time $C_j(\sigma)$ such that the capacity and availability constraints of the machine are not violated. We assume that the quality of a schedule is measured in terms of two criteria; the *scheduling cost* is measured by a function $F(f,g)$, where f and g are two *performance criteria* defined on σ. We consider the following performance criteria:
- the *sum of completion times* $\Sigma C_j = \Sigma_{j=1}^{n} C_j(\sigma)$,
- the *sum of weighted completion times* $\Sigma w_j C_j = \Sigma_{j=1}^{n} w_j C_j(\sigma)$,
- the *maximum lateness* $L_{\max} = \max_{1 \leqslant j \leqslant n} (C_j(\sigma) - d_j)$,
- the *maximum earliness* $E_{\max} = \max_{1 \leqslant j \leqslant n} (d_j - C_j(\sigma))$,
- the *maximum cost* $f_{\max} = \max_{1 \leqslant j \leqslant n} f_j(C_j(\sigma))$, where all penalty functions f_j $(j = 1, \ldots, n)$ are assumed to be nondecreasing in the job completion times.

We illustrate these notions by a 4-job example. The data are found in Table 1. An arbitrary schedule σ is represented in the *Gantt chart* in Figure 1. The values of σ for the performance measures ΣC_j and $L_{\max}$ are easily computed; $\Sigma C_j(\sigma) = 3+7+12+18 = 40$, and $L_{\max}(\sigma) = \max\{3-20, 7-16, 12-11, 18-5\} = 13$. Note that, if there is no machine idle time between the jobs, then a schedule is completely characterized by the order in which the jobs are executed. Such a schedule is called a *permutation schedule.*

	J_1	J_2	J_3	J_4
p_j	3	4	5	6
d_j	20	16	11	5

TABLE 1. Processing times and due dates.

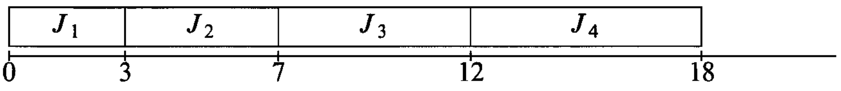


FIGURE 1. Gantt chart.

In this thesis, two methods are used to combine conflicting criteria: *hierarchical* minimization and *simultaneous* minimization. In case of hierarchical minimization, the performance criteria are ranked in order of importance; the less important criterion is minimized subject to the constraint that the schedule is optimal with respect to the more important criterion. In case of *simultaneous* minimization, the criteria are aggregated into a single composite objective function, which is then minimized. Note that simultaneous minimization turns into hierarchical minimization for an appropriate choice of the composite objective function.

We assume that any composite objective function is nondecreasing in both arguments. This assumption reflects the belief that a dominated solution should not be chosen as the optimal solution. We show that under this assumption there is a *Pareto optimal* point in which the minimum is attained.

DEFINITION 1. A feasible schedule σ is *Pareto optimal* with respect to the performance criteria f and g if there is no feasible schedule π such that $f(\pi) \leqslant f(\sigma)$ and $g(\pi) \leqslant g(\sigma)$, where at least one of the inequalities is strict.

THEOREM 1. *If the composite objective function F of (f,g) is nondecreasing in both arguments, then there exists a Pareto optimal point for (f,g) in which the function F attains its minimum.*

PROOF. Let (f_1,g_1) be a point in which F attains its minimum. If (f_1,g_1) is not Pareto optimal, then there exists a Pareto optimal point (f_2,g_2), with $f_2 \leqslant f_1$ and $g_2 \leqslant g_1$. Hence, $F(f_2,g_2) \leqslant F(f_1,g_1)$, implying that F also attains its minimum in (f_2,g_2). □

We use our example to illustrate this definition. We generate the set of all schedules that are possibly Pareto optimal. As it is easily seen that insertion of idle time cannot improve the quality of the schedule on ΣC_j or $L_{\max}$, we know that we can restrict ourselves to the $4! = 24$ permutation schedules. The values of these schedules with respect to $(L_{\max}, \Sigma C_j)$ have been plotted in Figure 2; each point corresponds to a schedule, and the Pareto optimal points are in bold. The essence of Pareto optimality is shown at the point (7,43): it is Pareto optimal as there is no point in its lower-left quadrant; this is the area to the south-west of the dotted lines. Note that we can profit from the knowledge that F is nondecreasing in both arguments if we are able to determine the set of Pareto optimal points in less time than needed for complete enumeration.

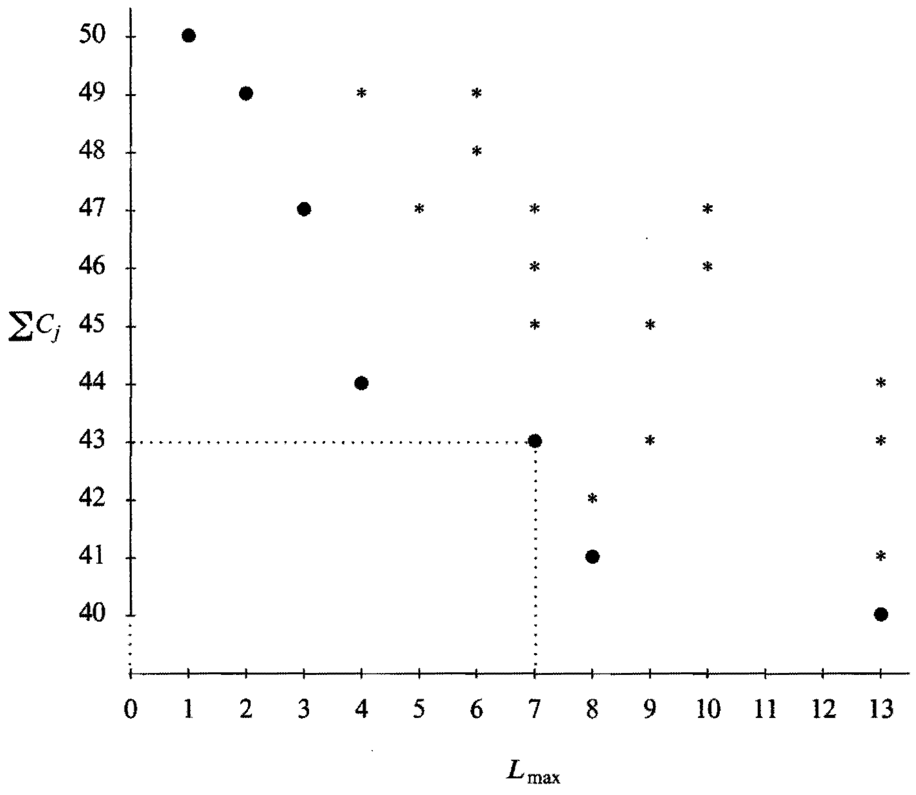


FIGURE 2. Outcomes of the example.

Once the set of Pareto optimal points has been obtained, we can solve the problem for any composite objective function F that is nondecreasing in each of its arguments. If we do not have any information on F except that F is nondecreasing in each of its arguments, then we are forced to determine the set of Pareto optimal points to solve the problem. If we have some additional information on F, then there may be quicker ways to solve the problem. A common property of F is linearity, that is, $F(f,g) = \alpha_1 f + \alpha_2 g$, where α_1, α_2 are assumed to be nonnegative. We show that, in case of a linear composite objective function, we can restrict ourselves to determining the set of *extreme* points with respect to (f,g). Before defining the concept extreme, we need two preliminary definitions.

DEFINITION 2. A feasible schedule σ is *efficient* with respect to (f_1, f_2) if there exist real nonnegative values (α_1, α_2) such that $\alpha_1 f_1(\sigma) + \alpha_2 f_2(\sigma) \leqslant \alpha_1 f_1(\pi) + \alpha_2 f_2(\pi)$ for all feasible schedules π.

DEFINITION 3. The *efficient frontier* is the shortest curve that connects all efficient points.

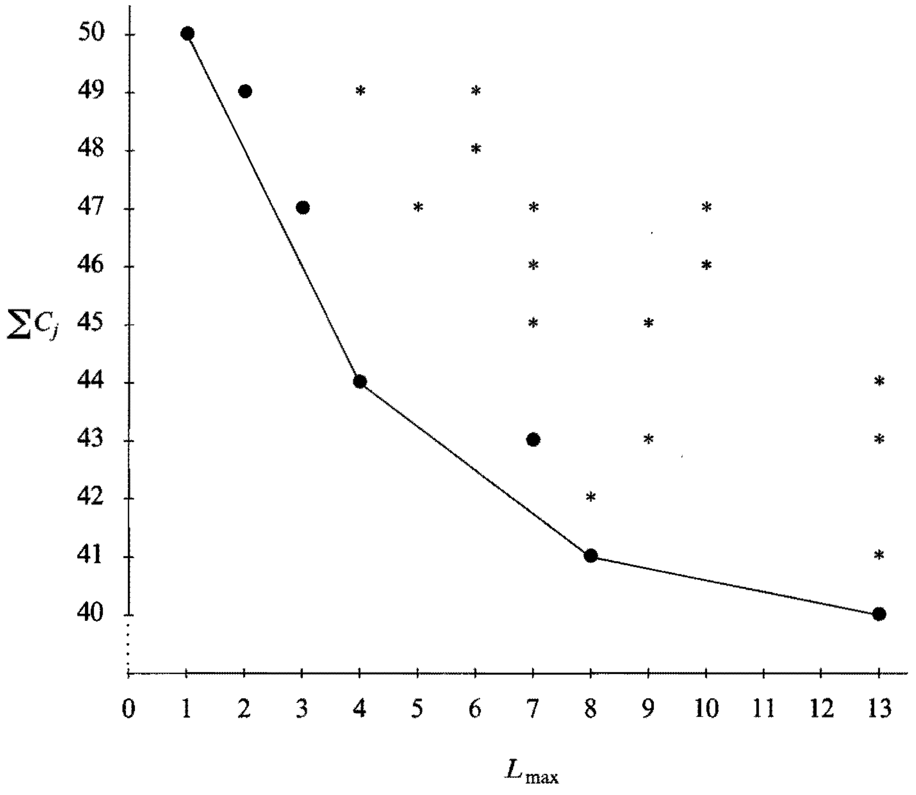


FIGURE 3. The efficient frontier for the example.

DEFINITION 4. A feasible schedule σ is *extreme* with respect to (f,g) if it corresponds to a vertex of the efficient frontier.

THEOREM 2. *If the composite objective function F of (f,g) is linear, then there exists an extreme point for (f,g) in which F attains its minimum.*

PROOF. Let (f_1, g_1) be a point in which F attains its minimum. If (f_1, g_1) is not extreme, then there exists a line segment of the efficient frontier containing a point (f_2, g_2) with $f_2 \leqslant f_1$ and $g_2 \leqslant g_1$; hence, $F(f_2, g_2) \leqslant F(f_1, g_1)$. Due to the linearity of F, at least one of the endpoints of the line segment must have cost no more than $F(f_2,g_2)$, implying that F attains it minimum in one of these extreme points. □

These notions are applied to our example in Figure 3. Note that the number of interesting points has decreased from 7 to 4. Obviously, we can gain in speed from the knowledge that F is linear if we are able to determine the set of extreme points faster than the set of Pareto optimal points.

If F is known exactly, then another solution strategy is to solve the problem directly, that is, without determining the set of extreme points. Unfortunately, this approach has been seldomly applied successfully.

Throughout this paper, we denote scheduling problems by the three-field notation scheme $\alpha \mid \beta \mid \gamma$ introduced by Graham, Lawler, Lenstra, and Rinnooy Kan (1979), where α describes the machine environment, β the job characteristics, and γ the objective function.

The most commonly used job characteristics are 'preemption', denoted by the acronym *pmtn*, 'no machine idle time', denoted by *nmit*, 'precedence constraints', denoted by *prec*, 'release dates', denoted by r_j, and 'deadlines', denoted by $\bar{d}_j$. In case preemption is allowed, the execution of a job can be interrupted and resumed later; in case of no machine idle time, all jobs have to be executed between time 0 and time Σp_j; in case of precedence constraints, for each job J_j ($j = 1, \ldots, n$), a set of jobs has been given that have to precede J_j and a set of jobs has been given that have to succeed J_j in any feasible schedule; in case of release dates, for each job J_j ($j = 1, \ldots, n$), a lower bound r_j on the start time S_j has been specified; in case of deadlines, for each job J_j ($j = 1, \ldots, n$), an upper bound $\bar{d}_j$ on the completion time C_j has been given.

For example, the single-machine scheduling problem in which the sum of the job completion times and the maximum lateness have to be minimized is denoted by $1 \mid \mid F(\Sigma C_j, L_{\max})$ if F is a general composite objective function, by $1 \mid \mid F_l(\Sigma C_j, L_{\max})$ if F is linear, and by either $1 \mid \mid F_h(\Sigma C_j, L_{\max})$ or by $1 \mid \mid F_h(L_{\max}, \Sigma C_j)$ if the minimization is hierarchical, where the first mentioned performance measure is assumed to be the more important one.

In case of a general composite objective function F, we will only consider solution approaches that determine all Pareto optimal points to solve the problem. In case of a linear composite objective function F_l, we will determine the set of extreme points to solve the problem, unless we are able to present a successful direct approach.

3. Outline of this thesis

The thesis consists of three parts: two introductory chapters, seven papers, and a 'samenvatting' (a summary in Dutch).

The second introductory chapter gives a survey of the complexity of the single-machine bicriteria problems that arise when two of the performance criteria mentioned in Section 2 of the present chapter are combined.

The first paper, 'Minimizing maximum promptness and maximum lateness on a single machine' (Hoogeveen, 1990), addresses the problems $1 \mid nmit \mid F(E_{\max}, L_{\max})$ and $1 \mid \mid F_l(E_{\max}, L_{\max})$. For both problems a polynomial-time algorithm is presented.

The second paper, 'Polynomial-time algorithms for single-machine multicriteria scheduling' (Hoogeveen and Van de Velde, 1990), addresses the problems $1 \mid \mid F(f_{\max}, \Sigma C_j)$, $1 \mid \mid F(L_{\max}, \Sigma C_j)$, $1 \mid pmtn \mid F_l(E_{\max}, \Sigma C_j)$, and $1 \mid \mid \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ with $\alpha_1 \geqslant \alpha_2$. Polynomial-time algorithms are presented for each of these problems.

The third paper, 'Single-machine scheduling to minimize a function of K maximum cost criteria' (Hoogeveen, 1991), presents polynomial-time algorithms for the problems $1 \mid \mid F(f_{\max}, g_{\max})$, where $f_{\max}$ and $g_{\max}$ are two arbitrary maximum cost criteria, and for $1 \mid nmit \mid F(E_{\max}, \tilde{E}_{\max})$, where $E_{\max}$ and $\tilde{E}_{\max}$ are two different maximum earliness criteria.

The fourth paper, 'A new lower bound approach for single-machine multicriteria scheduling' (Hoogeveen and Van de Velde, 1991A), presents a new lower bound approach, which we call 'objective splitting'. This lower bound approach can be applied to single-machine multicriteria scheduling problems with a linear composite objective function. It is shown to dominate previously proposed lower bound approaches for this problem in terms of both speed and quality.

The fifth and the sixth paper deal with *common due date* problems. The first of these, 'Scheduling around a small common due date' (Hoogeveen and Van de Velde, 1991B), addresses the $1 \mid \mid \Sigma w_j \mid C_j - d \mid$ problem. This problem is shown to be $\mathcal{NP}$-hard for general d, even if $w_j = 1$ for $j = 1, \ldots, n$; a pseudopolynomial-time optimization algorithm is presented. The next paper, 'New lower and upper bounds for scheduling around a small common due date' (Hoogeveen, Oosterhout, and Van de Velde, 1990), deals with the $1 \mid \mid \Sigma \mid C_j - d \mid$ problem. Lower and upper bounds are presented that coincide for virtually all instances, provided that the number of jobs is not too small.

In the last paper, 'Single machine scheduling to minimize total inventory cost' (Hoogeveen and Van de Velde, 1991C), the $1 \mid \mid \alpha \Sigma C_j + \beta \Sigma E_j$ problem with $0 \leqslant \alpha < \beta$ is solved by branch-and-bound. This problem turns out to be very hard to solve in practice. Although we present no less than six lower bound procedures, the largest instances solvable in reasonable time by our algorithm consist of no more than 20 jobs.

References

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.

J.A. Hoogeveen (1990). *Minimizing maximum earliness and maximum lateness on a single machine*, Report BS-R9001, CWI, Amsterdam.

J.A. Hoogeveen (1991). *Single-machine scheduling to minimize a function of K maximum cost criteria*, Report BS-R9113, CWI, Amsterdam.

J.A. Hoogeveen, H. Oosterhout, and S.L. van de Velde (1990). *New lower and upper bounds for scheduling around a small common due date,* Report BS-R9030, CWI, Amsterdam.

J.A. Hoogeveen and S.L. van de Velde (1990). *Polynomial-time algorithms for single-machine multicriteria scheduling,* Report BS-R9008, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1991A). A new lower bound approach for single-machine multicriteria scheduling. To appear in *Operations Research Letters.*

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1991B). Scheduling around a small common due date. *European Journal of Operational Research 55,* 237-242.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1991C). Minimizing total inventory cost on a single machine in just-in-time manufacturing, Report BS-R91xx, CWI, Amsterdam.

Complexity of single-machine bicriteria scheduling: a survey

As already mentioned in the previous chapter, it is of the utmost importance to know beforehand whether an optimal solution is guaranteed to be found within a reasonable amount of time; otherwise, one may decide to look for a quickly determined solution that is not necessarily optimal.

Of course, we first have to specify what amount of time should be considered reasonable to be spent on searching for an optimal solution. Obviously, the decision should involve the size of a problem, that is, the number of jobs and so on. An optimization problem is said to be easy if there exists an algorithm that solves the problem in time *polynomially* bounded in the input size. Next to the optimization variant of a problem, there exists the decision variant of a problem. The decision variant of a scheduling problem is defined as the following question: given an instance of the problem and a threshold value y, does there exist as schedule with value no more than y? All easy decision problems constitute the class $\mathcal{P}$. This class is a subset of the class $\mathcal{NP}$, which, in the present context, contains all decision problems for which it is possible to check in polynomial time if the answer is 'yes' for a given schedule. A decision problem is said to be $\mathcal{NP}$-complete if it belongs to $\mathcal{NP}$ and if every problem in $\mathcal{NP}$ is *polynomially reducible* to it. A problem A is said to be polynomially reducible to a problem B if and only if an arbitrary instance of A can be solved by solving a corresponding instance of B that is constructed in time polynomially bounded in the size of A. The optimization variant of an $\mathcal{NP}$-complete problem is called $\mathcal{NP}$-hard; these problems are at least as hard as all problems in $\mathcal{NP}$. For details concerning $\mathcal{NP}$-hardness, we refer to the excellent textbook by Garey and Johnson (1979).

In Section 1, we repeat some basic concepts for the single-machine single-criterion scheduling problems underlying our bicriteria problems. In Section 2, we present a survey of bicriteria scheduling problems, thereby providing some straightforward polynomial-time optimization algorithms and $\mathcal{NP}$-hardness proofs. These complexity results are displayed in three tables in Section 3. The first one concerns hierarchical minimization, while the second and third concern simultaneous optimization with a general and a linear composite objective function, respectively. Some concluding remarks are made in Section 4.

1. Basic concepts

Before analyzing the bicriteria problems, we first show how to solve the single-criterion problems in which one of the performance criteria mentioned in Section 1.2 is involved. Four of these problems are solved by scheduling the jobs according to a *priority order* that is specified in terms of the parameters of the problem; only the minimization of $f_{\max}$ needs a more intricate procedure.

Instead of the definition of $E_{\max}$ given in the previous chapter, we will use from now on a refinement of it, proposed by Garey, Tarjan, and Wilfong (1988). They define $E_j = d_j - p_j - S_j$, for $j = 1, \ldots, n$, where S_j denotes the start time of J_j, and $E_{\max} = \max_{1 \leqslant j \leqslant n} E_j$. If a job is executed during an uninterrupted period of time, then both definitions coincide. However, if we allow preemption, that is, a job may be interrupted and resumed later, then the new definition resembles the idea that earliness should be measured in terms of a deviation of the actual start time from a target start time.

Note that all performance criteria under consideration, except $E_{\max}$, are *regular*; that is, the objective function cannot be decreased by inserting idle time into a given feasible schedule. In case of a regular performance criterion, we can 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. Hence, if all jobs are allowed to start at time 0, then all jobs in an active schedule are processed in the interval $[0, \Sigma p_j]$. In order to avoid unbounded solutions in case of a nonregular performance measure like $E_{\max}$, we impose the constraint that no machine idle time is allowed, implying that all jobs have to be processed in the interval $[0, \Sigma p_j]$. This constraint is denoted by putting *nmit* ('no machine idle time') in the second field of the three-field notation scheme.

Theorem 1 (Smith, 1956). *The problem* $1||\Sigma C_j$ *is solved by scheduling the jobs according to the shortest processing time (SPT) rule, that is, in order of nondecreasing* p_j.

Proof. The proof is based upon showing that every schedule σ can be transferred into an *SPT* schedule by applying adjacent interchanges that do not increase ΣC_j. Consider an arbitrary schedule σ in which the jobs are not in *SPT* order. Hence, there exist two adjacent jobs J_i and J_j in σ that are not in *SPT* order, with $p_i > p_j$. It is easily checked that swapping J_i and J_j decreases $\Sigma C_j(\sigma)$ by $p_i - p_j$. This interchange argument can be applied until all jobs are in *SPT* order, implying that an *SPT* schedule is optimal. $\square$

Similar proofs hold for the Theorems 2 through 4.

THEOREM 2 (Smith, 1956). *The problem* $1 \mid \mid \Sigma w_j C_j$ *is solved by scheduling the jobs in order of nondecreasing ratios* p_j / w_j. □

THEOREM 3 (Jackson, 1955). *The problem* $1 \mid \mid L_{\max}$ *is solved by scheduling the jobs according to the earliest due date (EDD) rule, that is, in order of nondecreasing* d_j. □

THEOREM 4. *The problem* $1 \mid nmit \mid E_{\max}$ *is solved by scheduling the jobs according to the minimum slack time (MST) order, that is, in order of nondecreasing* $d_j - p_j$. □

Lawler (1973) presents an $O(n^2)$ algorithm for $1 \mid prec \mid f_{\max}$, where the acronym *prec* indicates that precedence constraints have been specified; that is, for each job J_j ($j = 1, \ldots, n$), a set of jobs has been given that have to precede J_j and a set of jobs has been given that have to succeed J_j in any feasible schedule. The algorithm is based upon the following observation. Let L denote the index subset of jobs that may be processed last, let T denote the sum of the processing times of all the jobs, and let J_k be a job in L such that $f_k(T) = \min_{j \in L} f_j(T)$. Then there exists an optimal schedule in which J_k is last.

LAWLER'S ALGORITHM

```
(0) $T \leftarrow \Sigma_{j=1}^{n} p_j$; $\mathcal{J} \leftarrow \{J_1, \ldots, J_n\}$.
(1) Determine the set $L$ containing the jobs that have no successors in $\mathcal{J}$.
(2) Choose from $L$ the job $J_j$ that has minimal $f_j(T)$ value, settling ties arbitrarily;
        $J_j$ is processed from time $T - p_j$ to time $T$.
(3) $T \leftarrow T - p_j$; $\mathcal{J} \leftarrow \mathcal{J} - \{J_j\}$.
(4) If $\mathcal{J} \neq \varnothing$, then go to Step 1; otherwise, stop.
```

THEOREM 5. *Lawler's algorithm solves* $1 \mid prec \mid f_{\max}$.

PROOF. We adopt the proof presented by Baker, Lawler, Lenstra, and Rinnooy Kan (1983). Let $N = \{1, 2, \ldots, n\}$ be the index set of all jobs, and let $L \subseteq N$ be the index set of jobs without successors. For any subset $S \subseteq N$, let $p(S) = \Sigma_{j \in S} p_j$ and let $f^*_{\max}(S)$ denote the cost of an optimal schedule indexed by S. Clearly, $f^*_{\max}(N)$ satisfies the following inequalities:

$$f^*_{\max}(N) \geqslant \min_{j \in L} f_j(p(N)),$$

$$f^*_{\max}(N) \geqslant f^*_{\max}(N - \{j\}) \text{ for all } j \in N.$$

Now let J_l with $l \in L$ be such that

$$f_l(p(N)) = \min_{j \in L} f_j(p(N)).$$

We have

$$f^*_{\max}(N) \geqslant \max\{f_l(p(N)), f^*_{\max}(N - \{l\})\}.$$

But the right-hand side of this inequality is precisely the cost of an optimal schedule subject to the condition that J_l is processed last. It follows that there exists an optimal schedule in which J_l is in the last position. As J_l is the job that is selected by Lawler's algorithm, repeated application of the above procedure yields the proof of correctness. □

Lawler's algorithm determines an optimal schedule in $O(n^2)$ time, as for each position the job in L with minimal cost has to be found.

2. A SURVEY OF SINGLE-MACHINE BICRITERIA SCHEDULING

In this section, we review results on single-machine bicriteria scheduling. We use the opportunity to provide a number of elementary results, in the form of either $\mathcal{NP}$-hardness proofs or polynomial optimization algorithms. It seems that these results, in spite of their simplicity, have not been published before. The more complicated results will be derived in the Papers I through III.

2.1. MINIMIZING A COMBINATION OF $\Sigma w_j C_j$ AND $\Sigma \hat{w}_j C_j$

We start by analyzing a combination of $\Sigma w_j C_j$ and $\Sigma \hat{w}_j C_j$. The results of this section can be easily extended to the special case that w_j or $\hat{w}_j$ is equal to 1, for $j = 1, \ldots, n$. Note that, if the p_j / w_j and the $p_j / \hat{w}_j$ ratios are similarly ordered, then there exists a schedule that is optimal for both criteria; hence, this schedule solves the problem of minimizing *any* nondecreasing function of $\Sigma w_j C_j$ and $\Sigma \hat{w}_j C_j$. Therefore, we assume from now on that the p_j / w_j and the $p_j / \hat{w}_j$ ratios are not similarly ordered.

First, we analyze the $1 | | F_h(\Sigma w_j C_j, \Sigma \hat{w}_j C_j)$ problem. Typical for a solution of the hierarchical minimization problem in which $\Sigma w_j C_j$ is involved as the primary criterion is that Smith's ratio rule leaves us no freedom to schedule the jobs, unless there are jobs with equal p_j / w_j ratio. In that case, however, we have complete freedom to schedule these jobs so as to minimize the secondary criterion. This implies that the problem of hierarchically minimizing the primary criterion $\Sigma w_j C_j$ and a secondary criterion f is as hard as minimizing f subject to the 'no machine idle time' constraint.

THEOREM 6. *The problem* $1 | | F_h(\Sigma w_j C_j, \Sigma \hat{w}_j C_j)$ *is solved by scheduling the jobs in order of nondecreasing* p_j / w_j *ratios, settling ties according to nondecreasing* $p_j / \hat{w}_j$ *ratios.* □

THEOREM 7. *The* $1 | | F(\Sigma w_j C_j, \Sigma \hat{w}_j C_j)$ *problem is* $\mathcal{NP}$*-hard in the ordinary sense.*

PROOF. Our proof is based upon showing that the number of Pareto optimal points is not polynomially bounded. Suppose that $w_j = 1$ and $\hat{w}_j = 2p_j - 1$, for $j = 1, \ldots, n$. As $\Sigma p_j C_j = \Sigma\Sigma_{j \leqslant k} p_j p_k = A$ is constant for every schedule without idle time, a schedule σ in which the jobs are executed in the interval $[0, \Sigma p_j]$ corresponds to a point $(\Sigma C_j(\sigma), 2A - \Sigma C_j(\sigma))$, and all such points are Pareto optimal. We will establish $\mathcal{NP}$-hardness of the problem by constructing an instance that yields 2^n consecutive Pareto optimal points; Schrijver (1989) proves that the problem of minimizing an arbitrary nondecreasing function over 2^n

consecutive integer points is $\mathcal{NP}$-hard in the strong sense (see Hoogeveen (1990); Paper I in this thesis). There are $2n+1$ jobs with processing times

$$p_1 = 1;\ p_{2i} = p_{2i+1} = 2^i,\quad \text{for } i = 1, \ldots, n$$

Let B be equal to the sum of the completion times when the jobs are in *SPT* order. Note that interchanging the jobs J_{2i-1} and J_{2i} increases ΣC_j by 2^{i-1}, for $i = 1, \ldots, n$. Hence, the points $(B+K, 2A-B-K)$ are all Pareto optimal, for $K = 0, \ldots, 2^n - 1$, and Schrijver's reduction can be executed. □

Note that this reduction only proves $\mathcal{NP}$-hardness in the ordinary sense as the processing times are exponential; this is all we can hope for when using this approach, as the number of Pareto optimal points for $(\Sigma w_j C_j, \Sigma \hat{w}_j C_j)$ is pseudopolynomially bounded by $n(\max_j w_j)\Sigma_j p_j$. It is yet an open question whether there exists a pseudopolynomial optimization algorithm for this problem.

Another discouraging result has been provided by Lenstra (1979), who proves that $1 \mid \Sigma w_j C_j \leqslant A \mid \Sigma \hat{w}_j C_j$ is already $\mathcal{NP}$-hard in the ordinary sense by a reduction from PARTITION.

THEOREM 8 (Lenstra, 1979). *The* $1 \mid \Sigma w_j C_j \leqslant y \mid \Sigma \hat{w}_j C_j$ *problem is* $\mathcal{NP}$*-hard in the ordinary sense.*

PROOF. We have to prove that the decision variant of the problem is $\mathcal{NP}$-complete. The decision variant of the problem is defined as the following question: given an arbitrary instance of the problem and a threshold value $\hat{y}$, does there exist a feasible schedule with cost no more than $\hat{y}$?

The decision variant of the problem belongs to $\mathcal{NP}$, as it is possible to check in polynomial time for a given solution whether it provides an affirmative answer. We now show that the PARTITION problem is polynomially reducible to it; PARTITION is $\mathcal{NP}$-complete in the ordinary sense.

PARTITION
Given a multiset $\mathcal{A} = \{a_1, \ldots, a_n\}$ of n integers, is it possible to partition $\mathcal{A}$ into two disjoint subsets that have equal sum?

Given an arbitrary instance $\{a_1, \ldots, a_n\}$ of PARTITION, define the constants B and C by $B = \Sigma\Sigma_{j \leqslant k} a_j a_k$ and $C = \Sigma_{j=1}^{n} a_j$, respectively, and construct the following instance of $1 \mid \Sigma w_j C_j \leqslant y \mid \Sigma \hat{w}_j C_j$:

$$p_j = w_j = a_j;\ \hat{w}_j = 0,\ \text{for } j = 1, \ldots, n,$$

$$p_0 = 1;\ w_0 = 0;\ \hat{w}_0 = 1,$$

$$y = B + C/2;\ \hat{y} = C/2 + 1.$$

The idea behind the reduction is the following: the constraint $\Sigma w_j C_j \leqslant y$ implies that J_0 cannot start before time $C/2$, the constraint $\Sigma \hat{w}_j C_j \leqslant \hat{y}$ implies that J_0 cannot be completed after time $C/2+1$, and the combination of the two constraints implies that the schedule cannot contain machine idle time. It is easy to see that a schedule that satisfies these three properties exists if and only if PARTITION is answered affirmatively.

We start by proving the third property. Consider an arbitrary schedule σ; $\Sigma w_j C_j(\sigma) + \Sigma \hat{w}_j C_j(\sigma) = \Sigma \tilde{w}_j C_j(\sigma)$. A lower bound for this last term is determined by scheduling the jobs according to Smith's ratio rule; this yields a solution equal to $y + \hat{y}$. Hence, if σ contains idle time, then $\Sigma w_j C_j(\sigma) + \Sigma \hat{w}_j C_j(\sigma) > y + \hat{y}$, implying that at least one of the constraints is not satisfied. The first and the second property are proven analogously; therefore, we only prove the first one. Let σ be an arbitrary schedule without idle, and suppose that J_0 is completed at time C_0. Let A denote the index set of jobs that are completed after J_0; $\Sigma_{j \in A} p_j = C - S_0$. Then $\Sigma w_j C_j(\sigma) = B + \Sigma_{j \in A} a_j = B + C - S_0$. Hence, $\Sigma w_j C_j(\sigma) \leqslant y$ implies $B + C - S_0 \leqslant B + C/2$ or $S_0 \geqslant C/2$. □

Note that the above reduction also proves that the $1 \mid \Sigma w_j C_j \leqslant y, pmtn \mid \Sigma \hat{w}_j C_j$ problem is $\mathcal{NP}$-hard, in which formulation the acronym *pmtn* denotes that preemption is allowed.

The situation becomes uncomparably brighter if we restrict ourselves to linear composite objective functions. Then the objective function is simply reformulated as $\Sigma \tilde{w}_j C_j$, with $\tilde{w}_j = \alpha w_j + \hat{\alpha} \hat{w}_j$ $(j = 1, \ldots, n)$, and the problem is solved through Smith's ratio rule.

If we do know that the composite objective function is linear but the values α and $\hat{\alpha}$ are not specified, then the problem is to specify the set of points $(\Sigma w_j C_j, \Sigma \hat{w}_j C_j)$ that correspond to an optimal solution for some choice of $(\alpha, \hat{\alpha})$; this set is exactly equal to the set of efficient points. By definition, we know that a point is efficient if there are nonnegative values α and $\hat{\alpha}$ such that $\alpha \Sigma w_j C_j + \hat{\alpha} \Sigma \hat{w}_j C_j = \Sigma \tilde{w}_j C_j$ is minimal. If the ratio $\alpha / \hat{\alpha}$ increases, then we move from one efficient point to another by interchanging two jobs J_i and J_j that have differently ordered weight over processing time ratios, that is, $w_i / p_i < w_j / p_j$ and $\hat{w}_j / p_j < \hat{w}_i / p_i$. This suggests determining the set of efficient points by computing for every pair of jobs with differently ordered weight over processing time ratios the values α and $\hat{\alpha}$ such that both $\tilde{w}_j / p_j$ ratios are equal. As a normalization constraint, we put $\hat{\alpha}$ equal to 1. We generate the efficient points in order of nondecreasing α value. Then we compute $\Sigma w_j C_j$ and $\Sigma \hat{w}_j C_j$ in constant time, given the previous efficient point. Hence, the set of efficient points is generated in $O(n^2 \log n)$ time, the time needed to order the $O(n^2)$ α-values. This approach was followed by Bagchi (1989) in the slightly different context of minimizing a nondecreasing linear function of ΣC_j and $\Sigma \mid C_j - C_i \mid$.

2.2. MINIMIZING A COMBINATION OF ΣC_j AND $f_{\max}$

As the results for minimizing a combination of ΣC_j and $L_{\max}$ are almost identical to the results for minimizing ΣC_j and $f_{\max}$, we devote a single section to these problems.

The first paper on a problem of this type is by Smith (1956), who shows that

$1 \mid L_{\max} \leqslant 0 \mid \Sigma C_j$ is solved by the following backward scheduling rule: *assign the job that has largest processing time from among the set of jobs that are allowed to be scheduled on that position.* Heck and Roberts (1972) solve $1 \mid \mid F_h(L_{\max}, \Sigma C_j)$ by this rule. Emmons (1975) applies Smith's rule to solve $1 \mid \mid F_h(f_{\max}, \Sigma C_j)$. 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 $(\Sigma C_j, L_{\max})$ can be determined. John (1989) extends their algorithm to determine the efficient frontier. Hoogeveen and Van de Velde (1990) (Paper II in this thesis) use a similar algorithm to solve $1 \mid \mid F(f_{\max}, \Sigma C_j)$. The running time of their algorithm for executing one iteration is $O(n \min\{n, \log \Sigma p_j\})$; furthermore, they show that every iteration yields a Pareto optimal point.

Therefore, the polynomiality of their algorithm depends upon the number of Pareto optimal points. With respect to $(L_{\max}, \Sigma C_j)$, 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 the number of Pareto optimal points for $(L_{\max}, \Sigma C_j)$ to be only pseudopolynomially bounded; hence, they presented their algorithm as being pseudopolynomial. This inspired Sen and Gupta (1983) to present a branch-and-bound algorithm for $1 \mid \mid L_{\max} + \Sigma C_j$. Hoogeveen and Van de Velde eventually verified validity and tightness of the claimed bound on the number of Pareto optimal points for $(L_{\max}, \Sigma C_j)$; they also proved that the same bound holds when $L_{\max}$ is replaced by $f_{\max}$. Therefore, $1 \mid \mid F(f_{\max}, \Sigma C_j)$ is solved in $O(n^3 \min\{n, \log \Sigma p_j\})$ time; $1 \mid \mid F(L_{\max}, \Sigma C_j)$ is solved in $O(n^3)$ time, due to appropriate preprocessing. For details, see Paper II in this thesis.

As already indicated in the previous subsection, the problems $1 \mid \mid F_h(\Sigma C_j, f_{\max})$ and $1 \mid \mid F_h(\Sigma C_j, L_{\max})$ are solved in $O(n^2)$ and $O(n \log n)$ time, respectively, by the following backward scheduling rule: *assign the job that has largest processing time, where ties are settled to minimize maximum cost and maximum lateness, respectively.*

2.3. Minimizing a combination of ΣC_j and $E_{\max}$

According to our knowledge, only one paper in which an objective function that is formed as a combination of these two criteria has appeared in the literature. For the case that preemption is allowed, Hoogeveen and Van de Velde (1990; Paper II in this thesis) show that, although the number of Pareto optimal points for $(\Sigma C_j, E_{\max})$ is not polynomially bounded, the number of efficient points is at most equal to $n(n-1)/2+1$ and that each of these points can be found in $O(n^2)$ time. Hence, $1 \mid pmtn \mid F_l(\Sigma C_j, E_{\max})$ is solved in $O(n^4)$ time. They also prove that $1 \mid pmtn \mid F_l(\Sigma C_j, E_{\max})$ has a nonpreemptive optimal solution when ΣC_j outweighs $E_{\max}$, implying that $1 \mid \mid \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ is solvable in $O(n^4)$ time if $\alpha_1 \geqslant \alpha_2$.

Allowing preemption is never advantageous in case of the hierarchical minimization problem with ΣC_j as the primary and $E_{\max}$ as the secondary criterion; $1 \mid \mid F_h(\Sigma C_j, E_{\max})$ is solvable in $O(n \log n)$ time through Smith's rule, where ties are settled according to nondecreasing slack time values. The prohibition of preemption does have impact on the other hierarchical minimization problem in which these two criteria are involved; the problem $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ is $\mathcal{NP}$-hard in the strong sense. We have included the 'no machine idle time' constraint to avoid unbounded solutions.

THEOREM 9. *The* $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ *problem is* $\mathcal{NP}$*-hard in the strong sense.*

PROOF. We establish $\mathcal{NP}$-hardness in the strong sense for $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ by showing that its decision variant is $\mathcal{NP}$-complete, or, in other words, that it belongs to $\mathcal{NP}$ and that an $\mathcal{NP}$-complete problem is polynomially reducible to it. The reduction is from the decision variant of $1 \mid r_j, nmit \mid \Sigma C_j$, which has been proven to be $\mathcal{NP}$-complete in the strong sense by Lenstra, Rinnooy Kan, and Brucker (1977). In this formulation, r_j denotes that for each job a *release date* has been specified; that is, job J_j cannot be started before time r_j $(j = 1, \ldots, n)$. As the machine is not available before time zero, we assume without loss of generality that the release dates are nonnegative. The decision variant of $1 \mid r_j, nmit \mid \Sigma C_j$ is defined as follows: given n jobs $J_1, \ldots, J_n$ with processing time p_j and release date $r_j \geqslant 0$ $(j = 1, \ldots, n)$, and a threshold value y, does there exist a feasible schedule with value no more than y?

Given an arbitrary instance of the decision variant of $1 \mid r_j, nmit \mid \Sigma C_j$, we construct the following instance of $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$. There are n jobs $J_1, \ldots, J_n$ that correspond to the jobs in the instance of $1 \mid r_j, nmit \mid \Sigma C_j$. The processing times of two corresponding jobs are equal; the due date of the job belonging to the instance for $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ is equal to the sum of the processing time and the release date of the corresponding job belonging to the instance for $1 \mid r_j, nmit \mid \Sigma C_j$.

Unless $1 \mid r_j, nmit \mid \Sigma C_j$ is infeasible, a schedule without idle time is obtained if we schedule the jobs in order of nondecreasing release dates. As $r_j \geqslant 0$ $(j = 1, \ldots, n)$, this implies that the outcome of $1 \mid nmit \mid E_{\max}$ for the instance constructed above is equal to 0, so that $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ is identical to $1 \mid E_{\max} \leqslant 0, nmit \mid \Sigma C_j$. Now consider the constraint $E_{\max} \leqslant 0$: it implies $E_j \leqslant 0$ $(j = 1, \ldots, n)$, and thereby $d_j - S_j - p_j \leqslant 0$ or $S_j \geqslant d_j - p_j$ for $j = 1, \ldots, n$. Therefore, this constraint induces a set of release dates $r_j = d_j - p_j$ $(j = 1, \ldots, n)$, so that the problems $1 \mid E_{\max} \leqslant 0, nmit \mid \Sigma C_j$ and $1 \mid r_j, nmit \mid \Sigma C_j$ are identical. If we choose equal thresholds for both decision problems, then we have that an affirmative answer for the one problem always corresponds to an affirmative answer for the other problem. The only thing left to prove is that the decision variant of $1 \mid nmit \mid F_h(E_{\max}, \Sigma C_j)$ belongs to $\mathcal{NP}$, which is obvious. □

The above result shows that $1 \mid nmit \mid F(\Sigma C_j, E_{\max})$ and the general $1 \mid nmit \mid F_l(\Sigma C_j, E_{\max})$ problem are $\mathcal{NP}$-hard in the strong sense. Hence, the problem becomes harder when $E_{\max}$ becomes more important. It is easy to show that $1 \mid \mid \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ and $1 \mid nmit \mid \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ become $\mathcal{NP}$-hard when some critical α_2 / α_1 ratio is exceeded; this ratio amounts to $(n-1)$ for the first problem and to $(n-1)(\max_j p_j - \min_j p_j)$ for the second problem. The question whether these critical values can be bounded more sharply is still open.

2.4. Minimizing a combination of $\Sigma w_j C_j$ and a maximum cost criterion

We now consider the problems that arise when combining $\Sigma w_j C_j$ with either $L_{\max}$, $f_{\max}$, or $E_{\max}$. The only well-studied problem that falls in this context is the $1 | L_{\max} \leqslant L | \Sigma w_j C_j$ problem, for some upper bound L on $L_{\max}$. Smith (1956) developed a heuristic for this problem proceeding in the same way as his algorithm to solve $1 | L_{\max} \leqslant 0 | \Sigma C_j$. Heck and Roberts (1972) claimed that Smith's heuristic always produces an optimal schedule. This claim was disproved by Burns (1976) by means of a counterexample; Lenstra, Rinnooy Kan, and Brucker (1977) proved that the problem was $\mathcal{NP}$-hard in the strong sense. Burns applied neighborhood search to improve Smith's heuristic for $1 | L_{\max} \leqslant 0 | \Sigma w_j C_j$. In its turn, this heuristic was improved by Miyazaki (1981). The general $1 | L_{\max} \leqslant L | \Sigma w_j C_j$ problem has been studied by Chand and Schneeberger (1986), who distinguish some special cases of the problem that are solved to optimality through Smith's heuristic.

We prove that, except for the hierarchical minimization problems with $\Sigma w_j C_j$ as the primary criterion that were already indicated as being easy, all variants of the problems with criteria $\Sigma w_j C_j$ and $L_{\max}$, $\Sigma w_j C_j$ and $f_{\max}$, and $\Sigma w_j C_j$ and $E_{\max}$ are strongly $\mathcal{NP}$-hard.

Theorem 10. *The problems* $1 | | F_h(\Sigma w_j C_j, L_{\max})$, $1 | | F_h(\Sigma w_j C_j, f_{\max})$, *and* $1 | | F_h(\Sigma w_j C_j, E_{\max})$ *are solved by sequencing the jobs according to nondecreasing* p_j / w_j *ratio, where ties are settled such that the secondary criterion is minimized.* □

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

Proof. We establish $\mathcal{NP}$-hardness in the strong sense for $1 | | F_h(L_{\max}, \Sigma w_j C_j)$ by showing that its decision variant belongs to $\mathcal{NP}$ and that an $\mathcal{NP}$-complete problem is polynomially reducible to it. The reduction is from the decision variant of the $1 | \bar{d}_j | \Sigma w_j C_j$ problem, which has been proven to be $\mathcal{NP}$-complete in the strong sense (Lenstra, Rinnooy Kan, and Brucker, 1977). In this formulation, $\bar{d}_j$ denotes that for each job a *deadline* has been specified; that is, J_j is not allowed to be completed after time $\bar{d}_j$ $(j = 1, \ldots, n)$. The decision variant of $1 | \bar{d}_j | \Sigma w_j C_j$ is defined as follows: given n jobs $J_1, \ldots, J_n$ with processing time p_j, weight w_j, and deadline $\bar{d}_j$ $(j = 1, \ldots, n)$, and a threshold value y, does there exist a feasible schedule with value no more than y?

As the weights are nonnegative, there is an optimal solution for $1 | \bar{d}_j | \Sigma w_j C_j$ without idle time. Therefore, we may assume without loss of generality that all deadlines are at most equal to Σp_j. Given an arbitrary instance of the decision variant of $1 | \bar{d}_j | \Sigma w_j C_j$, we construct the following instance of $1 | | F_h(L_{\max}, \Sigma w_j C_j)$. There are n jobs $J_1, \ldots, J_n$ that correspond to the jobs in the instance of $1 | \bar{d}_j | \Sigma w_j C_j$: the processing times and the weights are the same, and the due dates are equal to the deadlines.

Since all deadlines are assumed to be at most equal to Σp_j, the outcome of $1 | | L_{\max}$ is equal to 0, unless the instance of $1 | \bar{d}_j | \Sigma w_j C_j$ is infeasible. Hence, $1 | | F_h(L_{\max}, \Sigma w_j C_j)$ is identical to $1 | L_{\max} \leqslant 0, nmit | \Sigma w_j C_j$. Since the constraint

$L_{\max} \leqslant 0$ induces a deadline $d_j = \bar{d}_j$ for each job, the problems $1 \mid L_{\max} \leqslant 0 \mid \Sigma w_j C_j$ and $1 \mid \bar{d}_j \mid \Sigma w_j C_j$ are identical. Therefore, if we choose equal thresholds for both decision problems, then we have that an affirmative answer for the one problem always corresponds to an affirmative answer for the other problem. The only thing left to prove is that the decision variant of $1 \mid \mid F_h(L_{\max}, \Sigma w_j C_j$ belongs to $\mathcal{NP}$, which is obvious. □

THEOREM 12. *The* $1 \mid \mid F_h(f_{\max}, \Sigma w_j C_j)$ *problem is $\mathcal{NP}$-hard in the strong sense.*

PROOF. This follows immediately from Theorem 11. □

THEOREM 13. *The* $1 \mid nmit \mid F_h(E_{\max}, \Sigma w_j C_j)$ *problem is $\mathcal{NP}$-hard in the strong sense.*

PROOF. This follows immediately from Theorem 9. □

2.5. MINIMIZING A COMBINATION OF TWO MAXIMUM COST CRITERIA

We now consider the problems that arise when combining two criteria of the type $L_{\max}$, $f_{\max}$, and $E_{\max}$. The only combination that has attracted many researchers concerns $L_{\max}$ and $E_{\max}$. Two problems within this context have been studied extensively.

The first one is $1 \mid \mid \max\{E_{\max}, L_{\max}\}$. This problem has been addressed by Garey, Tarjan, and Wilfong (1988) as the problem of *sequencing tasks to minimize maximum discrepancy*; they show that it is solvable in $O(n \log \Sigma p_j)$ time.

The second one is $1 \mid nmit \mid E_{\max} + L_{\max}$. This problem is known as the problem of *minimizing the range of lateness*, since $E_{\max} = -L_{\min}$. It was introduced by Gupta and Sen (1984), who provide a branch-and-bound algorithm for it, with a lower bound based upon the *maximum improvement method.* Tegze and Vlach (1988) also provided a branch-and-bound algorithm for this problem with an improved lower bound based upon the method of *objective splitting*; see Hoogeveen and Van de Velde (1991; Paper IV in this thesis) for a comparison of these two lower bounding methods. A pseudopolynomial algorithm for this problem is due to Liao and Huang (1991). Hoogeveen (1990) proved that $1 \mid nmit \mid F(E_{\max}, L_{\max})$ is solved in $O(n^2)$ time, thereby making the aforementioned approaches obsolete. Furthermore, in this paper an $O(n^2 \log n)$ algorithm is given for determining the *trade-off curve* of $E_{\max}$ and $L_{\max}$, when idle time is allowed; the trade-off curve provides for each value E of $E_{\max}$ the outcome of $1 \mid E_{\max} \leqslant E \mid L_{\max}$. Hence, $1 \mid \mid F_l(E_{\max}, L_{\max})$ is solvable in $O(n^2 \log n)$ time.

The $1 \mid \mid F(f_{\max}, f_{\max})$ problem has been addressed by Tuzikov (1991), who proposed a method for determining the so-called ϵ-approximation of the set of Pareto optimal points, and by Hoogeveen (1991; Paper III in this thesis). Hoogeveen proves that $1 \mid \mid F(f_{\max}, f_{\max})$ can be solved in $O(n^4)$ time, even if there are arbitrary precedence constraints between the tasks. The special case $1 \mid \mid F(L_{\max}, f_{\max})$ is solved in $O(n^3 \log n)$ time; this algorithm can also be applied to solve $1 \mid nmit \mid F(E_{\max}, E_{\max})$. He further shows that this analysis can be extended to the K criteria case, thereby presenting an $O(n^{K(K+1)-6})$ algorithm for $1\,|\,|\,F(f^1_{\max},\ldots,f^K_{\max})$.

The only combination remaining is $f_{\max}$ and $E_{\max}$. We show that all variants of the problem combining these criteria are strongly $\mathcal{NP}$-hard by proving that $1\,|\,nmit\,|\,F_h(E_{\max},f_{\max})$ and $1\,|\,|\,F_h(f_{\max},E_{\max})$ are strongly $\mathcal{NP}$-hard.

THEOREM 14. *The problems* $1\,|\,nmit\,|\,F_h(E_{\max},f_{\max})$ *and* $1\,|\,|\,F_h(f_{\max},E_{\max})$ *are* $\mathcal{NP}$*-hard in the strong sense.*

PROOF. We simultaneously prove both problems to be $\mathcal{NP}$-hard in the strong sense. The reduction is from the strongly $\mathcal{NP}$-complete problem 3-PARTITION.

3-PARTITION
Given an integer B and a multiset $\mathcal{A}=\{a_1,\ldots,a_{3n}\}$ of $3n$ positive integers with $B/4<a_j<B/2$ $(j=1,\ldots,3n)$ and $\Sigma_{j=1}^{3n}a_j=nB$, is there a partition of $\mathcal{A}$ into n mutually disjoint subsets $\mathcal{A}_1,\ldots,\mathcal{A}_n$ such that the elements in $\mathcal{A}_j$ add up to B, for $j=1,\ldots,n$?

Given an arbitrary instance $\{a_1,\ldots,a_{3n}\}$ of 3-PARTITION, define the following problem instance. There are $4n$ tasks: n enforcer jobs $V_1,\ldots,V_n$, and $3n$ partition jobs $J_1,\ldots,J_{3n}$. The enforcer tasks have unit processing times, due dates equal to $d_j=j(B+1)$ $(j=1,\ldots,n)$, and penalty functions $f_j(T)=0$ for $0\leqslant T\leqslant j(B+1)$ and ∞ otherwise, for $j=1,\ldots,n$. The partition tasks have processing times $p_j=a_j$ $(j=1,\ldots,3n)$, due dates that are equal to processing times, and penalty functions $f_j(T)=0$ if $0\leqslant T\leqslant n(B+1)-1$, and ∞ otherwise, for $j=1,\ldots,3n$. Straightforward computations show that E^*, the outcome of $1\,|\,nmit\,|\,E_{\max}$, and f^*, the outcome of $1\,|\,|\,f_{\max}$, are both equal to zero. Hence, if we choose the thresholds for the decision variants of $1\,|\,nmit\,|\,F_h(E_{\max},f_{\max})$ and $1\,|\,|\,F_h(f_{\max},E_{\max})$ both equal to zero, then both decision variants boil down to the same question: does a schedule without idle time exist in which V_j is executed from time $j(B+1)-1$ to time $j(B+1)$, for $j=1,\ldots,n$? As such a schedule corresponds to a partitioning of the set $\mathcal{A}$ that provides an affirmative answer to 3-PARTITION, the decision variants of both scheduling problems are answered affirmatively if and only if 3-PARTITION is answered affirmatively. As both decision problems are in $\mathcal{NP}$, the problems $1\,|\,nmit\,|\,F_h(E_{\max},L_{\max})$ and $1\,|\,|\,F_h(f_{\max},E_{\max})$ are both $\mathcal{NP}$-hard in the strong sense. □

COROLLARY 15. *The problems* $1\,|\,|\,F(E_{\max},f_{\max})$, $1\,|\,nmit\,|\,F(E_{\max},f_{\max})$, $1\,|\,|\,F_l(E_{\max},f_{\max})$, $1\,|\,nmit\,|\,F_l(E_{\max},f_{\max})$ *are* $\mathcal{NP}$*-hard in the strong sense.* □

3. COMPLEXITY TABLES

Our complexity results are summarized in Tables 1, 2, and 3. Table 1 gives the results for hierarchical minimization, Table 2 for a general nondecreasing composite objective function, and Table 3 for a linear nondecreasing composite objective function. An exclamation point '!' indicates $\mathcal{NP}$-hardness in the ordinary sense; a double exclamation point '!!' indicates $\mathcal{NP}$-hardness in the strong sense.

secondary → primary ↓	$\sum_{j=1}^{n} C_j$	$\sum_{j=1}^{n} \hat{w}_j C_j$	$\hat{L}_{\max}$	$\hat{E}_{\max}$	$g_{\max}$
$\sum_{j=1}^{n} C_j$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
$\sum_{j=1}^{n} w_j C_j$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
$L_{\max}$	$O(n\log n)$	!!	$O(n\log n)$	$O(n^2\log n)$	$O(n^2)$
$E_{\max}$	!!	!!	$O(n\log n)$	$O(n\log n)$	!!
$f_{\max}$	$O(n^2)$	!!	$O(n^2)$	!!	$O(n^2)$

TABLE 1. Complexity results for hierarchical minimization.

	$\sum_{j=1}^{n} C_j$	$\sum_{j=1}^{n} \hat{w}_j C_j$	$\hat{L}_{\max}$	$\hat{E}_{\max}$	$g_{\max}$
$\sum_{j=1}^{n} C_j$	$O(n\log n)$	!	$O(n^3)$	!!	$O(n^4)$
$\sum_{j=1}^{n} w_j C_j$	!	!	!!	!!	!!
$L_{\max}$	$O(n^3)$	!!	$O(n^3\log n)$	!!	$O(n^3\log n)$
$E_{\max}$	!!	!!	!!	$O(n^3\log n)$	!!
$f_{\max}$	$O(n^4)$	!!	$O(n^3\log n)$	!!	$O(n^4)$

TABLE 2. Complexity results for a general composite objective function.

	$\sum_{j=1}^{n} C_j$	$\sum_{j=1}^{n} \hat{w}_j C_j$	$\hat{L}_{\max}$	$\hat{E}_{\max}$	$g_{\max}$
$\sum_{j=1}^{n} C_j$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$	!!	$O(n^4)$
$\sum_{j=1}^{n} \hat{w}_j C_j$	$O(n\log n)$	$O(n\log n)$	!!	!!	!!
$L_{\max}$	$O(n^3)$	!!	$O(n^3\log n)$	$O(n^2\log n)$	$O(n^3\log n)$
$E_{\max}$	!!	!!	$O(n^2\log n)$	$O(n^3\log n)$	!!
$f_{\max}$	$O(n^4)$	!!	$O(n^3\log n)$	!!	$O(n^4)$

TABLE 3. Complexity results for a linear composite objective function.

4. CONCLUDING REMARKS

In this survey, we have presented a review of the complexity results on single-machine bicriteria scheduling. This survey is complete with respect to the criteria under consideration. An interesting performance criterion that we not have dealt with concerns *the number of late jobs* ΣU_j, where U_j is an indicator function with value equal to 1 if J_j is late and 0 otherwise. The $1| \,|\Sigma U_j$ problem is solvable in $O(n\log n)$ time by an algorithm due to Moore and Hodgson (Moore, 1969). All of the single-machine bicriteria scheduling problems that are obtained by combining ΣU_j with another performance criteria are still open.

REFERENCES

U. BAGCHI (1989). Simultaneous minimization of mean and variation of flow time and waiting time in single machine systems. *Operations Research 37*, 118-125.

K.R. BAKER, E.L. LAWLER, J.K. LENSTRA, AND A.H.G. RINNOOY KAN (1983). Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Operations Research 31*, 381-386.

R.N. BURNS (1976). Scheduling to minimize the weighted sum of completion times with secondary criteria. *Naval Research Logistics Quarterly 23*, 125-129.

S. CHAND AND H. SCHNEEBERGER (1986). A note on the single-machine scheduling problem with minimum weighted completion time and maximum allowable tardiness. *Naval Research Logistics Quarterly 33*, 551-557.

H. EMMONS (1975). A note on a scheduling problem with dual criteria. *Naval Research Logistics Quarterly 22*, 615-616.

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

M.R. GAREY, R.E. TARJAN, AND G.T. WILFONG (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research 13*, 330-348.

S.K. GUPTA AND T. SEN (1984). Minimizing the range of lateness of a single machine. *Journal of the Operational Research Society 35*, 853-857.

H. HECK AND S. ROBERTS (1972). A note on the extension of a result on scheduling with secondary criteria. *Naval Research Logistics Quarterly 19*, 59-66.

J.A. HOOGEVEEN (1990). *Minimizing maximum earliness and maximum lateness on a single machine*, Report BS-R9001, CWI, Amsterdam.

J.A. HOOGEVEEN (1991). *Single-machine scheduling to minimize a function of K maximum cost criteria*, Report BS-R9113, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1990). *Polynomial-time algorithms for single-machine multicriteria scheduling*, Report BS-R9008, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1991). A new lower bound approach for single-machine multicriteria scheduling. To appear in *Operations Research Letters*.

J.R. JACKSON (1955). *Scheduling a production line to minimize maximum tardiness*, Research Report 43, Management Sciences Research Project, UCLA.

T.C. JOHN (1989). Tradeoff solutions in single machine production scheduling for minimizing flow time and maximum penalty. *Computers and Operations Research 16*, 471-479.

E.L. LAWLER (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science 19*, 544-546.

E.L. LAWLER, J.K. LENSTRA, AND A.H.G. RINNOOY KAN (1979). Unpublished manuscript.

J.K. LENSTRA (1979). Unpublished manuscript.

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

C.-J. LIAO AND R.-H. HUANG (1991). An algorithm for minimizing the range of lateness on a single machine. *Journal of the Operational Research Society 42*, 183-186.

S. MIYAZAKI (1981). One machine scheduling problem with dual criteria. *Journal of the Operations Research Society of Japan 24*, 37-50.

J.M. MOORE (1968). An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science 15*, 102-109.

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

A. SCHRIJVER (1989). Private communication.

T. SEN AND S.K. GUPTA (1983). A branch-and-bound procedure to solve a bicriterion scheduling problem. *IIE Transactions 15*, 84-88.

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

M. Tegze and M. Vlach (1988). Improved bounds for the range of lateness on a single machine. *Journal of the Operational Research Society 39*, 675-680.

A.V. Tuzikov (1990). *One approach to solving bicriterion scheduling problems,* Report 33, Academy of Sciences of Byelorussian SSR, Minsk.

L.N. Van Wassenhove and F. Gelders (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research 4*, 42-48.

This paper has been submitted for publication.

Minimizing maximum promptness and maximum lateness on a single machine

J.A. Hoogeveen

Minimizing maximum promptness and maximum lateness on a single machine

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

A set of n jobs has to be scheduled on a single machine that can handle only one job at a time. Each job J_i requires processing during a given positive uninterrupted time p_i, and has both a given target start time s_i and a given due date d_i, with $0 \leqslant d_i - s_i \leqslant p_i$. For each job J_i $(i = 1, \ldots, n)$, its promptness P_i is defined as the difference between the target start time and the actual start time, and its lateness L_i as the difference between the completion time and the due date. We consider the problem of finding a schedule that minimizes a function of maximum promptness $P_{\max} = \max_{1 \leqslant i \leqslant n} P_i$ and maximum lateness $L_{\max} = \max_{1 \leqslant i \leqslant n} L_i$, which 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. As a side-result, we prove that the special case of minimizing maximum lateness subject to release dates that lie in the interval $[d_i - p_i - A, d_i - A]$ $(i = 1, \ldots, n)$, for some constant A, is solvable in $O(n \log n)$ time if no machine idle time is allowed and in $O(n^2 \log n)$ time if machine idle time is allowed.

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

1. Introduction

Suppose that n independent jobs have to be scheduled on a single machine that 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 processing during a given positive uninterrupted time p_i, should ideally be started at a given *target start time* s_i, and should ideally be completed at a given *due date* d_i. A *schedule* defines for each job J_i a starting time S_i and a completion time $C_i = S_i + p_i$ such that the jobs do not overlap. Given a schedule σ, the *promptness* and the *lateness* of job J_i are defined by $P_i = s_i - S_i$ and $L_i = C_i - d_i$, respectively. Accordingly, the maximum promptness and the maximum lateness are defined by $P_{\max} = \max_{1 \leqslant i \leqslant n} P_i$ and $L_{\max} = \max_{1 \leqslant i \leqslant n} L_i$, respectively. If the target start times s_i are equal to $d_i - p_i$ for $i = 1, \ldots, n$, then the maximum promptness criterion coincides with the well-known maximum *earliness* criterion $E_{\max} = \max_{1 \leqslant i \leqslant n} (d_i - C_i)$. We consider the problem of finding a schedule σ

that minimizes the scheduling cost $f(\sigma) = F(P_{\max}(\sigma), L_{\max}(\sigma))$, where F is a given function that is nondecreasing in both arguments, subject to the constraint $s_j \in [d_j - p_j, d_j]$, for $j = 1, \ldots, n$. We will also consider a variant of the problem in which idle time is not allowed. Using the three-field notation scheme $\alpha \mid \beta \mid \gamma$ introduced by Graham, Lawler, Lenstra, and Rinnooy Kan (1979), where α describes the machine environment, β the job characteristics, and γ the objective function, the first problem is denoted by $1 \mid \mid F(P_{\max}, L_{\max})$ and the second by $1 \mid nmit \mid F(P_{\max}, L_{\max})$, where *nmit* expresses the no machine idle time restriction.

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 schedule is optimal with respect to the primary criterion. Examples are minimizing the sum of completion times subject to minimal maximum lateness [Smith, 1956], and minimizing maximum lateness subject to a minimal number of late jobs [Shanthikumar, 1983]. Only a few of the papers on bicriteria scheduling consider simultaneous optimization, in which the criteria are transformed into a single composite objective function. An example is minimizing the number of late jobs and maximum lateness simultaneously [Nelson et al., 1986]. Most contributions to the area of bicriteria scheduling concern branch and bound algorithms. There are some notable exceptions, however. Garey, Tarjan, and Wilfong [1988] present an $O(n(\log \Sigma p_i))$ algorithm to solve $1 \mid \mid \max\{E_{\max}, L_{\max}\}$. Hoogeveen and Van de Velde [1990] present an $O(n^3 \min\{n, \log \Sigma p_i\})$ time algorithm for $1 \mid \mid F(f_{\max}, \Sigma C_i)$; $f_{\max}$ is an arbitrary maximum cost function, defined by $f_{\max}(\sigma) = \max\{f_i(C_i(\sigma)) \mid i = 1, \ldots, n\}$, where all functions f_i are assumed to be nondecreasing in the job completion times. Furthermore, they present an $O(n^4)$ time algorithm for $1 \mid \mid \alpha E_{\max} + \Sigma C_i$, with $\alpha \leqslant 1$.

The organization of this paper is as follows. In Section 2, we repeat some basic theory, and we present a strategy to obtain the set of Pareto optimal points. In Section 3, we derive a dominance rule, which can be applied to both variants of the problem. We further show that the subclass of $1 \mid r_j, nmit \mid L_{\max}$ with $r_j \in [d_j - p_j - A, d_j - A]$ $(j = 1, \ldots, n)$, for an arbitrary constant A, is solvable in $O(n \log n)$ time. In Section 4, we apply the strategy formulated in Section 2 to determine the set of Pareto optimal points for the case in which no machine idle time is allowed. In Section 5, we drop this constraint and analyze the general problem, which we prove to be $\mathcal{NP}$-hard in Section 6, but solvable in $O(n^2 \log n)$ time if the function $F(P_{\max}, L_{\max})$ is linear.

2. Basic concepts

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

MINIMUM TARGET START TIME (MTST) RULE. *If no idle time is allowed, then $P_{\max}$ is minimized by sequencing the jobs in order of nondecreasing values of s_i.*

EARLIEST DUE DATE (EDD) RULE [Jackson, 1955]. *$L_{\max}$ is minimized by sequencing the jobs in order of nondecreasing due dates d_i.*

The *MTST* rule forms a generalization of the *EDD* rule (see Theorem 3). If both orderings coincide, then the corresponding job sequence solves both $1 \mid \mid F(P_{\max}, L_{\max})$ and $1 \mid nmit \mid F(P_{\max}, L_{\max})$. However, in general both orderings will differ and it is unlikely that a single sequence minimizes both $P_{\max}$ and $L_{\max}$. Hence, in order to solve $1 \mid \mid F(P_{\max}, L_{\max})$ 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 $P_{\max}$ and $L_{\max}$.

DEFINITION 1. A feasible schedule σ is *Pareto optimal* with respect to the objective functions $f_1,\ldots,f_K$ if there is no feasible schedule π with $f_k(\pi) \leqslant f_k(\sigma)$ for $k = 1, \ldots, K$, where at least one of the inequalities is strict.

THEOREM 1. *Consider the composite objective function $F(f_1(\sigma), \ldots, f_K(\sigma))$, where F is nondecreasing in each argument. Then there is a Pareto optimal schedule with respect to $f_1,\ldots,f_K$ 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.

We start by analyzing $1 \mid nmit \mid F(P_{\max}, L_{\max})$. In order to determine the set of Pareto optimal points with respect to $(P_{\max}, L_{\max})$ subject to the constraint *nmit*, we apply the following strategy. First, given a value P of $P_{\max}$ that corresponds to a possibly Pareto optimal point (P,L) for $(P_{\max}, L_{\max})$, we solve $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ to obtain L. Second, we determine the next $P_{\max}$-value that corresponds to a possibly Pareto optimal point.

There are three difficulties hidden in applying the above strategy. The first problem concerns the choice of the start value P of $P_{\max}$. This problem can easily be overcome by choosing P equal to the $P_{\max}$-value of the *MTST* schedule; obviously, there can be no Pareto optimal point with smaller $P_{\max}$-value.

The second problem is how to solve $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$. The constraint $P_{\max} \leqslant P$ induces for each J_j a release date r_j, that is, a lower bound for the start time S_j; $P_{\max} \leqslant P$ implies $s_j - S_j \leqslant P$, for $j = 1, \ldots, n$, and hence $S_j \geqslant s_j - P$, for $j = 1, \ldots, n$. Hence, the problems $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ and $1 \mid r_j = s_j - P, nmit \mid L_{\max}$ are identical. Although the $1 \mid r_j, nmit \mid L_{\max}$ problem with general release dates is $\mathcal{NP}$-hard in the strong sense [Lenstra, Rinnooy Kan, and Brucker, 1977], we show in Section 3 that $1 \mid r_j = s_j - P, nmit \mid L_{\max}$ is solvable in $O(n \log n)$ time if $s_j \in [d_j - p_j, d_j]$.

The third problem is how to determine the next $P_{\max}$-value that corresponds to a possibly Pareto optimal point in such a way that the total number of generated points is not too great. Obviously, if we increase the $P_{\max}$-value by one every time, then we will certainly determine all Pareto optimal points, but this approach will not yield a polynomial time algorithm.

Finally, we have to determine the complexity of the algorithm, and hence, we have to find a bound on the number of points that need to be generated in order to determine all Pareto optimal points.

We will extend the above strategy to deal with the $1 \mid \mid F(P_{\max}, L_{\max})$ problem in Section 5.

3. A polynomially solvable subclass of $1 \mid r_j, nmit \mid L_{\max}$

Lenstra, Rinnooy Kan, and Brucker [1977] prove that the general $1 \mid r_j, nmit \mid L_{\max}$ problem is $\mathcal{NP}$-hard in the strong sense. The problem is solvable in polynomial time, however, if all release dates are equal, if all due dates are equal, if all processing times are equal, or if preemption is allowed, that is, if the processing of a job can be stopped and resumed later. For a review, we refer to the survey by Lawler, Lenstra, Rinnooy Kan, and Shmoys [1989]. Furthermore, if the release dates and the due dates are similarly ordered, then the corresponding $1 \mid r_j \mid L_{\max}$ is solvable in $O(n \log n)$ time, as follows immediately from the analysis in Section 2 by the choice $s_j = r_j$ $(j = 1, \ldots, n)$, $F(P_{\max}, L_{\max}) = \infty$ if $P_{\max} > 0$, and $F(P_{\max}, L_{\max}) = L_{\max}$ if $P_{\max} \leqslant 0$.

In this section, we consider a subclass of the $1 \mid r_j\, nmit \mid L_{\max}$ problem in which the release dates do not depend on the jobs, but on the *position*; we use $[k]$ as a subscript to denote the kth position. Let $K = (K_1, \ldots, K_n)$, with $K_i \leqslant K_{i+1}$ $(i = 1, \ldots, n-1)$ denote a vector in $\mathbb{R}^n$. The problem under consideration is denoted as $1 \mid r_{[k]} = s_j - K_k, nmit \mid L_{\max}$; J_j can be started at the kth position in σ if $s_j - K_k \leqslant C_{[k-1]}(\sigma)$. Note that the $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ problem belongs to this subclass. We prove that this problem can be solved by the *extended Jackson rule: always keep the machine assigned to the available job with the smallest due date.* We have modified the rule such that ties are settled according to nondecreasing value of s_j.

Algorithm A.

```
(0) $T \leftarrow 0$; $k \leftarrow 1$; $U \leftarrow \{J_1, \ldots, J_n\}$; $V \leftarrow \varnothing$.
{Initialization: $T$ denotes the start time of the job in the $k$th position.}
(1) For each job $J_j \in U$: if $s_j - K_k \leqslant T$ then $V \leftarrow V \cup \{J_j\}$ and $U \leftarrow U \setminus \{J_j\}$.
{$U$ 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 $T$.}
(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 target
    completion time. If there are still ties, then choose the job with the smallest
    index. Suppose that $J_i$ is chosen. Assign $J_i$ to the $k$th position.
(3) $T \leftarrow T + p_i$; $k \leftarrow k + 1$; $V \leftarrow V - \{J_i\}$.
(4) If there are unassigned jobs left, then go to 1.
```

We need a preliminary lemma and a dominance rule before proving that Algorithm A solves $1 \mid r_{[k]} = s_j - K_k, nmit \mid L_{\max}$, LEMMA 1. *Consider an arbitrary schedule σ. Let J_i and J_j be two jobs, where J_i is scheduled before J_j in σ. If J_j cannot be started as soon as J_i is finished, or, if $L_i(\sigma) > L_j(\sigma)$, then both $s_j > s_i$ and $d_j > d_i$.*

PROOF. Let J_i be assigned to the kth position in σ. If J_j is not available at time $C_i(\sigma)$, then $s_j - K_{k+1} > C_i(\sigma)$. As $C_i(\sigma) - p_i \geqslant s_i - K_k$, and as $K_k \leqslant K_{k+1}$, we obtain $s_j > s_i + p_i$, and hence $d_j \geqslant s_j > s_i + p_i \geqslant d_i - p_i + p_i = d_i$. As to the second case, $L_i(\sigma) > L_j(\sigma)$ implies $C_i(\sigma) - d_i > C_j(\sigma) - d_j$. As J_i is scheduled before J_j, $C_j(\sigma) \geqslant C_i(\sigma) + p_j$. The combination of these two inequalities yields $d_j - p_j > d_i$, and hence $s_j \geqslant d_j - p_j > d_i \geqslant s_i$. □

DOMINANCE RULE. *Let J_i and J_j be two arbitrary jobs. If both $s_i \leqslant s_j$ and $d_i \leqslant d_j$, where at least one of the inequalities is strict, then there exists an optimal schedule for $1 \mid r_{[k]} = s_j - K_k, nmit \mid L_{\max}$ in which J_i precedes J_j.*

PROOF. We will show that an optimal schedule σ that does not satisfy the dominance rule can be transformed by applying interchanges (not necessarily adjacent) into a feasible schedule σ that is optimal and that satisfies the dominance rule.

Consider an optimal schedule σ that contains two jobs J_i and J_j that satisfy the conditions of the dominance rule, while J_j precedes J_i in σ. Let J_i and J_j be chosen such that the jobs scheduled between J_i and J_j in σ satisfy the order of the dominance rule, implying that there is no job J_l scheduled between J_i and J_j that has both $s_l < s_j$ and $d_l < d_j$, or both $s_l > s_i$ and $d_l > d_i$.

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

(1) $\bar{\sigma}$ is feasible with respect to the release dates.

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

Proof of (1). As $s_i \leqslant s_j$, $\bar{\sigma}$ is feasible with respect to the release dates of J_i and J_j. Suppose that there is a job J_l in $\bar{\sigma}$ scheduled between J_i and J_j that starts before its release date in $\bar{\sigma}$. Hence, J_l can not be started when J_i is completed, while J_i precedes J_l in $\bar{\sigma}$. Application of Lemma 1 yields $s_l > s_i$ and $d_l > d_i$, contradicting the assumption.

Proof of (2). As $d_i \leqslant d_j$; the second claim holds with respect to J_i and J_j. Suppose that there is a job J_l in $\bar{\sigma}$ scheduled between J_i and J_j, with $L_l(\bar{\sigma}) > L_i(\sigma)$. As $d_i \leqslant d_j$, we have $L_i(\sigma) = C_i(\sigma) - d_i \geqslant C_i(\sigma) - d_j = C_j(\bar{\sigma}) - d_j = L_j(\bar{\sigma})$, so the lateness of J_l is greater than the lateness of J_j in schedule $\bar{\sigma}$ while J_l precedes J_j. Application of Lemma 1 yields $s_j > s_l$ and $d_j > d_l$, contradicting the assumption.

The interchange argument can be repeated until a schedule is obtained that satisfies the dominance rule. This schedule is also feasible and optimal. □

THEOREM 2. *The* $1 \mid r_{[k]} = s_j - K_k, nmit \mid L_{\max}$ *problem is solved to optimality by Algorithm A.*

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

Compare σ and $\bar{\sigma}$, starting at the front. Suppose the first difference occurs in the kth position; let J_i be scheduled in the kth position in σ and let J_j be scheduled in the kth position in $\bar{\sigma}$.

Let $\hat{\sigma}$ be the schedule that results when J_i and J_j are interchanged in $\bar{\sigma}$. It now suffices to prove the following claims in order to prove that σ is an optimal schedule that is feasible with respect to the release dates.

(1) $\hat{\sigma}$ is feasible with respect to the release dates.

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

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

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

Proof of (2). Analogous to the proof of claim (2) in the dominance rule, this is proven by showing that the lateness in $\hat{\sigma}$ of J_j and the jobs scheduled between J_i and J_j in $\bar{\sigma}$ does not exceed $L_i(\bar{\sigma})$. Because of the construction of σ, we must have $d_i \leqslant d_j$, and hence, $L_j(\hat{\sigma}) \leqslant L_i(\bar{\sigma})$. Consider an arbitrary job J_l, scheduled between J_j and J_i in $\bar{\sigma}$. Suppose that $L_l(\hat{\sigma}) > L_i(\bar{\sigma})$. As $L_i(\bar{\sigma}) \geqslant L_j(\hat{\sigma})$, we then have $L_l(\hat{\sigma}) > L_j(\hat{\sigma})$, while J_l precedes J_j in $\hat{\sigma}$. Application of Lemma 1 yields $s_j > s_l$, and $d_j > d_l$, contradicting the assumption that $\bar{\sigma}$ satisfies the dominance rule. This implies that for each J_l, scheduled between J_j and J_i, the lateness of J_l in $\hat{\sigma}$ does not exceed the value of $L_{\max}(\bar{\sigma})$. This completes the proof of (2).

Proof of (3). Unfortunately, $\hat{\sigma}$ does not necessarily have to obey the dominance rule, as a job J_l can exist that is scheduled between J_j and J_i in $\bar{\sigma}$, with $d_l > d_j \geqslant d_i$ and $s_i > s_l > s_j$; in that case, interchanging J_j and J_i yields a schedule that does not satisfy the dominance rule. From the proof of the dominance rule, however, it follows immediately that $\hat{\sigma}$ can then be adjusted to a new schedule, named $\bar{\sigma}$ again, that is also feasible and optimal, that satisfies the dominance rule, and in which the first k jobs are the same as in σ.

The interchange argument can be repeated until the schedule σ and the newly obtained schedule $\bar{\sigma}$ are the same. This proves that σ, obtained through Algorithm A, is an optimal schedule that is feasible with respect to the release dates, and that satisfies the dominance rule. □

THEOREM 3. *Algorithm A solves the problems* $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ *and* $1 \mid L_{\max} \leqslant L, nmit \mid P_{\max}$ *to optimality.*

PROOF. The proof of the first part follows immediately from Theorem 2, as $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ is identical to $1 \mid r_j = s_j - P, nmit \mid L_{\max}$.

As to the second part of the proof, consider an arbitrary instance $V_1 = \{p_{11}, d_{11}, s_{11}, \ldots, p_{n1}, d_{n1}, s_{n1}, L\}$ of $1 \mid L_{\max} \leqslant L, nmit \mid P_{\max}$. Now construct the following instance $V_2 = \{p_{12}, d_{12}, s_{12}, \ldots, p_{n2}, d_{n2}, s_{n2}, P\}$ for $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$:

$$p_{i2} = p_{i1} \text{ for } i = 1, \ldots, n,$$

$$d_{i2} = \sum_{j=1}^{n} p_{j1} - s_{i1} \text{ for } i = 1, \ldots, n,$$

$$s_{i2} = \sum_{j=1}^{n} p_{j1} - d_{i1} \text{ for } i = 1, \ldots, n,$$

$$P = L.$$

Suppose that the application of Algorithm A to V_2 yields σ_2. An optimal schedule σ_1 for V_1 is then obtained by reversing σ_2, since

$$L_i(\sigma_2) = C_i(\sigma_2) - d_{i,2} = C_i(\sigma_2) - \underset{j=1}{\overset{n}{sum}} p_{j,1} + s_{i,1} = s_{i,1} - S_i\sigma_1) = P_i(\sigma_1).$$

$$P_i(\sigma_2) = s_{i,2} - S_i(\sigma_2) = \sum_{j=1}^{n} p_{j,1} - d_{i,1} - S_i(\sigma_2) = C_i(\sigma_1) - d_{i,1} = L_i(\sigma_1).$$

This implies that σ_1 is optimal and feasible if and only if σ_2 is optimal and feasible. □

4. Pareto optimal points if no idle time is allowed

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

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

Proof. Suppose to the contrary 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 σ, job J_k cannot be available when J_j is selected, and hence, it cannot be started as soon as J_i is finished. Application of Lemma 1 yields that therefore $d_k > d_i$, contradicting the assumption. □

Given an optimal schedule σ for $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ obtained by Algorithm A, where P is an arbitrary value of the upper bound on $P_{\max}$, it is possible to decrease $L_{\max}$ only by interchanging two jobs that are not scheduled in *EDD* order. We prove that σ 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 σ into blocks according to the following algorithm.

PARTITIONING ALGORITHM.
(0) Start at the beginning of the schedule.
(1) Select the next job J_i to be the first job in a block. Compare the due date of 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 J_i and all its successors up to J_k.
(2) Proceed until the schedule has been completely partitioned into blocks.

PROPOSITION 1. *Let σ be an optimal schedule for the $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ problem obtained by Algorithm A, where P is an arbitrary upper bound on $P_{\max}$, with $P \geqslant P^{MTST}$, where P^{MTST} is equal to $P_{\max}$ (MTST). Partition σ 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 σ 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 $d_j < d_i$ that is scheduled after J_i and 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 $d_l \geqslant d_i > d_j$. 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 P_1 and P_2 be two arbitrary values of the upper bound on $P_{\max}$, with $P_1 \leqslant P_2$. Let σ_1 and σ_2 be the optimal schedules obtained by applying Algorithm A to $1 \mid P_{\max} \leqslant P_1, nmit \mid L_{\max}$ and $1 \mid P_{\max} \leqslant P_2, nmit \mid L_{\max}$, respectively. Partition σ_1 into blocks according to the Partitioning algorithm. Let B be an arbitrary block of σ_1, let T_1 and T_2 be the start and completion time of block B in σ_1, respectively. Then the jobs belonging to B are processed in σ_2 during the interval $[T_1, T_2]$.*

PROOF. Consider the first block B of σ_1. Let J_j be an arbitrary job that does not belong to B. As $P_2 \geqslant P_1$, it is feasible to schedule all jobs of B in the same position as in σ_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 σ_1 and so on, until only the last block remains. □

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

PROOF. $L_{\max}(\sigma)$ 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 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 at most equal to n.

THEOREM 6. *Let σ_1 and σ_2 be two schedules obtained by Algorithm A, which both correspond to a Pareto optimal point, (P_1, L_1) and (P_2, L_2), respectively. Suppose $P_1 < P_2$. Partition σ_1 and σ_2 into blocks by applying the Partitioning algorithm. The number of blocks into which σ_1 has been partitioned is smaller than the number of blocks in which σ_2 has been partitioned.*

PROOF. Application of Theorem 4 proves that σ_2 has been partitioned in at least as many blocks as σ_1. Furthermore, Theorem 5 implies that, in order to achieve a lower value of $L_{\max}$, at least one of the blocks, in which σ_1 has been partitioned, must have been split in at least two blocks in σ_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 that have the corresponding schedules, obtained by Algorithm A, partitioned by the Partitioning algorithm into the same number of blocks. Hence, the number of Pareto optimal points is bounded by n.

The following example shows that the bound is tight:

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

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

It is easily verified that the schedules $(J_n, J_1, J_2, \ldots, J_{n-1})$, $(J_1, J_n, J_2, J_3, \ldots, J_{n-1}), \ldots, (J_1, J_2, J_3, \ldots, J_n)$ all correspond to Pareto optimal points. □

From Theorem 5 it follows immediately that a new Pareto optimal point can only be obtained by increasing the value P of the upper bound on $P_{\max}$ 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 σ for $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$, determines the next $P_{\max}$-value that corresponds to a Pareto optimal point.

ALGORITHM NEXT P

(0) Partition σ into blocks, according to the Partitioning algorithm.

(1) Determine for each block B that contains a job J_i with $L_i(\sigma) = L_{\max}(\sigma)$ the value of the upper bound P such that another job in B is allowed to be scheduled in the first position. If J_i is the only job in B, then $L_{\max}$ cannot be decreased; stop.

(2) Choose the maximum of the values found at Step 1. This maximum is the new value P.

A straightforward implementation of all properties derived above yields an algorithm that determines all Pareto optimal schedules in $O(n^2 \log n)$ time. We can, however, gain a little by not determining the Pareto optimal *schedules* but the Pareto optimal *points*; after selecting the point that yields minimal $F(P_{\max}, L_{\max})$ value, the corresponding schedule is easily obtained.

The algorithm highly depends upon the properties of the blocks. Assume that the jobs are numbered in order of nondecreasing due dates, where ties are settled according to nondecreasing target start times. Consider an arbitrary block B; suppose that it contains the jobs $\{J_i, \ldots, J_l\}$. Then the completion time of each one of the jobs $\{J_i, \ldots, J_{l-1}\}$ is equal to its completion time in the EDD schedule plus p_l. Hence, the blockwise maximum lateness for B, that is, the maximum lateness within B, is attained by the job in $\{J_i, \ldots, J_{l-1}\}$ that has maximum lateness in the EDD schedule. Furthermore, the job in B that will occupy the first position in B when the upper bound on $P_{\max}$ is increased minimally is the job in $\{J_i, \ldots, J_{l-1}\}$ that has minimum target start time; the necessary minimal increase of P is equal to $s_j - C_{i-1}(EDD) - P$.

The above observations show that, once the necessary orders are stored per block, we can determine the $L_{\max}$-value within B and the upper bound value that is needed to alter the sequence within B in constant time. Hence, if we store the blockwise $L_{\max}$-values and the next upper bound values in an ordered tree, then we can determine the next interesting $P_{\max}$-value and the corresponding $L_{\max}$-value in $O(\log n)$ time. As partitioning of the orders according to the blocks takes $O(n)$ time, the running time of Algorithm B is $O(n^2)$. The proof of correctness follows from the observations made above.

ALGORITHM B

(0) Solve $1 \mid nmit \mid P_{\max}$, yielding P^{MTST}, and solve $1 \mid P_{\max} \leqslant P^{MTST}, nmit \mid L_{\max}$, yielding σ; store $(P_{\max}(\sigma), L_{\max}(\sigma))$. Partition σ into blocks by applying the Partitioning algorithm.

(1) Determine the $MTST$-order, the $L_j(EDD)$-order, in which the jobs are ordered according to nonincreasing lateness in the EDD-schedule, and $C_j(EDD)$, for $j = 1, \ldots, n$. Partition the $MTST$ and the $L_j(EDD)$ order

according to the partitioning of σ. Determine for each block B its blockwise maximum lateness value and its next upper bound value. Store these values in an ordered tree.

(2) Determine the minimum element in the ordered tree containing the blockwise next upper bound values; let this be P. If $P = \infty$, then go to 5; all interesting points have been discovered. Suppose that P originates from job J_k; let this job be contained in the block B that contains the jobs $\{J_i, \ldots, J_l\}$.

(3) Split B, the *MTST*-order, and the L_j (*EDD*)-order in two parts; the first part contains the jobs $J_i, \ldots, J_k$, while the second part contains the jobs $J_{k+1}, \ldots, J_l$. Determine for both parts the blockwise maximum lateness value; store these values in the ordered tree and delete the former maximum lateness value corresponding to B from the tree. Determine for the second part the blockwise next upper bound value; let this be P_1. If $P_1 \leqslant P$, then the second part has to be split further. This can be done in the same fashion as described above. This process repeats until the current P_1-value has become greater than P. If $P_1 > P$, then we have obtained an interesting point with $P_{\max}$-value equal to P and $L_{\max}$-value equal to the maximal element in the ordered tree containing the blockwise maximum lateness values.

(4) Store this point, and go to 2.

(5) Compute for each of the interesting points its F-value, and choose the minimum. Suppose the minimum is attained by the point (P,L). The corresponding optimal schedule is then determined by solving $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ through the extended Jackson rule.

5. Pareto optimal schedules if idle time is allowed

We prove that the $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem can be solved in $O(n^2 \log n)$ time, given P. Furthermore, we show that the trade-off curve of $P_{\max}$ and $L_{\max}$, defined as the curve that connects all points (P,L), where L is the outcome of $1 \mid P_{\max} \leqslant P \mid L_{\max}$, is piecewise linear with gradient alternately -1 and 0 and that it can be computed in $O(n^2 \log n)$ time.

Consider the $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem. As $L_{\max}$ is a *regular* performance measure, implying that its value cannot be decreased by inserting idle time into a given schedule that is feasible with respect to $P_{\max} \leqslant P$, 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 available job that has the smallest due date in the kth position, it now may be advantageous to wait until another job with smaller due date becomes available. Although this looks similar to increasing the upper bound P to allow a job with smaller due date to be sequenced next, as in the previous section, both situations differ tremendously with respect to the consequences. Inserting idle time affects the completion times of *all* jobs still to come, even if the sequence in which the remaining jobs are scheduled stays the same, in contrast to increasing P in the previous section. We need *position* *dependent* release dates, as defined in Section 3, in order to prevent unnecessary changes in the beginning of the schedule that possibly increase $L_{\max}$. We show that we can use the insight gained in the analysis of the $1 \mid P_{\max} \leqslant P, nmit \mid L_{\max}$ problem to deal with the $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem.

The $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem is not easily accessible itself. Therefore, we will solve it by formulating it in a different way. To this representation we apply a strategy that is very similar to the one used in Section 4. Consider an optimal schedule σ for $1 \mid P_{\max} \leqslant P \mid L_{\max}$; let $\bar{\sigma}$ denote the corresponding sequence that remains after removing the idle time. Define $P_{[i]}(\bar{\sigma})$ as the promptness of the job in the ith position in $\bar{\sigma}$; define $K_i = \max_{1 \leqslant k \leqslant i} P_{[i]}$ $(i = 1, \ldots, n)$. The total amount of idle time that has to be inserted into $\bar{\sigma}$ before $J_{[i]}$ to make it feasible with respect to the constraint $P_{\max} \leqslant P$ is equal to $\max\{K_i - P, 0\}$. Hence, the $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem can alternatively be formulated as the problem $1 \mid P_{[i]} \leqslant K_i, nmit \mid \max_{1 \leqslant i \leqslant n} \{L_{[i]} + \max\{K_i - P, 0\}\}$, where the set of constraints $P_{[i]} \leqslant K_i$ induces the set of positional release dates $r_{[i]} = s_j - K_i$ $(i = 1, \ldots, n; j = 1, \ldots, n)$. One way to solve this problem is by using a step-wise approach: given a nondecreasing vector of upper bounds K^j that possibly corresponds to an optimal solution of the above problem, determine K^{j+1} by increasing at least one component of K^j such that $L_{[i]}$ is decreased, where $J_{[i]}$ is the job that attains $\max_{1 \leqslant k \leqslant n} \{L_{[k]} + \max\{K_k - P, 0\}\}$. Note that, given a vector $K^j = (K_1^j, \ldots, K_n^j)$ of upper bounds with $K_i^j \leqslant K_{i+1}^j$ for $i = 1, \ldots, n-1$, the optimal set of $L_{[i]}$-values is found by solving $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ through Algorithm A; hence, for simplicity, we denote the problem of determining the $L_{[i]}$-values given a vector K^j of upper bounds as $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$. As the sequence without idle time has to be made feasible with respect to $P_{\max} \leqslant P$ by inserting idle time, there is no use in considering vectors $K = (K_1, \ldots, K_n)$ that are not *nondecreasing*, that is, that do not satisfy $K_i \leqslant K_{i+1}$ for $i = 1, \ldots, n-1$. Further note that every component of K^{j+1} has to be greater than or equal to the corresponding component of K^j. Otherwise, the schedule that solves $1 \mid P_{[i]} \leqslant K_i^{j+1}, nmit \mid L_{\max}$ does not lead to a smaller $L_{[i]}$-value than the schedule that solves $1 \mid P_{[i]} \leqslant K_i^h, nmit \mid L_{\max}$, where $J_{[i]}$ is the job that attains $\max_{1 \leqslant k \leqslant n} \{L_{[k]} + \max\{K_k - P, 0\}\}$, and where $K_i^h = \min\{K_i^j, K_i^{j+1}\}$; hence, K^{j+1} then cannot correspond to an optimal schedule. Let $\{K^1, \ldots, K^m\}$ be the set of vectors that are obtained by applying the above step-wise approach. Define σ_j as the sequence obtained by solving $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ through Algorithm A; define $\sigma_j(P)$ as the active schedule that is obtained when σ_j is made feasible with respect to the constraint $P_{\max} \leqslant P$. Obviously, the vectors $\{K^1, \ldots, K^m\}$ have to be *minimal*, that is, if one of the components of K^j $(j = 1, \ldots, m)$ is decreased, then σ_j must become infeasible with respect to the constraints $P_{[i]} \leqslant K_i^j$.

We have now come to the point of implementing the above step-wise approach to determine the set of upper bound vectors $\{K^1, \ldots, K^m\}$ that yield a possibly optimal schedule $\sigma_j(P)$ for $1 \mid P_{\max} \leqslant P \mid L_{\max}$. We show that the set of vectors $\{K^1, \ldots, K^m\}$ is obtained in a similar fashion as the set of $P_{\max}$-values corresponding to a possibly Pareto optimal point in the previous section.

Consider a vector K from the set $\{K^1, \ldots, K^m\}$. Let k and l be such that $K_{k-1} < K_k = \ldots = K_l < K_{l+1}$. 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 $\sigma(P)$ contains no idle time within a group of jobs. Define the maximum lateness within the group G as $L(G)_{\max} = \max\{L_{[i]}(\sigma) \mid J_{[i]} \in G\}$ and define $K(G)$ as the common K-value for the group of positions corresponding to G.

PROPOSITION 2. *Consider a sequence σ_j obtained when solving $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ through Algorithm A, where $K^j \in \{K^1, \ldots, K^m\}$. Partition σ_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 $J_{[j]}$ be two arbitrary jobs in G_1 and G_2, respectively. Then $d_{[i]} < d_{[j]}$.*

PROOF. Let $J_{[l]}$ be the first job in G_2. As K^j is minimal, K_l^j cannot be decreased, hence, $J_{[l]}$ cannot be started as soon as $J_{[i]}$ is completed. Application of Lemma 1 yields $s_{[i]} < s_{[l]}$ and $d_{[i]} < d_{[l]}$. As σ_j is obtained by Algorithm A, either $s_{[j]} \geqslant s_{[l]}$ or $d_{[j]} \geqslant d_{[l]}$. As to the first case, $J_{[j]}$ cannot be started as soon as $J_{[i]}$ is completed, implying $d_{[j]} > d_{[i]}$; as to the second case, we obtain $d_{[j]} \geqslant d_{[l]} > d_{[i]}$. □

THEOREM 7. *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 σ_j and σ_k be the optimal sequences obtained by applying Algorithm A to $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ and $1 \mid P_{[i]} \leqslant K_i^k, nmit \mid L_{\max}$, respectively. Partition σ_j and σ_k in groups. Let G_1 and G_2 be two arbitrary groups of jobs in σ_j, where G_1 is completed before G_2 in σ_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 σ_k.*

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

From Theorem 7 it follows immediately that if we start with a vector K for the upper bound on $P_{[i]}$ 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.

(0) Let K be a given vector of upper bounds on $P_{[i]}$. Let σ be the sequence obtained by applying Algorithm A to $1 \mid P_{[i]} \leqslant K_i, nmit \mid L_{\max}$.

(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 the Partitioning algorithm.

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

(3) Determine for each block B in $\mathfrak{B}$ how much the upper bound $K(G)$ has to be increased to let another job within B be scheduled in the first position in B. Denote this value by $K(B)$. If B consists of a single job, then $K(B) = \infty$ and

hence, $L(G)_{\max}$ cannot be decreased; stop.

(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 these are the positions $k+1,\ldots,l$. The new upper bound value K_{i+1} becomes equal to $\max\{K_i, K(B)\}$, where B is the block that contains 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. The time complexity of the algorithm is $O(n)$.

We are now able to formulate an algorithm to determine all vectors $K^1,\ldots,K^m$ and the corresponding sequences $\sigma_1,\ldots,\sigma_m$. Let K^{MTST} denote the vector with $K_i^{MTST} = \max\{P_{[j]}(MTST) \mid j = 1,\ldots,i\}$, for $i = 1,\ldots,n$.

ALGORITHM C.

(0) Determine the vector K^{MTST}; $l \leftarrow 1$; $K^l \leftarrow K^{MTST}$.

(1) Solve $1 \mid P_{[i]} \leqslant K_i^l, nmit \mid L_{\max}$ by applying Algorithm A; this yields sequence σ_l.

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

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

Screen the set of vectors $\{K^1,\ldots,K^m\}$ in to remove all vectors that lead to dominated sequences. A sequence σ_{j+1} is dominated by sequence σ_j if $L_{\max}(\sigma_j) \leqslant L_{\max}(\sigma_{j+1})$. We now prove that at most n upper bound vectors are determined by Algorithm C.

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 σ_j and σ_k be the optimal sequences obtained by applying Algorithm A to $1 \mid P_{[i]} \leqslant K_i^j, nmit \mid L_{\max}$ and $1 \mid P_{[i]} \leqslant K_i^k, nmit \mid L_{\max}$, respectively. Partition σ_j and σ_k into blocks according to the Partitioning algorithm. Then σ_k is partitioned into more blocks than σ_j. Hence, Algorithm C computes at most n schedules, and therefore, its time complexity is $O(n^2 \log n)$.*

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

Let $\sigma_j(P)$ be the active schedule obtained by inserting idle time in σ_j to make σ_j feasible with respect to the constraint $P_{\max} \leqslant P$, for $j = 1,\ldots,m$. Let $J_{[i]}$ be an arbitrary job in $\sigma_j(P)$. We now have that $L_{[i]}(\sigma_j(P)) = L_{[i]}(\sigma_j) + \max\{0, K_i^j - P\}$. Therefore, the trade-off curve of P and $L_{\max}(\sigma_j(P))$ is obtained by combining the trade-off curves of $L_{[i]}(\sigma_j) + \max\{0, K_i^j - P\}$ for $i = 1,\ldots,n$; hence, the trade-off curve of P and $L_{\max}(\sigma_j(P))$ forms a continuous, piecewise linear step-function with alternate gradients -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 P the minimum value of $L_{\max}(\sigma_j(P))$ for $j = 1, \ldots, m$. The number of breakpoints in this trade-off curve is no more than $mn \leqslant n^2$. As $1 \mid P_{\max} \leqslant P \mid L_{\max}$ is solved by one of the schedules $\sigma_1(P), \ldots, \sigma_m(P)$ for every value of P, the trade-off curve derived above is exactly equal to the trade-off curve of P and $L_{\max}$, where $L_{\max}$ is equal to the outcome of the $1 \mid P_{\max} \leqslant P \mid L_{\max}$ problem. The trade-off curve is determined in $O(n^2 \log n)$ time, this is the time needed to determine the set of upper bound vectors $\{K^1, \ldots, K^m\}$ and to determine the set of sequences $\{\sigma_1, \ldots, \sigma_m\}$. Hence, $1 \mid P_{\max} \leqslant P \mid L_{\max}$ is solved in $O(n^2 \log n)$ time

THEOREM 9. *The* $1 \mid r_j \mid L_{\max}$ *problem is solvable in* $O(n^2 \log n)$ *time if* $r_j \in [d_j - p_j - C, d_j - C]$, *for* $j = 1, \ldots, n$, *for some constant C.* □

6. SOLVING THE $1 \mid\mid F(P_{\max}, L_{\max})$ PROBLEM IS $\mathcal{NP}$-HARD

In this section we prove that $1 \mid\mid F(P_{\max}, L_{\max})$ 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 U 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 U 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 $U = \{0, \ldots, 2^{|H|}\}$. Every integer $x \in U$ can be described by $|H|$ zeros and ones, by using binary encoding. Further define for every $x \in U$ a subset $U_x \subset H$ in the following way: the ith edge in H belongs to U_x if and only if the ith digit in the binary representation of x is equal to 1. Now we define the following function $f(x)$.

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

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* $1 \mid \mid F(P_{\max}, L_{\max})$ *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 $P_{\max}$ and $L_{\max}$ is unbounded, as every value $P \in \mathbb{Z}$, with P not larger than the P-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 hence, we can carry out the reduction from the Hamiltonian Circuit problem, with $F(P_{\max}, L_{\max}) = f(P - c)$, where c is such that $0 \leqslant P - c < 2^{|H|}$ for every selected P-value. □

Note that, if we impose the restriction that $P_{\max}$ has to be nonnegative, then only a pseudo-polynomial number of Pareto optimal points remains, and hence, $1 \mid \mid F(P_{\max}, L_{\max})$ is solvable in 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 $P_{\max}$ is bounded from below there is no polynomial algorithm for $1 \mid \mid F(P_{\max}, L_{\max})$, unless $\mathcal{P} = \mathcal{NP}$. Further, note that the reduction from the Hamiltonian circuit problem is not polynomial anymore, when $P_{\max}$ is assumed to be nonnegative and therefore, we cannot conclude that this special case of $1 \mid \mid F(P_{\max}, L_{\max})$ is $\mathcal{NP}$-hard in the strong sense.

Note that this example is artificial. In practice, it will be very seldom that a function F is considered such that $F(x, C - x)$ cannot be minimized in polynomial time. If we restrict ourselves to linear objective functions $F(P_{\max}, L_{\max}) = \alpha_1 P_{\max} + \alpha_2 L_{\max}$, 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; hence, $1 \mid \mid \alpha_1 P_{\max} + \alpha_2 L_{\max}$ is solved in $O(n^2 \log n)$ time, the time needed to determine the trade-off curve.

In case the function $F(P_{\max}, L_{\max})$ is convex, then we can solve solve $1 \mid \mid F(P_{\max}, L_{\max})$ by applying binary search in $O(\max\{n^2 \log n, \log \Sigma p_i\})$ time, provided that we impose the restriction that $P_{\max}$ is nonnegative.

ACKNOWLEDGEMENT
The author is grateful to 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.

M.R. GAREY, R.E. TARJAN, G.T. WILFONG (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research 13*, 330-348.

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.

J.A. HOOGEVEEN, S.L. VAN DE VELDE (1990). *Polynomial-time algorithms for single-machine multi criteria scheduling*, Report BS-R9008, CWI, 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.

E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS (1989). *Sequencing and scheduling: Algorithms and complexity*, Report BS-R8909, CWI, Amsterdam.

J.K. LENSTRA, A.H.G. RINNOOY KAN, P. BRUCKER (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 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.

Polynomial-time algorithms for single-machine bicriteria scheduling

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

This paper has been submitted for publication.

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 $O(n^3 \min\{(n, \log n + \log p_{\max}\})$ time, where $p_{\max}$ is the maximum job processing time, for every nondecreasing composite objective function; the algorithm can be improved to run in $O(n^3)$ time for the special case that the second criterion is maximum lateness. Second, we show that, if the second criterion is maximum earliness, 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 earliness.

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 earliness.

1. Introduction

A single-machine job shop can be 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. Without loss of generality, we assume that the processing times and due dates are integral. A *schedule* σ defines for each job J_j its completion time C_j such that the machine availability constraints are satisfied. A *performance measure* or *scheduling criterion* associates a value $f(\sigma)$ with each feasible schedule σ. Well-known measures are total completion time ΣC_j, maximum lateness $L_{\max}$, defined as $\max_{1 \leqslant j \leqslant n}(C_j - d_j)$, and maximum earliness $E_{\max}$, defined as $\max_{1 \leqslant j \leqslant n}(d_j - S_j - p_j)$, where S_j denotes the start time of J_j. In addition, we define $f_{\max}$ as $\max_{1 \leqslant j \leqslant n} f_j(C_j)$, where f_j is an arbitrary *regular* cost function for

J_j $(j = 1, \ldots, n)$; 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 σ^* is *optimal* for a given performance measure if $f(\sigma^*) = \min_{\sigma \in \Omega} f(\sigma)$, where Ω 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.

We consider the bicriteria problems that arise when the criterion ΣC_j is combined with one of the minmax criteria $f_{\max}$, $L_{\max}$, and $E_{\max}$. A second criterion is taken into account to prohibit a solution from being unbalanced; with unbalanced we mean that the schedule performs perfectly well on one criterion, whereas its performance on the other criterion is very poor.

The performance criteria under consideration are commonly used to model economic aspects. Total completion time is used to measure the work-in-process inventories; the elements needed in the processing of the job have to be stored until the job is completed. Maximum lateness measures the observance of due dates, whereas maximum earliness measures the observance of 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.

Basically, there are two methods to cope with multiple criteria. If the objectives are subject to a hierarchy, then the objectives are considered *sequentially* in order of relevance. An example hereof is the problem of minimizing maximum tardiness subject to the minimum number of tardy jobs (Shanthikumar, 1983); the primary criterion is to minimize the number of tardy jobs, and subject to this maximum tardiness is minimized. Note that in case of hierarchical minization we do not mind the schedule being unbalanced.

This paper, however, is concerned with the *simultaneous* optimization of several criteria. In this approach, the performance measures, specified by the functions f_k $(k = 1, \ldots, K)$, are transformed into a single *composite objective* function $F : \Omega \rightarrow \mathbb{R}$. With each schedule σ 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

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

where F is nondecreasing in each of its arguments. Minimizing the number of tardy jobs and maximum tardiness simultaneously (Nelson, Sarin, and Daniels, 1986) is an example of this method.

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 π with $f_k(\pi) \leqslant f_k(\sigma)$ 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 bounded by a polynomial in the problem size, then problem (P) is polynomially solvable.

An interesting subclass of (P) is one in which the composite objective function is *linear*. The associated problem, hereafter referred to as problem (P_α), is formulated as

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

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_α) for *any* weight vector $\alpha \geqslant 0$. 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_α) can be solved for any given $\alpha \geqslant 0$ by computing the cost of each extreme point and taking the minimum.

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

This paper is organized as follows. In Section 2, we present some fundamental algorithms for the underlying single-criterion problems. In Section 3, we consider the $1 | | F(\Sigma C_j, f_{\max})$ problem. We establish the polynomiality of Van Wassenhove and Gelders's conjecturedly pseudo-polynomial algorithm for $1 | | F(\Sigma C_j, L_{\max})$ (Van Wassenhove and Gelders, 1980), thereby proving a conjecture by Lawler, Lenstra, and Rinnooy Kan (1979). We show that $1 | | F(\Sigma C_j, f_{\max})$ is solvable in $O(n^3 \min\{n, \log n + \log p_{\max}\})$ time, where $p_{\max} = \max_j p_j$, and that $1 | | F(\Sigma C_j, L_{\max})$ is solvable in $O(n^3)$ time. These results make the branch-and-bound algorithms proposed by Sen and Gupta (1983) and Nelson et al. (1986) obsolete.

In Section 4, we consider $1 | pmtn | F(\Sigma C_j, E_{\max})$; the notation *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 $1 | pmtn | \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ and, in the case that $\alpha_1 \geqslant \alpha_2$, also $1 | | \alpha_1 \Sigma C_j + \alpha_2 E_{\max}$ are solvable in $O(n^4)$ time.

2. Fundamental algorithms and notation

There are four single-machine single-criterion scheduling problems related to the bicriteria problems under consideration. These involve the minimization of ΣC_j, $L_{\max}$, $E_{\max}$, and $f_{\max}$, respectively. The first three problems are solvable by arranging the jobs in a certain *priority order* that is specified in terms of the parameters of the problem type.

THEOREM 1 (Smith, 1956). *The problem of minimizing total completion time, denoted as* $1 | | \Sigma C_j$ *is solved by sequencing the jobs according to the shortest-processing-time (SPT) rule, that is, in order of nondecreasing* p_j. □

THEOREM 2 (Jackson, 1955). *The problem of minimizing maximum lateness, denoted as* $1 | | L_{\max}$, *is solved by sequencing the jobs according to the earliest-due-date (EDD) rule, that is, in order of nondecreasing* d_j. □

THEOREM 3. *The problem of minimizing maximum earliness subject to no machine idle time, denoted as* $1 | nmit | E_{\max}$, *is solved by sequencing the jobs according to the minimum-slack-time (MST) rule, that is, in order of nondecreasing* $d_j - p_j$. □

The no-machine-idle-time constraint *nmit* is imposed on the $1 | | E_{\max}$ problem to avoid unbounded solutions.

The fundamental argument that validates each algorithm is the following. Suppose that there is an optimal schedule with two adjacent jobs that are not scheduled according to the indicated priority order. The interchange of the jobs will possibly improve but certainly not worsen the objective value. An improvement contradicts the claimed optimality, whereas in the other case we can repeat the argument to obtain a schedule with equal objective value that matches the priority order.

THEOREM 4 (Lawler, 1973). *The* $1 | | f_{\max}$ *problem is solved as follows: while there are unassigned jobs, assign the job that has minimum cost when scheduled in the last unassigned position to that position.* □

3. Minimizing total completion time and maximum cost

Let $f_j : \mathbb{N} \rightarrow \mathbb{R}$ 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 $1 | | F(\Sigma C_j, f_{\max})$ is solvable in $O(n^3 \min\{n, \log n + \log p_{\max}\})$ time, with $p_{\max} = \max_j p_j$, for any function F that is nondecreasing in both ΣC_j and $f_{\max}$. Note that $1 | | F(\Sigma C_j, L_{\max})$ corresponds to a special case of $1 | | F(\Sigma C_j, f_{\max})$.

In Theorem 4, we recalled Lawler's $O(n^2)$ time algorithm for $1 | | f_{\max}$. An extension has been provided by Emmons (1975), who considered the hierarchical problem of minimizing ΣC_j subject to the constraint that $f_{\max}$ is minimal; this problem is denoted as $1 | f_{\max} \leqslant f^* | \Sigma C_j$, where f^* denotes the solution value of the outcome of $1 | | f_{\max}$. Once f^* has been determined by Lawler's algorithm, Emmons's algorithm requires $O(n^2)$ time to minimize total completion time subject to minimal maximum cost. Observe, however, that an upper bound on $f_j(C_j)$ induces a deadline $\bar{d}_j$ on the completion time of J_j. Each deadline can be determined in $O(\log(\Sigma p_j))$ time by binary search over the $O(\Sigma p_j)$ possible completion times. Furthermore, $\bar{d}_j$ 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 $1 | \bar{d}_j | \Sigma C_j$, which requires only $O(n \log n)$ time (Smith, 1956).

We state the algorithm for $1 | f_{\max} \leqslant f \; | \Sigma C_j$, 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 $\bar{d}_j$ induced by $f_j(C_j) \leqslant f$.
Step 2. $T \leftarrow \Sigma p_j$.
Step 3. Determine $U \leftarrow \{J_j \in J \mid \bar{d}_j \geqslant T\}$ as the set of jobs that are allowed to 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. $J \leftarrow J - \{J_j\}$; $T \leftarrow T - p_j$.
Step 6. If $T > 0$, then go to Step 3.
```

THEOREM 5. *Algorithm I determines a Pareto optimal point with respect to ΣC_j and $f_{\max}$.*

PROOF. It suffices to show that the algorithm generates a schedule σ that solves the problems $1 | f_{\max} \leqslant f \; | \Sigma C_j$ and $1 | \Sigma C_j \leqslant \Sigma C_j(\sigma) | f_{\max}$ simultaneously. Evidently, σ solves $1 | f_{\max} \leqslant f \; | \Sigma C_j$. Assume that not σ, but π is optimal for $1 | \Sigma C_j \leqslant \Sigma C_j(\sigma) | f_{\max}$. This implies that $f_{\max}(\pi) < f_{\max}(\sigma) \leqslant f$; hence, π is also feasible for $1 | f_{\max} \leqslant f \; | \Sigma C_j$. 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 kth position, which is occupied by jobs J_i and J_j in σ and π, respectively. Since $f_{\max}(\pi) < f$ and because of the choice of job J_i in the algorithm, we have $p_i \geqslant p_j$. If $p_i > p_j$, then π cannot be optimal, as the schedule that is obtained by interchanging J_i and J_j in π is feasible with respect to the constraint $f_{\max} \leqslant f$ 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, $f_i(C_i(\sigma)) \leqslant f_j(C_j(\pi))$. This implies, however, that the jobs J_i and J_j can be interchanged in π without affecting the cost of the schedule. Repetition of this argument shows that π can be transformed into σ without affecting the cost, thereby contradicting the assumption that $f_{\max}(\pi) < f_{\max}(\sigma)$. Therefore, σ also solves $1 | \Sigma C_j \leqslant \Sigma C_j(\sigma) | f_{\max}$, and is hence Pareto optimal for ΣC_j and $f_{\max}$. □

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 $1 \,|\,|\, F(\Sigma C_j, L_{\max})$, 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 $1 | f_{\max} \leqslant f_{\max}(SPT) | \Sigma C_j$; 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 $1 | f_{\max} < f_{\max}^{(k-1)} | \Sigma C_j$; 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

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

and $\Delta(\sigma) = \Sigma_{i,j}\, \delta_{ij}(\sigma)$. 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 $\Delta(\sigma) \leqslant \frac{1}{2}n(n-1)$ for any $\sigma \in \Omega$. In addition, we define a *neutral* interchange with respect to σ as the interchange of two jobs J_i and J_j with $p_i = p_j$.

LEMMA 1. *If schedule π can be obtained from schedule σ 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_l be an arbitrary job that is scheduled between J_i and J_j in σ. Then it is easy to verify that $\delta_{il}(\sigma) + \delta_{lj}(\sigma) \geqslant \delta_{jl}(\pi) + \delta_{li}(\pi)$. □

THEOREM 6. *Consider two arbitrary Pareto optimal schedules σ and π. If $\Sigma C_j(\sigma) < \Sigma C_j(\pi)$, then $\Delta(\sigma) < \Delta(\pi)$.*

PROOF. We show that schedule σ can be obtained from schedule π 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 kth position; J_i occupies the kth position in σ, whereas job J_j occupies the kth position in π. Because of the choice of J_i and J_j in Algorithm I, we have $p_i \geqslant p_j$; the interchange of J_i and J_i in π is therefore positive or neutral. We proceed in this way until we reach schedule σ. 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 7. *The number of Pareto optimal schedules is bounded by $\frac{1}{2}n(n-1)+1$, and this bound is tight.*

PROOF. The first part follows immediately from Theorem 9. For the second part, consider the following instance of $1\,||\,F(\Sigma C_j, L_{\max})$: 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 $\frac{1}{2}n(n-1)+1$ Pareto optimal schedules. □

COROLLARY 1. *The $1\,||\,F(\Sigma C_j, f_{\max})$ problem is solvable in $O(n^3 \min\{n, \log n + \log p_{\max}\})$ time.*

PROOF. Emmons's algorithm requires $O(n^2)$ time to solve $1\,|f_{\max} \leqslant f|\,\Sigma C_j$. 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 $1\,||\,F(\Sigma C_j, L_{\max})$ problem is solvable in $O(n^3)$ time.*

PROOF. First, note that an upper bound L on maximum lateness induces a deadline $\bar{d}_j = d_j + L$, which is determined in constant time. Furthermore, in view of Smith's algorithm, it suffices to sort the deadlines only once, since a change of the value of the upper bound L does not affect the order of the deadlines. Once the processing times and deadlines have been sorted, Algorithm II can be implemented to take only linear time per iteration. □

4. MINIMIZING TOTAL COMPLETION TIME AND MAXIMUM EARLINESS

In this section, we analyze the problem of minimizing total completion time and maximum earliness simultaneously. First, we make the additional assumption that machine idle time is forbidden, implying that all jobs are to be scheduled in the interval $[0, \Sigma p_j]$; we show how the insight gained from analyzing this special case can be used to deal with the general problem.

Due to the no-machine-idle-time constraint, it is evident that in each Pareto optimal schedule σ we have that $E_{\max}(\sigma)$ ranges from E^*, defined as the solution value of the outcome of $1\,|\,nmit\,|\,E_{\max}$, to $E_{\max}(SPT)$, and that $\Sigma C_j(\sigma)$ ranges from ΣC_j^* to $\Sigma C_j(MST)$, where ties in the *SPT* and *MST* schedule are settled in order to minimize slack time and completion time, respectively. Observe that an upper bound E on $E_{\max}$ induces for each job J_j a release time $r_j = \max\{0, d_j - p_j - E\}$. The associated value of ΣC_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. This is an important relaxation, since the $1|pmtn, r_j|\Sigma C_j$ 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 $E \geqslant E^*$.

The introduction of preemption has also a less convenient effect. *Any* value of $E_{\max}$ in the range $[E^*, E_{\max}(SPT)]$ is now attainable, and therefore corresponds to a Pareto optimal point. Since $E_{\max}(SPT) - E^* \leqslant \Sigma p_j$, the number of Pareto optimal schedules is only pseudo-polynomially bounded. These $O(\Sigma p_j)$ schedules are generated by the following algorithm.

```
ALGORITHM III
Step 1. Let $E^{(1)} \leftarrow E_{\max}(SPT)$ and $k \leftarrow 1$.
Step 2. Solve $1|pmtn, r_j = d_j - p_j - E^{(k)}|\Sigma C_j$; this yields the kth Pareto optimal
        schedule, denoted as $\sigma^{(k)}$.
Step 3. $k \leftarrow k+1$; $E^{(k)} \leftarrow E^{(k-1)} - 1$; if $E^{(k)} \geqslant E^*_{\max}$, then go to Step 2.
```

COROLLARY 1. *The* $1|pmtn,nmit|F(\Sigma C_j, E_{\max})$ *problem is solvable in* $O(n\Sigma p_j)$ *time.*

PROOF. A decrease of E does not affect the order of the release dates; hence, we have to sort the release dates only once. □

As to the complexity of $1|nmit,pmtn|F(\Sigma C_j, E_{\max})$, 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 y values is $\mathcal{NP}$-hard in the strong sense (Schrijver; see Hoogeveen, 1990), we have that $1|nmit,pmtn|F(\Sigma C_j, E_{\max})$ is $\mathcal{NP}$-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 $1|pmtn|F(\Sigma C_j, E_{\max})$ is $\mathcal{NP}$-hard in the strong sense.

In the remainder of this section, we restrict ourselves to linear objective functions $\alpha_1\Sigma C_j + \alpha_2 E_{\max}$. 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 $E_{\max}$ values in the interval $[E^*, E_{\max}(SPT)]$.

Let $\sigma(E)$ denote the schedule obtained by Baker's algorithm for $1|pmtn, E_{\max} \leqslant E|\Sigma C_j$; $\sigma(E)$ corresponds to $(E, \Sigma C_j(\sigma(E)))$. We say that a *complete interchange* has occurred in $\sigma(E)$ if there are two jobs J_i and J_j such that J_i is started before J_j in $\sigma(E-1)$, whereas J_j is started before J_i in $\sigma(E)$.

LEMMA 2. *An upper bound E on* $E_{\max}$ *can only correspond to an extreme point for* $(\Sigma C_j, E_{\max})$ *if a complete interchange has occurred in* $\sigma(E)$.

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

Obviously, the next step to determine the extreme set is to select the candidate values E; these should be such that a complete interchange takes place in $\sigma(E)$. Given a pair of jobs J_i and J_j with $p_i > p_j$ and J_i started before J_j in $\sigma(E)$, 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 $E_{\max} \leqslant E$ and the start time of J_i in $\sigma(E)$. However, if J_i is executed between the start and completion time of a preemptive job J_k, then an increase of E 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 IV that, given an upper bound value E and the corresponding schedule $\sigma(E)$, computes the smallest value $\bar{E} > E$ that possibly corresponds to an extreme point. The variable a_j $(j = 1, \ldots, n)$ signifies the increase of E necessary to let a complete interchange involving J_j take place.

```
ALGORITHM IV
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 $C_j(\sigma(E))$ with $p_l \geqslant a_j$. Set $T \leftarrow S_l(\sigma(E))$.
    (b) $J_j$ is not a preempted job. Then $a_j \leftarrow \min\{d_i - p_i - E - S_j(\sigma(E)) \mid J_i \in J\}$, where $J$ denotes the set of jobs for which $d_i - p_i - E > S_j(\sigma(E))$ and $p_j > p_i$. Set $T \leftarrow C_j(\sigma(E))$.
Step 3. If $T < \Sigma p_j$, then go to Step 2.
Step 4. Put $\bar{E} \leftarrow \min_j\{a_j\} + E$.
```

THEOREM 8. *All values E that may correspond to an extreme point $(E, \Sigma C_j(\sigma(E)))$ are generated by the iterative application of Algorithm IV.*

PROOF. Suppose that $\bar{E}$, although corresponding to an extreme point, was not determined by iteratively applying Algorithm IV. This implies that there is a value $E_1 < \bar{E}$ such that Algorithm IV determines a value $E_2 > \bar{E}$ when initialized with E_1. Hence, we have the situation that Algorithm IV did not notice the complete interchange of two jobs J_i and J_j, which implies that the start time of J_i in $\sigma(E)$ was not considered in Step 2. This, however, could take only place in Step 2(a): J_i is started in the time interval $[S_k(\sigma(E)), C_k(\sigma(E+1))]$, where J_k is some preempted job. 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 E of $E_{\max}$ generated through Algorithm IV is polynomially bounded, thereby establishing that $1|pmtn,nmit|\alpha_1\Sigma C_j+\alpha_2 E_{\max}$ is solved in polynomial time. We define for a given schedule σ 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 plus $\Sigma_{i=1}^{n}\delta_{ij}$, and $\Delta(\sigma)=\Sigma_{i,j}\delta_{ij}(\sigma)$.

THEOREM 9. *Let E_1 and E_2 be two $E_{\max}$ values that are generated through Algorithm IV, with $E_1>E_2$. Then $\Delta(\sigma(E_1))<\Delta(\sigma(E_2))$.*

PROOF. We start by showing that $\Delta(\sigma(E_1))\leq\Delta(\sigma(E_2))$. Suppose to the contrary that $\Delta(\sigma(E_1))>\Delta(\sigma(E_2))$. Then there must exist a job J_j for which $\Delta_j(\sigma(E_1))>\Delta_j(\sigma(E_2))$. There are two possibilities for an increase of Δ_j.

First, the number of preemptions of J_j in $\sigma(E_1)$ may be greater than in $\sigma(E_2)$. 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 $\sigma(E_2)$ but before C_j in $\sigma(E_1)$. We then have, however, that $\delta_{jk}(\sigma(E_1))=0$ and $\delta_{jk}(\sigma(E_2))=1$. This implies that an extra preemption of J_j decreases some Δ_k by the same amount; hence, an extra preemption does not increase Δ.

Second, we may have $\delta_{ij}(\sigma(E_1))=1$, whereas $\delta_{ij}(\sigma(E_2))=0$. This implies that there exists some job J_i with $p_i>p_j$ that is completed before J_j is started in $\sigma(E_1)$ but not in $\sigma(E_2)$. As $E_1>E_2$, this can only occur if there exists a job J_k that is completed before J_i in $\sigma(E_2)$ but after J_i in $\sigma(E_1)$. Hence, Δ_j is then increased by 1, but Δ_k is decreased by at least 1, implying that Δ does not increase. The same argument holds if there are some jobs scheduled between J_j and J_i in $\sigma(E_1)$. Note that the decrease of Δ_k is always greater than the increase of Δ_j, unless J_k is preempted at the start of J_i in $\sigma(E_1)$.

As E_1 has been determined by Algorithm IV such that either a preemption is removed or an interchange has been completed, we have that $\Delta(\sigma(E_1))<\Delta(\sigma(E_2))$. □

COROLLARY 2. *If preemption is allowed, then the number of extreme schedules with respect to $E_{\max}$ and ΣC_j is bounded by $\frac{1}{2}n(n-1)+1$.*

PROOF. It is easy to show that $\Delta(\sigma)$ is at most equal to $\frac{1}{2}n(n-1)$ for every schedule σ. Therefore, Theorem 9 yields the desired result. □

Although it is easy to construct an instance such that Algorithm IV determines $\frac{1}{2}n(n-1)+1$ different $E_{\max}$ values, it is yet an open question whether this bound is tight for the number of extreme points.

COROLLARY 3. *The* $1|pmtn,nmit|\alpha_1\Sigma C_j+\alpha_2 E_{\max}$ *problem is solvable in* $O(n^4)$ *time.* □

THEOREM 10. *If* $\alpha_1=\alpha_2$, *then there exists a nonpreemptive schedule that is optimal for* $1|pmtn,nmit|\alpha_1\Sigma C_j+\alpha_2 E_{\max}$. *If* $\alpha_1>\alpha_2$, *then any optimal schedule for* $1|pmtn,nmit|\alpha_1\Sigma C_j+\alpha_2 E_{\max}$ *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 Δ, then E_j is increased by Δ, while C_j is decreased by Δ. 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 $\alpha_1\Sigma C_j+\alpha_2 E_{\max}$ if total completion time outweighs maximum earliness, we have the following corollary.

COROLLARY 4. *If* $\alpha_1\geqslant\alpha_2$, *then* $1||\alpha_1\Sigma C_j+\alpha_2 E_{\max}$ *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 $\sigma(E)$ and $\sigma(E+1)$, with $E<E^*$. The idle time inserted between the jobs displays the same behavior as a preemptive job that is completed last: if E 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 $E_{\max}$-value $\bar{E}$ of the first extreme point we can determine the set of extreme points by adding an extra job J_0 to the instance with $p_0=d_0=E^*-\bar{E}+p_{\max}+1$. The value E 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 $q\leqslant n$, 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 $E_{\max}$ is easily determined.

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

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

References

K.R. Baker (1974). *Introduction to Sequencing and Scheduling*, Wiley, New York.

H. Emmons (1975). A note on a scheduling problem with dual criteria. *Naval Research Logistics Quarterly 22*, 615-616.

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

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.

J.A. Hoogeveen (1990). *On the minimization of maximum earliness and maximum lateness*, Report BS-R9001, CWI, Amsterdam.

J.R. Jackson (1955). *Scheduling a production line to minimize maximum tardiness*, Research Report 43, Management Sciences Research Project, UCLA.

E.L. Lawler (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science 19*, 544-546.

E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan (1979). Unpublished manuscript.

J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343-362.

R.T. Nelson, R.K. Sarin, and R.L. Daniels (1986). Scheduling with multiple performance measures: the one-machine case. *Management Science 32*, 464-479.

T. Sen and S.K. Gupta (1983). A branch-and-bound procedure to solve a bicriterion scheduling problem. *IIE Transactions 15*, 84-88.

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.

L.N. Van Wassenhove and F. Gelders (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research 4*, 42-48.

Single-machine scheduling to minimize a function of K maximum cost criteria

J.A. Hoogeveen

This paper has been submitted for publication.

Single-machine scheduling to minimize a function of K maximum cost criteria

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

A number of jobs has to be scheduled on a single machine, which can handle no more than one job at a time. Each job requires processing during a given positive uninterrupted time. For each job, there are K arbitrary non-decreasing penalty functions. The quality of a schedule is measured by K performance criteria, the kth one being given by the maximum value of the kth penalty function that is attained by any job. The problem is to find the set of Pareto optimal points with respect to these performance criteria. We present an algorithm for this problem that is polynomial for fixed K. We also show that these algorithms are still applicable if precedence constraints exist between the jobs or if all penalty functions are non-increasing in the job completion times.

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

1. Introduction

Since the introduction of scheduling theory in the 1950's, most research has been concentrated on single-criterion optimization. In real-life problems, however, multiple and usually conflicting criteria play a role. There are two methods to cope with conflicting criteria. The first one is *hierarchical* minimization. The performance criteria $f^1, \ldots, f^K$ are indexed in order of decreasing importance. First, f^1 is minimized. Next, f^2 is minimized subject to the constraint that the schedule has minimal f^1 value. Then, f^3 is minimized subject to the constraint that the values for f^1 and f^2 are equal to the values determined in the previous step; and so on. The first results on multicriteria scheduling (e.g., Smith, 1956) concern this approach. The second method is *simultaneous* minimization. The criteria are aggregated into a single composite objective function $P(f^1, \ldots, f^K)$, which is then minimized.

In this paper, we choose the second approach. We address the following single-machine multicriteria scheduling problem. A set of n independent jobs has to be scheduled on a single machine, which can handle no more than 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 processing during a given positive uninterrupted time p_i. A *schedule* σ defines for each job J_i its completion time $C_i(\sigma)$ such that the jobs do not overlap in their execution. The cost of completing J_i $(i = 1, \ldots, n)$ is measured by K

penalty functions f_i^k $(k = 1, \ldots, K)$; each of these penalty functions is assumed to be non-decreasing in the job completion time. Given a schedule σ, the penalty functions induce K performance criteria $f_{\max}^k(\sigma)$ $(k = 1, \ldots, K)$, defined as $f_{\max}^k(\sigma) = \max_{1 \leqslant i \leqslant n} f_i^k(C_i(\sigma))$, respectively. Given a function $P: \mathbb{R}^K \rightarrow \mathbb{R}$, we wish to find a schedule σ that minimizes $P(f_{\max}^1(\sigma), \ldots, f_{\max}^K(\sigma))$. We additionally assume that P is non-decreasing in each of its arguments. We denote this problem by $1 \,|\,|\, P(f_{\max}^1, \ldots, f_{\max}^K)$.

Due to this additional assumption, we know that there is a *Pareto optimal* point with respect to $(f_{\max}^1, \ldots, f_{\max}^K)$ in which P attains the optimum. A schedule σ corresponds to a Pareto optimal point if there is no feasible schedule π with $f_{\max}^k(\pi) \leqslant f_{\max}^k(\sigma)$, for $k = 1, \ldots, K$, where at least one of the inequalities is strict; in this case, we say that σ is not dominated.

The organization of the paper is as follows. In Section 2, we recall Lawler's algorithm (Lawler, 1973) for $1 \,|\, prec \,|\, f_{\max}$, where the acronym *prec* indicates that precedence constraints have been specified; that is, for each job J_i $(i = 1, \ldots, n)$ there is a set of jobs that have to precede J_i and a set of jobs that have to succeed J_i in any feasible schedule. Furthermore, we show that we can solve $1 \,|\, \bar{d}_j, prec \,|\, f_{\max}$ by adjusting Lawler's algorithm appropriately, where $\bar{d}_j$ indicates that for each job a deadline on the completion time has been specified. In Section 3, we present an $O(n^4)$ time algorithm to determine all Pareto optimal points for the two-criteria problem. In Section 4, we analyze the three-criteria problem, and show how this analysis can be extended to the K-criteria problem, for any fixed $K \geqslant 4$. Finally, in Section 5, we consider two problems that are solved analogously. The first problem allows precedence constraints; the second one has *non-increasing* penalty functions.

2. Lawler's algorithm to minimize maximum cost for one criterion

Lawler (1973) presented an $O(n^2)$ algorithm to solve $1 \,|\, prec \,|\, f_{\max}$. The algorithm is based upon the following observation. Let S denote the subset of jobs that may be processed last, let T denote the sum of the processing times of all jobs, and let J_k be a job in S such that $f_k(T) = \min_{j \in S} \{f_j(T)\}$. Then there exists an optimal schedule in which J_k is last.

Lawler's algorithm

```
(0) $T \leftarrow \Sigma_{j=1}^n p_j$; $\mathcal{J} \leftarrow \{J_1, \ldots, J_n\}$.
(1) Determine the set $\mathcal{U}$ containing the jobs that have no successors in $\mathcal{J}$.
(2) Choose from $\mathcal{U}$ the job $J_j$ that has minimal $f_j(T)$ value, settling ties arbitrarily; $J_j$ is
    processed from time $T - p_j$ to time $T$.
(3) $T \leftarrow T - p_j$; $\mathcal{J} \leftarrow \mathcal{J} - \{J_j\}$.
(4) If $\mathcal{J} \neq \varnothing$, then go to Step 1; otherwise, stop.
```

THEOREM 1. *Lawler's algorithm solves* $1 \,|\, prec \,|\, f_{\max}$.

PROOF. Let σ be the schedule obtained by Lawler's algorithm, and let $\bar{\sigma}$ be an optimal schedule for $1 \,|\, prec \,|\, f_{\max}$. Compare both schedules, starting at the end. Suppose that the first difference occurs at the kth position; let J_i occupy the kth position in σ. Adjust $\bar{\sigma}$ by assigning J_i to the kth position; the sequence of the other jobs stays the same. The new schedule $\bar{\sigma}$ is feasible; J_i can be assigned to that position as σ is feasible, and the sequence of the other jobs has not been changed. Furthermore, its cost has not been increased; J_i was chosen by Lawler's algorithm, so it must have minimal cost among the unassigned jobs that could be scheduled in that position. Proceed in the same way until σ and $\bar{\sigma}$ are identical, implying that $f_{\max}(\sigma) \leqslant f_{\max}(\bar{\sigma})$. Hence, σ is optimal. □

Lawler's algorithm is easily adjusted to deal with $1 \mid \bar{d}_j, prec \mid f_{\max}$. If a job J_k is a candidate for the last position, then we have to check whether J_k has no successors and $\bar{d}_j \geqslant T$. Hence, the set $\mathcal{U}$ contains the jobs that have no successors in $\mathcal{J}$ and that have a deadline greater than or equal to T. Alternatively, we can incorporate the deadlines by redefining $f_j(T) \leftarrow \infty$ for $T > \bar{d}_j$ $(j = 1, \ldots, n)$ and apply Lawler's algorithm to the adjusted $1 \mid prec \mid f_{\max}$ problem.

The deadlines do not have to be given explicitly, but may be induced by given upper bounds on other criteria. For example, if g_i is a non-decreasing penalty function, for $i = 1, \ldots, n$, then the constraint $g_{\max} \leqslant G$ induces a deadline for each job J_i.

3. Analysis of the two-criteria problem

For notational convenience, we denote the penalty functions for J_i $(i = 1, \ldots, n)$ by f_i and g_i, respectively. Correspondingly, the maximum cost criteria are called $f_{\max}$ and $g_{\max}$, respectively. There are basically two ways to deal with the $1 \mid \mid P(f_{\max}, g_{\max})$ problem. The first one is to solve it directly, for instance through branch-and-bound. The second one is to solve it in a roundabout way by determining the Pareto optimal set, that is, the set of points that are Pareto optimal with respect to $(f_{\max}, g_{\max})$, and then selecting the one that minimizes $P(f_{\max}, g_{\max})$. We take the second option. From now on, whenever we refer to the problem $1 \mid \mid P(f^1_{\max}, \ldots, f^K_{\max})$, it is assumed that we are going to determine all Pareto optimal points with respect to $(f^1_{\max}, \ldots, f^K_{\max})$. For instance, $1 \mid \bar{d}_j \mid P(f_{\max}, g_{\max})$ denotes the problem of determining all Pareto optimal points with respect to $(f_{\max}, g_{\max})$ subject to deadlines.

In order to determine the Pareto optimal set, we apply the following strategy. We start by solving $1 \mid \mid f_{\max}$; this yields the first value F that corresponds to a candidate Pareto optimal point (F, G). Next, we determine the corresponding value G by solving $1 \mid f_{\max} \leqslant F \mid g_{\max}$ through Lawler's algorithm. Then, we determine the next larger value F that corresponds to a possibly Pareto optimal point (F, G), solve $1 \mid f_{\max} \leqslant F \mid g_{\max}$ to obtain G, and so on.

There are two difficulties with the application of this strategy. The first one is how is the next value of F determined. The second one concerns the question of how many of these values are to be computed before all Pareto optimal points have been found.

We start by addressing the first problem. Let σ be the schedule obtained by solving $1 \mid f_{\max} \leqslant F \mid g_{\max}$ through Lawler's algorithm, and let J_j be a job that attains $g_{\max}(\sigma)$, that is, $g_j(C_j(\sigma)) = \max_{1 \leqslant i \leqslant n} g_i(C_i(\sigma))$. As g_j is non-decreasing, a Pareto optimal point with smaller $g_{\max}$ value can be obtained only if the completion time of J_j is decreased. Hence, some job J_i that is before J_j in σ and that has $g_i(C_j(\sigma)) < g_{\max}$ has to be completed no earlier than time $C_j(\sigma)$.

This observation provides the basis for our algorithm to determine the increase of F that is necessary to reach a new candidate Pareto optimal point.

ALGORITHM NEXT UPPER BOUND (NUB)

(0) Given a schedule σ obtained by Lawler's algorithm, determine the set $\mathcal{J}$ of jobs that attain $g_{\max}(\sigma)$.

(1) Determine for each $J_j \in \mathcal{J}$ the set $\mathcal{U}_j$ of jobs J_i that are scheduled before J_j in σ and that have $g_i(C_j(\sigma)) < g_{\max}(\sigma)$. If $\mathcal{U}_j = \varnothing$ for some $J_j \in \mathcal{J}$, then $g_{\max}(\sigma)$ cannot be decreased; stop. For each job $J_j \in \mathcal{J}$, define $F_j = \min\{f_i(C_j(\sigma)) \mid J_i \in \mathcal{U}_j\}$.

(2) The new upper bound F on $f_{\max}$ is the maximum of the values F_j.

THEOREM 2. *Let $(\hat{F}, \hat{G})$ be a Pareto optimal point with respect to $(f_{\max}, g_{\max})$, and let σ be the corresponding schedule. Let F be the new upper bound on $f_{\max}$ that is obtained by applying Algorithm* NUB, *given σ. There is no Pareto optimal point corresponding to a value $\tilde{F}$, with $F > \tilde{F} > \hat{F}$.* □

A decrease of C_j does not necessarily induce a decrease in $g_j(C_j)$, and hence the new upper bound F does not necessarily correspond to a Pareto optimal point. The remaining question is how many values F are determined by Algorithm NUB, before all Pareto optimal points have been found.

THEOREM 3. *The Algorithm* NUB *determines at most $n(n-1)/2$ values F.*

PROOF. Every new value F obtained by applying Algorithm NUB to σ corresponds to a combination of two jobs $\{J_i, J_j\}$, where $g_j(C_j(\sigma)) = g_{\max}(\sigma)$ and $f_i(C_j(\sigma)) = F$. We will show that every upper bound value F that is obtained by Algorithm NUB corresponds to a different combination of jobs.

Define f^* and g^* as the outcome of $1 \| f_{\max}$ and $1 \| g_{\max}$, respectively. Let σ_1 be the schedule obtained by Lawler's algorithm when solving $1 | f_{\max} \leqslant f^* | g_{\max}$. For $a = 1, \ldots, A$, apply Algorithm NUB to schedule σ_a to obtain F^{a+1}, and determine σ_{a+1} by solving $1 | f_{\max} \leqslant F^{a+1} | g_{\max}$; A is such that $g_{\max}(\sigma_A) = g^*$. Suppose that the combination $\{J_i, J_j\}$ corresponds to both F^{a+1} and F^{b+1}, with $a < b$. Without loss of generality, let J_j attain $g_{\max}(\sigma_a)$; $f_i(C_j(\sigma_a)) = F^{a+1}$. We have to consider two cases: either J_i or J_j attains $g_{\max}(\sigma_b)$. First, suppose that $g_j = g_{\max}(\sigma_b)$; $f_i(C_j(\sigma_b)) = F^{b+1}$. As $g_j(C_j(\sigma_a)) = g_{\max}(\sigma_a) \geqslant g_{\max}(\sigma_b) = g_j(C_j(\sigma_b))$, we must have $C_j(\sigma_a) \geqslant C_j(\sigma_b)$. This implies that J_i is allowed to be completed at time $C_j(\sigma_b)$ when σ_b is constructed, because $F^b > F^{a+1}$. As Lawler's algorithm selected J_j to be completed at time $C_j(\sigma_b)$, we must have that either J_i had already been scheduled or $g_j(C_j(\sigma_b)) \leqslant g_i(C_j(\sigma_b))$. In both cases, Algorithm NUB will not take J_i into consideration, when applied to σ_b. In the same fashion, we prove that J_j will not be taken into consideration by the algorithm if J_i attains $g_{\max}(\sigma_b)$. Hence, every pair of jobs (J_i, J_j) corresponds to at most one of the values F obtained by Algorithm NUB. This proves the theorem. □

COROLLARY 1. *The number of Pareto optimal points with respect to $(f_{\max}, g_{\max})$ is at most equal to $n(n-1)/2 + 1$.* □

The following example shows that this bound is tight, even if both maximum cost functions are of the maximum lateness type, that is, $f_i : C_i \rightarrow C_i - d_i$, and $g_i : C_i \rightarrow C_i - e_i$, for $i = 1, \ldots, n$.

$$d_i = (n-i)(n-i+3)/2, \quad \text{for } i = 1, \ldots, n,$$

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

$$e_n = e_{n-1} + 1,$$

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

$$p_n = d_n - (n-1)(n-2)/2;$$

It is easy to check that the Pareto optimal schedules for this example are: $(J_n, \ldots, J_2, J_1), (J_n, \ldots, J_1, J_2), \ldots, (J_1, J_n, \ldots, J_3, J_2), \ldots,$ $(J_1, J_2, J_n, \ldots, J_4, J_3), \ldots,\ (J_1, \ldots, J_n)$.

For sake of completeness, we list the algorithm to determine all Pareto optimal points and the optimal solution value. Its correctness follows from Theorems 1 and 2.

ALGORITHM A

(0) Determine f^* and g^* by solving $1 \,|\,|\, f_{\max}$ and $1 \,|\,|\, g_{\max}$, respectively; put $F \leftarrow f^*$.

(1) Solve $1 \,|\, f_{\max} \leqslant F \,|\, g_{\max}$; let G denote the outcome. Add (F, G) to the set of Pareto optimal points, unless it is dominated by the previously obtained Pareto optimal point. If $G = g^*$, then go to Step 3.

(2) Determine the next value of F by applying Algorithm NUB to the schedule obtained in the previous step. Go to Step 1.

(3) The Pareto optimal set has been obtained. The $1 \,|\,|\, P(f_{\max}, g_{\max})$ problem is solved by computing the value of the objective function for each point of the Pareto optimal set, and by choosing the optimum.

The running time of Algorithm A is $O(n^4)$; this is the time needed for solving $O(n^2)$ instances of the $1 \,|\, f_{\max} \leqslant F \,|\, g_{\max}$ problem.

4. ANALYSIS OF THE K-CRITERIA PROBLEM

We prove that the K-criteria problem can be solved by solving a polynomial number of $(K-1)$-criteria problems. First, we analyze the three-criteria problem; later on, we show how this analysis can be extended to the K-criteria problem. For notational convenience, the criteria are called $f_{\max}$, $g_{\max}$, and $h_{\max}$, respectively; correspondingly, the penalty functions for J_i $(i = 1, \ldots, n)$ are called f_i, g_i, and h_i, respectively.

Note that each Pareto optimal point (F, G) for $(f_{\max}, g_{\max})$ yields a Pareto optimal point (F, G, H) for $(f_{\max}, g_{\max}, h_{\max})$, where H is obtained by solving $1 \,|\, f_{\max} \leqslant F, g_{\max} \leqslant G \,|\, h_{\max}$, and that each dominated point (F, G) can only correspond to a Pareto optimal point (F,G,H) if H is attractive enough. Note further that, if (F,G,H) is Pareto optimal, then (F,G) is a solution of $1 \mid h_{\max} \leqslant H \mid P(f_{\max}, g_{\max})$. These observations provide the basis for our strategy to solve the three-criteria problem $1 \mid \mid P(f_{\max}, g_{\max}, h_{\max})$.

We will determine all Pareto optimal points (F,G,H) for $(f_{\max}, g_{\max}, h_{\max})$ by considering all $h_{\max}$ values that correspond to a candidate Pareto optimal point. The $h_{\max}$ values under consideration lie in the interval $[h^*, \bar{H}]$; h^* is the solution of $1 \mid \mid h_{\max}$, and $\bar{H}$ is an upper bound on the $h_{\max}$ value of any Pareto optimal point that we will establish now. Obviously, $\bar{H}$ should be such that solving $1 \mid h_{\max} \leqslant \bar{H} \mid P(f_{\max}, g_{\max})$ yields the set of Pareto optimal points for $(f_{\max}, g_{\max})$. Hence, $\bar{H}$ is determined by solving $1 \mid f_{\max} \leqslant F, g_{\max} \leqslant G \mid h_{\max}$ for every Pareto optimal point (F,G) for $(f_{\max}, g_{\max})$ and selecting the maximum. If we want to determine new Pareto optimal points (F,G,H), then we have to decrease the upper bound on $h_{\max}$, such that at least one of the currently determined Pareto optimal points is eliminated. This leads to the following outline for an algorithm to determine all Pareto optimal points for $(f_{\max}, g_{\max}, h_{\max})$.

(0) Determine the set of Pareto optimal points (F,G) with respect to $(f_{\max}, g_{\max})$. For each of these points, compute the corresponding $h_{\max}$ value, say, H. Store these Pareto optimal points (F,G,H) in a set U_1.
(1) Determine $\bar{H}$ as the maximum of the $h_{\max}$ values in U_1. Remove the Pareto optimal points with $h_{\max}$ value equal to $\bar{H}$ and store them in a set U_2.
(2) If $\bar{H} = h^*$, then stop; the set of Pareto optimal points is equal to $U_1 \cup U_2$.
(3) Solve $1 \mid h_{\max} < \bar{H} \mid P(f_{\max}, g_{\max})$, and compute for these points (F,G) the corresponding $h_{\max}$ values H. Add these points (F,G,H) to U_1. Go to 1.

We solve $1 \mid h_{\max} < \bar{H} \mid P(f_{\max}, g_{\max})$ by adjusting Algorithm A such that every solution that is generated by Algorithm A satisfies $h_{\max} < \bar{H}$. As observed before, this is easily achieved by adjusting the penalty functions appropriately.

Before proving that this strategy determines all points (F,G,H) that are Pareto optimal with respect to $(f_{\max}, g_{\max}, h_{\max})$, we prove two preliminary results.

THEOREM 4. *Let (F,G) be an arbitrary Pareto optimal point that is obtained when solving $1 \mid h_{\max} < \bar{H} \mid P(f_{\max}, g_{\max})$. Solve $1 \mid f_{\max} \leqslant F, g_{\max} \leqslant G \mid h_{\max}$, let H be the outcome. The point (F,G,H) is Pareto optimal for $(f_{\max}, g_{\max}, h_{\max})$.*

PROOF. Suppose that there exists a Pareto optimal point $(\tilde{F}, \tilde{G}, \tilde{H})$ that dominates (F,G,H). This implies that $(\tilde{F}, \tilde{G})$ is obtained when solving $1 \mid h_{\max} \leqslant \tilde{H} \mid P(f_{\max}, g_{\max})$. As $\tilde{H} \leqslant H < \bar{H}$, the constraint $h_{\max} \leqslant \tilde{H}$ is at least as restrictive as $h_{\max} < \bar{H}$, implying that the point $(\tilde{F}, \tilde{G})$ is also obtained when solving $1 \mid h_{\max} < \bar{H} \mid P(f_{\max}, g_{\max})$. Hence, $\tilde{F} = F$ and $\tilde{G} = G$, implying that $\tilde{H} = H$, as both values are equal to the outcome of $1 \mid f_{\max} \leqslant F, g_{\max} \leqslant G \mid h_{\max}$. □

COROLLARY 2. *Let (F,G,H) be an arbitrary point with $H < \bar{H}$ that is not found when solving $1 \mid h_{\max} < \bar{H} \mid P(f_{\max}, g_{\max})$. Then there exists a Pareto optimal point $(\tilde{F}, \tilde{G}, \tilde{H})$ with $\tilde{H} < \bar{H}$ such that $\tilde{F} \leqslant F$ and $\tilde{G} \leqslant G$, where at least one of the inequalities is strict.* □

THEOREM 5. *Every Pareto optimal point with respect to $(f_{\max}, g_{\max}, h_{\max})$ is found.*

PROOF. Let (F,G,H) be an arbitrary Pareto optimal point with respect to $(f_{\max}, g_{\max}, h_{\max})$. If (F,G) is Pareto optimal with respect to $(f_{\max}, g_{\max})$, then (F,G,H) is determined at the initialization. Otherwise, there must exist a Pareto optimal point that dominates (F,G,H) with respect to $(f_{\max}, g_{\max})$. Suppose that $(\tilde{F},\tilde{G},\tilde{H})$ is the Pareto optimal point with the smallest $h_{\max}$ value that dominates (F,G,H) with respect to $(f_{\max}, g_{\max})$. Hence, (F,G,H) will be generated as soon as the upper bound on $h_{\max}$ has become smaller than $\tilde{H}$. □

A straightforward implementation of the strategy leads to an $O(n^4 \mid Z \mid)$ time algorithm, where $\mid Z \mid$ denotes the cardinality of the set of Pareto optimal points. The factor $O(n^4)$ stems from solving an $1 \mid f_{\max} \leqslant F, g_{\max} \leqslant G \mid h_{\max}$ problem for every point (F,G) that is obtained when solving $1 \mid h_{\max} < H \mid P(f_{\max}, g_{\max})$. Note that we have not yet taken precautions to avoid a point (F,G) being generated more than once. Hence, we may improve the time complexity by determining a quick way to generate every Pareto optimal point only once. This is achieved by generating only the Pareto optimal points that are not present in the current set U_1, when solving $1 \mid h_{\max} < \overline{H} \mid P(f_{\max}, g_{\max})$; these are exactly the points that are dominated with respect to $(f_{\max}, g_{\max})$ by a Pareto optimal point $(\overline{F},\overline{G},\overline{H})$ with $H = \overline{H}$, but not by any other Pareto optimal point $(\hat{F},\hat{G},\hat{H})$ in U_1 with $\hat{H} < \overline{H}$. In order to determine only these Pareto optimal points, we derive lower and upper bounds on the $f_{\max}$ value such that a new Pareto optimal point must have a $f_{\max}$ value that is in between. We then search within this region for $f_{\max}$ values that correspond to a possibly Pareto optimal point by applying Algorithm NUB. The schedule we need to start with is obtained simultaneously with the bounds.

Order the Pareto optimal points in the set U_1 lexicographically, that is, put the points in non-decreasing order of $f_{\max}$ value, settling ties according to non-decreasing $g_{\max}$ values. Let (F^1, G^1, H^1) be the last point before (F,G,H) in the list with $H^1 < H$, and let σ_1 be the corresponding schedule. If there is no such point, then F^1 is equal to the outcome of $1 \mid h_{\max} < \overline{H} \mid f_{\max}$, G^1 to the outcome of $1 \mid f_{\max} \leqslant F^1, h_{\max} < \overline{H} \mid g_{\max}$, and H^1 to the outcome of $1 \mid f_{\max} \leqslant F^1, g_{\max} \leqslant G^1 \mid h_{\max}$, respectively; σ^1 is then the corresponding schedule. Let (F^2, G^2, H^2) be the first point after (F,G,H) with $h_{\max}$ value smaller than H. If such a point does not exist, then $F^2 = \infty$.

The new Pareto optimal points are determined by an iterative procedure. Apply Algorithm NUB to σ_1; this yields an $f_{\max}$ value $\hat{F}$. Determine $\hat{G}$ by solving $1 \mid f_{\max} \leqslant \hat{F}, h_{\max} < H \mid g_{\max}$, and $\hat{H}$ by solving $1 \mid f_{\max} \leqslant \hat{F}, g_{\max} \leqslant \hat{G} \mid h_{\max}$; call the corresponding schedule $\hat{\sigma}$. If $\hat{F} \geqslant F^2$, then stop; otherwise repeat the above procedure, in which Algorithm NUB is applied to $\hat{\sigma}$.

THEOREM 6. *Let $(\hat{F},\hat{G},\hat{H})$ be an arbitrary Pareto optimal point that is dominated with respect to $(f_{\max}, g_{\max})$ by (F,G,H), but that is not dominated with respect to $(f_{\max}, g_{\max})$ by a point in U_1 with $h_{\max}$ value smaller than H. Then $F \leqslant \hat{F} < F^2$; these Pareto optimal points $(\hat{F},\hat{G},\hat{H})$ are all determined by the procedure described above.*

PROOF. First, we prove the validity of the bounds on $\hat{F}$. The lower bound F follows by definition; the upper bound follows from the observation that $G^2 < G$ as (F^2, G^2, H^2) is not dominated by (F,G,H) with respect to $(f_{\max}, g_{\max})$, and hence $G^2 < \hat{G}$. As $(\hat{F},\hat{G},\hat{H})$ is not dominated by (F^2, G^2, H^2) with respect to $(f_{\max}, g_{\max})$, $\hat{F}$ must be smaller than F^2.

Second, the point (F^1, G^1) is a solution of $1|h_{\max} < H|P(f_{\max}, g_{\max})$. Applying Algorithm NUB as described above, starting with σ^1, yields a set of values F each corresponding to a possibly Pareto optimal point. None of these points is dominated with respect to $(f_{\max}, g_{\max})$ by a point already in U_1, as $\hat{F}$ is chosen such that $G \leqslant G^1$, while (F^1, G^1, H^1) is not dominated by a point in U_1 with respect to $(f_{\max}, g_{\max})$. □

Note that we have to check whether the $f_{\max}$ value determined by Algorithm NUB corresponds to a Pareto optimal point (F,G) with respect to $(f_{\max}, g_{\max})$ subject to the constraint $h_{\max} < H$, as an increase of the $f_{\max}$ value does not necessarily induce a decrease of the $g_{\max}$ value.

The theorems above show that our strategy can be implemented in such a way that all Pareto optimal points with respect to $(f_{\max}, g_{\max}, h_{\max})$ are found in $O(n^2 |P|)$ time. The $O(n^2)$ component per Pareto optimal point is needed to solve $1|h_{\max} < H|P(f_{\max}, g_{\max})$, to solve $1|f_{\max} \leqslant F, g_{\max} \leqslant G|h_{\max}$, to order the Pareto optimal points in U_1 lexicographically, and to determine the maximum of the $h_{\max}$ values in U_1. It remains to be shown that $|P|$ is polynomially bounded in n.

THEOREM 7. *There are at most $n^2(n-1)^2/4 + n(n-1) + 1$ Pareto optimal points with respect to $(f_{\max}, g_{\max}, h_{\max})$.*

PROOF. Immediately after the initialization, U_1 contains at most $n(n-1)/2 + 1$ points. Every other Pareto optimal point (F,G,H) is dominated with respect to $(f_{\max}, g_{\max})$, and hence is generated in the remainder of the algorithm.

Consider an arbitrary point (F,G,H) that is generated from $(\bar{F},\bar{G},\bar{H})$. Let $\bar{\sigma}$ be the schedule corresponding to $(\bar{F},\bar{G},\bar{H})$, and let σ be the schedule corresponding to (F,G,H). Let $h_{\max}$ be attained by J_j in $\bar{\sigma}$. As $\bar{H} > H$, J_j must be completed earlier in σ, and hence, there must be a job J_i preceding J_j in $\bar{\sigma}$ that in σ is completed at or after time $C_j(\bar{\sigma})$. As $F \geqslant \bar{F}$ and $G \geqslant \bar{G}$, we can prove along the same lines as in the proof of Theorem 3 that this combination $\{J_i, J_j\}$ will not occur again, when $h_{\max}$ is decreased. Hence, every point obtained in Step 0 dominates at most $n(n-1)/2$ Pareto optimal points with respect to $(f_{\max}, g_{\max})$, which implies that there are at most $n^2(n-1)^2/4 + n(n-1) + 1$ Pareto optimal points. □

For sake of completeness, we list the $O(n^6)$ time algorithm to determine all Pareto optimal points. Its correctness follows from Theorems 5 through 9.

ALGORITHM B

(0) Solve $1||P(f_{\max}, g_{\max})$. Determine for each point (F,G) the corresponding $h_{\max}$ value by solving $1|f_{\max} \leqslant F, g_{\max} \leqslant G|h_{\max}$. Store these points in set U_1; determine $\bar{H}$ as the maximum of the $h_{\max}$ values of the points in U_1.

(1) Order the points in U_1 lexicographically and let $\bar{H}$ be the maximum $h_{\max}$ value. Let

(F,G,H) be the first point in the list with $h_{\max}$ value equal to $\overline{H}$. Determine the bound F^2 and the schedule σ^1, and solve $1|F \leqslant f_{\max} < F^2, h_{\max} < \overline{H}|P(f_{\max}, g_{\max})$ as described on the previous page, and scan the solution set for Pareto optimal points $(\hat{F},\hat{G})$ with respect to $(f_{\max}, g_{\max})$. Determine the corresponding $h_{\max}$ value by solving $1|f_{\max} \leqslant \hat{F}, g_{\max} \leqslant \hat{G}|h_{\max}$. Remove (F,G,H) from U_1 and store it in the set U_2. Add the newly obtained points to U_1.

(2) If $\overline{H}$ is greater than the outcome of $1||h_{\max}$, then go to Step 1. Otherwise, the union of the sets U_1 and U_2 contains all Pareto optimal points (F,G,H) with respect to $(f_{\max}, g_{\max}, h_{\max})$.

It is easily checked that the strategy that was followed to solve the three-criteria problem can also be applied to solve the K-criteria problem, as the Theorems 4 and 5 and Corollary 2 still hold for the K-criteria case. Unfortunately, Theorem 6 no longer holds, so we can no longer guarantee that each Pareto optimal point is generated only once. We now solve $O(|P|)$ times the problem of determining all Pareto optimal points for a $(K-1)$-criteria problem with a given upper bound on $f_{\max}^K$; for each of the obtained points, we determine the corresponding $f_{\max}^K$ value.

The proof of Theorem 7 can be extended to the K-criteria case, showing that there are at most $(n(n-1)/2+1)^{K-1}$ Pareto optimal points. Therefore, our strategy can be implemented to run in $O(n^{K(K+1)-6})$ time, for $K \geqslant 4$.

5. Related problems

We finally consider the problems $1|prec|P(f_{\max}^1, \ldots, f_{\max}^K)$ and $1||P(g_{\max}^1, \ldots, g_{\max}^K)$, where the functions $g_{\max}^k$ are induced by penalty functions g_j^k that are non-increasing in the job completion times, for $k = 1, \ldots, K$. We show that we can solve both problems by Algorithm B by modifying the penalty functions appropriately.

First, we deal with $1|prec|P(f_{\max}^1, \ldots, f_{\max}^K)$. Let $\mathcal{S}_i$ denote the set of jobs J_j that have to succeed J_i in any feasible schedule σ. As $C_i(\sigma) < C_j(\sigma)$ for each job $J_j \in \mathcal{S}_i$, the cost of σ does not increase if at time T $(T \in [0, \Sigma p_j])$ the value of the penalty functions f_i^k $(k = 1, \ldots, K)$ is set equal to $\max\{f_i^k(T), f_j^k(T)\}$, for each $J_j \in \mathcal{S}_i$. The next theorem shows that the precedence constraints can be handled by adjusting the penalty functions as described above.

Theorem 8. *The* $1|prec|P(f_{\max}^1, \ldots, f_{\max}^K)$ *problem is solved by adjusting the penalty functions as described above, provided that ties in Lawler's algorithm are settled according to the precedence constraints.*

Proof. Let g_i^k $(k = 1, \ldots, K)$ denote the adjusted penalty functions. The proof consists of two parts. First, we have to show that every Pareto optimal point for $1||P(g_{\max}^1, \ldots, g_{\max}^K)$ corresponds to a schedule that is feasible with respect to the precedence constraints. This is guaranteed by the requirement to settle ties in Lawler's algorithm according to the precedence constraints.

Second, we have to prove that the sets of Pareto optimal points for $1|prec|P(f_{\max}^1, \ldots, f_{\max}^K)$ and $1||P(g_{\max}^1, \ldots, g_{\max}^K)$ are the same. Note that a point

$(F^1, \ldots, F^K)$ is Pareto optimal with respect to $(f^1_{\max}, \ldots, f^K_{\max})$ subject to precedence constraints if and only if, for $k = 1, \ldots, K$, the outcome of the problem of minimizing $f^k_{\max}$ subject to the constraints $f^i_{\max} \leqslant F^i$ $(i = 1, \ldots, K; i \neq k)$ and precedence constraints is equal to F^k. Furthermore, a point $(F^1, \ldots, F^K)$ is Pareto optimal with respect to $(g^1_{\max}, \ldots, g^K_{\max})$ if and only if, for $k = 1, \ldots, K$, the outcome of the problem of minimizing $g^k_{\max}$ subject to the constraints $g^i_{\max} \leqslant F^i$ $(i = 1, \ldots, K; i \neq k)$ is equal to F^k. Hence, if we prove that the problems $1 \mid f^1_{\max} \leqslant F^1, \ldots, f^{K-1}_{\max} \leqslant F^{K-1}, prec \mid f^K_{\max}$ and $1 \mid g^1_{\max} \leqslant F^1, \ldots, g^{K-1}_{\max} \leqslant F^{K-1} \mid g^K_{\max}$ yield the same outcome, then we are done. To that end, we have to justify the following three claims.

(1) The outcome of the problem $1 \mid f^1_{\max} \leqslant F^1, \ldots, f^{K-1}_{\max} \leqslant F^{K-1}, prec \mid f^K_{\max}$ stays the same if we replace the constraints $f^k_{\max} \leqslant F^k$ by $g^k_{\max} \leqslant F^k$, for $k = 1, \ldots, K-1$.

(2) The outcome of the problem $1 \mid g^1_{\max} \leqslant F^1, \ldots, g^{K-1}_{\max} \leqslant F^{K-1}, prec \mid f^K_{\max}$ stays the same if we replace the objective function $f^K_{\max}$ by $g^K_{\max}$.

(3) The precedence constraints in the problem $1 \mid g^1_{\max} \leqslant F^1, \ldots, g^{K-1}_{\max} \leqslant F^{K-1}, prec \mid g^K_{\max}$ are redundant.

Proof of (1). The first claim is proven by showing that the sets of feasible schedules are identical for both problems. The nontrivial part consists of showing that every schedule $\sigma \in \{\sigma \mid f^1_{\max} \leqslant F^1, \ldots, f^{K-1}_{\max} \leqslant F^{K-1}, prec\}$ has $g^k_{\max} \leqslant F^k$ $(k = 1, \ldots, K-1)$. Suppose to the contrary that there exists a feasible schedule σ with $g^k_i(C_i(\sigma)) > F^k$ for some job J_i, for some k. Then J_i must have a successor J_j such that $g^k_i(C_i(\sigma)) = f^k_j(C_i(\sigma)) \leqslant f^k_j(C_j(\sigma)) \leqslant F^k$, contradicting the assumption.

Proof of (2). The second claim is proven by showing that $f^K_{\max}(\sigma) = g^K_{\max}(\sigma)$ for every feasible schedule σ. By definition, $f^K_{\max}(\sigma) \leqslant g^K_{\max}(\sigma)$. Let $g^K_{\max}$ be attained by J_i; suppose that $f^K_{\max}(\sigma) < g^K_{\max}(\sigma)$. Hence, J_i must have a successor J_j with $f^K_j(C_i(\sigma)) = g^K_i(C_i(\sigma))$. In that case, however, $f^K_{\max} \geqslant f^K_j(C_j(\sigma)) \geqslant f^K_j(C_i(\sigma)) = g^K_i(C_i(\sigma)) = g^K_{\max}$, contradicting the assumption.

Proof of (3). Consider an arbitrary job J_i; let J_j be a successor of J_i. As $g^k_i(T) \geqslant g^k_j(T)$ $(k = 1, \ldots, K-1;\ T = 1, \ldots, \Sigma p_j)$, job J_j will be available for processing if job J_i is. Hence, Lawler's algorithm yields an optimal schedule for $1 \mid g^1_{\max} \leqslant F^1, \ldots, g^{K-1}_{\max} \leqslant F^{K-1} \mid g^K_{\max}$ that satisfies the precedence constraints, provided that ties are settled according to the precedence constraints. □

COROLLARY 3. *The* $1 \mid \bar{d}_j, prec \mid P(f^1_{\max}, \ldots, f^K_{\max})$ *problem can be solved in* $O(n^{2K})$ *time for* $K = 2,3$, *and in* $O(n^{K(K+1)-6})$ *time for* $K \geqslant 4$. □

The second problem we consider in this section is $1 \mid nmit \mid P(g^1_{\max}, \ldots, g^K_{\max})$, where the maximum cost functions $g^k_{\max}$ $(k = 1, \ldots, K)$ are induced by penalty functions g^k_j $j = 1, \ldots, n; k = 1, \ldots, K)$ that are non-increasing functions of the job completion times. In order to avoid unbounded solutions, we make the additional assumption that no machine idle time is allowed. This assumption is denoted by the acronym *nmit*, and implies that all jobs are processed in the time interval $[0, \Sigma p_j]$. We show that this problem can be transformed into a problem that fits in the existing framework, and hence, that it is solved in $O(n^{2K})$ time for $K = 2,3$, and in $O(n^{K(K+1)-6})$ time for $K \geqslant 4$.

THEOREM 9. *Lawler's algorithm solves* $1 \mid g^1_{\max} \leqslant G^1, \ldots, g^{K-1}_{\max} \leqslant G^{K-1}, nmit \mid g^K_{\max}$ *to optimality.*

PROOF. Consider an arbitrary instance of the $1|g_{\max}^1 \leqslant G^1, \ldots, g_{\max}^{K-1} \leqslant G^{K-1}, nmit|g_{\max}^K$ problem. Now construct the following instance of $1|f_{\max}^1 \leqslant F^1, \ldots, f_{\max}^{K-1} \leqslant F^{K-1}|f_{\max}^K$. The processing times are identical for both problems, $f_i^k(T) = g_i^k(\Sigma p_j + p_i - T)$ $(i = 1, \ldots, n; k = 1, \ldots, K; T = 1, \ldots, \Sigma p_j)$, and $F^k = G^k$, for $k = 1, \ldots, K-1$. Suppose that Lawler's algorithm yields schedule σ for $1|f_{\max}^1 \leqslant F^1, \ldots, f_{\max}^{K-1}|f_{\max}^K$. An optimal schedule $\bar{\sigma}$ for $1|g_{\max}^1 \leqslant G^1, \ldots, g_{\max}^{K-1} \leqslant G^{K-1}, nmit|g_{\max}^K$ is obtained by reversing σ; $C_i(\bar{\sigma}) = \Sigma p_j + p_i - C_i(\sigma)$ $(i = 1, \ldots, n)$, and hence, $g_i^k(C_i(\bar{\sigma})) = f_i^k(C_i(\sigma))$ $(i = 1, \ldots, n; k = 1, \ldots, K)$. This implies that σ is optimal and feasible if and only if $\bar{\sigma}$ is optimal and feasible. □

COROLLARY 4. *A point* $(F^1, \ldots, F^K)$ *is Pareto optimal with respect to* $(f_{\max}^1, \ldots, f_{\max}^K)$ *if and only if this point is Pareto optimal with respect to* $(g_{\max}^1, \ldots, g_{\max}^K)$. □

From Corollary 4 it follows immediately that we can solve $1|nmit|P(g_{\max}^1, \ldots, g_{\max}^K)$ by transforming it to an $1||P(f_{\max}^1, \ldots, f_{\max}^K)$ problem as described in Theorem 9, and by applying Algorithm B to this instance. As a deadline $\bar{d}_j$ for the $1||P(f_{\max}^1, \ldots, f_{\max}^K)$ problem corresponds to a release date r_j, that is, a lower bound on the start time for J_j, $1|r_j, nmit, prec|P(g_{\max}^1, \ldots, g_{\max}^K)$ is solvable in $O(n^{2K})$ time for $K = 2,3$, and in $O(n^{K(K+1)-6})$ time for $K \geqslant 4$.

ACKNOWLEDGEMENT
The author is grateful to Jan Karel Lenstra and Steef van de Velde for their helpful comments.

REFERENCES

E.L. LAWLER (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science 19*, 544-546.

W.E. SMITH (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 3*, 59-66.

A new lower bound approach for single-machine multicriteria scheduling

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

This paper will appear in Operations Research Letters.

A New Lower Bound Approach for Single-Machine Multicriteria 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

The concept of maximum potential improvement has played an important role in computing lower bounds for single-machine scheduling problems with composite objective functions that are linear in the job completion times. We introduce a new method for lower bound computation: objective splitting. We show that it dominates the maximum potential improvement method in terms of speed and quality.

(1980) Mathematics Subject Classification (Revision 1985): 90B35.
Keywords & Phrases: single-machine scheduling, multicriteria scheduling, maximum potential improvement, objective splitting.

1. Introduction

A single-machine job shop can be 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 no more than one job at a time. Each job $J_i\ (i = 1, \ldots, n)$ requires processing during a positive time p_i. In addition, it has a due date d_i, at which it should ideally be completed. A *schedule* defines for each job J_i its completion time C_i such that no two jobs overlap in their execution. A *performance measure* or *scheduling criterion* associates a value $f(\sigma)$ with each feasible schedule σ. Some well-known measures are the sum of the job completion times ΣC_i, the maximum job lateness $L_{\max} = \max_{1 \leqslant i \leqslant n}(C_i - d_i)$, and the maximum job earliness $E_{\max} = \max_{1 \leqslant i \leqslant n}(d_i - C_i)$.

In this paper, we adopt the terminology of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) to classify scheduling problems. Scheduling problems are classified according to a three-field notation $\alpha | \beta | \gamma$, where α specifies the machine environment, β the job characteristics, and γ the objective function. For instance, $1 | nmit | E_{\max}$ denotes the single-machine problem of minimizing maximum earliness, where *nmit* denotes that no machine idle time is allowed.

Most research has been concerned with a single criterion. In real life scheduling, however, it is necessary to take several performance measures into account. There are basically two approaches to cope with multiple criteria. If the scheduling criteria are subject to a well-defined hierarchy, they can be considered *sequentially* in order of relevance. An example is the problem of minimizing maximum lateness subject to the minimum number of tardy jobs, for which Shanthikumar (1983) presents a branch-and-bound algorithm.

The second approach is *simultaneous* optimization of several criteria. The K performance measures specified by the functions f_k $(k = 1, \ldots, K)$ are then transformed into one single *composite objective* function $F : \Omega \rightarrow \mathbb{R}$, where Ω denotes the set of all feasible schedules. We restrict ourselves to the case that F is a linear composition of the individual performance measures. This leads to the problem class (P) that contains all problems that can be formulated as

$$\min_{\sigma \in \Omega} \sum_{k=1}^{K} \alpha_k f_k(\sigma), \qquad \text{(P)}$$

where $\alpha = (\alpha_1, \ldots, \alpha_K)$ is a given vector of real nonnegative weights. The problem of minimizing a linear function of the number of tardy jobs and maximum lateness, denoted as $1 \,|\,|\, \Sigma U_i + L_{\max}$, is a member of this class. Nelson, Sarin, and Daniels (1986) present a branch-and-bound algorithm for its solution.

In addition to solving some problem in (P) for a given $\alpha \geqslant 0$, it may be of interest to determine the extreme set. The *extreme set* for given functions $f_1, \ldots, f_K$ is defined as the minimum cardinality set that contains an optimal schedule for *any* weight vector $\alpha \geqslant 0$. The elements of this set are the *extreme schedules*. If this set has been identified, then we can solve any problem for these functions by computing the function value for each extreme schedule and choosing the best. Hence, if the cardinality of the extreme set is polynomially bounded in n, the number of jobs, and if each extreme schedule can be found in polynomial time, then any problem in (P) with respect to these functions $f_1, \ldots, f_K$ can be solved in polynomial time.

Suppose that some problem in (P) is $\mathcal{NP}$-hard and that one wishes to design a branch-and-bound method for its solution. In that case, good lower bounds are required. Until now, virtually all lower bound computations for problems in (P) are based upon the so-called *maximum potential improvement* method. We prove in Section 2 that these bounds are dominated in terms of quality and computational effort by a much simpler method that we name *objective splitting*. In Section 3, we refine the basic objective splitting method.

The problem $1 \,|\,|\, \Sigma C_i + L_{\max} + E_{\max}$ is our benchmark in comparing the two lower bound approaches. It is still an open question whether this problem is $\mathcal{NP}$-hard. Sen, Raiszadeh, and Dileepan (1988) develop a branch-and-bound algorithm and derive lower bounds by means of the maximum potential improvement method. There is an optimal schedule for this problem without machine idle time, although $E_{\max}$ is nonincreasing in the job completion times. It is not meaningful to insert idle time, as the gain for $E_{\max}$ will at least be compensated by the increase of ΣC_i. We recall the following fundamental algorithms for the three embedded subproblems.

THEOREM 1 (Smith, 1956). *The* $1 | | \Sigma C_i$ *problem is solved by sequencing the jobs according to the shortest-processing-time (SPT) rule, that is, in order of nondecreasing* p_i.

THEOREM 2 (Jackson, 1955). *The* $1 | | L_{\max}$ *problem is solved by sequencing the jobs according to the earliest-due-date (EDD) rule, that is, in order of nondecreasing* d_i.

THEOREM 3. *The* $1 | nmit | E_{\max}$ *problem is solved by sequencing the jobs according to the minimum-slack-time (MST) rule, that is, in order of nondecreasing* $d_i - p_i$.

The proof of each of these algorithms proceeds by a straightforward interchange argument. Note that each of these problems is solved by arranging the jobs in a certain *priority order* that can be specified in terms of the parameters of the problem type.

The optimal solution values for these single-machine scheduling problems will be denoted by ΣC_i^*, $L_{\max}^*$, and $E_{\max}^*$, respectively. Furthermore, $\Sigma C_i(\sigma)$, $L_{\max}(\sigma)$, and $E_{\max}(\sigma)$ are the objective values for the schedule σ. In analogy, $C_i(\sigma)$, $L_i(\sigma)$, and $E_i(\sigma)$ denote the respective measures for job J_i $(i = 1, \ldots, n)$. Whenever (σ) is omitted, we are considering the performance measure in a generic sense, or there is no confusion possible as to the schedule we are referring to. The schedules that minimize ΣC_i, $L_{\max}$, and $E_{\max}$ are referred to as *SPT, EDD,* and *MST* respectively. In addition, $v(\cdot)$ denotes the optimal objective value for problem $\cdot$.

2. Maximum potential improvement versus objective splitting

Townsend (1978) proposed the maximum potential improvement method to compute lower bounds for minimizing a *quadratic* function of the job completion times. Since then, the method has been extended to problems in (P), including $1 | | \Sigma C_i + L_{\max}$ (Sen and Gupta, 1983), $1 | nmit | L_{\max} + E_{\max}$ (Gupta and Sen, 1984), and $1 | | \Sigma C_i + L_{\max} + E_{\max}$ (Sen, Raiszadeh, and Dileepan, 1988). To our knowledge, there is only one publication on objective splitting *avant la lettre*: Tegze and Vlach (1988) obtained an extremely simple, but provably stronger lower bound for $1 | nmit | L_{\max} + E_{\max}$.

Meanwhile, Hoogeveen (1990) and Hoogeveen and Van de Velde (1990) have found polynomial-time algorithms for $1 | nmit | \alpha_1 L_{\max} + \alpha_2 E_{\max}$ and $1 | | \alpha_1 \Sigma C_i + \alpha_2 L_{\max}$. The former problem has $O(n)$ extreme schedules, each of which is found in $O(n \log n)$ time. The latter problem has $O(n^2)$ extreme schedules, each of which is determined in $O(n)$ time after appropriate preprocessing. However, it is an interesting issue how to derive lower bounds for $\mathcal{NP}$-hard problems in (P). The maximum potential improvement method is a cumbersome procedure. However, by viewing it from a different angle, we derive a closed expression for the resulting lower bound. It is then immediately clear that the maximum potential improvement method is completely dominated by the much simpler objective splitting method.

Objective splitting is based upon the observation that

$$\min_{\sigma \in \Omega} \left[\sum_{k=1}^{K} \alpha_k f_k(\sigma) \right] \geqslant \sum_{k=1}^{K} \alpha_k \left[\min_{\sigma \in \Omega} f_k(\sigma) \right],$$

if $\alpha_k \geqslant 0$ for $k = 1, \ldots, K$. The application of this idea to $1 \,|\,|\, \Sigma C_i + L_{\max} + E_{\max}$ yields the problems $1 \,|\,|\, \Sigma C_i$, $1 \,|\,|\, L_{\max}$, and $1 \,|\, nmit \,|\, E_{\max}$. Each problem is polynomially solvable, and we obtain the bound $LB^{OS} = \Sigma C_i^* + L_{\max}^* + E_{\max}^*$. This bound is computed in $O(n)$ time in each node of the search tree, provided that the *SPT*, *EDD*, and *MST* sequences have been stored and that we employ a convenient branching strategy.

It is relatively easy to apply the *maximum potential improvement* method to problems in (P) for which each embedded single-machine problem has a priority order. The $1 \,|\,|\, \Sigma C_i + L_{\max} + E_{\max}$ problem has three: the *SPT* order for ΣC_i, the *EDD* order for $L_{\max}$, and the *MST* order for $E_{\max}$. Clearly, we have solved an instance of this problem in case these orders concur; in general though, the priority orders are conflicting.

Suppose we start with the *MST* schedule, which we refer to as the *primary* priority order. The scheduling cost induced by the *MST* schedule is $\Sigma C_i(MST) + E_{\max}^* + L_{\max}(MST)$; this is obviously an upper bound on the optimal solution value. In addition, we know that any optimal schedule σ^* must have $E_{\max}(\sigma^*) \geqslant E_{\max}^*$, and $\Sigma C_i(\sigma^*) + L_{\max}(\sigma^*) \leqslant \Sigma C_i(MST) + L_{\max}(MST)$. The maximum potential improvement method assesses the current schedule with respect to the maximum improvement that can be obtained for each of the performance measures *separately*. Accordingly, we get a lower bound by subtracting the total maximum potential improvement from the upper bound.

First, consider the maximum lateness criterion, which is the *secondary* priority order. If we interchange every pair of adjacent jobs J_i and J_j for which $d_i > d_j$ and $C_i < C_j$, then we need to conduct $O(n^2)$ interchanges before we have transformed the *MST* schedule into an *EDD* schedule. The actual effect on the objective value by one particular interchange depends on the interchanges that have been conducted thus far. It might have no effect whatsoever on the performance of the schedule; this is true if both the maximum lateness and the maximum earliness remain unchanged. The *maximum possible decrease* of the scheduling cost, however, is $d_i - d_j$; if σ and π denote the schedule before and after the interchange, respectively, then the maximum decrease is realized if $L_{\max}(\sigma) = L_j(\sigma)$, $L_{\max}(\pi) = L_i(\pi)$ and $E_{\max}(\pi) = E_{\max}(\sigma)$. The effect that the interchange might have on the sum of the job completion times is not considered here and dealt with separately. Any interchange conducted to transform the *MST* schedule into the *EDD* schedule may improve the maximum lateness by the corresponding maximum possible decrease. The sum of these is the *maximum potential improvement* with respect to the initial lateness $L_{\max}(MST)$. It is given by

$$MPI_2 = \sum_{i,j:\, d_i > d_j,\, C_i < C_j} (d_i - d_j).$$

Note that the maximum potential improvement does not depend on the order in which the interchanges are conducted.

Second, the sum of the job completion times, which is the *tertiary* priority order, is reduced by interchanging two adjacent jobs J_i and J_j with $p_i > p_j$ and $C_i < C_j$. The maximum potential improvement is then $p_i - p_j$, which is also the *true* improvement. The maximum potential improvement with respect to

$\Sigma C_j(MST)$ is then

$$MPI_3 = \sum_{i,j:p_i > p_j, C_i < C_j} (p_i - p_j).$$

The lower bound LB^{MPI} suggested by Sen, Raiszadeh, and Dileepan (1988) for $1 \mid \mid \Sigma C_i + L_{\max} + E_{\max}$ is then

$$LB^{MPI} = E^*_{\max} + L_{\max}(MST) - MPI_2 + \Sigma C_i(MST) - MPI_3.$$

Since $\Sigma C_i(MST) - MPI_3 = \Sigma C_i(SPT) = \Sigma C^*_i$ and $L_{\max}(MST) - MPI_3 \leqslant L^*_{\max}$, as we have systematically overestimated the reduction in maximum lateness, we conclude that

$$LB^{MPI} = E^*_{\max} + \Sigma C^*_i + L_{\max}(MST) - MPI_2 \leqslant LB^{OS}.$$

The maximum potential improvement method can be generalized to problems in (P) as follows. Let σ^*_k denote an optimal schedule for the kth individual objective. Furthermore, let the optimal sequence that goes with the kth objective be the kth preference order. The first step is then to sequence the jobs according to the primary preference order, which gives the upper bound $\alpha_1 f_1(\sigma^*_1) + \Sigma_{k=2}^{K} \alpha_k f_k(\sigma^*_1)$. We then have to transform the primary preference order into the kth preference order, for $k = 2, \ldots, K$, and determine the corresponding maximum potential improvement MPI_k. The lower bound is then given by

$$LB^{MPI} = \alpha_1 f_1(\sigma^*_1) + \sum_{k=2}^{K} \alpha_k (f_k(\sigma^*_1) - MPI_k).$$

Note that this procedure requires $O(n^2)$ time for fixed K in addition to the time required to determine σ^*_k, for $k = 1, \ldots, K$. Since $f_k(\sigma^*_1) - MPI_k \leqslant f_k(\sigma^*_k)$ for each $k = 1, \ldots, K$, we have the following theorem.

THEOREM 4. *For any problem in* (P), *the lower bound obtained by the maximum potential improvement method is dominated in terms of both quality and speed by the lower bound obtained by the objective splitting method.* □

Consider the following example that is taken from Sen, Raiszadeh, and Dileepan (1988) for the problem $1 \mid \mid q\Sigma C_i + (1-q)(L_{\max} + E_{\max})$ with $0 \leqslant q \leqslant 1$.

J_i	J_1	J_2	J_3	J_4
p_i	14	7	6	7
d_i	20	14	15	17
$d_i - p_i$	6	7	9	10

By means of the maximum potential improvement method, we obtain the lower bound $LB^{MPI} = 64q + 9$. It is easy to verify that $\Sigma C^*_j = 73$, $L^*_{\max} = 14$, and $E^*_{\max} = 6$. This gives the bound $LB^{OS} = 53q + 20$. Note that $53q + 20 \geqslant 64q + 9$ for all q with $0 \leqslant q \leqslant 1$.

3. Improving the objective splitting procedure

The objective splitting procedure above was given in its simplest form: we separated the composite objective function into K single-criterion scheduling problems. We now propose a refinement that gives us a lower bound that is at least as good, but requires more time. Our more general approach allows combinations of objective functions. Let $(T_1, \ldots, T_H)$ be a partition of the set $\{1, \ldots, K\}$, i.e., the sets T_h are mutually disjoint and $\cup_{h=1}^{H} T_h = \{1, \ldots, K\}$. For any problem A in the class (P) we clearly have

$$v(A) \geqslant \sum_{h=1}^{H} \left[\min_{\sigma \in \Omega} \sum_{k \in T_h} \alpha_k f_k(\sigma_k) \right] \geqslant \sum_{k=1}^{K} \alpha_k \left[f_k(\sigma_k^*) \right] = LB^{OS}.$$

This idea can be refined even further, since it is not obligatory to match each performance criterion f_k with only one set T_h. Hence, let us relax the assumption that $(T_1, \ldots, T_H)$ is a partition of $\{1, \ldots, K\}$, and let α_{kh} denote the *fraction* of f_k that is assigned to T_h. We must have that $\Sigma_h \alpha_{kh} = \alpha_k$ for each $k = 1, \ldots, K$, and also that $\alpha_{kh} \geqslant 0$, since the composite objective function associated with the set T_h has to be nondecreasing in each of its arguments, for $h = 1, \ldots, H$. We can compute the lower bound for *given* values of α_{kh} as

$$v(OS) = \sum_{h=1}^{H} \left[\min_{\sigma \in \Omega} \sum_{k \in T_h} \alpha_{kh} f_k(\sigma) \right]. \qquad \text{(OS)}$$

An interesting question is how to determine the values of α_{kh} that maximize the lower bound $v(OS)$. This problem, referred to as problem (D), is to

$$\text{maximize } v(OS) \qquad \text{(D)}$$

subject to

$$\sum_{h=1}^{H} \alpha_{kh} = \alpha_k \quad \text{for } k = 1, \ldots, K,$$

$$\alpha_{kh} \geqslant 0 \quad \text{for } k = 1, \ldots, K,\ h = 1, \ldots, H.$$

A sufficient condition for solving problem (D) in polynomial time (for fixed K) is that the extreme set for each problem induced by T_h $(h = 1, \ldots, H)$ can be determined in polynomial time. In that case, there is only a polynomial number of extreme schedules involved, and problem (D) can then be formulated as a linear programming problem with a polynomial number of constraints and variables. Let $N(h)$ be the number of extreme schedules for the problem associated with T_h $(h = 1, \ldots, H)$, and let $\sigma_{j(h)}$ denote the jth extreme schedule for the problem associated with T_h. There are at most $2^K - 2$ sets T_h $(|T_h| < K$ and $T_h \neq \varnothing)$. The linear program is then to

$$\text{maximize } w$$

subject to

$$w \leqslant \sum_{h=1}^{H} \sum_{k \in T_h} \alpha_{kh} f_k(\sigma_{j(h)}) \qquad \text{for } j(h) = 1, \ldots, N(h),\ h = 1, \ldots, H,$$

$$\sum_{h=1}^{H} \alpha_{kh} = \alpha_k \qquad \text{for } k = 1, \ldots, K,$$

$$\alpha_{kh} \geqslant 0 \qquad \text{for } k = 1, \ldots, K,\ h = 1, \ldots, H.$$

In general, it would be unreasonable to presume that each of the possible $2^K - 2$ sets T_h would result into a polynomially solvable problem; it may be a formidable challenge to identify those that will. If we touch upon a problem that appears to be hard to solve, then we may relax the assumptions by allowing preemption. (I.e., the processing of jobs may be interrupted and resumed at a later moment in time; this is denoted by *pmtn*.) This may be useful with respect to the computational complexity, but also with respect to the lower bound quality. The latter follows particularly from the following theorem.

THEOREM 6. *The optimal objective value of* $1|pmtn|\Sigma_{k=1}^{K}\alpha_k f_k$ *is greater than or equal to* $\Sigma\alpha_k f_k(\sigma_k^*)$, *where* σ_k^* *is the optimal value for* $1||f_k$ $(k = 1, \ldots, K)$.

PROOF. The proof follows from the observation that σ_k^* also solves $1|pmtn|f_k$, if f_k is either monotonically nondecreasing or monotonically nonincreasing in the job completion times. □

If we apply the refined objective splitting procedure to $1||\Sigma C_i + L_{\max} + E_{\max}$, then, except for the obvious single-criterion problems, we have to consider three problems: $1||\alpha_1\Sigma C_i + \alpha_2 L_{\max}$, $1|nmit|\alpha_1\Sigma C_i + \alpha_2 E_{\max}$, and $1|nmit|\alpha_1 L_{\max} + \alpha_2 E_{\max}$. Hoogeveen (1990) presents an $O(n^2 \log n)$ time algorithm for $1|nmit|\alpha_1 L_{\max} + \alpha_2 E_{\max}$ to find the $O(n)$ extreme schedules, and Hoogeveen and Van de Velde (1990) present an $O(n^3)$ time algorithm for $1||\alpha_1\Sigma C_i + \alpha_2 L_{\max}$, which has $O(n^2)$ extreme schedules. For the problem $1|nmit|\alpha_1\Sigma C_i + \alpha_2 E_{\max}$, there is only a polynomial-time algorithm available if $\alpha_1 \geqslant \alpha_2$ (Hoogeveen and Van de Velde, 1990). The complexity of the case $\alpha_1 < \alpha_2$ is unknown. However, $1|nmit,pmtn|\alpha_1\Sigma C_i + \alpha_2 E_{\max}$ is solvable in $O(n^4)$ time and has $O(n^2)$ extreme schedules.

If we reconsider the example, we find that there is one extreme schedule for ΣC_i and $L_{\max}$ with $\Sigma C_i = 73$ and $L_{\max} = 14$; there are two extreme schedules for $L_{\max}$ and $E_{\max}$ with values $L_{\max} = 14$ and $E_{\max} = 7$, and $L_{\max} = 17$ and $E_{\max} = 6$; there are three extreme schedules for $E_{\max}$ and ΣC_i if we allow preemption with values $E_{\max} = 6$ and $\Sigma C_i = 96$, $E_{\max} = 7$ and $\Sigma C_i = 74$, and $E_{\max} = 9$ and $\Sigma C_i = 73$, respectively.

The lower bound that is obtained by the improved objective splitting method depends on the parameter q. Suppose $q = \frac{1}{2}$. Then we obtain $LB^{MPI} = 41$ and $LB^{OS} = 46\frac{1}{2}$. It is easy to verify that the improved objective splitting method gives $47\frac{1}{2}$ as a lower bound. This bound is tight, since the optimal sequence (J_2, J_3, J_4, J_1) has the same value.

ACKNOWLEDGEMENT

The authors are grateful to Jan Karel Lenstra for his comments on earlier drafts of this paper.

REFERENCES

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.

S.K. GUPTA AND T. SEN (1984). Minimizing the range of lateness on a single machine. *Journal of the Operational Research Society 35*, 853-857.

J.A. HOOGEVEEN (1990). *Minimizing maximum earliness and maximum lateness on a single machine*, Report BS-R9001, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1990). *Polynomial-time algorithms for multicriteria scheduling problems*, Report BS-R9008, CWI, Amsterdam.

J.R. JACKSON (1955). *Scheduling a production line to minimize maximum tardiness*, Research Report 43, Management Sciences Research Project, UCLA.

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

T. SEN AND S.K. GUPTA (1983). A branch-and-bound procedure to solve a bicriterion scheduling problem. *IIE Transactions 15*, 84-88.

T. SEN, F.M.E. RAISZADEH AND P. DILEEPAN (1988). A branch-and-bound approach to the bicriterion scheduling problem involving total flowtime and range of lateness. *Management Science 34*, 254-260.

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.

M. TEGZE AND M. VLACH (1988). Improved bounds for the range of lateness on a single machine. *Journal of the Operational Research Society 39*, 675-680.

W. TOWNSEND (1978). The single machine problem with quadratic penalty function of completion times: a branch-and-bound solution. *Management Science 24*, 530-534.

Scheduling around a small common due date

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

This paper will appear in the European Journal of Operational Research.

Scheduling Around a Small Common Due Date

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 Eindhoven, 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 and has a positive weight. The problem is to find a schedule that minimizes the sum of weighted deviations of the job completion times from a given common due date d, which is smaller than the sum of the processing times. We prove that this problem is $\mathcal{NP}$-hard even if all job weights are equal. In addition, we present a pseudopolynomial algorithm that requires $O(n^2 d)$ time and $O(nd)$ space.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single machine scheduling, $\mathcal{NP}$-hardness, pseudopolynomial algorithm.

1. Introduction

Recently, we have seen a growing interest in just-in-time manufacturing. This concept decrees that products should be completed as close to their due dates as possible in order to avoid both storage costs as a result of early completions and penalty costs inflicted on account of late deliveries. This might induce the following type of problems for the single machine job shop.

A set of n independent jobs has 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 ($i = 1, \ldots, n$) has a given positive uninterrupted processing time p_i and should ideally be completed at a given due date d_i. Without loss of generality, we assume that the processing times and the due dates are integral. A *schedule* defines for each job J_i a completion time C_i such that the jobs do not overlap in their execution. Given a schedule S, the earliness and tardiness of job J_i are defined as $E_i = \max\{d_i - C_i, 0\}$ and $T_i = \max\{C_i - d_i, 0\}$, respectively. The just-in-time philosophy is reflected in the objective function

$$f(S) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i).$$

For a review on problems with this type of objective function, see Baker and Scudder (1990).

An important subclass contains the set of problems that deal with a common due date d for all jobs. The common due date is either specified as part of the problem instance, or is a decision variable that has to be optimized with the job sequence simultaneously. As the first job may start later than time 0, the optimal schedule is identical for both problems unless the common due date d is restrictively small ($d < \Sigma p_i$). Therefore, the first variant is referred to as the restricted problem and the second variant as the unrestricted problem.

We will call the earliness and tardiness penalty weights symmetric if $\alpha_i = \beta_i$ for each $i = 1, \ldots, n$. For the case of nonsymmetric weights, only one problem type has been investigated, namely the case in which all α_i are equal and all β_i are equal. Bagchi, Chang and Sullivan (1987) and Emmons (1987) present an $O(n \log n)$ algorithm for the unrestricted variant, while Bagchi et al. (1987) propose a branch-and-bound algorithm for the restricted problem.

If the earliness and tardiness penalty weights are symmetric, then the problem reduces to finding a schedule S that minimizes the weighted sum of the deviations of the completion times from the common due date:

$$f(S) = \sum_{i=1}^{n} w_i \,|\, C_i - d \,|.$$

There are two notable results for the case that $d \geqslant \Sigma p_i$. Kanet (1981) gives an $O(n \log n)$ time algorithm to find an optimal schedule, if all weights are equal. Hall and Posner (1991) show that the problem with symmetric weights is $\mathcal{NP}$-hard.

In contrast, we focus our attention on the case that $d < \Sigma p_i$. In Section 2 we prove some properties of an optimal schedule. In Section 3 we establish $\mathcal{NP}$-hardness of the problem, even for the case that all job weights are equal. We note that Hall, Kubiak and Sethi (1991) independently obtained this result by a slightly more complicated proof. This result justifies the development of enumerative algorithms by Bagchi, Sullivan and Chang (1986) and by Szwarc (1989) for minimizing $\Sigma_{i=1}^{n} | C_i - d |$ subject to a common due date $d < \Sigma p_i$. In contrast, we present a pseudopolynomial algorithm in Section 4 for $1 | | \Sigma w_i | C_i - d |$, which requires $O(n^2 d)$ time and $O(nd)$ space. Our algorithm is applicable to a more general problem type than the pseudopolynomial algorithm of Hall et al. (1991), which can only handle equal job weights. In Section 5 we present some well-solvable cases.

2. Basic concepts

It is straightforward to verify that no optimal solution has any idle time between the execution of jobs. In case there were idle time, the scheduling cost could be reduced by closing the gap. The next two theorems further characterize any optimal solution.

THEOREM 1. *In any optimal schedule S, the jobs J_i that are completed before or at the common due date d are scheduled in order of nondecreasing values of w_i / p_i, and the jobs that are started at or after d are scheduled in order of nonincreasing values of w_i / p_i.*

PROOF. This follows immediately from Smith's ratio rule (Smith, 1956). □

THEOREM 2. *In each optimal schedule S, either the first job starts at time* 0 *or the due date d coincides with the start time or completion time of the job with the largest ratio* w_i/p_i.

PROOF. For a given schedule S, let $B(S)$ denote the set of jobs that are completed before or at the common due date and $A(S)$ the set of jobs completed after the due date. Define $\Delta = \Sigma_{J_i \in B(S)} w_i - \Sigma_{J_i \in A(S)} w_i$. We consider the cases in which $\Delta < 0$ and $\Delta \geqslant 0$ separately.

Suppose first $\Delta < 0$. If S starts at time $T > 0$, determine $t = \min\{T, \min_{J_i \in A(S)} C_i - d\}$. If the entire schedule is put t time units earlier, then the reduction in cost equals $-t\Delta > 0$. In the new situation either schedule S starts at time $T = 0$ or one job has moved from $A(S)$ to $B(S)$. If still $T > 0$ and $\Delta < 0$, we repeat the procedure until we arrive at a situation in which $T = 0$ or $\Delta \geqslant 0$, and no further improvement is possible. The latter case implies that the due date coincides with the completion time of one job and the start time of another. Because of Theorem 1, one of these jobs must be the job with the largest ratio w_i/p_i.

On the other hand, in the case of $\Delta \geqslant 0$, reverse arguments can be applied to show that the due date coincides with the completion or start time of the job with the largest ratio w_i/p_i. □

Note that Theorem 1 does not impose any restrictions on a job that is started before and completed after the due date. Consider the following instance with $n = 3, p_1 = 8, p_2 = 10, p_3 = 4, w_1 = 5, w_2 = 7, w_3 = 3$, and $d = 15$. The optimal solution is shown in Figure 1 and demonstrates that such a job can exist, and that it can even have the smallest ratio w_i/p_i.

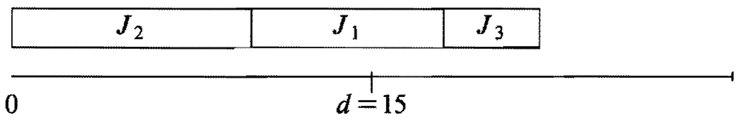


FIGURE 1

3. SCHEDULING AROUND A SMALL COMMON DUE DATE IS $\mathcal{NP}$-HARD

In this section we prove that this problem is $\mathcal{NP}$-hard even if $w_i = 1$ for each job J_i, by showing that the corresponding decision problem is $\mathcal{NP}$-complete. The reduction is from Even-Odd Partition.

EVEN-ODD PARTITION (Garey, Tarjan and Wilfong, 1988): Given a set of $2n$ positive integers $B = \{b_1, \ldots, b_{2n}\}$ such that $b_i > b_{i+1}$ for each $i = 1, \ldots, 2n-1$, is there a partition of B into two subsets B_1 and B_2 such that $\Sigma_{b_i \in B_1} b_i = \Sigma_{b_i \in B_2} b_i = A$ and such that B_1 contains exactly one of $\{b_{2i-1}, b_{2i}\}$ for each $i = 1, \ldots, n$?

We start by describing a reduction from the Even-Odd Partition problem to the small common due date problem with $w_i = 1$ for all J_i. Let $B = \{b_1, \ldots, b_{2n}\}$ be an arbitrary instance of the Even-Odd Partition problem, with $A = \Sigma b_i / 2$. Construct the following set of jobs: $2n$ 'partition' jobs J_i with processing times $p_i = b_i + nA$ for each $i = 1, \ldots, 2n$, an additional job J_0 with $p_0 = 3(n^2+1)A$, weights $w_i = 1$ for $i = 0, \ldots, 2n$, and a common due date $d = (n^2+1)A$. In addition, we define a threshold value $y_0 = \Sigma_{i=1}^{n}[(i+1)(p_{2i-1} + p_{2i})] + d$ on the scheduling cost.

Consider a partitioning of the set of partition jobs $\{J_1, \ldots, J_{2n}\}$ into the sets $B_1 = \{J_{11}, J_{21}, \ldots, J_{n1}\}$ and $B_2 = \{J_{12}, J_{22}, \ldots, J_{n2}\}$, where $\{J_{i1}, J_{i2}\} = \{J_{2i-1}, J_{2i}\}$ for each $i = 1, \ldots, n$.

LEMMA 1. *If the partitioning into the sets B_1 and B_2 corresponds to a solution of the Even-Odd Partition problem, then the cost of schedule S_0 constructed as shown in Figure 2 equals the threshold value y_0.*

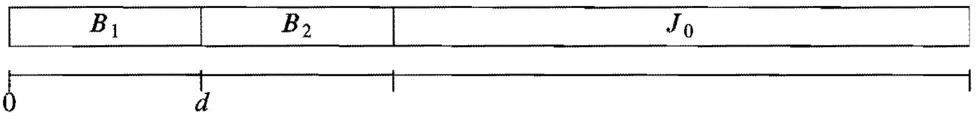


FIGURE 2: SCHEDULE S_0

PROOF. Note that the jobs in B_1 and B_2 are scheduled as indicated in Theorem 1. The verification then only requires straightforward computations. □

We now prove that, conversely, any schedule S with $f(S) \leqslant y_0$ must have the same structure as S_0, and that the subsets B_1 and B_2 must correspond to a solution of the Even-Odd Partition problem.

PROPOSITION 1. *Suppose S is a schedule with scheduling cost $f(S) \leqslant y_0$. Then S has the following properties.*
(1) *At most n jobs can be completed before the due date d.*
(2) *The first job must start at time* 0.
(3) *The additional job J_0 is scheduled last.*
(4) *At least $n-1$ jobs must be completed before the due date d.*

PROOF.
(1) This is due to the choice of the processing times.
(2) This follows immediately from the first property and the proof of Theorem 2.
(3) Suppose J_0 is not scheduled last. Then, because of Theorem 1, J_0 must start before the common due date d. Since at most n jobs can be scheduled before job J_0, for at least $n+1$ jobs in S we have $C_i - d \geqslant p_0 - d = 2d$. This implies that $f(S) \geqslant 2(n+1)d \geqslant (n+4)d$. However, as each of the multipliers of $p_1, \ldots, p_n$ in y_0 is at most $\frac{1}{2}(n+3)$, while $\Sigma_{i=1}^{n}(i+1) = \frac{1}{2}n(n+3)$, we have the following inequality:

$$y_0 = \sum_{i=1}^{n}[(i+1)(p_{2i-1}+p_{2i})] + d < \tfrac{1}{2}(n+3)\sum_{i=1}^{2n}p_i + d = (n+4)d \leqslant f(S),$$

which contradicts the assumption.
(4) This follows immediately from the first three properties and the choice of the processing times. □

LEMMA 2. *Suppose S is an optimal schedule with $f(S) \leqslant y_0$. Then the due date d must coincide with the completion time of the n-th job in the schedule S, the schedule S must have the same structure as the schedule S_0, and provide an affirmative answer to the Even-Odd Partition problem.*

PROOF. Assume that $s(i)$ denotes the index of the job that is scheduled on position i in schedule S. We compute the scheduling cost relative to the imaginary due date $k = p_{s(1)} + \dots + p_{s(n)}$. Then we have

$$\begin{aligned}\sum_{i=0}^{2n} |C_i - k| &= \sum_{i=1}^{n}[(i-1)p_{s(i)}] + \sum_{i=n+1}^{2n}[(2n+2-i)p_{s(i)}] + 3d = \\ &\sum_{i=1}^{n}[(i+1)p_{s(i)}] + \sum_{i=n+1}^{2n}[(2n+2-i)p_{s(i)}] + 3d - 2k = \\ &\sum_{i=1}^{n}[(i+1)p_{s(i)}] + \sum_{i=1}^{n}[(i+1)p_{s(2n+1-i)}] + 3d - 2k \geqslant \\ &\sum_{i=1}^{n}[(i+1)(p_{2i-1}+p_{2i})] + 3d - 2k = y_0 + 2d - 2k.\end{aligned}$$

The true scheduling cost $f(S)$ can be written as

$$f(S) = \sum_{i=0}^{2n} |C_i - d| = \sum_{i=0}^{2n} |C_i - k| + (d-k)(card(B(S)) - card(A(S))),$$

where *card* denotes the cardinality function. Because of Proposition 1, we have only three cases to consider:
- if $d = k$, then $f(S) \geqslant y_0$,
- if $d > k$, then $card(B(S)) = n$, and therefore $f(S) \geqslant y_0 + d - k > y_0$,
- if $d < k$, then $card(B(S)) = n - 1$, and hence $f(S) \geqslant y_0 + k - d > y_0$.

This implies that if $f(S) \leqslant y_0$, then $C_{s(n)} = d$, that is, the completion time of the n-th job in S must coincide with the due date. Furthermore, $f(S) \leqslant y_0$ implies $\{J_{s(i)}, J_{s(2n+1-i)}\} = \{J_{2i-1}, J_{2i}\}$ for $i = 1, \dots, n$. Therefore, the schedule S has the same structure as the schedule S_0 depicted in Figure 2. This means that the original Even-Odd Partition problem has an affirmative answer. □

THEOREM 3. *Given a set of jobs and a nonnegative integer y, the problem of deciding whether there exists a schedule S_0 with $f(S_0) \leqslant y$ is $\mathcal{NP}$-complete.*

PROOF. The decision problem is clearly in $\mathcal{NP}$. For any given instance of the Even-Odd Partition problem, we construct a set of jobs as described above and set $y = y_0$. This reduction requires polynomial time. Theorem 3 now follows from Lemmas 1 and 2. □

4. A DYNAMIC PROGRAMMING ALGORITHM

Theorem 3 implies that, unless $\mathcal{P} = \mathcal{NP}$, no polynomial algorithm exists for solving the small common due date problem. We present a pseudopolynomial algorithm that requires $O(n^2d)$ time and $O(nd)$ space, for which Theorems 1 and 2 provide the basis. According to Theorem 2 we must consider two cases: one in which the job with the largest weight to processing time ratio is scheduled such that either its completion or its start time coincides with the due date, and one in which all the jobs are scheduled in the interval $[0, \Sigma p_i]$.

For the first option, we renumber the jobs according to nonincreasing weight to processing time ratios. Let $F_j(t)$ denote the optimal objective value for the first j jobs subject to the condition that the interval $[d-t,\ d+\Sigma_{i=1}^{j} p_i - t]$ is occupied by the first j jobs. Then the initialization is

$$F_j(t)=\begin{cases}0 & \text{for } t=0, j=0,\\ \infty & \text{otherwise,}\end{cases}$$

and the recursion for $j = 1, \ldots, n$ is given by

$$F_j(t) = \min\{F_{j-1}(t-p_j) + w_j(t-p_j),\ F_{j-1}(t) + w_j(\textstyle\sum_{i=1}^{j} p_i - t)\} \text{ for } 0 \leqslant t \leqslant d.$$

In the second case, all jobs are scheduled in the interval $[0, \Sigma p_i]$. In such a situation it might occur that one of the jobs is started before and yet completed after the due date (see Figure 1). To allow for this possibility, we leave one job out of the recursion, and repeat the recursion n times, once for each job. Since the cost of the schedule can now only be computed relative to the endpoints of the interval, it is assumed that the jobs have been renumbered according to nondecreasing values of w_i/p_i. Consequently, we know that the first job either starts at time 0 or finishes at time Σp_i.

Assume that J_h is the job that will be scheduled around the due date. Let $G_j^h(t)$ denote the optimal cost for the first j jobs subject to the condition that the intervals $[0,t]$ and $[\Sigma_{i=j+1}^{n} p_i + t,\ \Sigma p_i]$ are occupied by the first j jobs. The initialization is

$$G_j^h(t)=\begin{cases}0 & \text{for } t=0, j=0,\\ \infty & \text{otherwise,}\end{cases}$$

and the recursion for $j = 1, \ldots, n$ is

$$G_j^h(t)=\begin{cases}G_{j-1}^h(t) & \text{if } j=h,\\ G_{j-1}^h(t) + w_j(\sum_{i=j}^{n} p_i + t - d) & \text{if } d-p_j \leqslant t \leqslant d,\\ G_{j-1}^h(t-p_j) + w_j(d-t) & \text{if } \Sigma_{j+1}^{n} p_i < d-t,\\ \min\{G_{j-1}^h(t) + w_j(\sum_{i=j}^{n} p_i + t - d), G_{j-1}^h(t-p_j) + w_j(d-t)\} & \text{otherwise.}\end{cases}$$

The recursion leaves the interval $[t, t+p_h]$ idle, and it is here that we insert the job J_h and compute the resulting cost as

$$G_n^h(t)=\begin{cases}G_n^h(t) + w_h(t+p_h-d) & \text{if } d-p_h \leqslant t \leqslant d,\\ \infty & \text{otherwise.}\end{cases}$$

The optimal solution is then found as

$$f(S) = \min\{\min_{1 \leqslant h \leqslant n} \min_{d-p_h \leqslant t \leqslant d} G_n^h(t), \min_{0 \leqslant t \leqslant d} F_n(t)\},$$

by which we have established the following result.

THEOREM 4. *The dynamic programming algorithm solves the problem in $O(n^2 d)$ time and $O(nd)$ space.*

Note that the dynamic programming algorithm can be modified to cope with any common due date problem with nonsymmetric earliness and tardiness penalty weights that allow for a prespecified processing order of the jobs that are completed before or started after the common due date. This includes the problem with all α_i equal and all β_i equal, for which Bagchi et al. (1987) presented a branch-and-bound algorithm. In addition, the weighted tardiness problem with a common due date possesses this property.

5. POLYNOMIALLY SOLVABLE CASES

5.1 *Identical jobs*

If the jobs are identical, we have $p_i = p$ for each job J_i. Since the processing times and due date are assumed to be integral, this situation is more general than the one in which all $p_i = 1$. Suppose the jobs have been renumbered according to nonincreasing weights.

If $d \geqslant p\lceil n/2 \rceil$, then it is easy to show that Emmons' matching approach (Emmons, 1987) generates an optimal schedule S by partitioning the jobs into sets $A(S) = \{J_{2i} \mid i = 1, \ldots, \lfloor n/2 \rfloor\}$ and $B(S) = \{J_{2i-1} \mid i = 1, \ldots, \lceil n/2 \rceil\}$, where the first job in $B(S)$ starts at time $t = d - \Sigma_{J_i \in B(S)} p_i = d - p\lceil n/2 \rceil$. In this notation, $\lfloor n/2 \rfloor$ denotes the largest integer smaller than or equal to $n/2$, and $\lceil n/2 \rceil$ denotes the smallest integer greater than or equal to $n/2$.

Conversely, if $d < p\lceil n/2 \rceil$, then there are two options: either the first job starts at time 0 or the last job in $B(S)$ is completed at time d. It is easy to see that in both cases Emmons' matching approach generates optimal schedules, and the problem is solved by choosing the better one.

5.2 *The jobs have equal weight to processing time ratios*

THEOREM 5. *In the event that $p_i = w_i$ for each job J_i, there is an optimal schedule for any value of d in which the jobs are scheduled according to nonincreasing processing times.*

PROOF. Consider two adjacent jobs that are not scheduled according to the indicated order. If both jobs are completed before or started after the common due date, then these jobs can be interchanged without affecting the cost of the schedule S, unless the due date lies in the interval between the start time of the first and the completion time of the other job. We prove that even in that case, an interchange of these two jobs does not increase the scheduling cost. Without loss of generality, let J_1 and J_2 be the two jobs that have to be interchanged, with

$p_1 \geqslant p_2$, and let J_2 start at time t. We have to investigate the following three situations.

(1) $t \leqslant d \leqslant t + p_2$. Then the interchange leads to a schedule with the same cost.

(2) $t + p_2 < d \leqslant t + p_1$. Then the interchange lowers the cost by $2p_2(d - p_2) \geqslant 0$.

(3) $t + p_1 < d \leqslant t + p_1 + p_2$. Then the interchange decreases the cost by $2dp_2 - 2dp_1 + 2p_1^2 - 2p_2^2 = 2(p_1 + p_2 - d)(p_1 - p_2) \geqslant 0$. □

Assume that the jobs have been renumbered in order of nonincreasing processing times. Suppose r is the smallest index for which $\Sigma_{i=1}^{r} p_i \geqslant \Sigma_{i=r+1}^{n} p_i$. Theorem 5 then implies that, if $d \geqslant \Sigma_{i=1}^{r} p_i$, the problem is solved by putting $B(S) = \{J_i \mid i = 1, \ldots, r\}$ and $A(S) = \{J_i \mid i = r+1, \ldots, n\}$. If $d < \Sigma_{i=1}^{r} p_i$, the first job needs to start at time 0, and the jobs are processed in order of nondecreasing processing times.

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

REFERENCES

U. BAGCHI, R.S. SULLIVAN, Y.L. CHANG (1986). Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics Quarterly 33*, 227-240.

U. BAGCHI, Y.L. CHANG, R.S. SULLIVAN, (1987). Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics 34*, 739-751.

K. BAKER, G. SCUDDER. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, to appear.

H. EMMONS (1987). Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics 34*, 803-810.

M.R. GAREY, R.E. TARJAN, G.T. WILFONG (1988). One-processor scheduling with earliness and tardiness penalties. *Mathematics of Operations Research 13*, 330-348.

N.G. HALL, W. KUBIAK, AND S.P. SETHI (1991). Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research 39*, 847-856.

N.G. HALL, M.E. POSNER (1991). Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date. *Operations Research 39*, 836-846.

J.J. KANET (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly 28*, 643-651.

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

W. SZWARC (1989). Single machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics 36*, 663-673.

New lower and upper bounds for scheduling around a small common due date

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

This paper will appear in Operations Research.

New lower and upper bounds for scheduling around a small common due date

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

H. Oosterhout
*Department of Economics, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands*

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

Suppose a set of n jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The problem is to find a schedule that minimizes the sum of the deviations of the job completion times from a given common due date that is smaller than the sum of the processing times. This problem is known to be $\mathcal{NP}$-hard. There exists a pseudo-polynomial algorithm that is able to solve instances up to 1000 jobs. Branch-and-bound algorithms can solve instances up to only 25 jobs. We apply Lagrangian relaxation to find in $O(n\log n)$ time new lower and upper bounds. Based upon this upper bound, we develop a heuristic whose solution value is guaranteed to be no more than 4/3 times the optimal solution value. We identify conditions under which the lower and upper bound concur; these conditions can be expected to be satisfied by many instances with n not too small. For processing times drawn from a uniform distribution, all our computational experiments exhibit that the bounds concur already for $n \geqslant 40$. For the case these bounds do not concur, we present a refinement of the lower bound. This is obtained by solving a subset-sum problem that is of considerably smaller dimension then the common due date problem to optimality by a pseudo-polynomial algorithm. Finally, we indicate to what extent the analysis also applies to the case that all early completions are weighted by a common weight α and all tardy completions by a common weight β.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single-machine scheduling, common due date, Lagrangian relaxation,, approximation algorithm, worst-case behavior.

1. Introduction

The just-in-time concept for manufacturing has induced a new type of machine scheduling problem in which both early and tardy completions of jobs are penalized. We consider the following single-machine scheduling problem that is associated with this concept.

A set of n independent jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The machine is assumed to be continuously available from time zero onwards only. Job J_j requires processing during a given uninterrupted time p_j and should ideally be completed at a given due date d_j. Without loss of generality, we assume that the processing times and the due dates are integral. We assume furthermore that the jobs are indexed in order of nonincreasing processing times. A *schedule* σ defines for each job J_j a completion time C_j, such that the jobs do not overlap in their execution. The earliness and tardiness of J_j are defined as $E_j = \max\{d_j - C_j, 0\}$ and $T_j = \max\{C_j - d_j, 0\}$, respectively. The just-in-time philosophy is reflected in the objective function

$$f(\sigma) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j),$$

where the deviation of C_j from d_j is penalized by either α_j or β_j, depending on whether J_j is early or tardy, for $j = 1, \ldots, n$. For a review of problems with this type of objective function, we refer to the survey by Baker and Scudder (1990).

An important subclass contains problems with a due date d that is common to all jobs. The common due date is either specified as part of the problem instance, or is a decision variable that has to be optimized simultaneously with the job sequence. As the first job may start later than time zero, the optimal schedule is identical for both problems unless the common due date d is restrictively small. The first variant is therefore referred to as the restricted problem and the second variant as the unrestricted problem.

Bagchi, Chang, and Sullivan (1987) propose a branch-and-bound approach for the restricted variant with all earliness penalties equal to α and with all tardiness penalties equal to β. Szwarc (1989) presents a branch-and-bound approach for the case that $\alpha = \beta$. These branch-and-bound algorithms are able to solve instances up to 25 jobs. Sundararaghavan and Ahmed (1984) present an approximation algorithm for the case $\alpha = \beta$ that shows a remarkably good performance from an empirical point of view. Lee and Liman (1991) present an approximation algorithm with *performance guarantee* 3/2; this means that for any instance their approximation algorithm produces a solution with value no more than 3/2 times the optimal solution value. Hall, Kubiak and Sethi (1991) and Hoogeveen and Van de Velde (1991) establish the $\mathcal{NP}$-hardness of the problem, even if $\alpha = \beta$, thereby justifying the enumerative and approximative approaches. Furthermore, Hall et al. (1991) propose a pseudo-polynomial time algorithm running in $O(n \Sigma p_j)$ time and space, and provide computational results for instances up to 1000 jobs. Their experiments, however, show that the algorithm is limited by space, not time.

We present a Lagrangian-based branch-and-bound algorithm for the case $\alpha = \beta$. Using Lagrangian relaxation, we find new lower and upper bounds in $O(n \log n)$ time. We identify conditions under which the lower and upper bound concur; these conditions can be expected to be satisfied by many instances with n

not too small. This is confirmed by our computational results when the processing times are drawn from a uniform distribution.

For the case that these bounds do not concur, we present a refinement of the lower bound, which is obtained by solving a subset-sum problem to optimality by a pseudo-polynomial algorithm. This can be done very fast, since the subset-sum problem in our application is of a considerably smaller dimension than the common due date problem. Computational experiments show that, if any, only a small number of nodes are examined in the branch-and-bound algorithm.

In addition, we develop a heuristic that is based upon the Lagrangian upper bound with performance guarantee 4/3. This means that the heuristic produces a solution with value guaranteed to be no more than 4/3 times the optimal solution value.

This paper is organized as follows. In Section 2, we review Emmons's matching algorithm (Emmons, 1987) for the unrestricted variant of the common due date problem with general α and β. In Section 3, we develop a lower bound based upon Lagrangian relaxation for the restricted variant with $\alpha = \beta$. In Section 4, we use the insight gained in Section 3 to develop a heuristic for the restricted variant. In Section 5, we show that this heuristic has performance guarantee 4/3. In Section 6, we describe the branch-and-bound algorithm, and in Section 7, we present some computational results. Finally, we briefly indicate to what extent the analysis applies to the case $\alpha \neq \beta$.

2. Emmons's matching algorithm for the unrestricted problem

Kanet (1981) presents an $O(n \log n)$ algorithm for the unrestricted variant with $\alpha = \beta$. Bagchi et al. (1987) and Emmons (1987) propose $O(n \log n)$ algorithms for the case $\alpha \neq \beta$. We briefly review the concepts of Emmons's matching algorithm, since they provide the insight needed for the subsequent sections.

Theorem 1 (Kanet, 1981). *No optimal schedule has idle time between the execution of the jobs.* □

Theorem 2 (Kanet, 1981). *There is an optimal schedule for the unrestricted variant in which the due date d coincides with the start time or completion time of the job with the smallest processing time.* □

Emmons's matching algorithm is based upon the concept of positional weights. The scheduling problem reduces then to an assignment problem where jobs have to be assigned to positions. The cost of assigning J_j to the kth early position is equal to $\alpha(k-1)p_j$; the cost of assigning J_j to the kth tardy position is equal to $\beta k p_j$. The assignment problem is solved in $O(n \log n)$ time by matching the job that has the jth largest processing time with the position that has the jth smallest weight, for $j = 1, \ldots, n$.

Emmons's matching algorithm shows that in any optimal schedule the jobs completed before or at d are scheduled in order of nonincreasing processing times and the jobs started at or after d in order of nondecreasing processing times. Due to this structure, optimal schedules are said to be *V-shaped.*

Optimal schedules for the restricted variant have the same structure, albeit that there may be one job that is scheduled around d. For this particular job, it holds that the early or tardy jobs have larger processing times.

3. A NEW LOWER BOUND FOR THE RESTRICTED VARIANT

We look upon this $\mathcal{NP}$-hard problem as an 'easy' problem complicated by the 'nasty' constraint that the machine is only available from time zero onwards. If this constraint were not present, then the problem could easily be solved through Emmons's algorithm. This is exactly the approach Szwarc (1989) follows to determine a lower bound. The structure of the problem, however, suggests that the technique of Lagrangian relaxation might be more successful. We remove the nasty constraint, and put it into the objective function, weighted by a nonnegative Lagrangian multiplier. The resulting problem is easy to solve. It will be referred to as the Lagrangian problem; its solution provides a lower bound for the original problem.

The nasty constraint can be formulated as

$$W \leqslant d,$$

where W denotes the total amount of work that is processed up to time d. If we introduce a Lagrangian multiplier $\lambda \geqslant 0$ and bring this constraint weighted by λ into the objective function, then we get the following Lagrangian problem, referred to as problem (L_λ): find the value $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n}(E_j + T_j) + \lambda(W - d), \tag{L_λ}$$

for a given $\lambda \geqslant 0$. Obviously, $L(\lambda)$ is a lower bound for the original problem. There are two questions that immediately arise: Given a value of λ, can $L(\lambda)$ be determined in polynomial time? If so, can the value λ^* that maximizes the lower bound $L(\lambda)$ be found in polynomial time? The latter problem is referred to as the *Lagrangian dual problem.* The following two theorems provide affirmative answers to both questions.

THEOREM 3. *For a given λ, the Lagrangian problem is solved by applying Emmons's matching algorithm with the weights of the early positions increased by λ.*

PROOF. Straightforward arguments show that there exists an optimal schedule for the Lagrangian problem in which some job is completed exactly on time d. Hence, there is an optimal schedule with $W = \Sigma_{J_j \in \mathcal{E}} p_j$, where $\mathcal{E}$ denotes the set of jobs that are scheduled in the early and just-in-time positions. The Lagrangian objective function can then alternatively be written as

$$\{\sum_{j=1}^{n}(E_j + T_j) + \sum_{J_j \in \mathcal{E}} \lambda p_j\} - \lambda d.$$

Since the last term is a constant for a given λ, we need to minimize only the expression inside the braces. This is achieved by applying Emmons's matching algorithm to the case where the weight of the kth early position is equal to $k-1+\lambda$. □

THEOREM 4. *The optimal value λ^*, that is, the value that maximizes the Lagrangian lower bound, is equal to the index λ for which*

$$\sum_{j=0}^{\lfloor(n-\lambda)/2\rfloor} p_{\lambda+2j} > d \geqslant \sum_{j=0}^{\lfloor(n-\lambda-1)/2\rfloor} p_{\lambda+1+2j},$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x. If no such index exists, then $\lambda^* = 0$.

PROOF. Consider an arbitrary value λ. If λ is not integral, then all optimal schedules for (L_λ) have equal W. If λ is integral, then there are multiple optimal schedules with different W; these are found by breaking ties differently in Emmons's algorithm. Define for each integer λ ($\lambda = 0, \ldots, n$) $\sigma_\lambda^{\min}$ as the optimal schedule for the Lagrangian problem (L_λ) with W minimal. In the same fashion, the schedule $\sigma_\lambda^{\max}$ is defined as the optimal schedule for the Lagrangian problem (L_λ) with W maximal, for $\lambda = 0, \ldots, n$. We define $W_\lambda^{\min}$ and $W_\lambda^{\max}$ as the amount of work processed before time d in $\sigma_\lambda^{\min}$ and $\sigma_\lambda^{\max}$, respectively. Straightforward calculations show that $\sigma_\lambda^{\min}$ remains optimal if the Lagrangian multiplier is increased by ϵ, with $0 \leqslant \epsilon \leqslant 1$; hence, we have that $\sigma_\lambda^{\min}$ is identical to $\sigma_{\lambda+1}^{\max}$ and $W_\lambda^{\min} = W_{\lambda+1}^{\max}$. This implies that $L(\lambda)$ is a piecewise-linear and concave function of λ. The breakpoints correspond to the integral values $\lambda = 1, \ldots, n$, and the gradient of the function between the integral breakpoints λ and $\lambda+1$ is equal to $W_\lambda^{\min} - d$, for $\lambda = 0, \ldots, n-1$. The Lagrangian dual problem is therefore solved by putting λ^* equal to the index λ for which $W_\lambda^{\max} > d \geqslant W_\lambda^{\min}$. Due to the indexing of the jobs, the theorem follows. □

Let σ^* be an optimal schedule for the Lagrangian dual problem. If $\lambda^* = 0$, then $\sigma^* = \sigma_0^{\min}$ is feasible for the original problem, and hence optimal. Note that this also implies that $d \geqslant p_1 + p_3 + \cdots + p_n$ if n is odd, and $d \geqslant p_1 + p_3 + \cdots + p_{n-1}$ if n is even. This agrees with the observation by Bagchi et al. (1987) that the schedules $(J_1, J_3, \ldots, J_n, J_{n-1}, \ldots, J_2)$ and $(J_1, J_3, \ldots, J_{n-1}, J_n, \ldots, J_2)$ are optimal under the respective conditions.

In the remainder, we assume that $\lambda^* \geqslant 1$. Depending on whether $n - \lambda^*$ is odd or even, σ^* has the following structure. First, suppose $n - \lambda^*$ is odd. Then the jobs $J_1, \ldots, J_{\lambda^*-1}$ occupy the last $\lambda^* - 1$ positions in σ^*, the pair $\{J_{\lambda^*}, J_{\lambda^*+1}\}$ occupies the first early position and the λ^*th tardy position, the pair $\{J_{\lambda^*+2}, J_{\lambda^*+3}\}$ occupies the second early position and the (λ^*+1)th tardy position, and so on. Finally, the pair $\{J_{n-1}, J_n\}$ occupies the positions around the due date. Second, if $n - \lambda^*$ is even, then σ^* has the same structure, except that J_n is positioned between J_{n-2} and J_{n-1}, and is started somewhere in the interval $[d - p_n, d]$.

PROPOSITION 1. *If there exists a schedule σ^* that is optimal for the Lagrangian dual problem in which the first job is started at time zero, then the Lagrangian lower bound $L(\lambda^*)$ is tight and σ^* is an optimal schedule for the original problem.*

PROOF. In this case we have $L(\lambda^*) = \Sigma(E_j + T_j) + \lambda^*(W - d) = \Sigma(E_j + T_j) = f(\sigma^*)$. □

If no such schedule σ^* exists, then there is a gap between the optimal value for the original problem and the Lagrangian lower bound. We get a better lower bound, however, by solving the *modified* Lagrangian problem, which is to find a schedule that minimizes

$$\sum_{j=1}^{n} |C_j - d| + \lambda^*(W - d) + |W - d|.$$

Clearly, the modified Lagrangian problem yields a lower bound for the original problem for any $\lambda^* \geqslant 1$.

THEOREM 5. *The modified Lagrangian problem is solved by a schedule from among the optimal schedules for the Lagrangian dual problem that has minimal $|W - d|$.*

PROOF. Suppose that π is a schedule that has minimal Lagrangian cost from among the optimal schedules for the modified Lagrangian problem; suppose further that π is not optimal for the Lagrangian dual problem. Then either the jobs are not assigned to the optimal set of positions, or that there are at least two jobs J_i and J_j with $p_i > p_j$ that are not optimally assigned. As to the first case, assigning J_i to a position with smaller weight decreases the Langrangian cost by at least p_i, while $|W - d|$ is increased by at most p_i. As to the second case, the interchange of J_i and J_j decreases the Lagrangian cost by at least $p_i - p_j$, while $|W - d|$ is increased by at most $p_i - p_j$. Therefore, in both cases π is easily transformed into a schedule $\bar{\pi}$ that is also optimal for the modified Lagrangian problem but that has smaller Lagrangian cost than π. This contradicts the assumption that π has minimal Lagrangian cost. Hence, π must be also optimal for the Lagrangian dual problem. □

The problem of minimizing $|W - d|$ is transformed into a considerably smaller instance of subset-sum in the following way. Renumber the jobs such that $J_{k-1+\lambda^*}$ becomes J_k for $k = 1, \ldots, n - \lambda^* + 1$; n becomes equal to $n - \lambda^* + 1$; the jobs previously denoted by $J_1, \ldots, J_{\lambda^*-1}$ are now simply referred to as the 'remaining' jobs. Hence, the jobs $\{J_{2k-1}, J_{2k}\}$ form a pair in the Lagrangian dual for $k = 1, \ldots, l$, with $l = 1, \ldots, \lfloor n/2 \rfloor$. Define a_j as the difference in processing time between the jobs of the jth pair ($j = 1, \ldots, l$), and define $D = d - W_{\lambda^*}^{\min}$. Remove the values a_j that are zero; suppose that m of them remain. Define $\mathcal{A}$ as the multiset containing the m remaining a_j-values; let $a_{[j]}$ denote the jth largest element in $\mathcal{A}$.

If n is even, then the problem of minimizing $|W - d|$ is equivalent to determining a subset $A \subseteq \mathcal{A}$, whose sum is as close to D as possible. If n is odd, then an optimal schedule for the Lagrangian dual problem is optimal for the original problem in case $W \in [d - p_n, d]$. Finding such a schedule is equivalent to determining a subset $A \subseteq \mathcal{Q}$ whose sum falls in the interval $[D - p_n, D]$. If no such subset exists, then the goal is to find a subset A whose sum is as close as possible to either $D - p_n$ or D. This problem, known as the optimization version of SUBSET-SUM, is $\mathcal{NP}$-hard in the ordinary sense (Garey and Johnson, 1979).

The instance of subset-sum can then be solved to optimality by dynamic programming requiring $O(mD)$ time and space. Note that $D \leqslant \Sigma a_j \leqslant p_{\max}$; hence, the subset-sum problem is of a smaller dimension than the underlying common due date problem.

4. A NEW UPPER BOUND FOR THE RESTRICTED VARIANT

Consider an optimal schedule for the Lagrangian dual problem. If $W \leqslant d$, then it is also a feasible schedule for the common due date problem; if $W > d$, then we defer the schedule to make it feasible. The analysis in the previous section suggests that we should look for an optimal schedule for the Lagrangian dual problem with $|W - d|$ minimal. However, only if $W = d$, then we have a guaranteedly optimal schedule for the common due date problem.

We develop an approximation algorithm for the common due date problem based upon Johnson's approximation algorithm (Johnson, 1974) for subset-sum, which runs in $O(m)$ time.

JOHNSON'S ALGORITHM

```
Step 1. $\mathcal{Q} = \varnothing$; $j \leftarrow 1$.
Step 2. If $a_{[j]} \leqslant D$, then $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{a_{[j]}\}$ and $D \leftarrow D - a_{[j]}$.
Step 3. $j \leftarrow j + 1$; if $j \leqslant m$, then go to Step 1.
```

Using an approximation algorithm for subset-sum rather than an optimization algorithm does not affect the worst-case behavior; see Section 5. As to the empirical behavior, our computational results suggest that the loss in accuracy, if any, is small.

Furthermore, we can identify a class of instances for which Johnson's algorithm always finds a solution value equal to the *target sum D*. This class comprises the instances possessing the so-called *divisibility property*; this class is important in our application, as many instances can be expected to belong to it.

DEFINITION. *A multiset of values $\{a_1, \ldots, a_m\}$, with $1 = a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$ is said to possess the divisibility property if for every j ($j = 1, \ldots, m$) and for every value $D \in \{1, 2, \ldots, \Sigma_{i=1}^{j} a_i\}$ there exists a subset $A \subseteq \{a_1, \ldots, a_j\}$, whose sum is equal to D.*

THEOREM 6. *A multiset of values $\{a_1, \ldots, a_m\}$, with $1 = a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$ possesses the divisibility property if and only if $a_{j+1} \leqslant \Sigma_{i=1}^{j} a_i + 1$, for $j = 1, \ldots, n-1$.* □

THEOREM 7. *If an instance of subset-sum satisfies the divisibility property, then Johnson's algorithm finds a subset with sum equal to D.* □

In our application, each a_j is equal to the difference in processing times between two successive jobs in the shortest processing time order. If the number of jobs with different processing times is not too small, then the values a_j tend to be small. This intuitive reasoning suggests that many instances possess the divisibility property.

Johnson's algorithm always yields a subset with sum no more than D. This handicap is overcome by applying the algorithm also to the target sum $\overline{D} = \Sigma_{j=1}^{m} a_j - D$ and taking the complement of the resulting subset with respect to $\mathcal{A}$. We use the subscripts 1 and 2 to distinguish the approximation from below and from above: A_1 and D_1 denote the resulting subset and the gap for the approximation from below, and A_2 and D_2 denote the resulting subset and the gap for the approximation from above.

If both $D_1 > 0$ and $D_2 > 0$, then we apply the next algorithm to derive feasible schedules for the common due date problem from the subsets A_1 and A_2.

ALGORITHM TRANSFORM

Step 1. Consider A_1. Starting with $\sigma_{\lambda^*}^{\min}$, interchange the jobs that correspond to $a_j \in A_1$ for $j = 1, \ldots, m$, thereby increasing W by a_j per interchange. Determine the schedule corresponding to A_2 in a similar fashion, starting from $\sigma_{\lambda^*}^{\max}$. Let the resulting schedules be σ_1 and σ_2.

Step 2. The schedule σ_1 is started at time D_1. Shift the schedule to the left until the first job is started at time 0 or the number of jobs completed before or at d exceeds the number of jobs completed after d by two. Rearrange the jobs to make the schedule V-shaped again. The resulting schedule is denoted by $\overline{\sigma}_1$.

Step 3. The schedule σ_2 is started at time $-D_2$. Defer the schedule such that the first job is started at time zero, and rearrange the jobs to make the schedule V-shaped again; this schedule is denoted as $\overline{\sigma}_2^0$. If some J_k is scheduled around d, then defer $\overline{\sigma}_2^0$ until J_k is started exactly at d. Rearrange the jobs to make the schedule V-shaped; let the resulting schedule be $\overline{\sigma}_2$.

We now present our approximation algorithm for the common due date problem; in the remainder, we refer to it as the Even-Odd Heuristic.

EVEN-ODD HEURISTIC

Step 0. Given an instance of the common due date problem, solve the Lagrangian dual problem, and apply Johnson's algorithm to the corresponding instance of subset-sum.

Step 1. If $D_1 \leqslant D_2$, then apply Algorithm Transform; go to Step 5.

Step 2. Let $Q = \{a_j \mid a_j \geqslant D_1\}$. If $Q \neq \{a_1\}$, then apply Algorithm Transform, and go to Step 5.

Step 3. If $p_1 > d$, then apply Algorithm Transform to determine $\bar{\sigma}_2^0$. Furthermore, solve the Lagrangian dual problem under the condition that J_1 and all the 'remaining' jobs occupy the last positions; go to Step 5.

Step 4. Solve the Lagrangian dual problem under the condition that J_1 and the 'remaining' jobs are assigned to positions after d, and solve the Lagrangian dual problem with J_1 assigned to a position before d. Apply Johnson's algorithm and Algorithm Transform to all these solutions.

Step 5. Choose a schedule with minimal cost.

5. Worst-case behavior

For any instance I of the common due date problem, let $EOH(I)$ denote the solution value determined by the Even-Odd Heuristic, and let $OPT(I)$ denote the optimal solution value. We define ρ as

$$\rho = \inf_I \{EOH(I)/OPT(I)\}.$$

In this section, we prove that $\rho \leqslant 4/3$, that is, that the Even-Odd Heuristic has performance guarantee 4/3.

Suppose first that Johnson's algorithm does not solve the corresponding instance of subset-sum to optimality, that is, D_1 or D_2 is not minimal. This means that we do not know the minimal value of $W-d$, and therefore cannot use the strengthened lower bound in our analysis.

Lemma 1. *If Johnson's algorithm does not solve the resulting instance of subset-sum to optimality, then* $\rho \leqslant 8/7$.

Proof. A straightforward analysis shows that, if Johnson's algorithm leaves a gap G that is not minimal, then at least 3 a_j-values greater than G have to be involved; this means that are at least six jobs with processing times at least equal to $3G, 2G, 2G, G, G$, and 0, respectively. Furthermore, due to the structure of the solution of the Lagrangian problem, the $\lambda^* - 1$ 'remaining' jobs must have processing times at least $3G$.

First, assume $D_1 \leqslant D_2$. Then we have for any instance I that

$$EOH(I) \leqslant f(\sigma_1) = L(\lambda^*) + \lambda^* D_1 \leqslant OPT(I) + \lambda^* D_1.$$

Inspecting $\sigma_{\lambda^*}^{\max}$, we see that $L(\lambda^*) \geqslant D_1(5 + 3\lambda^*(\lambda^* + 1)/2)$. Hence, $\rho \leqslant 1 + (2\lambda^*/(10 + 3\lambda^*(\lambda^* + 1))) \leqslant 8/7$ for any $\lambda^* \geqslant 1$.

Second, assume $D_1 > D_2$. If D_1 is not minimal, then we use the above analysis and find $\rho \leqslant 8/7$. If D_1 is minimal, then D_2 is not. Consider an element $a_j \notin A_2$ and suppose that $a_j < D_1 + D_2$. This implies that

$$\bar{D} < \sum_{k \in A_2} a_k + a_j < \bar{D} + D_1;$$

as a consequence, the sizes of the elements in $\mathcal{Q} - A_2 - \{a_j\}$ add up to a value between $D - D_1$ and D, contradicting the minimality of D_1. Hence, $a_j \geqslant D_1 + D_2$, and the above analysis can be applied to establish $\rho \leqslant 8/7$. □

So, if Johnson's algorithm does not give minimal values D_1 and D_2, then we surely have $\rho \leqslant 4/3$. From now on, we assume that D_1 and D_2 are minimal; hence, we can now use the strengthened lower bound.

LEMMA 2. *If $D_1 \leqslant D_2$, then $\rho \leqslant 4/3$.*

PROOF. Again, we have that $EOH(I) \leqslant L(\lambda^*) + \lambda^* D_1$. Furthermore, from Theorem 5 it follows that $OPT(I) \geqslant L(\lambda^*) + D_1$. Every element $a_j \notin A_1$ must have size $a_j \geqslant D_1 + D_2 \geqslant 2D_1$. Inspecting $\sigma_{\lambda^*}^{\max}$, we see that $L(\lambda^*) \geqslant \lambda^*(\lambda^* - 1)D_1$; this gives $\rho \leqslant 1 + ((\lambda^* - 1)/(1 + \lambda^*(\lambda^* - 1))) \leqslant 4/3$ for any $\lambda \geqslant 1$. □

Now suppose that $D_1 > D_2$. It is easy to show $\rho = 4/3$ if there exists an element $a_k \geqslant D_1$ with $k \geqslant 3$. If no such element exists, then we consider the costs of all schedules determined by Algorithm Transform. To that end, we need an upper bound on $\Delta = f(\bar{\sigma}_2) - f(\sigma_2)$.

PROPOSITION 3. *Suppose that the first job in σ_2 has processing time no more than d. Then Δ is no more than the sum of the positional costs in $\bar{\sigma}_2$ of the last k jobs before d and the first $k+1$ jobs after d, where k is the number of jobs that have been transferred from a position before d to a position after d.*

PROOF. Without loss of generality, we assume that n is even; if not, then we add a dummy job with zero processing time. For matter of convenience, renumber the jobs temporarily such that $J_1, \ldots, J_k$ are the jobs that are transferred from positions before d to positions after d (J_k is completed at time d), and $J_{2k}, \ldots, J_{k+1}, J_0$ are the first $k+1$ jobs after d (J_{2k} is started at time d). Note that the jobs J_i and J_{k+i} ($i = 1, \ldots, k$) form a pair in the Lagrangian dual; hence, we must have that $\min\{p_i, p_{k+i}\} \geqslant \max\{p_{i+1}, p_{k+i+1}\}$, for $i = 1, \ldots, k-1$.

Suppose that J_0 occupies position $\lambda^* + \mu$ in $\bar{\sigma}_2$, with $\mu \geqslant 0$. Twice the positional cost of the jobs $J_0, \ldots, J_{2k}$ in σ_2 is then equal to

$$
\begin{aligned}
&2((\mu+1)p_1 + \cdots + (\mu+k)p_k + (\lambda^* + \mu)p_0 + \cdots + (\lambda^* + \mu + k)p_{2k}) \geqslant \\
&(\lambda^* + \mu)p_0 + ((\lambda^* + \mu)p_0 + (\lambda^* + \mu + 1)p_{k+1} + 2p_1) + \cdots \\
&+((\lambda^* + \mu + k - 1)p_{2k-1} + (\lambda^* + \mu + k)p_{2k} + 2kp_k) \geqslant \\
&(\lambda^* + \mu)p_0 + (\lambda^* + \mu + 1)(p_1 + p_{k+1}) + \min\{p_1, p_{k+1}\} + (\lambda^* + \mu + 3)(p_2 + p \\
&+ \min\{p_2, p_{k+2}\} + \cdots + (\lambda^* + \mu + 2k - 1)(p_k + p_{2k}) + \min\{p_k, p_{2k}\}.
\end{aligned}
$$

The last expression is exactly equal to the positional cost due to the jobs $J_0, \ldots, J_{2k}$ in $\bar{\sigma}_2$. □

LEMMA 3. *Suppose that a_1 and a_2 are the only elements larger than D_1. Then $\rho \leqslant 4/3$.*

PROOF. First, suppose that $p_1+p_3 \leqslant d$. Partition the jobs in two subsets: the first one is $\{J_3,\ldots,J_n\}$, the second one consists of J_1, J_2, and the 'remaining' jobs. As $p_1+p_3 \leqslant d$, it follows immediately from Proposition 3 that for σ_2 the sum of the positional costs of the jobs in $\{J_3,\ldots,J_n\}$ is at least equal to Δ. The sum of the positional costs of the jobs in the other subset is at least $(1+\lambda^{*2})D_1 \geqslant 2\lambda^* D_1$. Hence, $OPT(I) \geqslant 2\lambda^* D_1 + \Delta$, implying that $\rho \leqslant 4/3$.

Second, suppose that $p_1+p_3 > d$. As a_1 and a_2 are the only two elements greater than D_1, it follows immediately that $D_1 = d-p_1-p_4-p_5-\cdots$ if $a_1 \geqslant a_2$, and that $D_1 = d-p_2-p_3-p_5\cdots$ otherwise; $D_2 = p_1+p_3+p_6+\cdots-d$. An easy interchange argument, validated by the inequality $D_1 > D_2$, proves that $J_1, J_3, J_n, J_{n-1}, \cdots$ is an optimal schedule for the case that J_1 and J_3 are started before time d. Hence, we are done unless J_1 or J_3 is started at or after time d in any optimal schedule. In this case, however, we impose the additional constraint to the common due date problem that J_1 or J_3 is started at or after time d. Consider the modified Lagrangian problem with such an additional constraint. Along the lines of the proof of Theorem 5, we can show that this problem is solved by an optimal schedule for the Lagrangian dual problem with J_1 or J_3 scheduled after d for which $|W-d|$ is minimal; this is exactly the schedule σ_1. We have therefore that $OPT(I) \geqslant L(\lambda^*)+D_1 \geqslant (\lambda^{*2}+2)D_1$. As $EOH(I) \leqslant L(\lambda^*)+\lambda^* D_1$, we obtain $\rho \leqslant 1+((\lambda^*-1)/(\lambda^{*2}+2)) < 4/3$. □

The analysis of the case that a_2 is the only element greater than D_1 proceeds along the same lines.

LEMMA 4. *Suppose $D_1 > D_2$, a_1 is the only element greater than D_1, and $p_1 > d$. Then $EOH(I) = OPT(I)$.*

PROOF. An easy interchange argument, validated by the inequality $D_1 > D_2$, proves that in any optimal schedule J_1 is either started at time 0 or scheduled immediately before the 'remaining' jobs. The inequality $D_1 > D_2$ also implies that Emmons's matching algorithm determines a feasible and hence optimal schedule for the case that J_1 and the 'remaining' jobs are started at or after d. □

If $p_1 \leqslant d$, then we solve both the Lagrangian dual problem with the additional constraint that J_1 and all 'remaining' jobs are scheduled after d and the Lagrangian dual problem with the additional constraint that J_1 is scheduled before d.

LEMMA 5. *Suppose that $D_1 > D_2$, that a_1 is the only element greater than D_1, and that $p_1 \leqslant d$. Then we have $\rho \leqslant 4/3$.*

PROOF. First, suppose that there is an optimal schedule in which J_1 and the 'remaining' jobs are started at or after d. Suppose that solving the Lagrangian dual problem under the condition that J_1 and all 'remaining' jobs are assigned to positions after d gives $\overline{\lambda}^*$, $\overline{D}_1$ and $\overline{D}_2$. If $\overline{\lambda}^* = 0$, then we have found an optimal schedule. If $\overline{\lambda}^* \geqslant 1$, then the schedule that corresponds to $\overline{D}_2$ must begin with J_2, J_3, and J_4; if not, then W not sum up to $d+\overline{D}_2$. Hence, we have $a_1 \geqslant p_4$. This gives $EOH(I) \leqslant L(\bar{\lambda}^*) + \bar{\lambda}^*\bar{D}_1$, $OPT(I) \geqslant L(\bar{\lambda}^*)$, and $L(\bar{\lambda}^*) \geqslant (3 + ((\bar{\lambda}^* + 1)(\bar{\lambda}^* + 2)/2)D_1$, from which $\rho \leqslant 4/3$ follows.

Second, suppose there is an optimal schedule in which J_1 or some 'remaining' job is not started after d. The optimal solution for the Lagrangian dual problem with the additional constraint that J_1 or some 'remaining' job is not started after d is such that J_1 is started before d and all the 'remaining' jobs after d; this is easily proven by an interchange argument. Suppose that solving this Lagrangian problem gives $\bar{\lambda}^*$, $\bar{D}_1$, and $\bar{D}_2$. Consider the schedule σ that corresponds to $\bar{D}_2$. Since $\bar{\lambda}^* \geqslant \lambda^* + 1$, the first job after J_1 must be some J_k with $k \geqslant 4$; hence, we have $\bar{D}_1 \leqslant p_4$. The case $\bar{D}_1 \leqslant \bar{D}_2$ is easy to handle; assume therefore that $\bar{D}_1 > \bar{D}_2$. Along the lines of Lemma 3, it can then be proven that $\rho \leqslant 4/3$. □

THEOREM 8. *The Even-Odd Heuristic has performance guarantee* 4/3, *and this bound can be approximated arbitrarily close.*

PROOF. The first part follows immediately from the Lemmas 1 to 5. The following example, based upon the case that only $a_2 > D_1$, shows that we can get arbitrarily close to this bound. Let D be an arbitrary positive integer. There are $n = 2D + 6$ jobs $\{J_1, \ldots, J_n\}$ with processing times

$$p_1 = p_2 = p_3 = D^2 + 2D,$$

$$p_4 = p_5 = p_6 = D,$$

$$p_{6+i} = 1 \text{ for } i = 1, \ldots, 2D,$$

and with common due date

$$d = 2D^2 + 5D.$$

The Even-Odd Heuristic gives the schedules $J_1, J_4, J_5, J_7, \ldots, J_n, J_6, J_3, J_2$ with J_1 started at time D^2, and $J_1, J_3, J_5, J_7, \ldots, J_n, J_6, J_4, J_2$ with J_1 started at time zero. Both schedules have cost $4D^2 + 18D$. The optimal schedule $J_1, J_3, J_7, \ldots, J_n, J_6, J_5, J_4, J_2$, has cost $3D^2 + 19D$, however. Hence, we get arbitrarily close to 4/3 by choosing D sufficiently large. □

6. BRANCH-AND-BOUND

First, we solve the Lagrangian dual problem. If $\lambda^* = 0$, then $\sigma^* = \sigma_0^{\min}$ is an optimal solution for the common due date problem, and we are done. Otherwise, we determine upper bounds as described in Section 4; we also apply the heuristic presented by Sundararaghavan and Ahmed. If the lower and the best upper bound do not concur, then we solve the subset-sum problem to optimality by dynamic programming. If the bounds still do not concur, then we apply branch-and-bound.

For the design of the search tree we make use of the V-shapedness of optimal schedules. Assume the jobs have been reindexed in order of nonincreasing processing times. A node at level j ($j = 1, \ldots, n$) of the search tree corresponds to a partial schedule in which the completion times of the jobs $J_1, \ldots, J_j$ are fixed.

Each node at level j has at most $(n-j)$ descendants. In the kth $(k=1,\ldots,n-j)$ descendant, J_k is started before d and the jobs $J_{j+1},\ldots,J_{j+k-1}$ are to be completed after d. Given the partial schedule for $J_1,\ldots,J_j$, a partial schedule for $J_1,\ldots,J_{j+k}$ can easily be computed.

The algorithm that we propose is of the 'depth-first' type. We employ an *active node* search: at each level we choose one node to branch from. We consistently choose the node, whose job has the smallest remaining index. A simple but powerful rule to restrict the growth of the search tree is the following. A node at level j $(j=1,\ldots,n)$ corresponding to some J_k can be discarded if another node at the same level corresponding to some J_l with $p_k=p_l$ has already been considered. This rule obviously avoids duplication of schedules.

In the nodes of the tree, we only compute the lower bound $L(\lambda^*)$; we neither solve the modified Lagrangian dual problem nor compute additional upper bounds.

7. Computational results

The processing times were drawn from the uniform distribution [1, 100]. Computational experiments were performed with $d=\lfloor t\Sigma p_j\rfloor$ for $t=0.1, 0.2, 0.3, 0.4$, respectively, and with the number of jobs ranging from 10 to 1000. For each combination of n and t we generated 100 instances. The algorithm was coded in the computer language C; the experiments were conducted on a Compaq-386 personal computer.

The results are shown in Table 1; its design reflects our three-phase approach. The third column '# $O(n\log n)$' shows the number of times (out of 100) that the Even-Odd Heuristic finds a schedule with cost equal to the Lagrangian lower bound $L(\lambda^*)$; this is the number of times that the common due date problem was provably solved to optimality in $O(n\log n)$ time. The fourth column '# DP' shows how many of the remaining instances were provably solved to optimality by dynamic programming applied to subset-sum. The fifth column '# Even-Odd optimal' shows the number of times that the Even-Odd Heuristic found an optimal schedule. The sixth column '# SA optimal' gives the same information for the approximation algorithm presented by Sundararaghavan and Ahmed. The last column '# LB tight' shows the number of times that the lower bound (strengthened or not) was equal to the optimal solution value.

From these results we may draw the conclusion that the common due date problem for randomly generated problem instances is extremely easy to solve from a practical point of view. If $n\geqslant 40$, then the $O(n\log n)$ algorithm solves all randomly generated instances to optimality; for $n\geqslant 30$, dynamic programming applied to subset-sum suffices to solve the remaining instances; for $n\leqslant 20$, branch-and-bound is occasionally needed, but requires only a very small number of nodes, and always less than 1 second of running time.

n	t	$\#O(n\log n)$	#DP	#Even-Odd optimal	#SA optimal	#LB tight
10	0.1	66	20	72	77	86
10	0.2	69	20	72	58	89
10	0.3	68	23	68	59	93
10	0.4	82	1	85	62	85
20	0.1	81	12	84	51	94
20	0.2	94	5	94	43	99
20	0.3	99	0	100	42	99
20	0.4	99	1	99	35	100
30	0.1	100	0	100	50	100
30	0.2	98	2	98	51	100
30	0.3	100	0	100	57	100
30	0.4	100	0	100	68	100
40	0.1	100	0	100	63	100
40	0.2	100	0	100	64	100
40	0.3	100	0	100	63	100
40	0.4	100	0	100	54	100
50	0.1	100	0	100	72	100
50	0.2	100	0	100	63	100
50	0.3	100	0	100	69	100
50	0.4	100	0	100	75	100
100	0.1	100	0	100	81	100
100	0.2	100	0	100	86	100
100	0.3	100	0	100	78	100
100	0.4	100	0	100	78	100

TABLE 1. Computational results.

8. EXTENSIONS

The lower bound approach can be extended to the restricted variant of each problem that is solvable by Emmons's matching algorithm. The most important problem in this context is the $1 \mid d_j = d \mid \Sigma(\alpha E_j + \beta T_j)$ problem.

Without loss of generality, we assume that α and β are integral and relatively prime. A similar analysis shows that the optimal value λ^* is the value $\lambda^* \in \{1, \ldots, n\beta\}$ for which $W_{\lambda^*}^{\max} \geqslant d > W_{\lambda^*}^{\min}$. Furthermore, Theorem 5 still holds.

It is straightforward to develop a heuristic for the common due date problem with $\alpha \neq \beta$ by applying Johnson's algorithm and Algorithm Transform; its worst-case performance, however, is still an open question.

ACKNOWLEDGEMENT

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

REFERENCES

U. BAGCHI, Y.L. CHANG, AND R.S. SULLIVAN (1987). Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics 34*, 739-751.

K. BAKER AND G. SCUDDER (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research 38*, 22-57.

H. EMMONS (1987). Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics 34*, 803-810.

N.G. HALL, W. KUBIAK, AND S.P. SETHI (1991). Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date.*Operations Research 39*, 847-856.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1990). Scheduling around a small common due date. *European Journal of Operational Research 55*, 237-242.

D.S. JOHNSON (1974). Approximation algorithms for Combinatorial Problems. *Journal of Computer and System Sciences 9*, 256-278.

J.J. KANET (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly 28*, 643-651.

C-Y. LEE AND S.D. LIMAN (1991). Error bound for the heuristic on the common due date scheduling problem. To appear in *ORSA Journal on Computing*.

P.S. SUNDARARAGHAVAN AND M.U. AHMED (1984). Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. *Naval Research Logistics Quarterly 31*, 325-333.

W. SZWARC (1989). Single machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics 36*, 663-673.

Minimizing total inventory cost a single machine in just-in-time manufacturing

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

This paper has been submitted for publication.

Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing

J.A. Hoogeveen
Department of Mathematics and Computing Science, 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

The just-in-time concept decrees not to accept ordered goods before their due dates in order to avoid inventory cost. This bounces the inventory cost back to the manufacturer: products that are completed before their due dates have to be stored. Reducing this type of storage cost by preclusion of early completion conflicts with the traditional policy of keeping work-in-process inventories down. This paper addresses a single-machine scheduling problem with the objective of minimizing total inventory cost, comprising cost associated with work-in-process inventories and storage cost as a result of early completion. The cost components are measured by the sum of the job completion times and the sum of the job earlinesses. This problem differs from more traditional scheduling problems, since the insertion of machine idle time may reduce total cost. The search for an optimal schedule, however, can be limited to the set of job sequences, since for any sequence there is a clear-cut way to insert machine idle time in order to minimize total inventory cost. We apply branch-and-bound to identify an optimal schedule. We present five approaches for lower bound calculation, based upon relaxation of the objective function, of the state space, and upon Lagrangian relaxation.

1980 Mathematics Subject Classification (1985): 90B35.
Key Words and Phrases: just-in-time manufacturing, inventory cost, work-in-process inventory, earliness, tardiness, machine idle time, branch-and-bound algorithm, Lagrangian relaxation.

1. Introduction

The just-in-time concept has affected the attitude towards inventories significantly. In order to keep inventories down, there is a reluctance to accept ordered goods prior to their due dates. This implies that manufacturers have to store early completed goods before they can be shipped to their destinations. This has added a relatively new aspect to machine scheduling theory: the preclusion of earliness. In principle, earliness can be avoided by allowing machine idle time, thereby deferring jobs. Machine idleness, however, runs counter to the natural instinct to minimize work-in-process inventories, to maximize machine utilization, and to observe due dates.

Within this context, we address the following situation. A set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n independent jobs has to be scheduled on a single machine, which is continuously available from time zero onwards. The machine can handle at most one job at a time. Job J_j ($j = 1, \ldots, n$) requires a positive integral uninterrupted processing time p_j and should ideally be completed exactly on its due date d_j. A *schedule* specifies for each job J_j a completion time C_j such that the jobs do not overlap in their execution. The order in which the machine processes the jobs is called the *job sequence*. For a given schedule, the earliness of J_j is defined as $E_j = \max\{0, d_j - C_j\}$ and its tardiness as $T_j = \max\{0, C_j - d_j\}$. In addition, we define *maximum earliness* as $E_{\max} = \max_{1 \leqslant j \leqslant n} E_j$ and *maximum tardiness* as $T_{\max} = \max_{1 \leqslant j \leqslant n} T_j$. Accordingly, J_j is called *early*, *just-in-time*, or *tardy* if $C_j < d_j$, $C_j = d_j$, or $C_j > d_j$, respectively.

In this paper, we follow the terminology of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) to classify scheduling problems. Deterministic scheduling problems are classified according to a three-field notation $\alpha \mid \beta \mid \gamma$, where α specifies the machine environment, β the job characteristics, and γ the objective function. For instance, $\alpha = 1$ refers to a single machine, $\beta = pmtn$ signifies that the jobs may be preempted, that is, the processing of a job may be interrupted and resumed later, and $\gamma = \Sigma C_j$ means that the objective is to minimize the sum of the job completion times. Since earliness is nonincreasing in the job completion times, it may generally be advantageous to permit machine idle time. The inclusion of the acronym *nmit* in the second field signifies that no machine idle time is allowed.

Three types of single-machine scheduling problems involving job earliness have been considered in the literature. The best-known is the minimization of $E_{\max}$. If machine idle time is not allowed, then the problem is solved by scheduling the jobs in nondecreasing order of $d_j - p_j$; this is known as the *minimum slack time order*. If machine idle time is permitted, then the problem is trivial: for any given sequence, we defer the jobs until all are just-in-time or tardy. This approach also applies to $1 \mid \mid \Sigma E_j$, but, surprisingly, $1 \mid nmit \mid \Sigma E_j$ is $\mathcal{NP}$-hard in the ordinary sense (Du and Leung, 1990). The third problem is to maximize $\Sigma w_j E_j$, where w_j is the weight of job J_j, denoted as $1 \mid \mid -\Sigma w_j E_j$; this problem is solvable in pseudo-polynomial time by an algorithm due to Lawler and Moore (1969).

The combination of earliness with another performance measure, reflecting other considerations, takes us into the arena of bicriteria scheduling. The state of the art, as far as a measure of earliness is concerned, is as follows. For the $1 \mid pmtn, nmit \mid \alpha \Sigma C_j + \beta E_{\max}$ problem, Hoogeveen and Van de Velde (1990) present an algorithm that runs in $O(n^4)$ time. They show that the same algorithm also solves $1 \mid \mid \alpha \Sigma C_j + \beta E_{\max}$ in case $\alpha \geqslant \beta$. Hoogeveen (1990) presents algorithms that solve $1 \mid \mid \alpha E_{\max} + \beta T_{\max}$ and $1 \mid nmit \mid F(E_{\max}, T_{\max})$ in $O(n^2 \log n)$ and $O(n^2)$ time; F is here an arbitrary nondecreasing function of $E_{\max}$ and $T_{\max}$. For the $1 \mid nmit \mid \Sigma(\alpha_j E_j + \beta T_j)$ problem, Ow and Morton (1989) propose a local search method to generate approximate solutions. A voluminous part of research is concerned with common due date scheduling. Here, we have $d_j = d$

($j = 1, \ldots, n$); the objective is to minimize some function of earliness and tardiness. A survey of problems, algorithms, and computational complexity is provided by Baker and Scudder (1990).

In this paper, we consider the problem of minimizing total inventory cost, which is supposed to comprise two components: cost due to work-in-process inventory and storage cost as a result of early completions. These components are assumed to depend linearly on the sum of job completion times and the sum of job earliness. If we let α and β denote the cost per unit time for work-in-process inventory and storage of finished product, respectively, then the total inventory cost for a given schedule σ is

$$f(\sigma) = \alpha \sum_{j=1}^{n} C_j + \beta \sum_{j=1}^{n} E_j.$$

Without loss of generality, we assume α and β to be integral, positive, and relatively prime. Since we have by definition that $E_j = T_j - C_j + d_j$ for $j = 1, \ldots, n$, the objective function can alternatively be written as

$$(\alpha-\beta) \sum_{j=1}^{n} C_j + \beta \sum_{j=1}^{n} (T_j + d_j).$$

If $\alpha \geqslant \beta$, then this a regular objective function, and hence there is an optimal schedule without machine idle time. The case $\alpha = \beta$ reduces to $1 \,|\,|\, \Sigma T_j$, which is $\mathcal{NP}$-hard in the ordinary sense (Du and Leung, 1990). Garey, Tarjan, and Wilfong (1988) prove that the case $\alpha < \beta$ is $\mathcal{NP}$-hard, too. We note that the case $\beta > n\alpha$ reduces to $1 \,|\, r_j \,|\, \Sigma C_j$, which is also $\mathcal{NP}$-hard in the strong sense (Lenstra, Rinnooy Kan, and Brucker, 1977).

We address the case $\beta \geqslant \alpha$, in which the insertion of machine idle time may be advantageous. Our purpose is to find a feasible schedule σ that minimizes $f(\sigma)$. This problem was introduced by Fry and Keong Leong (1987A), who formulate it as an integer linear program. They used a standard code to find an optimal schedule. Not surprisingly, the proposed method solves problems up to 12 jobs only.

The search for an optimal schedule, however, can be reduced to a search over the $n!$ different job sequences, as there is a clear-cut method to insert machine idle time to minimize total cost for a *given* sequence. This method, which requires $O(n^2)$ time, is described in Section 2.

The freedom to leave the machine idle singles out our problem from most concurrent research on scheduling problems with earliness penalties. To our knowledge, this is the first paper that presents a branch-and-bound algorithm for a single-machine scheduling problem with a nonregular objective function, where insertion of machine idle time is allowed. Machine idle time affects the design of a branch-and-bound algorithm significantly. In Section 3, we discuss some components of the algorithm such as the upper bound, the branching rule, the search strategy, and the dominance rules. Lower bounds are presented in Section 4. The range of the due dates in proportion to the processing times mainly dictates when the first job is started and how much machine idle time is inserted between the execution of the jobs. To cope with the variety of due date patterns, we propose five approaches for lower bound computation. Each of these methods seems to be suitable for a certain class of instances. Some computational results are reported in Section 5; conclusions are presented in Section 6.

2. THE INSERTION OF IDLE TIME FOR A GIVEN SEQUENCE

The search for an optimal schedule can be reduced to a search over the $n!$ different job sequences, as there is a clear-cut procedure to insert machine idle time so as to minimize total cost for a *given* sequence.

This procedure, however, is not new. Similar methods have been presented (cf. Baker and Scudder, 1990), including the ones proposed by Fry and Keong Leong (1987B) for the $1 \mid \mid \Sigma(\alpha C_j + \beta E_j + \gamma T_j)$ problem and by Garey, Tarjan, and Wilfong (1988) for the $1 \mid \mid \Sigma(E_j + T_j)$ problem. This is not surprising: as we have already noted, $T_j = C_j + E_j - d_j$ for all j; for specific choices for α and β, our problem is equivalent with theirs.

Suppose that the scheduling order is $\sigma = (J_n, \ldots, J_1)$. Accordingly, $\bar{C}_j = \Sigma_{k=j}^{n} p_k$ is the earliest possible completion time of J_j in this sequence. We introduce a vector $x = (x_1, \ldots, x_n)$ of variables, with x_j ($j = 1, \ldots, n$) denoting the amount of idle time immediately before the execution of J_j. The actual completion time of J_j is then $C_j = \bar{C}_j + \Sigma_{k=j}^{n} x_k$. The problem of minimizing inventory cost for the given job sequence is then equivalent to determining values x_j ($j = 1, \ldots, n$) that minimize

$$\alpha \sum_{j=1}^{n} (\bar{C}_j + \sum_{k=j}^{n} x_k) + \beta \sum_{j=1}^{n} \max(0, d_j - \bar{C}_j - \sum_{k=j}^{n} x_k)$$

subject to

$$x_j \geqslant 0, \qquad \text{for } j = 1, \ldots, n.$$

By the introduction of auxiliary variables E_j denoting the earliness of J_j ($j = 1, \ldots, n$), we can easily transform this problem into a linear programming problem. We know therefore that the optimum is attained in a vertex of the unspecified LP polytope. In addition, we know that the optimal x_j are integral, since the due dates, the processing times, α, and β are integral. A necessary condition for x to be optimal is that all existing *primitive directional derivatives* at x are non-negative. The primitive directional deratives are equal to the change of the scheduling cost if x_j is increased by one unit, and the change of the scheduling cost if x_j is decreased by one unit, for $j = 1, \ldots, n$. The increase of x_j by one unit only affects J_j and the jobs succeeding J_j up to the first period of machine idle time after J_j. We call these jobs the *immediate successors* of J_j. Let Q_j denote the set containing J_j and its immediate successors, let n_j be the number of early jobs in Q_j, and let g_j be the primitive directional derivative for increasing x_j. We have then that $g_j = \alpha \mid Q_j \mid - \beta n_j$. Recall that each J_j is ideally completed on its due date d_j.

Using the above observations, we develop an inductive procedure for finding an optimal schedule for σ. This procedure finds an optimal schedule for the subsequence $(J_l, \ldots, J_1)$, given an optimal schedule for the subsequence $(J_{l-1}, \ldots, J_1)$, for $l = 2, \ldots, n$. The first step is to find out whether putting $C_l = d_l$ is feasible; if so, then we have an optimal schedule for $(J_l, \ldots, J_1)$. Suppose $C_l = d_l$ is not feasible, because J_l overlaps with some other job. We then tentatively put $C_l = C_{l-1} - p_{l-1}$, and compute the optimal deferral of the jobs in Q_l, disregarding the jobs not in Q_l. The optimal deferral, denoted by δ, is dictated by the first point where g_l becomes non-negative. This deferral is feasible if δ is no larger than the length of the period of idle time immediately after the last job in Q_l; let this length be $\delta_{\max}$. If $\delta \leqslant \delta_{\max}$, then we get an optimal schedule for $(J_l, \ldots, J_1)$ by deferring the jobs in Q_l by δ. If $\delta > \delta_{\max}$, then we defer the jobs in Q_l by $\delta_{\max}$. At this point, we repeat the process for J_l: we update Q_l, and evaluate if additional deferral of the jobs in Q_l is advantageous. We now give a step-wise description of the idle time insertion algorithm.

IDLE TIME INSERTION ALGORITHM

```
Step 0. $C_1 \leftarrow d_1$; $l \leftarrow 2$.
Step 1. If $l = n + 1$, go to Step 9.
Step 2. Put $C_l \leftarrow \min\{d_l, C_{l-1} - p_{l-1}\}$. If $C_l = d_l$, then go to Step 8.
Step 3. Determine $Q_l$ and evaluate $g_l$. If $g_l \geqslant 0$, then go to Step 8.
Step 4. Compute $E_j$ for each job $J_j \in Q_l$.
Step 5. Compute $\delta_{\max}$, i.e., the length of the period of idle time immediately after
the last job in $Q_l$.
Step 6. Let $a \leftarrow \lfloor (|Q_l|)\alpha/\beta \rfloor$, and $k \leftarrow |Q_l| - a$. Determine the $k$th smallest
value of the earlinesses of the jobs in $Q_l$; this value is denoted as $E_{[k]}$. If the jobs
in $Q_l$ are deferred by $\delta = E_{[k]}$, then at most $a$ jobs in $Q_l$ remain early; due to the
choice of $a$, $g_l$ then becomes non-negative.
Step 7. Defer the jobs in $Q_l$ by $\Delta = \min\{\delta, \delta_{\max}\}$. If $\delta > \delta_{\max}$, then go to Step 3.
Step 8. $l \leftarrow l + 1$; go to Step 1.
Step 9. An optimal schedule for the sequence $(J_n, \ldots, J_1)$ has been determined.
```

THEOREM 1. *The idle time insertion algorithm generates an optimal schedule for a given sequence.*

PROOF. The proof proceeds by induction. The algorithm clearly produces the optimal schedule in case of a single job. Suppose that we want to find an optimal schedule for the sequence $(J_l, \ldots, J_1)$, having an optimal schedule for the sequence $(J_{l-1}, \ldots, J_1)$ available. There are two cases to consider. First, suppose $d_l \leqslant C_{l-1} - p_{l-1}$; in this case, we let $C_l = d_l$, and retain the completion times of the other jobs; this specifies an optimal schedule for the sequence $(J_l, \ldots, J_1)$. Suppose now $d_l > C_l - p_l$; for this case, deferring J_{l-1} and thereby its immediate successors, i.e., the jobs contained in the set Q_{l-1}, may be advantageous. We can compute the cost of deferring Q_{l-1} by one unit; the benefit of deferring J_l by one unit is equal to $\beta - \alpha$. If the cost is higher than or equal to the benefit, then we put $C_l = C_{l-1} - p_{l-1}$, and we have an optimal schedule for $(J_l, \ldots, J_1)$; otherwise, we defer the jobs in Q_{l-1} by one unit, and evaluate whether additional deferral is advantageous. The idle time insertion algorithm shortcuts this procedure by computing the break-even point, that is, the point where additional deferral is not advantageous. □

Consider the example for which the data are given in Table 1. Let $\alpha = 1$ and $\beta = 4$. We construct the optimal schedule for the sequence (J_3, J_2, J_1). First, we put $C_1 = d_1 = 15$. Next, we let $C_2 = d_2 = 10$, as $d_2 \leqslant C_1 - p_1$. Note that $d_3 > C_2 - p_2$. Therefore, we tentatively put $C_3 = C_2 - p_2 = 7$, and consider deferring J_3 and J_2. Apparently, we have $Q_3 = \{J_3, J_2\}$, $n_3 = 1$, $g_3 = 2\alpha - \beta < 0$, and $E_{[2]} = 3$. However, $\delta_{\max} = C_1 - p_1 - C_2 = 2$, therefore, we defer J_2 and J_3 by 2 units. At this point, the three jobs are processed consecutively. Now we have $g_3 = 3\alpha - \beta$, and additional deferral is still advantageous. As $E_{[3]} = 1$, we insert one more unit of machine idle time. The optimal schedule for each subproblem is depicted in Figure 1.

J_j	p_j	d_j
J_1	3	15
J_2	3	10
J_3	6	10

TABLE 1. Data for the example.

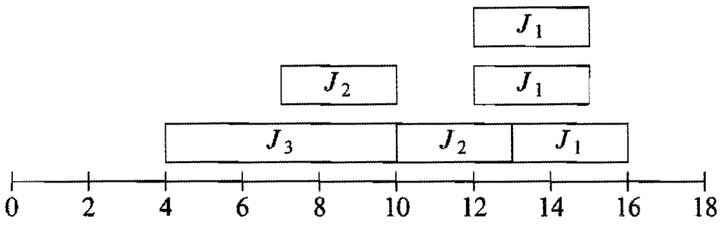


FIGURE 1. Schedules for the example.

The algorithm runs in $O(n^2)$ time. A complete run through the main part of the algorithm, i.e., steps 2 through 8, takes $O(n)$ time: this is needed to identify the set Q_l, to compute the primitive directional derivative g_l, the values $\delta_{\max}$ and δ, and to defer the jobs, if necessary. The value δ is determined in $O(n)$ time through a median-finding technique; see Aho, Hopcroft, and Ullman (1982). After each run through the main part of the algorithm, a gap between two successive jobs is closed. As at most $n-2$ such gaps exist, the algorithm runs in $O(n^2)$ time. For the case $2\alpha = \beta$, i.e., for the problem $1 \| \Sigma(E_j + T_j)$, Garey, Tarjan, and Wilfong (1988) show that the idle time insertion procedure can be implemented to run in $O(n \log n)$ time.

The problem of inserting machine idle time can also be solved by a symmetric procedure starting with the first job in σ. Because of our specific branching rule, however, we choose to start at the end.

In the remainder, we use the terms sequence and schedule interchangeably.

Unless stated otherwise, σ also refers to the optimal schedule for the sequence σ and to the set of jobs in the sequence σ. From now on, we let $p(\sigma) = \Sigma_{J_j \in \sigma} p_j$.

3. The branch-and-bound algorithm

We adopt a *backward sequencing branching rule*: a node at level k of the search tree corresponds to a sequence π with k jobs fixed in the last k positions. We assume from now on that the first job in a partial schedule π is not started before time $p(\mathcal{J}-\pi)$; this additional restriction, imposed to leave space for the remaining jobs, is easily incorporated in the idle time insertion algorithm. Let $f(\pi)$ denote the minimal inventory cost for π. Let $\underline{f}(\pi)$ denote the minimal inventory cost for π if the first job may be started before time $p(\mathcal{J}-\pi)$; the notation $\underline{f}(\pi)$ is only needed in this section. For any partial schedule π, we have $f(\pi) \geqslant \underline{f}(\pi)$.

We employ a *depth-first* strategy to explore the tree: at each level, we generate the descendant nodes for only one node at a time. At level k, there are $n-k$ descendant nodes: one for each unscheduled job. The completion times for the jobs in π are only temporary. Branching from a node that corresponds to π, we add some job J_j leading to the sequence $J_j\pi$. Subsequently, we determine the associated optimal schedule for $J_j\pi$, and possibly defer some jobs in π. We branch from the nodes in order of non-increasing due dates of the associated jobs. Before entering the search tree, we determine an upper bound on the optimal solution value. We use the optimal schedule corresponding to the minimum slack time sequence as an initial solution, and try to reduce its cost by pairwise adjacent interchanges.

A node is discarded if its associated partial schedule π cannot lead to a complete schedule with cost less than UB; UB denotes the currently best solution value. Let $LB(\mathcal{J}-\pi)$ be some lower bound on the minimal cost of scheduling the jobs in the set $\mathcal{J}-\pi$. Obviously, we discard a node if $f(\pi)+LB(\mathcal{J}-\pi) \geqslant UB$. The following rule is usually overlooked. Let $g(\sigma_1,\sigma_2)$ be a lower bound on the cost for scheduling the jobs in σ_1 given the final partial schedule σ_2.

Theorem 2. *The partial schedule π can be discarded if there exists a $J_j \in \mathcal{J}-\pi$ for which $\underline{f}(J_j\pi) + g(\mathcal{J}-\pi-J_j, \pi) \geqslant UB$.*

Proof. Consider a complete sequence σ that has π as final subsequence. Thus, σ can be written as $\sigma = \pi_1 J_j \pi_2 \pi$. Accordingly, we have

$$f(\sigma) = f(\pi_1 J_j \pi_2 \pi) \geqslant \underline{f}(J_j\pi) + g(\pi_1\pi_2, \pi) \geqslant UB. \quad \square$$

It is essential that $g(\mathcal{J}-\pi-J_j,\pi)$ depends only on π and not on $J_j\pi$, and that we use $\underline{f}(J_j\pi)$ instead of $f(J_j\pi)$. We derive two corollaries from Theorem 2.

Corollary 1. *If for a given partial schedule π, we have that $\underline{f}(J_jJ_k\pi) + g(\mathcal{J}-\pi-J_j-J_k,\pi) \geqslant UB$ for some $J_j \in \mathcal{J}-\pi$ and $J_k \in \mathcal{J}-\pi$, then J_k precedes J_j in any complete schedule $\sigma\pi$ with $f(\sigma\pi) < UB$.* $\square$

COROLLARY 2. *The partial schedule π can be discarded if two jobs $J_j \in \mathcal{J} - \pi$ and $J_k \in \mathcal{J} - \pi$ exist with $g(\mathcal{J} - \pi - J_j - J_k, \pi) + \min\{f(J_j J_k \pi), f(J_k J_j \pi)\} \geqslant UB$.* □

If a partial schedule $\pi^* \neq \pi$ exists comprising the same jobs as π and having $f(\sigma\pi^*) \leqslant f(\sigma\pi)$ for any sequence σ for the remaining $n-k$ jobs, then we can also discard π. If $f(\sigma\pi^*) < f(\sigma\pi)$ for some σ, then π is *dominated* by π^*. If $f(\sigma\pi^*) = f(\sigma\pi)$ for every σ, then we discard either π^* or π. The dominance condition above can be narrowed by the requirement that $f(\pi^*) \leqslant f(\pi)$ and that the circumstances to add the remaining $n-k$ jobs to π^* are at least as good as the circumstances to add the remaining jobs to π. The question whether such a sequence π^* exists is of course $\mathcal{NP}$-complete. We strive therefore to identify sufficient conditions to discard π. The temporary nature of the job completion times for π complicates the achievement of this goal. We have to be careful with dominance conditions that are based on interchange arguments: the conditions must remain valid if the jobs in π are deferred.

Suppose that the jobs in π have been reindexed in order of increasing completion times. In each of the following theorems, stating the dominance rules, the sequence π^* is obtained from π by swapping two jobs, say, J_j and J_k. We do not compute the optimal completion times for the sequence π^*. Instead, we determine the job completion times for the sequence π^* as follows. Let C_i and C_i^* be the completion time of J_i in the schedule π and π^*, respectively. Then we let

$$\begin{aligned} C_i^* &= C_i, && \text{for } i = 1, \ldots, j-1, i = k+1, \ldots, |\pi|, \\ C_i^* &= C_i - p_j + p_k, && \text{for } i = j+1, \ldots, k-1, \\ C_k^* &= C_j - p_j + p_k, \\ C_j^* &= C_k. \end{aligned}$$

Let $F(\pi^*)$ be the cost associated with the completion times C_i^*, for $i = 1, \ldots, |\pi|$. Hence, $F(\pi^*) \geqslant f(\pi^*)$. To validate the following dominance rules, we must verify that $f(\pi) \geqslant F(\pi^*)$, even if the jobs are deferred. Due to the relation between π and π^*, this comes down to verifying that for each set of nonnegative values Δ_i $(i = 1, \ldots, n)$

$$\alpha \sum_{i=j}^{k} C_i + \beta \sum_{i=j}^{k} \max\{0, d_i - C_i - \Delta_i\} \geqslant \alpha \sum_{i=j}^{k} C_i^* + \beta \sum_{i=j}^{k} \max\{0, d_i - C_i^* - \Delta_i\}.$$

We start with a straightforward result.

THEOREM 3. *There is an optimal schedule with J_j preceding J_k if $p_j = p_k$ and $d_j \leqslant d_k$.* □

THEOREM 4. *The partial sequence π can be discarded if there are two jobs J_j and J_k with $C_k = C_j + \Sigma_{i=j+1}^{k} p_i$ for which*

$p_j > p_k$, and

$$\alpha \sum_{i=j}^{k} C_i + \beta \sum_{i=j+1}^{k} \max\{0, d_i - C_i\} \alpha \sum_{i=j}^{k} C_i^* + \beta \sum_{i=j+1}^{k} \max\{0, d_i - C_i^*\}. \tag{2}$$

PROOF. As there is no idle time between the jobs in the block that begins with J_j and ends with J_k, the idle time insertion algorithm will defer all jobs in this block by the same amount of time Δ. Define $c(\Delta)$ as the change of cost due to the interchange, after deferring the jobs by $\Delta \geqslant 0$; i.e.,

$$c(\Delta)=\alpha\sum_{i=j}^{k} C_i+\beta\sum_{i=j}^{k}\max\{0,d_i-C_i-\Delta\}-\alpha\sum_{i=j}^{k} C_i^*-\beta\sum_{i=j}^{k}\max\{0,d_i-C_i^*-\Delta\}.$$

We prove that $c(\Delta)\geqslant 0$ for all $\Delta \geqslant 0$. From condition (2), it follows immediately that $c(0)\geqslant 0$. Furthermore, $C_j < C_j^*$ implies $\max\{0,d_j-C_j-\Delta\}\geqslant \max\{0,d_j-C_j^*-\Delta\}$ for all $\Delta\geqslant 0$; $C_i > C_i^*$ for $i=j+1,\ldots,k$ implies $\max\{0,d_i-C_i-\Delta\}-\max\{0,d_i-C_i^*-\Delta\}\geqslant \max\{0,d_i-C_i\}-\max\{0,d_i-C_i^*\}$ for all $\Delta\geqslant 0$. Combining the inequalities, we get the desired result. □

The possible increase of E_j is excluded here. The following theorem shows that in case no idle time exists between two adjacent jobs, then dominance already exists if condition (1) is satisfied for $\Delta = 0$.

THEOREM 5. *The partial sequence π can be discarded if there are two jobs J_j and J_k with $C_k = C_j + p_k$ for which*

$$p_j > p_k,$$

and

$$\alpha(p_j-p_k)+\beta\max\{0,d_j-C_j\}+\beta\max\{0,d_k-C_k\}\geqslant$$
$$\beta\max\{0,d_j-C_k\}+\beta\max\{0,d_k-C_k+p_j\}. \qquad (3)$$

PROOF. Define $c(\Delta)$ as the change of cost due to the interchange, after deferring the jobs by $\Delta\geqslant 0$; i.e.,

$$c(\Delta)=\alpha(p_j-p_k)+\beta\max\{0,d_j-C_j-\Delta\}-\beta\max\{0,d_j-C_k-\Delta\}+$$
$$\beta\max\{0,d_k-C_k-\Delta\}-\beta\max\{0,d_k-C_k+p_j-\Delta\}.$$

We need to show that condition (3), stating that $c(0)>0$, implies $c(\Delta)\geqslant 0$ for all $\Delta\geqslant 0$. Note that $\alpha<\beta$ implies that at least one due date is smaller than C_k; otherwise, condition (3) is not valid.

The expression $c(\Delta)$ has three components. The first component is $\alpha(p_j-p_k)$; it is a constant. The second component is $\beta\max\{0,d_j-C_j-\Delta\}-\beta\max\{0,d_j-C_k-\Delta\}$; it is a piecewise linear function of Δ. The function value is βp_k if $d_j\geqslant C_k+\Delta$, and 0 if $d_j\leqslant C_j+\Delta$. If $C_k+\Delta>d_j\geqslant C_j+\Delta$, then the gradient is -1. The third component is $\beta\max\{0,d_k-C_k-\Delta\}-\beta\max\{0,d_k-C_k+p_j-\Delta\}$; it is also a piecewise linear function of Δ. The function value is $-\beta p_j$ if $d_k\geqslant C_k+\Delta$, and 0 for $d_k\leqslant C_k-p_j+\Delta$. The gradient is 1 if $C_k+\Delta>d_k\geqslant C_k-p_j+\Delta$. Combining the three components yields a piecewise linear function whose behavior depends on the due dates. We now make the following observations. First, $c(\Delta)>0$ if $\Delta\geqslant d_k-C_k+p_j$. Second, if $c(t)>0$ for some $t\geqslant d_k-C_k$, then $c(\Delta)>0$ for all $\Delta\geqslant t$. As at least one due date is smaller than C_k, the second observation implies that, if $d_k \leqslant d_j$, then $c(\Delta) > 0$ for all $\Delta \geqslant 0$.

The only case left to consider is $d_j < d_k$ and $0 \leqslant \Delta \leqslant d_k - C_k$. Then, we have $c(\Delta) = \alpha(p_j - p_k) - \beta p_j + \beta \max\{0, d_j - C_j - \Delta\}$. As $d_j - C_j - \Delta \leqslant d_j - C_j = d_j - C_k + p_k \leqslant p_k$, we get $c(0) \leqslant (\alpha - \beta)(p_j - p_k) \leqslant 0$, which contradicts the assumption. This completes the proof. □

In Corollary 3, explicit conditions for the existence of dominance are derived from Theorem 5. This corollary is referred to when lower bounds are discussed in Section 4.

COROLLARY 3. *The partial sequence* π *can be discarded if there are two jobs* J_j *and* J_k *with* $C_k = C_j + p_k$ *such that*

$$p_j > p_k,$$

and one of the following conditions is satisfied:

$$C_k - p_j \geqslant d_k,$$

$$C_k - p_j < d_k,\ C_k \geqslant d_k, \alpha(p_j - p_k) \geqslant \beta(d_k - C_k + p_j),$$

$$C_k - p_j < d_k,\ C_k < d_k, \alpha(p_j - p_k) \geqslant \beta p_j,$$

$$C_k - p_j < d_k,\ C_k \geqslant d_k, \geqslant \beta(d_k - d_j - p_k + p_j). \square$$

THEOREM 6. *The partial sequence* π *with* J_k *scheduled last is dominated if there is a* J_j *such that*

$$p_j > p_k, \text{ and } C_j - p_j + p_k \geqslant d_k.$$

PROOF. Let $\pi = \pi_1 J_j \pi_2 J_k$ and $\pi^* = \pi_1 J_k \pi_2 J_j$. We compute the effect of the interchange on the scheduling cost. Since J_k is the last job in the optimal schedule π, we have $C_k \geqslant d_k$. In addition, we know $C_j^* = \max\{d_j, C_k - p_k + p_j\}$ and $C_k^* = C_j - p_j + p_k \geqslant d_k$. First, suppose $C_j^* = d_j$. The effect of the interchange is then equal to

$$\alpha(C_j + C_k - (C_j - p_j + p_k) - d_j) + \beta(d_j - C_j) \geqslant$$
$$\alpha(C_k + p_j - p_k - d_j) + \alpha(d_j - C_j) > 0,$$

as $C_k - p_k \geqslant C_j$. Second, suppose that $C_j^* = C_k - p_k + p_j$. The effect of the interchange is then equal to

$$\alpha[C_j + C_k - (C_k - p_k + p_j) - (C_j - p_j + p_k)] + \beta \max\{0, d_j - C_j\} \geqslant 0.$$

The effect remains non-negative if the jobs are deferred. □

THEOREM 7. *There is an optimal schedule in which* J_k *is not scheduled in the last position, if there is some* J_j *with* $p_j > p_k$ *and* $d_j - p_j \geqslant d_k - p_k$.

PROOF. We let $\pi = \pi_1 J_j \pi_2 J_k$ and $\pi^* = \pi_1 J_k \pi_2 J_j$ and compute the effect of the interchange. We have $C_k \geqslant d_k$ and $C_k - p_k \geqslant C_j$; in addition, we define here $C_j^* = \max\{d_j, C_k - p_k + p_j\}$. The effect of the interchange has to be non-negative; we therefore have to prove that

$$\alpha C_k + \beta \max\{0, d_j - C_j\} \geqslant \alpha(p_k - p_j + C_j^*) + \beta \max\{0, d_k - p_k + p_j - C_j\}. \quad (4)$$

First, we examine the case $C_j^* = C_k - p_k + p_j$. Expression (4) is then equivalent to

$$\beta \max\{0, d_j - C_j\} \geqslant \beta \max\{0, d_k - p_k + p_j - C_j\},$$

which is true for any C_j since $d_j - p_j \geqslant d_k - p_k$. Second, consider the case $C_j^* = d_j$. This implies $d_j > C_j$, since $d_j \geqslant C_k - p_k + p_j > C_j - p_k + p_j > C_j$. Hence, expression (4) is equivalent to

$$\alpha C_k + \beta(d_j - C_j) \geqslant \alpha(p_k - p_j + d_j) + \beta \max\{0, d_k - p_k + p_j - C_j\}.$$

Suppose $\max\{0, d_k - p_k + p_j - C_j\} = d_k - p_k + p_j - C_j$. We must then verify that

$$\alpha C_k + \beta d_j \geqslant \alpha(d_j - p_j + p_k) + \beta(d_k - p_k + p_j).$$

As $C_k \geqslant d_k$, we only need to prove that

$$0 \geqslant (\alpha - \beta)[(d_j - p_j) - (d_k - p_k)];$$

this expression is true since $\beta > \alpha$ and $d_j - p_j \geqslant d_k - p_k$. Conversely, suppose $\max\{0, d_k - p_k + p_j - C_j\} = 0$. Since $\alpha C_k + \beta(d_j - C_j) \geqslant \alpha(C_k + d_j - C_j) \geqslant \alpha(p_k + d_j) > \alpha(p_k - p_j + d_j)$, expression (4) is also true for this case. □

COROLLARY 4. *There is an optimal schedule in which J_j is scheduled last if $p_j \geqslant p_k$ and $d_j - p_j \geqslant d_k - p_k$ for each $J_k \in \mathcal{J}$.* □

4. LOWER BOUNDS

In this section, we present five lower bound procedures. It seems to be impossible to develop a lower bound procedure that copes satisfactorily with all conceivable due date patterns. For example, imagine an instance with due dates small with respect to the sum of the processing times; little idle time needs then to be inserted. In contrast, consider an instance with $d_k \gg \Sigma_{j=1}^{n} p_j$ for each J_k; the machine will then be idle for some time before processing the first job. Numerous variations and combinations of both patterns are possible.

Each of the lower bound methods is effective for a specific class of instances. Nonetheless, we use them supplementary rather than complementary. We partition the job set $\mathcal{J}$ into subsets, apply each lower bound method to each subset, and aggregate the best lower bounds. In this way, we hope to obtain a lower bound that is stronger than the separate lower bounds obtained for the entire set $\mathcal{J}$. The success of this strategy depends on the partitioning strategy. The jobs in a subset should be conflicting, that is, they should overlap when completed at their due date. If they are not, then we get the weak lower bound $\alpha \Sigma_{j=1}^{n} d_j$. In this sense, we prefer subsets such that the executions of the jobs in the same subset interfere with each other, but not with the execution of the jobs in the other subsets. We propose two partitioning strategies that pursue this effect.

The first strategy is motivated by the structure of any optimal schedule. The jobs that are consecutively processed between two periods of idle time interfere with each other, but not with the other jobs. Such a partitioning is hard to obtain. To mimic such a partitioning, we identify clusters. A *cluster* is a set of jobs such that for each job J_j in the cluster there is another job J_k in the cluster such that the intervals $[d_j-p_j, d_j]$ and $[d_k-p_k, d_k]$ overlap; hence, for each job in the cluster there exists a conflict with at least one other job in the cluster. However, clusters may interfere with each other in any optimal schedule.

The second strategy is the following. Given a partial schedule π, we try to identify the jobs not in π that will be early in any optimal complete schedule of the form $\sigma\pi$. We call these jobs *surely* early. The idea is to derive an upper bound T on the completion times of the unscheduled jobs; accordingly, $J_j \in \mathcal{J}-\pi$ is surely early if $d_j > T$. For instance, let g be the primitive directional derivative for deferring the first job in π by one unit. Suppose that $|\mathcal{J}-\pi|(\beta-\alpha) \leqslant g$. The current set of completion times for the jobs in π is then optimal for any schedule $\sigma\pi$; an upper bound T is then the start time of the first job in π. Other upper bounds are derived from the dominance rules. Suppose J_j and J_k are adjacent in π with $p_j > p_k$ and J_j preceding J_k. (It is not necessary that $C_k = C_j + p_k$.) The first condition of Corollary 3 indicates that π is dominated if $C_k \geqslant d_k + p_j$; hence, an upper bound is given by $d_k + p_j - 1 - \Sigma_{J_i \in \pi, C_i \leqslant C_k} p_i$. From the other criteria in Corollary 3 and from Theorem 7, similar upper bounds are derived. They can also be derived from Theorem 4, but this requires an intricate procedure. Finally, we set T equal to the minimum of all upper bounds. If no upper bound is specified, then we let $T = \infty$.

4.1. First method: relax the objective function

Let $\mathfrak{E}$ denote the set of surely early jobs; let $\mathfrak{R}$ be the set of remaining jobs. Observe that

$$\min_{\sigma \in \Omega} f(\sigma) \geqslant \min_{\sigma \in \Omega_{\mathfrak{R}}} \sum_{J_j \in \mathfrak{R}} \alpha C_j + \min_{\sigma \in \Omega_{\mathfrak{E}}} \sum_{J_j \in \mathfrak{E}} [\alpha C_j + \beta E_j],$$

where $\Omega_{\mathfrak{R}}$ and $\Omega_{\mathfrak{E}}$ denote the set of feasible schedules for the jobs in $\mathfrak{R}$ and $\mathfrak{E}$. The problem of minimizing $\Sigma_{J_j \in \mathfrak{E}} [\alpha C_j + \beta E_j]$ is solvable in polynomial time; we have $E_j = d_j - C_j$ for each $J_j \in \mathfrak{E}$, and hence, the scheduling cost reduces to $\Sigma_{J_j \in \mathfrak{E}} [(\alpha-\beta)C_j + \beta d_j]$. Applying an analogon of Smith's rule (Smith, 1956), we minimize this cost component by scheduling the jobs in $\mathfrak{E}$ in the interval $[T-p(\mathfrak{E}), T]$ in order of non-increasing processing times; the correctness of this rule is easily verified by an interchange argument. The other subproblem is solved by Smith's rule: simply schedule the jobs in $\mathfrak{R}$ in non-decreasing order of their processing times in the interval $[0, p(\mathfrak{R})]$. In the example, $\mathfrak{E} = \varnothing$, and the lower bound is 21α.

A slight improvement of the lower bound is possible. Let $E_{\max}{}^*$ be the minimum maximum earliness for the jobs in $\mathfrak{R}$ if they are processed in the interval $[0, p(\mathfrak{R})]$. We compute $E_{\max}{}^*$ from the minimum-slack-time sequence, that is, the sequence in which the jobs appear in order of non-decreasing values $d_j - p_j$. Avoiding $E_{\max}{}^*$ requires at least $E_{\max}{}^*$ units of machine idle time. The lower bound can therefore be improved by $\alpha E_{\max}{}^*$. If we have stored the shortest-processing-time sequence and the minimum-slack-time sequence, then we compute this lower bound in $O(n)$ time per node. In the example, we have $E_{\max}{}^* = 4$; hence, the lower bound is 25α. This lower bound approach can only be applied in conjunction with Theorem 2 if $\mathfrak{S} = \varnothing$.

Since all jobs in $\mathfrak{R}$ are scheduled in the interval $[0, p(\mathfrak{R})]$, and since only one early job in $\mathfrak{R}$ is taken into account, this lower bound is only effective if the due dates are small relative to the sum of the processing times.

4.2. *Second method: relax the machine capacity*

Recall that we write the objective function alternatively as $f(\sigma) = (\beta-\alpha)\Sigma_{j=1}^{n} E_j + \alpha\Sigma_{j=1}^{n} T_j + \alpha\Sigma_{j=1}^{n} d_j$ for each $\sigma \in \Omega$. Since the job earlinesses and tardinesses are non-negative by definition, we have that $f(\sigma) \geqslant \alpha\Sigma_{j=1}^{n} d_j$ for each $\sigma \in \Omega$.

We gain more insight if we derive this bound in the following way. Suppose that the machine can process an infinite number of jobs at the same time; this is a relaxation of the limited capacity of the machine. As $\alpha < \beta$, the optimal schedule has $C_j = d_j$ for each J_j; this gives rise to the lower bound $\alpha\Sigma_{j=1}^{n} d_j$. If no jobs overlap in their execution, then this schedule is feasible and hence optimal for the original problem. For the example, this relaxation gives the lower bound 35α. The corresponding schedule is not feasible: J_2 and J_3 overlap in their execution (see Figure 2).

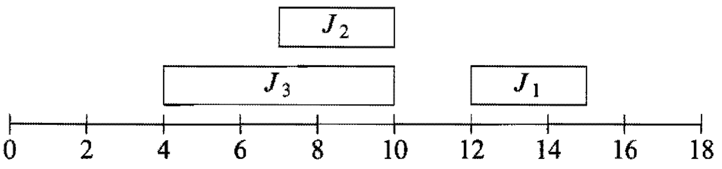


FIGURE 2. Gantt chart for machine with infinite capacity.

This conflict can be settled by executing J_3 before J_2, or, conversely, J_2 before J_3. If we intend to schedule J_2 after J_3, then we have basically two options: we retain either the completion time of J_3 or the completion time of J_2. For the first option, the additional cost is 3α; for the second option, the additional cost is $3(\beta-\alpha)$. Executing J_2 after J_3 costs therefore at least 3γ extra, where $\gamma = \min\{\alpha, \beta-\alpha\}$. Similarly, we find that executing J_3 after J_2 costs 6γ extra. Hence, the *minimum* additional cost required to settle the overlap is $\min\{3\gamma, 6\gamma\} = 3\gamma$. Accordingly, an improved lower bound is 38α.

We now describe a general procedure to improve the lower bound $\alpha\Sigma_{j=1}^{n} d_j$ by taking the overlap between jobs into consideration. Overlap of J_j and J_k ($J_j \neq J_k$) occurs if the intervals $[d_j - p_j, d_j]$ and $[d_k - p_k, d_k]$ overlap. Let $c_{jk} = \gamma \max\{0, d_j - (d_k - p_k)\}$ denote the *additional* cost to execute J_j immediately before J_k; let $\sigma(i) = j$ denote that J_j occupies the ith position in the sequence σ. For any optimal schedule σ, we have that $f(\sigma) \geqslant \alpha\Sigma_{j=1}^{n} d_j + \Sigma_{j=1}^{n-1} c_{\sigma(j)\sigma(j+1)}$; the last term is the length of the Hamiltonian path $\sigma(1) \cdots \sigma(n)$. The following procedure shows that the Hamiltonian path problem is solvable in $O(n \log n)$ time.

Partition the set of jobs into a set of clusters $Q_1, \dots, Q_m$ as described above. Let HP_l be the shortest Hamiltonian path for Q_l, and let $c(HP_l)$ denote its length. We have $c(HP_l) = \gamma(p(Q_l) - \max_{J_j, J_k \in Q_l, J_k \neq J_j} c_{jk})$, for each l ($l = 1, \dots, m$). We have also $\Sigma_{j=1}^{n-1} c_{\pi(j)\pi(j+1)} \geqslant \Sigma_{l=1}^{m} c(HP_l)$ for any sequence π, as can be easily verified. The individual Hamiltonian paths can be combined into one Hamiltonian path of length no more than the sum of the lengths of the separate paths.

4.3. Third method: relax the due dates

4.3.1. The common due date problem

Suppose the due dates have been replaced by a due date d common to all jobs. Consider the following *common due date problem*: for a given d, determine a schedule that minimizes

$$(\beta-\alpha)\sum_{j=1}^{n} E_j + \alpha\sum_{j=1}^{n} T_j + \alpha n d - \beta\sum_{j=1}^{n} \max\{0,d-d_j\}. \qquad \text{(CD)}$$

For any d, the optimal solution value is a lower bound for the original problem, since

$$\begin{aligned} f(\sigma) &= \alpha\sum_{j=1}^{n} C_j + \beta\sum_{j=1}^{n} \max\{0,d_j-C_j\} \\ &\geqslant \alpha\sum_{j=1}^{n} C_j + \beta\sum_{j=1}^{n} \max\{0,d-C_j\} - \beta\sum_{j=1}^{n} \max\{0,d-d_j\} \\ &= (\beta-\alpha)\sum_{j=1}^{n} E_j + \alpha\sum_{j=1}^{n} T_j + \alpha n d - \beta\sum_{j=1}^{n} \max\{0,d-d_j\}. \end{aligned}$$

There are two issues involved: (i) how to solve problem (CD)?, and (ii) how to find the value d maximizing the lower bound?

Problem (CD) consists of two parts. The first part is the problem of minimizing $(\beta-\alpha)\Sigma_{j=1}^{n} E_j + \alpha\Sigma_{j=1}^{n} T_j$. If the machine is only available from time 0 onwards and if d is given, then this problem is $\mathcal{NP}$-hard (Hall, Kubiak, and Sethi, 1991; Hoogeveen and Van de Velde, 1991). However, a strong lower bound $L(d)$ is derived by applying Lagrangian relaxation (see Hoogeveen, Oosterhout, and Van de Velde, 1990). The second part is the problem of maximizing the function $G: d \rightarrow \alpha n d - \beta\Sigma_{j=1}^{n} \max\{0,d-d_j\}$; this problem is solvable in polynomial time. Rather than solving problem (CD) to optimality and finding the best d, we maximize the lower bound $L(d) + G(d)$ over d.

First, we derive the best Lagrangian lower bound $L(d)$ for a given d. The derivation proceeds without details; we refer to Hoogeveen, Oosterhout, and van de Velde (1990) for an elaborate treatment. Let $\mathcal{E}$ denote the set of jobs that are not tardy. Since the machine is only available from time 0 onwards, we have the condition that $p(\mathcal{E}) \leqslant d$. We dualize this condition by use of the Lagrangian multiplier $\lambda \geqslant 0$. For a given $\lambda \geqslant 0$, the Lagrangian problem is then to find $L(d,\lambda)$, which is the minimum of

$$(\beta-\alpha)\Sigma_{j=1}^{n} E_j + \alpha\Sigma_{j=1}^{n} T_j + \lambda p(\mathcal{E}) - \lambda d.$$

The Lagrangian problem is solvable in polynomial time by Emmons's matching algorithm (Emmons, 1987), which proceeds by the concept of positional weights. Straightforward arguments show that there exists an optimal schedule with some job completed exactly on its due date. The weights for the early positions are then $\lambda, \lambda+(\beta-\alpha), \lambda+2(\beta-\alpha), \ldots, \lambda+(n-1)(\beta-\alpha)$; the smallest weight is for the first position in the schedule. The weights for the tardy positions are $\alpha, 2\alpha, \ldots, n\alpha$; the smallest weight is for the last position in the schedule. Emmons's matching algorithm assigns the job with the jth largest processing time to the position with the jth smallest weight, for $j = 1, \ldots, n$. Ties are settled to minimize the amount of work before d. Let σ_λ be the optimal schedule for the Lagrangian problem, and let $W(\sigma_\lambda)$ be the amount of work before d in σ_λ.

The best Lagrangian lower bound $L(d)$ is found as

$$L(d) = \max\{\, L(d,\lambda) \mid \lambda \geqslant 0 \,\}.$$

Due to the integrality of α and β, the optimization over $\lambda \geqslant 0$ may be reduced to the optimization over $\lambda \in \mathbb{N}_0$. The optimal choice for λ can be shown to be such that $W(\sigma_{\lambda-1}) > d \geqslant W(\sigma_\lambda)$; this choice gives us the Lagrangian lower bound $L(d)$.

We are now able to characterize the function $L: d \rightarrow L(d)$. The function L is continuous and piecewise linear; the value $L(d)$ depends on d only through the choice for λ. Hence, there are at most $\min\{n^2, n\alpha\}$ breakpoints: they correspond to the values $d = W(\sigma_\lambda)$, for $\lambda = 0, 1, \ldots, n\alpha$. The derivative of the trade-off curve between two consecutive breakpoints, the first corresponding to $W(\sigma_\lambda)$, is equal to $-\lambda$.

The function $G: d \rightarrow and - \beta\Sigma_{j=1}^{n} \max\{0, d-d_j\}$ is also continuous and piecewise linear; the breakpoints correspond to the values $d = d_j$, for $j = 1, \ldots, n$. The lower bound $L(d)+G(d)$ is therefore also continuous and piecewise linear in d; the value d maximizing this lower bound is found at a breakpoint.

For any given d, $L(d)$ is determined in $O(n \log n)$ time. The function L has $O(\min\{n^2, n\alpha\})$ breakpoints; the corresponding values are computed in $O(n^2)$ time. (Every new breakpoint is derived from the previous one by interchanging some jobs, which requires only constant time, and only $O(n^2)$ interchanges are needed to find all breakpoints.) The function G has $O(n)$ breakpoints. Hence, maximizing $L(d)+G(d)$ over d is achieved in $O(n^2)$ time.

In our 3-job example, we have $d = 10$. For the positions after d, the weights are 1,2, and 3; for the positions before d, the weights are 0, 3, and 6. An optimal schedule is depicted in Figure 3. Its objective value is 39α; this happens to be the optimal solution value for the original problem.

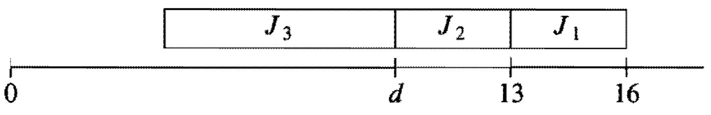


FIGURE 3. Optimal schedule for the common due date problem.

In a node of the search tree, there are two ways to implement this lower bound procedure. Let $\pi = \pi_1\pi_2$ be the partial schedule associated with the node. Disregarding π, we get the lower bound $f(\pi) + c(\mathcal{J}-\pi)$, where $c(\mathcal{J}-\pi)$ denotes the optimal solution value for the common due date problem for the jobs in $\mathcal{J}-\pi$. However, if π_1 and the optimal schedule for the common due date problem overlap in their execution, then it makes sense to take π_1 into regard. We do this in the following way. First of all, we require that d is common to each $J_j \notin \pi_2$. Subsequently, we solve the common due date problem under the condition that the jobs in π_1 retain their positions. Given the set of positions, it is easy to construct an optimal schedule: assign the jobs in π_1 to the last $|\pi_1|$ positions, and assign the other jobs to the remaining positions according to Emmons's algorithm. Lemma 1 states that we may use the same set of positions as for the case $\pi_1 = \varnothing$.

LEMMA 1. *The optimal schedule for the common due date problem with the last $|\pi_1|$ jobs fixed occupies the $\bar{n}$ positions with least positional weights, where $\bar{n} = n - |\pi_2|$.*

PROOF. Suppose to the contrary that the optimal schedule σ for the jobs $J_j \notin \pi_2$ does not occupy the $\bar{n}$ positions with least positional weights. Let n_1 jobs in σ be early or just-in-time and let $n_2 = \bar{n} - n_1$ jobs in σ be tardy. Suppose the set of optimal weights corresponds to $\bar{n}_1$ positions before d, and to $\bar{n}_2 = \bar{n} - \bar{n}_1$ positions after d. Suppose $n_1 < \bar{n}_1$. We then transfer the job occupying the n_2th tardy position in σ (the first tardy job) to the $(n_1 + 1)$th early position. The latter position is in the optimal set; the former is not. Hence, this transfer reduces the objective value, thereby contradicting the optimality of σ. If $n_1 > \bar{n}_1$, then a similar argument applies. □

The common due date lower bound can only be used in conjunction with Theorem 2 if the lower bound is independent from the partial sequence $j\pi$. It is effective if the due dates are close to each other.

4.3.2. The common slack time problem

Consider the special case of the $1||\alpha\Sigma C_j + \beta\Sigma E_j$ problem where all jobs have equal slack time s; i.e., $d_j - p_j = s$ for each J_j $(j = 1, \ldots, n)$. This problem has the same features as the common due date problem. The best Lagrangian lower bound is also computed in $O(n\min\{\alpha, n\})$ time; there are the same options to implement the lower bound. The common slack time lower bound is effective if all slack times are close to each other.

4.4. Fourth method: relax the processing times

Again, we consider a special case of the $1||\alpha\Sigma C_j + \beta\Sigma E_j$ problem. Assume that all processing times are equal. Theorem 3 indicates that the *earliest-due-date* sequence (i.e., the sequence with the jobs in order of non-decreasing due dates) is optimal. This special case is solved in $O(n^2)$ time, which is needed to compute the optimal schedule for a given sequence.

Let us return to our original problem. Define $p_{\min} = \min_{1 \leq j \leq n} p_j$. The optimal solution value of the relaxed problem $1|p_j = p_{\min}|\alpha\Sigma C_j + \beta\Sigma E_j$ provides a lower bound for the original problem: each set of job completion times that is feasible for the original problem is also feasible for the relaxed problem and has equal cost.

Given a partial schedule π, let σ be the earliest-due-date sequence for the jobs in $\mathcal{J} - \pi$, and let $g(\sigma)$ be the optimal solution value for the relaxed problem. Disregarding π, we get the lower bound $f(\pi) + g(\sigma)$. We can marginally improve on this lower bound. Suppose we have reindexed the jobs in order of non-decreasing due dates. Corollary 4 indicates that J_n is also scheduled last if we put its processing time equal to $\min\{p_n, p_{\min} + d_n - d_{n-1}\}$. An improved lower bound is therefore given by $f(\pi) + g(\sigma) + \alpha[\min\{p_n, p_{\min} + d_n - d_{n-1}\} - p_{\min}]$.

If the execution of jobs in σ overlap with the execution of jobs in π, then it pays to take π into account. The lower bound is then equal to the cost for the sequence $\sigma\pi$ with the jobs in π still having their original processing times.

Both bounds are computed in $O(n^2)$ time and dominate the lower bound $\alpha\Sigma_{j=1}^n d_j$. Only the first version can be used in conjunction with Theorem 2. The common processing time lower bounds are only effective if the processing times are close to each other.

In our 3-job example, we have $p_{\min} = 3$, $d_1 = 15$, and $d_2 = d_3 = 10$. An optimal schedule for the common processing time problem is depicted in Figure 4. Its objective value is 39α; this is equal to the optimal solution value for the original problem.

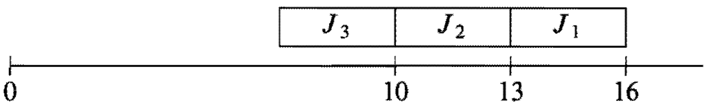


FIGURE 4. Optimal schedule for the common processing time problem.

4.5. *Fifth method: Lagrangian relaxation*

The problem of minimizing total inventory cost, referred to as problem (P), can be formulated as follows. Determine values C_j and E_j ($j = 1, \ldots, n$) that minimize

$$\alpha \sum_{j=1}^{n} C_j + \beta \sum_{j=1}^{n} E_j \tag{P}$$

subject to

$$E_j \geqslant 0, \qquad \text{for } j = 1, \ldots, n, \tag{5}$$

$$E_j \geqslant d_j - C_j, \qquad \text{for } j = 1, \ldots, n, \tag{6}$$

$$C_j \geqslant C_k + p_j \text{ or } C_k \geqslant C_j + p_k, \text{for } j, k = 1, \ldots, n, j \neq k, \tag{7}$$

$$C_j - p_j \geqslant 0, \qquad \text{for } j = 1, \ldots, n. \tag{8}$$

The conditions (5) and (6) reflect the definition of job earliness, while the conditions (7) ensure that the machine executes at most one job at a time. The conditions (8) express that the machine is available only from time 0 onwards.

We introduce a non-negative vector $\lambda = (\lambda_1, \ldots, \lambda_n)$ of Lagrangian multipliers in order to dualize the conditions (5). For a given vector $\lambda \geqslant 0$, the Lagrangian problem is to determine the value $L(\lambda)$, which is the minimum of

$$\alpha \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} (\beta - \lambda_j) E_j$$

subject to the conditions (6), (7), and (8). We know that for any given $\lambda \geqslant 0$ the value $L(\lambda)$ provides a lower bound to problem (P). If $\beta - \lambda_j < 0$ for some J_j, we get $E_j = \infty$, which disqualifies the lower bound. We therefore assume that

$$\lambda_j \leqslant \beta, \qquad \text{for } j = 1, \ldots, n. \tag{9}$$

This, in turn, implies that, for any solution to the Lagrangian problem, conditions (6) hold with equality: $E_j = d_j - C_j$ for each j ($j = 1, \ldots, n$). Hence, the Lagrangian problem, referred to as problem (L_λ), transforms into the problem of minimizing

$$\sum_{j=1}^{n} (\alpha - \beta + \lambda_j) C_j + \sum_{j=1}^{n} (\beta - \lambda_j) d_j \tag{L_λ}$$

subject to

$$C_j \geqslant C_k + p_j \text{ or } C_k \geqslant C_j + p_k, \text{for } j, k = 1, \ldots, n, j \neq k, \tag{7}$$

$$C_j - p_j \geqslant 0, \qquad \text{for } j = 1, \ldots, n. \tag{8}$$

If $\alpha - \beta + \lambda_j < 0$ for some J_j, we get $C_j = \infty$, which makes the lower bound rather weak. However, as demonstrated at the beginning of Section 4, we can determine an upper bound T on the job completion times, which implies that

$$C_j \leqslant T, \qquad \text{for } j = 1, \ldots, n. \tag{10}$$

Although the conditions (10) are redundant for the primal problem (P), they are essential to admit values $\lambda_j < \beta - \alpha$. For solving problem ($\mathrm{L}_\lambda$) under these additional conditions, we first determine the sets of jobs $\mathcal{J}^+ = \{J_j \mid \lambda_j > \beta - \alpha\}$, $\mathcal{J}^- = \{J_j \mid \lambda_j < \beta - \alpha\}$, and $\mathcal{J}^0 = \{J_j \mid \lambda_j = \beta - \alpha\}$. The following theorem stipulates that problem (L_λ) is solved by a simple extension of Smith's rule (Smith, 1956) for solving the $1 \| \Sigma w_j C_j$ problem; the proof proceeds by an elementary interchange argument.

THEOREM 9. *Problem* (L_λ) *with the additional conditions* (10) *is solved by scheduling the jobs in* $\mathcal{J}^+$ *in non-increasing order of ratios* $(\alpha - \beta + \lambda_j)/p_j$ *in the interval* $[0, p(\mathcal{J}^+)]$, *and scheduling the jobs in* $\mathcal{J}^-$ *in non-increasing order of ratios* $(\alpha - \beta + \lambda_j)/p_j$ *in the interval* $[T - p(\mathcal{J}^-), T]$. *The remaining jobs can be scheduled in any order in the interval* $[p(\mathcal{J}^+), T - p(\mathcal{J}^-)]$. □

We are interested in determining the vector $\lambda^* = (\lambda_1^*, \ldots, \lambda_n^*)$ of Lagrangian multipliers that induces the best Lagrangian lower bound. The vector λ^* stems from solving the *Lagrangian dual problem*, referred to as problem (D): maximize

$$L(\lambda) \qquad \text{(D)}$$

subject to

$$0 \leqslant \lambda_j \leqslant \beta, \quad \text{for } j = 1, \ldots, n.$$

Problem (D) is solvable to optimality in polynomial time by use of the ellipsoid method; see Van de Velde (1991). Since the ellipsoid method is very slow in practice, we take our resort to an approximation algorithm for problem (D).

First, we identify the *primitive directional derivatives*. In the solution to the Lagrangian problem (L_λ), the position of J_j depends on the ratio $(\alpha-\beta+\lambda_j)/p_j$; we call this ratio the *relative weight* of J_j. The larger this relative weight, the smaller the completion time of J_j. If other jobs have precisely the same relative weight as J_j, then the exact position of J_j is determined by settling ties. Let now $C_j^+(\lambda)$ denote the earliest possible completion time of J_j in an optimal schedule for problem (L_λ); let $C_j^-(\lambda)$ denote the latest possible completion time of J_j in an optimal schedule for problem (L_λ). If we increase λ_j by $\epsilon > 0$, then we can choose ϵ small enough to make sure that at least one optimal schedule for problem (L_λ) remains optimal; for a proof, see Van de Velde (1991). In fact, all such optimal schedules must have J_j completed on time $C_j^+(\lambda)$. If we increase λ_j by such a sufficiently small $\epsilon > 0$, then the Lagrangian objective value is affected by $\epsilon(C_j^+(\lambda) - d_j)$. The primitive directional derivative for increasing λ_j, as denoted by $l_j^+(\lambda)$, is therefore simply

$$l_j^+(\lambda) = C_j^+(\lambda) - d_j, \quad \text{for } j = 1, \ldots, n.$$

Hence, if $l_j^+(\lambda) > 0$, then increasing λ_j is an ascent direction: we get an improved lower bound by moving some scalar step size along this direction. In a similar fashion, we derive that the primitive directional derivative for decreasing λ_j, denoted by $l_j^-(\lambda)$, is

$$l_j^-(\lambda) = d_j - C_j^-(\lambda), \quad \text{for } j = 1, \ldots, n.$$

If $l_j^-(\lambda) > 0$, then decreasing λ_j is an ascent direction. Note that directional derivatives may not exist at the boundaries of the feasible region of λ; for instance, $l_i^-(\lambda)$ is undefined for $\lambda = (0, \ldots, 0)$, for any $i = 1, \ldots, m$.

Second, we determine an appropriate step size $\Delta > 0$ to move by along a chosen ascent direction. We compute the step size that takes us to the first point where the corresponding primitive directional derivative is no longer positive. If no such point exists, then we choose the step size as large as possible while maintaining feasibility.

Suppose $l_j^+(\lambda) > 0$: J_j is tardy in any optimal schedule for problem (L_λ). Increasing λ_j, thereby putting J_j earlier in the schedule, is an ascent direction. We distinguish the cases $p_j - d_j > 0$, $p_j - d_j = 0$, and $p_j - d_j < 0$. Consider the case $p_j - d_j > 0$. Hence, J_j is unavoidably tardy, and $l_j^+(\lambda) > 0$ for all $\lambda \geqslant 0$ with $\lambda_j < \beta$. Therefore, we take the step size $\Delta = \beta - \lambda_j$. Accordingly, we must also have that $\lambda_j^* = \beta$; otherwise, increasing λ_j^* would be an ascent direction. If $p_j = d_j$, then there exists an optimal solution to problem (D) with $\lambda_j^* = \beta$. Find $\mathcal{T} = \{J_j \mid p_j \geqslant d_j\}$. We have proven the following result.

THEOREM 10. *There exists an optimal solution for the Lagrangian dual problem* (D) *with* $\lambda_j^* = \beta$ *for each* $J_j \in \mathcal{T}$. □

Suppose now $p_j < d_j$. The step size Δ must satisfy $\lambda_j + \Delta \leqslant \beta$. We identify the first job in the schedule, say, J_k, for which $C_k - p_k + p_j \leqslant d_j$. Since $p_j < d_j$, such a J_k always exists. If J_j is scheduled in J_k's position, then J_j is not tardy. Hence, if there were no upper bound on λ, then increasing λ_j would be an ascent direction up to the point where the relative weight of J_j becomes equal to the relative weight of J_k. Hence, the maximum step size along this ascent direction is the largest value Δ such that

$$(\alpha - \beta + \lambda_j + \Delta)/p_j \leqslant (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$
$$\lambda_j + \Delta \leqslant \beta.$$

Let now $\bar{\lambda} = (\lambda_1, \ldots, \lambda_j + \Delta, \ldots, \lambda_n)$. Suppose $\bar{\lambda}_j + \Delta < \beta_j$. Since the relative weights for all jobs but J_j have remained the same, optimal solutions for the problems ($L_{\bar{\lambda}}$) and (L_λ) exist with the same jobs scheduled before J_k. Now J_j and J_k have equal relative weights: in any optimal solution to problem ($L_{\bar{\lambda}}$), J_j can be scheduled before J_k or after J_k. If J_j is scheduled before J_k, then J_j is not tardy; if J_j is scheduled after J_k, then J_j is not early. Hence, we have that $C_j^+(\bar{\lambda}) \leqslant d_j \leqslant C_j^-(\bar{\lambda})$; the step size Δ has taken us to the first point where the primitive directional derivative for increasing λ_j is no longer positive. If $\bar{\lambda}_j = \beta$, then the step size has been chosen as large as possible.

Suppose now $l_j^-(\lambda) < 0$: J_j is early in any optimal schedule for problem (L_λ). Decreasing λ_j, thereby deferring J_j, is an ascent direction. We distinguish the cases $d_j > T$, $d_j = T$, and $d_j < T$. Consider the case $d_j > T$; hence, J_j is unavoidably early, and $l_j^-(\lambda) > 0$ for all λ with $\lambda_j > 0$. Therefore, we choose the step size as large as possible: $\Delta = \lambda_j$. Accordingly, we also must have that $\lambda_j^* = 0$; otherwise, decreasing λ_j^* would be an ascent direction. If $d_j = T$, then there exists an optimal schedule to problem (D) with $\lambda_j^* = 0$. Identify $\mathcal{E} = \{J_j \mid d_j \geqslant T\}$. We have proven the following result.

THEOREM 11. *There exists an optimal solution for the Lagrangian dual problem* (D) *with* $\lambda_j^* = 0$ *for each* $J_j \in \mathcal{E}$. □

Consider now the case $d_j < T$. The procedure to compute the appropriate step size Δ proceeds in a similar fashion as above. We identify some J_k as the first job in the schedule with $C_k \geqslant d_j$. If J_j is scheduled in J_k's position, then J_j is not early. Hence, if there were no lower bound on λ, then decreasing λ_j would be an ascent direction up to the point where the relative weight of J_j becomes equal to the relative weight of J_k. Hence, the maximum step size along this ascent direction is the largest value Δ for which

$$(\alpha - \beta + \lambda_j - \Delta)/p_j \geqslant (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$
$$\lambda_j - \Delta \geqslant 0.$$

Let $\bar{\lambda} = (\lambda_1, \ldots, \lambda_j - \Delta, \ldots, \lambda_n)$. Suppose $\bar{\lambda}_j > 0$. Since the relative weights for all jobs but J_j have remained the same, optimal solutions for the problems ($L_{\bar{\lambda}}$) and (L_λ) exist with the same jobs scheduled after J_k. Since J_j and J_k have now equal weights, J_j can be scheduled after J_k or before J_k in any optimal schedule for problem ($L_{\bar{\lambda}}$). If J_j is scheduled after J_k, then J_j is not early; if J_j is scheduled before J_k, then J_j is not tardy. Hence, we find that $C_j^+(\bar{\lambda}) \leqslant d_j \leqslant C_j^-(\bar{\lambda})$. If $\bar{\lambda}_j = 0$, then the step was taken as large as possible.

Termination of the ascent direction procedure occurs at some $\bar{\lambda}$ where all existing primitive directional derivatives are non-positive. If all primitive directional derivatives exist at such a $\bar{\lambda}$, we have

$$C_j^+(\bar{\lambda}) \leqslant d_j \leqslant C_j^-(\bar{\lambda}), \quad \text{for } j = 1, \ldots, n.$$

These termination conditions also apply to λ^*, since they are necessary for optimality. They are, however, not sufficient for optimality; hence, termination may occur having $\bar{\lambda} \neq \lambda^*$, i.e., before finding the optimal vector of Lagrangian multipliers. Before implementing the ascent direction algorithm, we make use of this fact to decompose the Lagrangian dual problem (D) into two subproblems. This decomposition is achieved by partitioning $\mathcal{J}$ into four subsets, including the sets $\mathcal{T}$ and $\mathcal{E}$ we already identified.

Consider some job $J_j \in \mathcal{J} - \mathcal{E}$ with $d_j > p(\mathcal{J} - \mathcal{E})$. If $\lambda_j > \beta - \alpha$, then J_j will be early in any optimal solution to problem (L_λ). This means that $l_j^-(\lambda) > 0$, and hence we must have that $0 \leqslant \lambda_j^* \leqslant \beta - \alpha$. The set $\mathcal{F}$ of jobs that share this property is determined by the following procedure.

Partitioning Algorithm 1

Step 0. $\mathcal{F} \leftarrow \varnothing$, and reindex the jobs in $\mathcal{J} - \mathcal{E}$ according to non-increasing due dates. Let $k \leftarrow 1$.

Step 1. If $k > n - |\mathcal{E}|$ or if $d_k < p(\mathcal{J} - \mathcal{E} - \mathcal{F})$, then stop. Else $\mathcal{F} \leftarrow \mathcal{F} \cup \{J_k\}$.

Step 2. Set $k \leftarrow k + 1$; go to Step 1.

Suppose some job $J_j \in \mathcal{F}$ exists with $d_j > T - p(\mathcal{E})$. If we let $\lambda_j = \beta - \alpha$, then $C_j^-(\lambda) < d_j$; hence, decreasing λ_j is an ascent direction. Decreasing λ_j gives $(\alpha - \beta + \lambda_j)/p_j < 0$, as a result of which the execution of J_j interferes with the execution of the jobs in $\mathcal{E}$. We now partition the set $\mathcal{F}$ into subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ ($\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$) such that $d_j \leqslant T - p(\mathcal{E} \cup \mathcal{F}_2)$ for each $J_j \in \mathcal{F}_1$, and such that $d_j > T - p(\mathcal{E} \cup \mathcal{F}_2)$ for each $J_j \in \mathcal{F}_2$. To achieve this, we use the following partitioning procedure; it is similar to the first one.

Partitioning Algorithm 2

Step 0. Put $\mathcal{F}_2 \leftarrow \varnothing$, let $P \leftarrow T - p(\mathcal{E})$, and reindex the jobs in $\mathcal{F}$ according to non-increasing due dates. Let $k \leftarrow 1$.

Step 1. If $k > |\mathcal{F}|$, then stop. If $d_k \leqslant P$, then let $\mathcal{F}_1 \leftarrow \{J_k, \ldots, J_{|\mathcal{F}|}\}$, and stop. Otherwise, $\mathcal{F}_2 \leftarrow \mathcal{F}_2 \cup \{J_k\}$, and set $P \leftarrow P - p_k$.

Step 2. Set $k \leftarrow k + 1$; go to Step 1.

Let $\mathcal{R} = \mathcal{J} - \mathcal{T} - \mathcal{E} - \mathcal{F}$.

THEOREM 12. *For each $J_j \in \mathcal{F}_1$, we have that $\lambda_j{}^* = \beta - \alpha$.*

PROOF. Since we have $p(\mathcal{T} \cup \mathcal{R}) \leqslant d_j \leqslant T - p(\mathcal{E} \cup \mathcal{F}_2)$, the result follows. □

At this stage, we can decompose the Lagrangian dual problem (D) into two subproblems. Since $(\alpha - \beta + \lambda_j{}^*)/p_j = 0$ for each $J_j \in \mathcal{F}_1$, the jobs in $\mathcal{F}_1$ do not interfere with the execution of the other jobs. However, $\mathcal{T}$ and $\mathcal{R}$ interfere with each other, and $\mathcal{E}$ and $\mathcal{F}_2$ interfere with each other. On the one hand, we have the dual problem restricted to the sets $\mathcal{T}$ and $\mathcal{R}$; on the other hand, we have the dual problem restricted to the sets $\mathcal{F}_2$ and $\mathcal{E}$. In each optimal schedule for problem (D), the jobs in $\mathcal{T}$ and $\mathcal{R}$ are scheduled in the interval $[0, p(\mathcal{T} \cup \mathcal{R})]$, and the jobs in $\mathcal{F}$ and $\mathcal{E}$ are scheduled in the interval $[T - p(\mathcal{E} \cup \mathcal{F}_2), T]$. We give step-wise descriptions of the ascent direction algorithms for these two subproblems. Both are based upon the primitive directional derivatives and the step sizes we discussed earlier. The jobs in $\mathcal{F}_1$ are scheduled somewhere in the interval $[p(\mathcal{T} \cup \mathcal{R}), T - p(\mathcal{E} \cup \mathcal{F}_2)]$; they are left out of consideration. We introduce some new notation. Let $(\mathrm{L}_\lambda^{\mathcal{R} \cup \mathcal{T}})$ and $(\mathrm{L}_\lambda^{\mathcal{E} \cup \mathcal{F}_2})$ denote the Lagrangian problem restricted to the set $\mathcal{R} \cup \mathcal{T}$ and to the set $\mathcal{E} \cup \mathcal{F}_2$; let $L^{\mathcal{R} \cup \mathcal{T}}(\lambda)$ and $L^{\mathcal{E} \cup \mathcal{F}_2}(\lambda)$ denote their optimal solution values.

ASCENT DIRECTION ALGORITHM FOR THE SET $\mathcal{T} \cup \mathcal{R}$

Step 0. For each $J_j \in \mathcal{T}$, set $\lambda_j \leftarrow \lambda_j{}^* = \beta$; for each $J_j \in \mathcal{R}$, set $\lambda_j \leftarrow \beta$. Solve $(\mathrm{L}_\lambda^{\mathcal{R} \cup \mathcal{T}})$, settling ties arbitrarily; compute the job completion times.

Step 1. For each $J_j \in \mathcal{R}$, do the following:

(a) If $C_j^-(\lambda) < d_j$, identify J_k as the first job in the schedule with $C_k \geqslant d_j$. Compute the largest value Δ such that

$$(\alpha - \beta + \lambda_j - \Delta)/p_j \geqslant (\alpha - \beta + \lambda_k)/p_k, \text{ and} \tag{11}$$

$$\lambda_j - \Delta \geqslant \beta - \alpha. \tag{12}$$

Decrease λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

(b) If $C_j^+(\lambda) > d_j$, identify J_k that is the first job in the schedule with $C_k - p_k + p_j \leqslant d_j$. Compute the largest value for Δ such that

$$(\alpha - \beta + \lambda_j + \Delta)/p_j = (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$

$$\lambda_j + \Delta \leqslant \beta.$$

Increase λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

Step 2. If no multiplier adjustment has taken place, then compute $L^{\mathcal{R} \cup \mathcal{T}}(\lambda)$ and stop. Otherwise, go to Step 1.

THEOREM 13. *The procedure described above generates a series of monotonically increasing values $L^{\mathcal{R} \cup \mathcal{T}}(\lambda)$.*

PROOF. First, consider some $J_j \in \mathfrak{R}$ with $C_j^-(\lambda) < d_j$: decreasing λ_j is an ascent direction. For brevity, we let $\mu_j = \alpha - \beta + \lambda_j$ for each j ($j = 1, \ldots, |\mathfrak{R} \cup \mathfrak{T}|$). We reindex the jobs in order of non-increasing values μ_j / p_j, settling all ties arbitrarily except for J_j: we give J_j the largest index possible. Accordingly, we obtain the sequence $(J_1, \ldots, J_{|\mathfrak{R} \cup \mathfrak{T}|})$, which is optimal for problem $(\mathrm{L}_\lambda^{\mathfrak{R} \cup \mathfrak{T}})$, with job completion times $C_1, \ldots, C_{|\mathfrak{R} \cup \mathfrak{T}|}$. We note that $C_j = C_j^-(\lambda)$. Let Δ be the step size computed as prescribed in the ascent direction algorithm, and let $\bar{\lambda} = (\lambda_1, \ldots, \lambda_j - \Delta_1, \ldots, \lambda_{|\mathfrak{R} \cup \mathfrak{T}|})$.

We distinguish the case that condition (11) holds with equality from the case that condition (12) holds with equality. Consider the first case; accordingly let J_k be the job specified in the ascent direction procedure. In more detail, the sequence under consideration is $(J_1, \ldots, J_{j-1}, J_j, J_{j+1}, \ldots, J_{k-1}, J_k, J_{k+1}, \ldots, J_{|\mathfrak{R} \cup \mathfrak{T}|})$; an optimal sequence for problem $(\mathrm{L}_{\bar{\lambda}}^{\mathfrak{R} \cup \mathfrak{T}})$ is then $(J_1, \ldots, J_{j-1}, J_{j+1}, \ldots, J_k, J_j, J_{k+1}, \ldots, J_{|\mathfrak{R} \cup \mathfrak{T}|})$. The job completion times for the latter sequence can conveniently be expressed in terms of $C_1, \ldots, C_{|\mathfrak{R} \cup \mathfrak{T}|}$. We now prove that $L^{\mathfrak{R} \cup \mathfrak{T}}(\bar{\lambda}) > L^{\mathfrak{R} \cup \mathfrak{T}}(\lambda)$. We have

$$\begin{aligned}
L^{\mathfrak{R} \cup \mathfrak{T}}(\bar{\lambda}) &= \sum_{i=1}^{j-1} \mu_i C_i + (\mu_j - \Delta)(C_j^-(\lambda) + \sum_{i=j+1}^{k} p_i) + \\
&\quad \sum_{i=j+1}^{k} \mu_i (C_i - p_j) + \sum_{i=k+1}^{|\mathfrak{R} \cup \mathfrak{T}|} \mu_i C_i + \sum_{i=1}^{|\mathfrak{R} \cup \mathfrak{T}|} (\beta - \lambda_i) d_i + \Delta d_j \\
&= L^{\mathfrak{R} \cup \mathfrak{T}}(\lambda) - p_j \sum_{i=j+1}^{k} \mu_i + \mu_j \sum_{i=j+1}^{k} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k} p_i - d_j) \\
&= L(\lambda) - p_j \sum_{i=j+1}^{k-1} \mu_i + \mu_j \sum_{i=j+1}^{k-1} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k-1} p_i - d_j) + \\
&\quad (\mu_j - \Delta) p_k - p_j \mu_k .
\end{aligned}$$

Note that $(\mu_j - \Delta)/p_j = \mu_k / p_k$; hence, we have $(\mu_j - \Delta) p_k - p_j \mu_k = 0$. This implies that

$$L(\bar{\lambda}) \geqslant L(\lambda) + p_j \sum_{i=j+1}^{k-1} \left[p_i (\mu_j / p_j - \mu_i / p_i) \right] - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k-1} p_i - d_j).$$

Since $d_j > C_j^-(\lambda) + \Sigma_{i=j+1}^{k-1} p_i$, $\mu_j / p_j > \mu_i / p_i$ for each i ($i = j+1, \ldots, k-1$), and $\Delta > 0$, we have that $L^{\mathfrak{R} \cup \mathfrak{T}}(\bar{\lambda}) > L^{\mathfrak{R} \cup \mathfrak{T}}(\lambda)$.

Now assume that the condition (12) holds with equality and the condition (11) does not: $\Delta = \alpha - \beta + \lambda_j$. This implies that J_j will now be placed after some job J_h, with $j \leqslant h < k$. For this case, the second sequence is $(J_1, \ldots, J_{j-1}, J_{j+1}, \ldots, J_h, J_j, J_{h+1}, \ldots, J_k, \ldots, J_{|\mathfrak{R} \cup \mathfrak{T}|})$. We perform a similar analysis as above to obtain

$$\begin{aligned}
L^{\mathfrak{R} \cup \mathfrak{T}}(\bar{\lambda}) &= L^{\mathfrak{R} \cup \mathfrak{T}}(\lambda) - p_j \sum_{i=j+1}^{h} \mu_i + \mu_j \sum_{i=j+1}^{h} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{h} p_i - d_j) = \\
&= L^{\mathfrak{R} \cup \mathfrak{T}}(\lambda) + p_j \sum_{i=j+1}^{h} \left[p_i (\mu_j / p_j - \mu_i / p_i) \right] - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{h} p_i - d_j).
\end{aligned}$$

At this point, similar arguments as before apply to show that $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda)$.

Second, consider the case that $C_j^+(\lambda) > d_j$ for some $J_j \in \mathcal{R}$: increasing λ_j is an ascent direction. Let Δ be the desired step size, computed as described in the ascent direction algorithm. The proof to show that $L^{\mathcal{R}\cup\mathcal{T}}(\lambda_1,\ldots,\lambda_j+\Delta,\ldots,\lambda_{|\mathcal{R}\cup\mathcal{T}|}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda_1,\ldots,\lambda_j,\ldots,\lambda_{|\mathcal{R}\cup\mathcal{T}|})$ follows the same lines as above. □

Ascent Direction Algorithm for the Set $\mathcal{F}_2 \cup \mathcal{S}$

Step 0. Set $\lambda_j \leftarrow \beta-\alpha$ for each $J_j \in \mathcal{F}_2$, and $\lambda_j \leftarrow \lambda_j^* = 0$ for each $J_j \in \mathcal{S}$. Solve $(L_\lambda^{\mathcal{S}\cup\mathcal{F}_2})$, settling ties arbitrarily; compute the job completion times.

Step 1. For each $J_j \in \mathcal{F}_2$, do the following:

(a) If $C_j^-(\lambda) < d_j$, identify J_k as the first job in the schedule with $C_k \geqslant d_j$. Compute the largest value Δ such that

$$(\alpha-\beta+\lambda_j-\Delta)/p_j \geqslant (\alpha-\beta+\lambda_k)/p_k, \text{ and}$$
$$\Delta \leqslant \lambda_j.$$

Decrease λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

(b) If $C_j^+(\lambda) > d_j$, identify J_k that is the first job in the schedule with $C_k \leqslant d_j + p_k - p_j$. Compute the largest value for Δ such that

$$(\alpha-\beta+\lambda_j+\Delta)/p_j = (\alpha-\beta+\lambda_k)/p_k, \text{ and}$$
$$\lambda_j + \Delta \leqslant \beta-\alpha.$$

Increase λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

Step 2. If no multiplier adjustment has taken place, then compute $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda})$ and stop. Otherwise, go to Step 1.

Theorem 14. *The procedure described above generates a series of monotonically increasing values* $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda})$.

Proof. The proof proceeds along the same lines as the proof of Theorem 13. □

For each $J_j \in \mathcal{J}-\mathcal{F}_1$, let C_j and $\bar{\lambda}_j$ denote the completion time and the Lagrangian multiplier upon termination of the appropriate ascent direction algorithm. We note that $\bar{\lambda}_j = \beta_j$ for each $J_j \in \mathcal{T}$, $\bar{\lambda}_j = \beta-\alpha$ for each $J_j \in \mathcal{F}_1$, and $\bar{\lambda}_j = 0$ for each $J_j \in \mathcal{S}$. Hence, the overall Lagrangian lower bound is given by

$$L(\bar{\lambda}) = \sum_{J_j\in\mathcal{T}} \alpha C_j + \sum_{J_j\in\mathcal{F}_1} \alpha d_j + \sum_{J_j\in\mathcal{S}} \Big[(\alpha-\beta)C_j + \beta d_j\Big] + \\ + \sum_{J_j\in\mathcal{R}\cup\mathcal{F}_2} \Big[(\alpha-\beta+\bar{\lambda})C_j - (\beta-\bar{\lambda}_j)d_j\Big]$$

5. Computational results

The algorithm was coded in the computer language C; the experiments were conducted on a Compaq-386/20 Personal Computer. The algorithm was tested on instances with 8, 10, 12, 15, and 25 jobs. The processing times were generated from the uniform distribution [10,100]. The due dates were generated from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$, where $P = \Sigma_{j=1}^{n} p_j$ and where R and T are parameters. For both parameters, we considered the values 0.2, 0.4, 0.6, 0.8, and 1.0. This procedure to generate due dates parallels the procedure described by Potts and Van Wassenhove (1985) for the weighted tardiness problem. For each combination of T, P, and n, we generated 5 instances. Each instance was considered with $\alpha = 1$ and with β running from 2 to 5.

The general impression was that instances become difficult with smaller values of T, with smaller values of R, and with smaller values of β. A small value of T induces relative large due dates, implying that the machine will be idle for some time before processing the first job. A small value of R induces due dates that are close to each other; it is then harder to partition the jobs. A large value of β implies that earliness is severely penalized; most jobs will therefore be tardy. Accordingly, the instances with $T=0.2$, $R=0.2$, and $\beta = 5$ are the hardest; the instances with $T = 1.0$, $R = 1.0$, and $\beta = 2$ are the easiest.

Table 2 exhibits a summary of our computational results; we only report the results for the instances with T and R equal. It shows that instances with up to 10 jobs are easy. For $n = 12$, the instances with $T=R=0.2$ require already considerable effort. For $n=20$, only the choice $T=R=1.0$ induces instances that are solvable within reasonable time limits. It is likely, however, that the performance of the algorithm is considerably enhanced by fine-tuning the algorithm to specific instances. Currently, all lower bounds are computed in each node of the tree; Lagrangian relaxation, for instance, is useless for instances with $T=R=0.2$.

6. Conclusions

Although machine idle time is a practical instrument to reduce inventory cost, a considerable lack of theoretical analysis of related machine scheduling problems exists. Within this context, we have addressed the $1 \| \alpha\Sigma C_j + \beta\Sigma E_j$ problem for the case that $\alpha < \beta$. It is a very difficult problem from a practical point of view.

Acknowledgement

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

References

A.V. Aho, J.E. Hopcroft, and J.D. Ullman (1982). *Data Structures and Algorithms,* Addison-Wesley, Reading, Massachusetts.

K. Baker and G. Scudder (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research 38,* 22-36.

J. Du and J. Y-T. Leung (1990). Minimizing total tardiness on one machine is *NP*-hard. To appear in *Mathematics of Operations Research.*

H. Emmons (1987). Scheduling to a common due date on parallel uniform

		$\beta=2$		$\beta=3$		$\beta=4$		$\beta=5$	
n	T,R	nodes	sec	nodes	sec	nodes	sec	nodes	sec
8	0.2	417	2	406	2	301	2	58	1
8	0.4	131	1	198	1	185	1	31	1
8	0.6	34	1	48	1	29	1	5	1
8	0.8	23	1	37	1	14	1	8	1
8	1.0	20	1	36	1	33	1	15	1
10	0.2	2438	8	2525	9	2088	7	484	2
10	0.4	266	2	689	3	570	3	202	2
10	0.6	123	1	110	1	88	1	52	1
10	0.8	126	1	122	1	107	1	64	1
10	1.0	109	1	140	1	78	1	40	1
12	0.2	30182	103	26676	106	18358	78	10487	48
12	0.4	15176	66	20756	100	15613	75	10391	50
12	0.6	212	2	262	2	53	1	10	1
12	0.8	380	2	576	4	300	2	170	1
12	1.0	432	2	527	3	226	2	96	1
15	0.2	-	-	-	-	-	-	(2)	-
15	0.4	(3)	-	(2)	-	(2)	-	(30	-
15	0.6	1414	10	2407	17	927	7	339	2
15	0.8	1665	13	1865	15	1647	14	540	5
15	1.0	493	6	402	17	2063	17	1082	9
20	0.2	-	-	-	-	-	-	-	-
20	0.4	-	-	-	-	-	-	-	-
20	0.6	7991	80	13169	136	5529	62	2048	24
20	0.8	8183	85	7244	84	4016	55	1318	21
20	1.0	5127	49	5243	41	2191	32	651	12

TABLE 2. Computational results. For each combination of n ($n=8,10,12,15,20$), of T and R ($T=R=0.2,0.4,0.6,0.8,1.0$), and of β ($\beta=2,3,4,5$), we present the average number of nodes and the average number of seconds; the average was computed over 5 instances. All averages were rounded up to the nearest integer. The sign '-' indicates that not all instances of this particular combination could be solved without examining more than 100,000 nodes.

processors. *Naval Research Logistics 34*, 803-810.

T.D. FRY AND G. KEONG LEONG (1987A). Single machine scheduling: a comparison of two solution procedures. *Omega 15*, 277-282.

T.D. FRY AND G. KEONG LEONG (1987B). A bi-criterion approach to minimizing inventory costs on a single machine when early shipments are forbidden. *Computers and Operations Research 14*, 363-368.

M.R. GAREY, R.E. TARJAN, AND G.T. WILFONG (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research 13*, 330-348.

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.

N.G. HALL, W. KUBIAK, AND S.P. SETHI (1991). Earliness-tardiness scheduling problems, II: Deviation of completion times about a common due date. To appear in *Operations Research*.

J.A. HOOGEVEEN (1990). *Minimizing maximum earliness and maximum lateness on a single machine*, Report BS-R9001, CWI, Amsterdam.

J.A. HOOGEVEEN, H. OOSTERHOUT, AND S.L. VAN DE VELDE (1990). *New lower and upper bounds for scheduling around a small common due date*, Report BS-R9030, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1990). *Polynomial-time algorithms for single-machine multicriteria scheduling*, Report BS-R9008, CWI, Amsterdam.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE (1991). Scheduling around a small common due date. To appear in *European Journal of Operational Research*.

L.G. KHACHIYAN (1979). A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR 244*, 1093-1096 (English translation: *Soviet Mathematics Doklady 20*, 191-194).

E.L. LAWLER AND J.M. MOORE (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science 16*, 77-84.

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

P.S. OW AND T.E. MORTON (1989). The single-machine early/tardy problem. *Management Science 35*, 177-191.

C.H. PAPADIMITRIOU AND K. STEIGLITZ (1982). *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey.

C.N. POTTS AND L.N. VAN WASSENHOVE (1985). A branch-and-bound algorithm for the total weighted tardiness problem. *Operations Research 33*, 363-377.

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

S.L. VAN DE VELDE (1991). *Machine scheduling and Lagrangian relaxation* , Doctoral Thesis, CWI, Amsterdam.

Samenvatting

Dit proefschrift is gebaseerd op onderzoek dat is verricht op het gebied van machinevolgordeproblemen. Het onderzoek op dit gebied heeft zich lange tijd beperkt tot problemen waarbij slechts één beslissingscriterium een rol speelt. Aangezien bij de beoordeling van een productieschema niet alleen de productiekosten maar ook de wensen van de klanten een rol spelen, groeit het besef dat praktijkproblemen niet adequaat worden beschreven door een één-criterium machinevolgordeprobleem; vandaar dat het onderzoeksgebied van de bicriteria-machinevolgordeproblemen zich in een groeiende populariteit mag verheugen.

De door mij bestudeerde bicriteria-machinevolgordeproblemen kunnen als volgt worden beschreven. Een verzameling van taken moet worden verwerkt door een machine. Deze machine is beschikbaar vanaf tijdstip 0 en kan ten hoogste één taak tegelijkertijd uitvoeren. De verwerkingsduur voor iedere taak is gegeven. Een *schedule* definieert voor iedere taak een voltooiingstijd zodanig dat aan de beschikbaarheidsrestricties van de machine is voldaan. Een schedule wordt beoordeeld aan de hand van twee verschillende criteria; de doelstellingsfunctie van het probleem is een combinatie van beide criteria.

De doelstellingsfunctie van een machinevolgordeprobleem met als criteria f en g wordt gevormd door een functie $F(f,g)$; het doel is een schedule te vinden dat tot een minimale waarde van de doelstellingsfunctie leidt. Afhankelijk van de vorm van F kunnen drie verschillende problemen worden onderscheiden. Het *eerste* behelst *hiërarchisch* minimaliseren. Hierbij wordt aangenomen dat criterium f belangrijker is dan g; het probleem is nu om een schedule te vinden dat g minimaliseert onder voorwaarde dat het optimaal is met betrekking tot f. Het *tweede* probleem betreft het algemene geval. Om het probleem op te lossen moet nu de verzameling van *niet-gedomineerde* schedules worden bepaald; een schedule heet niet-gedomineerd als er geen schedule bestaat dat het beter doet op ten minste één criterium en niet slechter op het andere. Het *derde* probleem betreft het geval waarbij F lineair wordt verondersteld.

Het proefschrift bestaat uit drie delen. Het eerste deel bestaat uit een inleiding op het gebied van één-machine scheduling met verschillende criteria en, voor een aantal praktische beslissingscriteria, een overzicht van de complexiteit van de machinevolgordeproblemen met een doelstellingsfunctie die is samengesteld uit twee van deze criteria. De resultaten betreffen *polynomiale* optimaliseringsalgoritmen of *$\mathcal{NP}$-lastigheidsbewijzen*; $\mathcal{NP}$-lastigheid van een probleem ontkent met een grote mate van waarschijnlijkheid het bestaan van een polynomiaal algoritme.

Het tweede deel bestaat uit zeven artikelen op het gebied van bicriteria-machinevolgordeproblemen.

In het eerste artikel worden de problemen geanalyseerd waarbij de doelstellingsfunctie een combinatie is van de criteria *maximaal verschil tussen de gewenste starttijd en de feitelijke starttijd* en *maximaal verschil tussen de gewenste voltooiingstijd en de feitelijke voltooiingstijd*.

In het tweede artikel komen de problemen aan de orde met als doelstellingsfunctie een combinatie van het criterium *som van de voltooiingstijd* met één van de volgende criteria: *maximaal verschil tussen de gewenste voltooiingstijd en de feitelijke voltooiingstijd*, *maximale kosten ten gevolge van de voltooiing van een taak*, en *maximaal verschil tussen de gewenste starttijd en de feitelijke starttijd*.

In het derde artikel worden de problemen behandeld met als doelstellingsfunctie een combinatie van verschillende maximale kosten criteria.

In het vierde artikel wordt een *ondergrensstrategie* gepresenteerd die kan worden gebruikt om ondergrenzen, benodigd voor een *branch-and-bound* algoritme, af te leiden voor machinevolgordeproblemen met een *lineaire* samengestelde doelstellingsfunctie. Aangetoond wordt dat deze strategie de in de literatuur bekende ondergrensstrategieën domineert.

In het vijfde en zesde artikel worden problemen geanalyseerd waarbij alle taken dezelfde gewenste aflevertijd hebben. In het eerste van de twee wordt het probleem geanalyseerd met als doelstellingsfunctie *de gewogen som van de afwijkingen van de gewenste voltooiingstijden van de feitelijke voltooiingstijden*. Van dit probleem wordt bewezen dat het $\mathcal{NP}$-lastig is, zelfs als alle gewichten gelijk zijn, als de gewenste voltooiingstijd klein is ten opzichte van de totale verwerkingsduur van de taken; een *pseudo-polynomiaal* algoritme wordt gepresenteerd om het probleem op te lossen. In het andere artikel wordt het probleem van het minimaliseren van de som van de afwijkingen van de feitelijke voltooiingstijden van de gewenste voltooiingstijd, die klein is verondersteld ten opzichte van de totale verwerkingsduur, aangepakt met behulp van een branch-and-bound algoritme. De benodigde onder- en bovengrenzen worden afgeleid met behulp van *Lagrangiaanse relaxatie*; voor willekeurig gegeneerde problemen blijken deze grenzen bijna altijd samen te vallen als het aantal taken meer dan vijftig bedraagt.

In het laatste artikel wordt onderzoek gedaan naar een machinevolgordeprobleem dat voortkomt uit de zogenaamde *net-op-tijd* (NOP) benadering, waarbij iedere taak geacht wordt precies op het gewenste tijdstip voltooid te worden. De doelstellingsfunctie die gebruikt wordt om deze benadering te weerspiegelen is gedefinieerd als een lineaire combinatie van de som van de voltooiingstijden en de som van de positieve afwijkingen van de gewenste starttijden en de feitelijke starttijden. Hierbij wordt het gewicht van het tweede criterium groter verondersteld dan het gewicht van het eerste criterium; dit heeft tot gevolg dat in een optimaal schedule de machine stil kan staan terwijl er nog werk te doen is. Voor dit probleem is nog geen bruikbaar oplossingsalgoritme ontworpen: hoewel we niet minder dan vijf verschillende ondergrensstrategieën afleiden, zijn we niet in staat om problemen met meer dan twintig jobs in een redelijke tijd op te lossen.

Het derde deel bestaat uit een samenvatting.

STELLINGEN

behorende bij het proefschrift van

JOHANNES ADZER HOOGEVEEN

SINGLE-MACHINE BICRITERIA SCHEDULING

I

Beschouw het volgende probleem. Gegeven zijn een graaf $G = (V,E)$, twee punten $u,v \in V$, en een lengte voor iedere kant in E; bepaal een pad dat ieder punt ten minste één maal bevat, u en v als eindpunten heeft, en een minimale lengte heeft. Voor dit probleem kan met behulp van een Christofides-achtige algoritme een oplossing worden gevonden die niet langer is dan $5/3$ maal het optimum.

J.A. HOOGEVEEN (1991). Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Operations Research Letters 10*, 291-295.

D.S. JOHNSON, C.H. PAPADIMITRIOU (1985). Performance guarantees for heuristics. E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS (eds.). *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*. Wiley, Chichester, 145-180.

II

Het criterium van Hässelbarth is een noodzakelijke en voldoende voorwaarde voor het grafisch zijn van een getallenreeks.

G. SIERKSMA AND J.A. HOOGEVEEN (1991). Seven criteria for integer sequences being graphic. *Journal of Graph Theory 15*, 223-231.

III

Beschouw het volgende probleem. Een verzameling van n opdrachten met gegeven geheeltallige bewerkingstijden p_j en aflevertijden d_j moet worden bewerkt door één machine die beschikbaar is vanaf tijdstip 0 en ten hoogste één opdracht tegelijkertijd kan uitvoeren; bepaal een bewerkingsvolgorde zodanig dat $\Sigma_j \mid C_j - d_j \mid$ wordt geminimaliseerd. Dit probleem is oplosbaar in $O(n^2)$ tijd als iedere aflevertijd groter is dan de som van de bewerkingsduren en als ieder tweetal tijdsintervallen $[d_i - p_i, d_i]$ en $[d_j - p_j, d_j]$ elkaar overlapt.

J.A. HOOGEVEEN AND S.L. VAN DE VELDE *(1992). Scheduling around an almost common due date.* Unpublished manuscript.

IV

Het probleem van het minimaliseren van de maximale voltooiingstijd in een flow shop bestaande uit twee fasen met twee identieke machines in de eerste fase en één machine in de tweede is $\mathcal{NP}$-lastig in de sterke zin, zelfs als onderbreking van de taken is toegestaan.

J.A. HOOGEVEEN, J.K. LENSTRA, AND B. VELTMAN (1992). *Minimizing makespan in a multiprocessor flowshop is strongly $\mathcal{NP}$-hard.* Unpublished manuscript.

V

Het probleem van het minimaliseren van de som van de voltooiingstijden in een twee-machine flow shop is $\mathcal{NP}$-lastig in de sterke zin, zelfs als de bewerkingstijden van alle taken op de eerste machine gelijk zijn. Het probleem is oplosbaar in $O(n^4)$ tijd indien voor alle taken de bewerkingstijd op de eerste machine niet groter is dan de bewerkingstijd op de tweede machine.

T. KAWAGUCHI (1987). Bounds on permutation schedules for the two-processor mean finishing time flowshop problem. Unpublished manuscript.

VI

Beschouw het volgende probleem. Een verzameling van n opdrachten moet worden verwerkt door drie machines die beschikbaar zijn vanaf tijdstip 0 en ten hoogste één opdracht tegelijkertijd kunnen bewerken, waarbij gegeven is door welke machines een opdracht moet worden uitgevoerd; bepaal een rooster zodanig dat de maximale voltooiingstijd wordt geminimaliseerd. Dit probleem is $\mathcal{NP}$-lastig in de sterke zin indien sommige taken twee machines tegelijkertijd nodig hebben.

J.A. HOOGEVEEN, S.L. VAN DE VELDE, AND B. VELTMAN (1992). *Complexity of scheduling multiprocessor tasks with prespecified processor allocations.* Unpublished manuscript.

VII

Onrechtmatig verkregen bewijs is ook bewijs.

VIII

De wijze waarop geld wordt ingezameld bij de kinderpostzegelactie is onrechtmatig, aangezien zij is gebaseerd op het maffia-principe *geld voor bescherming*: alleen door postzegels te kopen kan men zich beschermen tegen de overlast van op verkoop beluste kinderen.

IX

Bij het bridgen dient op de systeemkaart te worden vermeld of er een voorkeur bestaat voor actief of passief starten.

X

Het voetbal als kijkspel kan aantrekkelijker worden gemaakt door de volgende veranderingen in te voeren.

(a) Het simuleren van een overtreding dient te worden bestraft met rood, ook indien de overtreding na afloop op grond van videobeelden wordt geconstateerd. Het op onrechtmatige wijze verhinderen van een doelrijpe kans buiten het strafschopgebied moet worden bestraft met een zogenaamd *penaltyshot*, zoals in de Amerikaanse voetbalcompetitie.

(b) De uitspelende ploeg moet meer betrokken raken bij het amusementsniveau van de wedstrijd. Dit kan worden bereikt door de tegenstander een deel te geven van de recette behaald uit de losse verkoop.

(c) Het moet minder rendabel worden gemaakt met tien man voor het doel te hangen. Dit kan worden bereikt door een bonuspunt toe te kennen voor vier thuis- of drie uitgescoorde doelpunten.

(d) Om in de Europa-cup bij de thuisclub de verlammende angst voor een dubbeltellend tegendoelpunt weg te nemen moet het grootste aantal thuisgescoorde doelpunten de doorslag geven bij een gelijke eindstand na twee wedstrijden.