KYBERNETIKA — VOLUME 23 (1987), NUMBER 4

ONE-MACHINE SCHEDULING WITH ALLOCATION OF CONTINUOUSLY-DIVISIBLE RESOURCE AND WITH NO PRECEDENCE CONSTRAINTS

ADAM JANIAK

The efficiently solved one-machine scheduling problems with no precedence constraints are generalized to the case with allocation of continuously-divisible constrained nonrenewable resource. Models of operation are assumed to be duration versus resource amount linear functions. The following optimality criteria are considered: maximum completion time, maximum lateness, maximum cost and weighted sum of completion times. For the problems discussed polynomial-time algorithms are found.

1. INTRODUCTION

Up to now, one-machine scheduling problems with different optimality criteria were considered under assumption that processing times of operations are constant. For the most of them polynomial-time algorithms exist (cf. [3]). In this paper some of these problems (i.e. problems with polynomial-time algorithms) are generalized to the case when operations processing times depend linearly on the amounts of continuously-divisible nonrenewable, doubly constrained resource, e.g., energy, fuel, oxygen, catalyst, raw materials, money. They can be precisely formulated as follows.

There are n jobs $J_1, \ldots, J_j, \ldots, J_n$ that are to be processed on one machine. The machine can handle only one job at a time. Each job J_j consists of one operation that corresponds to the processing of the job J_j on the machine during an uninterrupted processing time p_j. We shall assume that $p_j \triangleq p_j(u_j) = b_j - a_j u_j, j = 1, 2, \ldots, n$, where u_j is the amount of resource allotted to J_j; $a_j > 0$ may be interpreted as the unit cost of shortening the job J_j processing time and $b_j > 0$ — as the upper bound on p_j. We assume, moreover, the following set of feasible allocations of resource:

$$U \triangleq \{\bar{u} \in \mathbb{R}^n | \bar{u} = [u_1, \ldots, u_j, \ldots, u_n]' \wedge \sum_{j=1}^{n} u_j \leqq \hat{U} \wedge 0 \leqq u_j \leqq \beta_j, \\ j = 1, 2, \ldots, n\},$$

where $\hat{U}$ is the global amount of resource and $\beta_j < b_j/a_j$ $(j = 1, 2, \ldots, n)$ is a given

technological constraint on the maximal amount of resource allotted to the job J_j. (The value $b_j - a_j\beta_j$ is the lower bound on p_j). Let $\pi = \langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ be a permutation of $\{1, 2, \ldots, n\}$ and $\Pi = \{\pi\}$ be the family of all permutations.

We shall consider the maximum cost criterion $- c_{\max} \triangleq \max_{1 \leqq j \leqq n} c_j(C_j)$, where $c_j(t)$ is a nondecreasing function in the time variable t and $C_j \triangleq C_j(\pi, \bar{u})$ is the completion time of the job J_j in permutation $\pi \in \Pi$ under resource allocation $\bar{u} \in U$. The following particular forms of this criterion will be also considered: the maximum completion time $- C_{\max} \triangleq \max_{1 \leqq j \leqq n} \{C_j\}$ and the maximum lateness $- L_{\max} \triangleq \max_{1 \leqq j \leqq n} \{L_j\} \triangleq \triangleq \max_{1 \leqq j \leqq n} \{C_j - d_j\}$, where d_j is the due date, i.e. the moment by which the job J_j should be completed. Moreover, we shall consider the weighted sum of completion times criterion $- \sum w_j C_j \triangleq \sum_{j=1}^{n} w_j C_j$, where $w_j > 0$ is the weight attached to J_j.

One-machine scheduling problem with resource constraints consists in finding such a permutation $\pi^* \in \Pi$ and such an allocation of resource $\bar{u}^* \in U$ that the considered criterion (i.e. one of the above criteria) is minimized. In the next section we shall show that for these problems polynomial-time algorithms exist. Computational time (or a number of computational steps) for such algorithms is bounded by a polynomial $w(n)$ in the size of the problem considered (i.e. the number n of jobs). The computational complexity of such an algorithm (problem) is said to be $0(w(n))$ (cf. [3]).

In the sequel we shall use the following notation scheme of machine scheduling problems (see [3]) $- \alpha|\beta|\gamma$, where α specifies the machine environment (for one-machine scheduling problems $\alpha = 1$); $\beta \subset \{r_j, p_j(u_j)\}$ indicates certain jobs characteristics, where: r_j denotes that release dates (i.e. the moments at which jobs are available for processing) are unequal (the absence of r_j denotes that release dates are equal), $p_j(u_j)$ denotes that $p_j(u_j) = b_j - a_j u_j$ ($p_j(u_j) = b - ua_j$ denotes that all models of operations are of the same form); and γ indicates the optimality criterion.

2. MAIN RESULTS

The $1|p_j(u_j)| C_{\max}$ problem

It is obvious that for any allocation of resource $\bar{u} \in U$ $C_{\max}$ is equal to $\sum_{j=1}^{n} p_j(u_j)$ despite of permutation $\pi \in \Pi$.

Property 1. The following procedure generates (in $0(n \cdot \log n)$ steps) the optimal allocation of resource $\bar{u}^* \in U$ to the $1|p_j(u_j)| C_{\max}$ problem:

Procedure 1. From among all jobs, i.e. from the set $S \triangleq \{1, 2, \ldots, n\}$ choose the job J_h with the greatest parameter a_h and assign to J_h the resource amount $u_h^* = \min \{\beta_h, \hat{U}\}$. Then repeat on the set $S := S - \{h\}$ with $\hat{U} := \hat{U} - u_h^*$, and so on.

It is easy to notice that for the case $1|p_j(u_j) = b - au_j|\, C_{\max}$ the above procedure reduces to the following one (with $0(n)$-complexity): for J_1 assign the resource amount $u_1^* = \min\{\beta_1, \hat{U}\}$. Then repeat for J_2 with $\hat{U} := \hat{U} - u_1^*$, and so on.

The $1|r_j, p_j(u_j)|\, C_{\max}$ problem

The following property holds.

Property 2. The $1|r_j, p_j(u_j)|\, C_{\max}$ problem is solved (in $0(n^2)$ steps) by ordering the jobs π^* according to increasing r_j (i.e. by the Jackson rule [1]) and by the resource allocation $\bar{u}^* \in U$ obtained in agreement with the following procedure.

Procedure 2.

Step 1: Set $u_i := 0$ for $i = 1, 2, \ldots, n$ and $l := 1$. Find the starting times of jobs using the following recursive formulas:

$$S_{\pi^*(1)} := r_{\pi^*(1)}, \quad S_{\pi^*(i)} := \max\left[r_{\pi^*(i)}, S_{\pi^*(i-1)} + b_{\pi^*(i-1)}\right],$$

$i = 2, 3, \ldots, n$ and go to Step 2;

Step 2: Find the greatest index k, $l \leqq k \leqq n$ that satisfies the equation $r_{\pi^*(k)} = S_{\pi^*(k)}$. Then find the set

$$P := \{\pi^*(i) \mid k \leqq i \leqq n,\ u_{\pi^*(i)} < \beta_{\pi^*(i)}\}.$$

If the set P is empty or $\hat{U} = 0$, then Stop — u_i^*, $i = 1, 2, \ldots, n$ is the optimal resource allocation, otherwise go to Step 3.

Step 3: Find an index t that satisfies

$$a_{\pi^*(t)} := \max_{\pi^*(i) \in P} a_{\pi^*(i)}$$

and, then set

$$y := \min_{t < j \leqq n} \left(S_{\pi^*(j)} - r_{\pi^*(j)}\right), \quad x := \min\{\beta_{\pi^*(t)}, \hat{U}, y/a_{\pi^*(t)}\}.$$

Next bring up to date resource allocation and starting times by setting

$$u_{\pi^*(t)} := u_{\pi^*(t)} + x, \quad \hat{U} := \hat{U} - x,$$

$$S_{\pi^*(j)} := S_{\pi^*(j)} - x \cdot a_{\pi^*(t)} \quad \text{for} \quad t < j \leqq n.$$

Finally set $l := k$ and go to Step 2.

Proof. The ordering rule is self-evident since the Jackson rule is independent of the operations processing times. It is obvious that the global amount of resource should be allocated among the jobs composing the critical path. The maximal deterioration of the length of this path is obtained by assignment of feasible resource amounts to the jobs with the maximal parameters a_j values. □

The $1|p_j(u_j)|\, L_{\max}$ problem

As is known (e.g. [3]), the $1|p_j(u_j)|\, L_{\max}$ problem may be treated as the inverse

version of the $1|r_j, p_j(u_j)| C_{\max}$ problem. Thus the $1|p_j(u_j)| L_{\max}$ problem is solved (in $0(n^2)$ steps) by ordering the jobs according to non-decreasing d_i and by allocation of resource generated by the procedure analogical to that from Property 2.

The $1|p_j(u_j) = b - au_j| c_{\max}$ problem

Property 3. If all models of operations are of the same form: $p_j(u_j) = b - au_j$, $0 \leqq u_j \leqq \beta_j$, $j = 1, 2, \ldots, n$, then the following procedure generates (in $0(n^2)$ steps) the optimal solution to the $1|p_j(u_j) = b - au_j| c_{\max}$ problem.

Procedure 3. From among all jobs, i.e. from the set $S := \{1, 2, \ldots, n\}$, put the job J_h with the smallest value $c_h(\sum p)$ in the last position, where at the first step $\sum p :=$ $:= n \cdot b - a \cdot \min\{\sum_{j=1}^{u} \beta_j, \hat{U}\}$, and assign to J_h the resource amount

$$u_h^* = \min\{\beta_j, \max\{0, \hat{U} - \sum_{j \in S - \{h\}} \beta_j\}\}.$$

Then repeat on the set $S := S - \{h\}$ with $\hat{U} := \hat{U} - u_h^*$, $\sum p := \sum p - (b - au_h^*)$, $n := n - 1$, and so on.

The proof follows immediately from Theorem 1 in [2] and the fact that for any global amount of resource being fixed the optimal resource allocation is the same for any permutation (i.e. $u_{\pi(i)}^* = u_{\pi'(i)}^*$, $i = 1, 2, \ldots, n$, $\pi, \pi' \in \Pi$).

The $1|p_j(u_j)| \sum w_j C_j$ problem

For the $1|p_j(u_j)| \sum w_j C_j$ problem a polynomial-time algorithm (with $0(n \cdot \log n)$ complexity) may be found under a strong assumptions on the job parameters. However, the computational complexity of this problem in the general case remains an open question. The following property holds independently of the computational complexity of this problem:

Property 4. The problem $1|p_j(u_j) \sum C_j$ (i.e. for $w_j = 1$, $j = 1, 2, \ldots, n$) has the same optimal solution (i.e. processing order and resource allocation) as that obtained by replacing the processing times $p_j = b_j - a_j u_j$ by $p_j' = p_j + c$, where $c \geqq -$ $- \min_{1 \leqq j \leqq n} (b_j - a_j \cdot \beta_j)$ is a fixed constant.

Proof. Consider a permutation π with a resource allocation $\bar{u} \in U$; the completion time $C_{\pi(i)}(\pi, \bar{u})$ of the job $J_{\pi(i)}$ is given by $C_{\pi(i)}(\pi, \bar{u}) = \sum_{j=1}^{i} p_{\pi(j)}(u_{\pi(j)})$. Replacing $p_{\pi(j)}(u_{\pi(j)})$ by $p_{\pi(j)}(u_{\pi(j)}) + c$ the completion time is given by $C'_{\pi(i)}(\pi, \bar{u}) =$ $= \sum_{j=1}^{i} (p_{\pi(j)}(u_{\pi(j)}) + c) = C_{\pi(i)}(\pi, \bar{u}) + i \cdot c$. Hence the new criterion value $\sum C_i'$ is related to the old one $\sum C_i$ by $\sum C_i' = \sum_{i=1}^{n} C'_{\pi(i)}(\pi, \bar{u}) = \sum C_i + \frac{1}{2} n \cdot (n + 1) \cdot c$. □

3. REMARKS

It is easy to verify that the proposed algorithms solving the problems: $1|p_j(u_j)| C_{\max}$, $1|r_j, p_j(u_j)| C_{\max}$, $1|p_j(u_j)| L_{\max}$, $1|p_j(u_j) = b - au_j| c_{\max}$, may be generalized to the case with given precedence constraints.

Separate elaboration is desired for other one-machine scheduling problems with allocation of resource, e.g., for problems with such criteria as: the total cost, the weighted sum of tardinesses and the weighted sum of completion times and under precedence constraints. Most of them are probably NP-hard (see e.g. [3] for definition). Especially interesting seems to be this subset of the above NP-hard problems for which classical equivalents (without resource) have polynomial-time algorithms.

ACKNOWLEDGEMENT

This research was partially supported by the scientific program R.P.I. 02: "Theory of Control and Optimization of Continuous Systems and Discrete Processes".

(Received July 30, 1985.)

REFERENCES

[1] J. R. Jackson: Scheduling a Production Line to Minimize Maximum Tardiness. Research Report, University of California at Los Angeles 1955.

[2] L. Lawler: Optimal sequencing of a single machine subject to precedence constraints. Management Sci. 19 (1973), 544—546.

[3] H. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan: Recent developments in deterministic sequencing and scheduling: a survey. In: Deterministic and Stochastic Scheduling, (M. A. H. Dempster, J. K. Lenstra and A. H. G. Rinnooy Kan, eds.), Dordrecht 1982.

Dr. Adam Janiak, Technical University of Wroclaw, Institute of Engineering Cybernetics, ul. Wybrzeże Wyspiańskiego 27, 50 370 Wroclaw. Poland.