ELSEVIER Operations Research Letters 17 (1995) 205–208

operations research letters

Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time

J.A. Hoogeveen [a,*], S.L. van de Velde [b]

[a] *Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands*

[b] *Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands*

Received 1 July 1994; revised 1 April 1995

Abstract

We prove that the bicriteria single-machine scheduling problem of minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. Our result settles a long-standing open problem.

Keywords: Single-machine scheduling; Bicriteria scheduling; Total completion time; Maximum cost; Maximum lateness

1. Introduction

Little has been done in the area of multicriteria scheduling theory, in spite of its practical potential; after all, a schedule is evaluated on several performance measures in practice. We refer to Dileepan and Sen [1] for an overview of the literature on multicriteria scheduling problems and to Hoogeveen [4] for a survey of the few non-trivial polynomial-time algorithms and complexity results. In this paper, we prove that a fundamental and probably the most obvious and practical bicriteria problem is solvable in polynomial time.

This problem is described as follows. A set of n independent jobs has to be scheduled on a single machine that is continuously available from time zero onwards and that can process at most one job at a time. Each job J_j ($j = 1,\ldots,n$) requires an uninterrupted positive processing time p_j and has a due date d_j. Without loss of generality, we assume that the processing times and due dates are integral. A *schedule* σ specifies for each job when it is executed while observing the machine availability constraints. Hence, a schedule σ defines for each job J_j its completion time $C_j(\sigma)$, which we sometimes simply write as C_j.

The bicriteria problem that we consider concerns the simultaneous minimization of the performance measures *total completion time* $\sum_{j=1}^n C_j$ and *maximum cost* $f_{\max}$. Maximum cost is defined as $\max_{1\leqslant j\leqslant n} f_j(C_j)$, where each f_j denotes an arbitrary *regular* cost function for J_j; regular means that $f_j(C_j)$ does not decrease when C_j is increased. Note that *maximum lateness* $L_{\max}$, defined as $\max_{1\leqslant j\leqslant n}(C_j - d_j)$, is an important special case of $f_{\max}$. A performance measure is called regular if it is non-decreasing in each of the job completion times. Total completion time $\sum_{j=1}^n C_j$ and maximum cost are both regular performance measures. This implies that for either criterion there is an optimal

* Corresponding author.

solution in which the jobs are scheduled in the interval $[0, \sum_{j=1}^n p_j]$.

With each schedule σ we associate a point $(\sum C_j(\sigma), f_{\max}(\sigma))$ in $\mathbb{R}^2$ and a value $F(\sum_{j=1}^n C_j(\sigma), f_{\max}(\sigma))$, where F is non-decreasing in either of its arguments; that is, for any two schedules σ and π with $\sum_{j=1}^n C_j(\sigma) \leqslant \sum_{j=1}^n C_j(\pi)$ and $f_{\max}(\sigma) \leqslant f_{\max}(\pi)$, we have that $F(\sum_{j=1}^n C_j(\sigma), f_{\max}(\sigma)) \leqslant F(\sum_{j=1}^n C_j(\pi), f_{\max}(\pi))$. In the remainder, we use the terms schedule and point interchangeably. Our bicriteria problem is then formulated as

$$\min \left\{ F\left(\sum_{j=1}^n C_j(\sigma), f_{\max}(\sigma) \right) \middle| \sigma \in \Omega \right\},$$

where Ω denotes the set of feasible schedules. Extending the three-field notation scheme of Graham et al. [3], we denote this problem as $1||F(\sum_{j=1}^n C_j, f_{\max})$.

We can solve the $1||F(\sum_{j=1}^n C_j, f_{\max})$ problem in polynomial time if we can identify all of the so-called *Pareto optimal* schedules in polynomial time.

Definition 1. A schedule $\pi \in \Omega$ is Pareto optimal with respect to the objective functions $(\sum_{j=1}^n C_j, f_{\max})$ if there exists no feasible schedule σ with $\sum_{j=1}^n C_j(\sigma) \leqslant \sum_{j=1}^n C_j(\pi)$ and $f_{\max}(\sigma) < f_{\max}(\pi)$, or $\sum_{j=1}^n C_j(\sigma) < \sum_{j=1}^n C_j(\pi)$ and $f_{\max}(\sigma) \leqslant f_{\max}(\pi)$.

Once the *Pareto optimal set*, that is, the set of all schedules that are Pareto optimal with respect to the functions $(\sum_{j=1}^n C_j, f_{\max})$, has been determined, problem (P) can be solved for any function F that is non-decreasing in each of its arguments by computing the cost of each Pareto optimal point and taking the minimum. As a consequence, if each Pareto optimal schedule can be found in polynomial time and if the cardinality of the Pareto optimal set is polynomially bounded in the input size, then problem (P) is polynomially solvable.

Van Wassenhove and Gelders [11] give an iterative algorithm for $1||F(\sum_{j=1}^n C_j, L_{\max})$ that finds each Pareto optimal point in $O(n \log n)$ time; see also Nelson et al. [8]. John [5] extends their algorithm to determine the set of Pareto optimal points for $(\sum_{j=1}^n C_j, f_{\max})$. The complexity of these algorithms depends on the number of Pareto optimal points. This number has been subject of a lot of misunderstanding. Lawler et al. [7] conjectured that this number is equal to $n(n-1)/2+1$ for $(\sum_{j=1}^n C_j, L_{\max})$. Van Wassenhove and Gelders, on the other hand, supposed that the number of Pareto optimal points for $(\sum_{j=1}^n C_j, L_{\max})$ be bounded only pseudo-polynomially; hence, they presented their algorithm as being pseudo-polynomial. This inspired Sen and Gupta [9] to present a branch-and-bound algorithm for $1||\sum_{j=1}^n C_j + L_{\max}$.

We prove that the number of Pareto optimal points for $(\sum_{j=1}^n C_j, f_{\max})$ is at most equal to $n(n-1)/2+1$. As a consequence, $1||F(\sum_{j=1}^n C_j, f_{\max})$ is polynomially solvable: we present an algorithm for $1||F(\sum_{j=1}^n C_j, f_{\max})$ that runs in $O(n^3 \min\{n, \log(\sum_{j=1}^n p_j)\})$ time; it can be implemented to run in $O(n^3 \log n)$ time if $f_{\max} = L_{\max}$.

2. Total completion time and maximum cost

Emmons [2] addresses the hierarchical problem of minimizing $\sum_{j=1}^n C_j$ subject to the constraint that $f_{\max}$ is minimal; this problem is denoted as $1|f_{\max} \leqslant f^*|\sum_{j=1}^n C_j$, where f^* denotes the optimal solution value of the $1||f_{\max}$ problem. The $1||f_{\max}$ problem is solved in $O(n^2)$ time by Lawler's rule [6]: *while there are unassigned jobs, assign the job that has minimum cost when scheduled at the last unassigned position to that position.* Once f^* has been determined, Emmons's algorithm requires $O(n^2)$ time to minimize total completion time subject to minimum maximum cost. Observe, however, that an upper bound on $f_j(C_j)$ induces a deadline $\bar{d}_j$ on the completion time of J_j. Each deadline can be determined in $O(\log(\sum_{j=1}^n p_j))$ time by binary search over the $\sum_{j=1}^n p_j$ 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 $1|\bar{d}_j|\sum_{j=1}^n C_j$. This problem is solvable in $O(n \log n)$ time by Smith's rule [10]: *while there are unscheduled jobs, schedule from among all jobs that can be scheduled last a job with the largest processing time.* This rule is easily validated by an interchange argument. Note that if there are no deadlines, this rule comes down to sequencing the jobs in order of non-decreasing p_j.

We give a step-wise description of the algorithm based on deadline determination for $1|f_{\max} \leqslant f|\sum_{j=1}^n C_j$, where f is some upper bound on the cost of the schedule. Steps 2–7 then, actually, describe Smith's rule.

Algorithm I.

Step 1: Compute for each job J_j the deadline $\bar{d}_j$ induced by $f_j(C_j) \leqslant f$.

Step 2: $T \leftarrow \sum_{j=1}^n p_j$.

Step 3: Determine $U \leftarrow \{J_j \in J | \bar{d}_j \geqslant T\}$ as the set containing the jobs that may be completed at time T.

Step 4: Determine J_k such that $p_k = \max\{p_j | J_j \in U\}$; in case of ties, J_k is chosen to be the job with smallest cost when completed at time T.

Step 5: Schedule J_k in the time interval $[T - p_k, T]$.

Step 6: $J \leftarrow J - \{J_k\}$; $T \leftarrow T - p_k$.

Step 7: If $T > 0$, then go to Step 3.

Theorem 1. *Algorithm I determines a Pareto optimal point for $\sum_{j=1}^n C_j$ and $f_{\max}$.*

Proof. It suffices to show that the algorithm generates a schedule σ that solves the problems $1|f_{\max} \leqslant f|\sum_{j=1}^n C_j$ and $1|\sum_{j=1}^n C_j \leqslant \sum_{j=1}^n C_j(\sigma)|f_{\max}$ simultaneously. Evidently, σ solves $1|f_{\max} \leqslant f|\sum_{j=1}^n C_j$. Assume that π is optimal for $1|\sum_{j=1}^n C_j \leqslant \sum_{j=1}^n C_j(\sigma)|f_{\max}$, not σ. This implies that $f_{\max}(\pi) < f_{\max}(\sigma) \leqslant f$; hence, π is also feasible for $1|f_{\max} \leqslant f|\sum_{j=1}^n C_j$. Therefore, we have $\sum_{j=1}^n C_j(\pi) = \sum_{j=1}^n C_j(\sigma)$. 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 $1|\sum_{j=1}^n C_j \leqslant \sum_{j=1}^n C_j(\sigma)|f_{\max}$; hence, σ is Pareto optimal for $\sum_{j=1}^n C_j$ and $f_{\max}$. □

The maximum cost of each Pareto optimal schedule ranges from f^* to $f_{\max}(SPT)$, where SPT refers to the schedule obtained by settling ties in the *Shortest-Processing-Time*-order to minimize maximum cost. The next algorithm, which is similar to Van Wassenhove and Gelders's algorithm for $1||F(\sum_{j=1}^n C_j, L_{\max})$, 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 $1|f_{\max} \leqslant f_{\max}(SPT)|\sum_{j=1}^n C_j$; this produces the first Pareto optimal schedule $\sigma^{(1)}$, and the first Pareto optimal point $(\sum_{j=1}^n C_j(\sigma^{(1)}), f_{\max}(\sigma^{(1)}))$.

Step 3: $k \leftarrow k + 1$. Solve $1|f_{\max} < f_{\max}\sigma^{(k-1)}|\sum_{j=1}^n C_j$; this produces the kth Pareto optimal schedule $\sigma^{(k)}$, and the kth Pareto optimal point $(\sum_{j=1}^n C_j(\sigma^{(k)}), f_{\max}(\sigma^{(k)}))$.

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. Let $S_j(\sigma)$ be the start time of job J_j in schedule σ. 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 $\Delta(\sigma) = \sum_{i,j} \delta_{ij}(\sigma)$. Note that $\delta_{ij}(\sigma) = 1$ implies that the interchange of the jobs J_i and J_j in σ will decrease total completion time. In that respect, $\delta_{ij}(\sigma) = 1$ signals a *positive* interchange. Observe that $\Delta(SPT) = 0$ and $\Delta(\sigma) \leqslant n(n-1)/2$ 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 2. *Consider two arbitrary Pareto optimal schedules σ and π. If $\sum_{j=1}^n C_j(\sigma) < \sum_{j=1}^n C_j(\pi)$, 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_j in π is therefore positive or neutral. We proceed in this way until we reach schedule σ. As $\sum_{j=1}^n C_j(\sigma) < \sum_{j=1}^n C_j(\pi)$, at least one of the interchanges must have been positive, and application of Lemma 1 yields the desired result. □

Theorem 3. *The number of Pareto optimal schedules is bounded by $n(n-1)/2+1$, and this bound is tight.*

Proof. The first part follows immediately from Theorem 2. For the second part, consider the following instance of $1||F(\sum_{j=1}^n C_j, L_{\max})$: there are n jobs with processing times $p_j = n-2+j$ and due dates $d_j = \sum_{i=j}^n p_i + n - j$, for $j = 1,\ldots,n$. Straightforward computations show that Algorithm II generates $n(n-1)/2+1$ Pareto optimal schedules for this example. □

Corollary 1. *The $1||F(\sum_{j=1}^n C_j, f_{\max})$ problem is solvable in $O(n^3 \min\{n, \log(\sum_{j=1}^n p_j)\})$ time.*

Proof. We have to solve $O(n^2)$ problems of the type $1|f_{\max}\leqslant f|\sum_{j=1}^n C_j$. To solve such a problem, we can either apply Emmons's algorithm, which requires $O(n^2)$ time, or determine the induced deadlines, which requires $O(\log(\sum_{j=1}^n p_j))$ time, and apply Smith's algorithm. □

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

Proof. We have to solve $O(n^2)$ problems of the type $1|L_{\max}\leqslant L|\sum_{j=1}^n C_j$. Since the constraint $L_{\max}\leqslant L$ yields the set of deadlines $\bar{d}_j = d_j + L$ $(j = 1,\ldots,n)$, which are computed in constant time per job, each of these problems is solved in $O(n \log n)$ time through Smith's algorithm. □

References

[1] P. Dileepan and T. Sen, "Bicriterion static scheduling research for a single machine", *Omega* **16**, 53–59 (1988).

[2] H. Emmons, "A note on a scheduling problem with dual criteria", *Naval Res. Logis. Quarterly* **22**, 615–616 (1975).

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

[4] J.A. Hoogeveen, *Single-machine bicriteria scheduling*, Doctoral Thesis, CWI, Amsterdam, 1992.

[5] T.C. John, "Tradeoff solutions in single machine production scheduling for minimizing flow time and maximum penalty", *Comput. Oper. Res.* **16**, 471–479 (1989).

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

[7] E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, unpublished manuscript, 1979.

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

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

[10] W.E. Smith "Various optimizers for single-stage production", *Naval Res. Logist. Quarterly* **1**, 59–66 (1956).

[11] L.N. Van Wassenhove and F. Gelders, "Solving a bicriterion scheduling problem", *European J. Oper. Res.* **4**, 42–48 (1980).