MANAGEMENT SCIENCE
Vol. 32, No. 3, March 1986
Printed in U.S.A.

PREEMPTIVE SCHEDULING OF UNIFORM MACHINES BY ORDINARY NETWORK FLOW TECHNIQUES*

A. FEDERGRUEN AND H. GROENEVELT
Graduate School of Business, Columbia University, New York, New York 10027
Graduate School of Management, University of Rochester, Rochester, New York 14627

We consider the problem of scheduling n jobs, each with a specific processing requirement, release time and due date on m uniform parallel machines. It is shown that a feasible schedule can be obtained by determining the maximum flow in a network, thus permitting the use of standard network flow codes. Using a specialized maximum flow procedure, the complexity reduces to $O(tn^3)$ operations when t is the number of distinct machine types. Previous algorithms solve the feasibility problem in $O((m + \log n)(m^2n^3 + n^4))$ operations. In addition to the feasibility problem, we describe algorithms for the maximum lateness criterion. Here we develop a bound which compares even more favorably to the best previous bound. We also show how other criteria with respect to the amount of work completed on each job prior to its due date or the amount of work scheduled in each of a sequence of periods can be optimized by similar path augmenting techniques.
(PREEMPTIVE SCHEDULING; POLYMATROIDAL NETWORK FLOW; NETWORK FLOW)

1. Introduction and Summary

We consider the problem of scheduling n jobs, each with a specific processing requirement, release time and due date on m parallel machines. The machines are assumed to be uniform in the sense that they merely differ in processing speed. Machines can work on only one job at a time and each job can be processed by at most one machine at a time; however preemptions are allowed. The feasibility problem consists of determining a feasible schedule (if one exists). In this paper we show that the feasibility problem can be solved as a classical maximal flow problem. A specialized maximal flow algorithm by Gusfield et al. (1985) allows for a complexity bound of $O(tn^3)$ where t is the number of different machine types (speeds).

We also show how various performance measures may be optimized by similar path augmenting algorithms applied to an appropriately chosen network. These include:

(a) *the maximum lateness problem*: minimize the maximum lateness (= completion time − due date) over all jobs, in case no feasible schedule exists. This problem may be solved by verifying the existence of a feasible schedule for at most $O(\log n + \log p_{\max} + \log s_1)$ lateness values where $p_{\max}$ denotes the maximum integer processing time and s_1 the largest integer machine speed. This results in an $O(tn^3(\log n + \log p_{\max} + \log s_1))$ algorithm.

(b) *the (weighted) minimum completion problem*: For each job, define the completion rate as the fraction of work completed prior to its due date. Find a schedule which maximizes the (weighted) minimum completion rate. Preemptive scheduling models apply e.g. to batch production systems where each batch consists of a large number of units. The minimum completion criterion is particularly relevant for such applications especially when due dates are unextendable. If a feasible schedule completing all jobs fails to exist, a fair rationing scheme is sought by allocating the scarce productive capacity so as to fill as large a (weighted) percentage of as many jobs (orders) as

* Accepted by Alexander H. G. Rinnooy Kan; received May 1984. This paper has been with the authors 4 months for 2 revisions.

0025-1909/86/3203/0341$01.25

possible. A path augmenting algorithm applied to an appropriately chosen network solves the problem in $O(tn^3 p_{\max})$ operations.

(c) *the* (*weighted*) *maximum utilization problem*: For a given collection of disjoint time-intervals, find a feasible schedule which minimizes the (weighted) maximum amount of work assigned to each of these intervals. This criterion is particularly relevant when attempting to create a maximum amount of slack capacity in each of the considered time-intervals, e.g. as a buffer against underestimated processing times or unanticipated "last-minute" jobs. (See §4 for additional applications.) A path augmenting algorithm (again applied to a slightly modified network) solves the problem in $O(tn^3 p_{\max})$ operations.

For the special case of identical machines, our feasibility algorithm reduces to the network flow procedure of Horn (1974) which determines a feasible schedule in $O(n^3)$ operations. Labetoulle et al. (1979) have shown that this feasibility algorithm can be used in a $O(n^3 \min\{n^2, \log n + \log p_{\max}\})$ procedure to resolve the lateness problem using binary search on the optimal value of the maximum lateness. Gonzalez and Johnson (1980) derived an $O(mn)$ procedure for the lateness problem with arbitrary release times but *identical* machines and due dates. Sahni and Cho (1980) deal with the special case of common due dates (but uniform machines); their feasibility algorithm requires $O(n \log n + mn)$ operations. Sahni and Cho (1979) and Labetoulle et al. (1979) propose an $O(n \log n + mn)$ procedure for the lateness problem with identical release and due dates. Gonzalez and Sahni (1978) have an $O(n + m \log m)$ procedure for the feasibility problem with identical release and due dates. Bruno and Gonzalez (1976) and Labetoulle et al. (1979) also deal with the two-machine model.

For the general problem with arbitrary machine speeds, processing requirements, due and release dates, the first polynomial algorithm is due to Martel (1982a, b) who showed that a feasible schedule may be determined by solving a polymatroidal network flow problem, cf. Lawler and Martel (1982), an extension of the classical maximal flow problem in which the flows are constrained by capacities of *sets* of arcs in addition to capacities on individual arcs. The resulting algorithm solves the feasibility problem in $O((m + \log n)(m^2 n^3 + n^4))$ operations. The same feasibility algorithm can also be used for the maximum lateness problem, performing a binary search over possible lateness values. The resulting maximum lateness algorithm terminates in $O((m + \log n)(m^2 n^5 + n^6))$ operations. The complexity bounds of our algorithms compare favorably with Martel's. The fact that our formulation permits the use of standard network flow codes is, of course, an additional practical advantage.

The minimum completion and the maximum utilization criteria have, to our knowledge, not been studied in the scheduling literature. We refer to Lawler et al. (1982) for an excellent survey of the literature on alternative criteria (such as the makespan, weighted sum of completion times and due date violation penalties) as well as the complications that arise due to precedence constraints among the jobs.

§2 describes our feasibility algorithm and §3 discusses the maximum lateness problem. Alternative criteria are discussed in §4.

2. Solution of the Feasibility Problem

We first introduce some notation. Each job $j = 1, \ldots, n$ is specified by a processing requirement p_j, a release time r_j and a due date d_j. There are t types of machines with speeds $s_1 > s_2 > \cdots > s_t$ and cardinalities $m_1, \ldots, m_t$; thus $m = \sum_{i=1}^{t} m_i$. (A job requiring p units of work can be processed on machine i in p/s_i units of time.) All parameters are assumed to be integer.

Rank the release and due dates in ascending order and determine the (at most $2n - 1$) intervals between consecutive milestones. Let τ_i be the starting time of the ith interval (i.e. τ_i is the ith smallest value of the release and due dates). A job j is available for processing (at time t) if $r_j \leqslant t$ and $d_j \geqslant t$. Note that within each interval the set of available jobs does not change. A feasible schedule can now be constructed with the following two-stage procedure: first determine the amount of work to be performed on each job in each interval; next, construct a feasible schedule for each interval separately given the amount of work to be processed on each of the available jobs. Observe that scheduling the jobs within an interval is an instance of the special case of our problem where all release and due dates are identical and the Gonzalez–Sahni algorithm (1979) solves this problem in $O(m \log m + n)$ operations.

Determination of the amount of work to be performed on each job in each interval is facilitated by the following result in Horvath, Lam and Sethi (1977): Let S_l represent the cumulative speed of the l fastest machines:

$$S_l = \begin{cases} l s_1, & l \leqslant m_1, \\ \sum_{i=1}^{r} m_i s_i + \left(l - \sum_{i=1}^{r} m_i\right) s_{r+1}, & \sum_{i=1}^{r} m_i < l \leqslant \sum_{i=1}^{r+1} m_i; \quad r = 1, \dots, t-1, \\ S_m, & l > m \end{cases}$$

Let y_j denote the amount of work to be processed on job j in a given interval of length T. A feasible schedule exists for this interval if and only if the following set of constraints is satisfied:

$$\sum_{j \in A} y_j \leqslant T S_{|A|}, \qquad A \subset \{1, \dots, n\}. \tag{1}$$

(In other words, the sum of the l largest y_j values needs to be less than or equal to TS_l. The necessity of (1) is immediate; for the sufficiency see Horvath et al. 1977.)

Consider now the following tripartite network (N, E) with an extra source O and sink O'. The first level has a node for every job (*job nodes*), the second level has a node for every period and machine type (*machine-period nodes*), and the third level has a node for every period (*period nodes*). The source is connected with each job node, the arc to the jth node having capacity p_j, $j = 1, \dots, n$. A job node is connected to all machine-period nodes for all periods during which the job is available. All arcs which lead to a machine-period node corresponding to machine type r and an interval of length T have capacity $(s_r - s_{r+1})T$, with the convention $s_{t+1} = 0$. Every node (r, i)

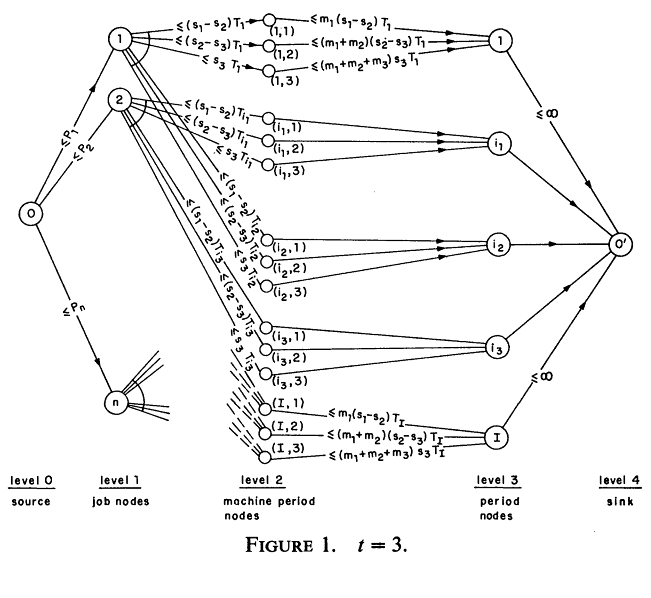

FIGURE 1. $t = 3$.

corresponding to machine type r and period i of length T_i is connected with the period node i and has capacity $(\sum_{n=1}^{r} m_n)(s_r - s_{r+1})T_i$, see Figure 1. Finally, all period nodes are connected to the sink with uncapacitated arcs.

LEMMA. *A feasible schedule exists if and only if the maximum flow in the network* (N, E) *equals* $\sum_j p_j$.

PROOF. Fix $\mathbf{f}$, a maximal flow in the network and assume the flow value equals $\sum_j p_j$. Let x_{ij} denote the flow from job node j to period node i. We show

$$\sum_{j \in A} x_{ij} \leqslant T_i S_{|A|}, \qquad A \subset \{1, \ldots, n\}; \quad \text{all } i, \tag{2}$$

so that, in view of (1), a feasible schedule exists. Thus, fix i and let J be the set of job nodes, and I the set of machine-period nodes corresponding to period i. Consider the subnetwork comprised by $J \cup I \cup \{i\}$ as well as the arcs between them. Also reverse the direction of the arcs, so that period node i appears as the source and the job nodes as sinks (see Figure 2). Since the restriction of $\mathbf{f}$ to the subnetwork is feasible, it follows for all $A \subset J$, that $\sum_{j \in A} x_{ij}$ is bounded by the maximum flow from period node i to the job nodes in A. The maximum flow through machine-period node (i, r) is bounded by (see Figure 2):

$$\begin{array}{ll} (m_1 + \cdots + m_r)(s_r - s_{r+1})T_i, & \text{if} \quad m_i + \cdots + m_r < |A|, \\ |A|(s_r - s_{r+1})T_i, & \text{otherwise.} \end{array}$$

Thus,

$$\sum_{j \in A} x_{ij} \leqslant T_i \sum_{r=1}^{t} \min\left\{ |A|; \sum_{p=1}^{r} m_p \right\}(s_r - s_{r+1}) = T_i S_{|A|}.$$

(The last identity can be shown by complete induction on $|A|$.) This verifies (2) and the sufficiency part of the lemma.

Conversely, for a feasible schedule, let x_{ij} denote the amount of work to be performed on job j in period i. In view of (1), (2) holds. Similar applications of the max flow min cut theorem verify the existence of a feasible flow in the network (N, E) with x_{ij} the total flow from job node j to period node i (see Lemma 4.1 in Megiddo (1974) for details). Q.E.D.

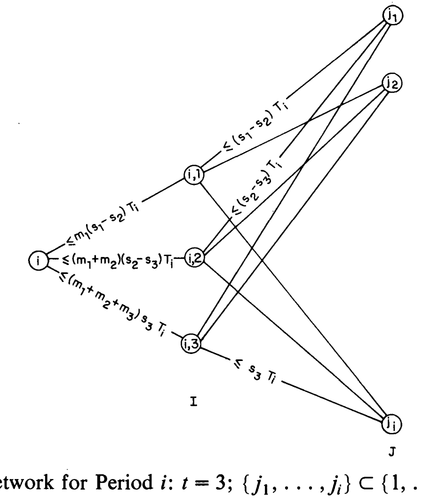

FIGURE 2. Reversed Subnetwork for Period i: $t = 3$; $\{j_1, \ldots, j_i\} \subset \{1, \ldots, n\}$ Set of Jobs Available in Period i.

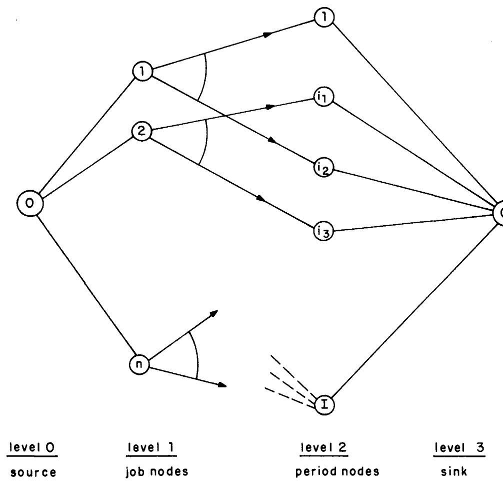

FIGURE 3. Polymatroidal Network.

Our network flow model is closely related to the one suggested by Martel (1982a, b). Martel's network avoids the level of machine-period nodes but imposes the upper bounds (2) on the collection of arcs pointing to the same period node, see Figure 3. Our expanded network can thus be viewed as a device to avoid the upper bounds on *sets* of arcs as given by (2). General transformations of this type are discussed in Federgruen and Groenevelt (1984b). Observe that our network has $O(tn)$ nodes and $O(tn^2)$ arcs. Classical maximal flow algorithms, e.g. Malhotra et al. (1978) thus solve the feasibility problem in $O(t^3n^3)$ operations, a significant reduction compared to the $O((m + \log n)(m^2n^3 + n^4))$ bound in Martel's procedure. (Note also that, in general, $t \ll m$.) A further reduction in the complexity bound to $O(tn^3)$ may be achieved by applying Gusfield et al.'s (1985) maximal flow procedure for multipartite gr^phs. In addition, our procedure allows for the use of standard maximal flow codes or even standard primal simplex network codes. (For a discussion of the practical advantages of the latter, see Glover et al. 1984a, b.)

The feasibility algorithm can be used to solve several generalizations of the considered problem. For example, each job may have a list of time intervals within which it can be processed (rather than a single such interval) and the machines may be available in certain time intervals only. We can divide the time line into a sequence of consecutive intervals such that within each interval the jobs and machines which are available, do not change. A network flow problem (constructed as above) solves the feasibility problem.

Note that for the special case of unit processing times, the solution generated by any maximum flow algorithm avoids preemptions, hence is optimal in a nonpreemptive setting as well. (With integer speeds, release and due times, an integer solution is optimal.) Our feasibility algorithm solves this case in $O(tn^3/m)$ operations. (One easily verifies, cf. Lemma 2 in Simons and Sipser 1984 that the total length of all time intervals within which a specific job can be run, may be bounded by $\lfloor n/m \rfloor \leqslant n/m + 1$. Each job node is thus connected with the machine-period nodes corresponding to at most $(n/m + 1)$ periods. As $n \geqslant m$ without loss of generality, the number of arcs in the network is thus $O(tn^2/m)$ and since the maximum flow is bounded by n, the complexity bound $O(tn^3/m)$ follows immediately, see Lawler 1976, p. 116.) Simons and Sipser (1984) treated the even more restricted case of identical machines ($t = 1$). Horn's original maximal flow algorithm (to which ours reduces) thus exhibits superior performance compared to their $O(n^3)$ matching algorithm. Our algorithm, in addition, solves the extension of the Simons and Sipser problem to the case of uniform machines.

3. The Maximum Lateness Problem

The lateness of a job is defined as the completion time − due date. The maximum lateness problem consists of finding a schedule which minimizes the maximum lateness over all jobs. As pointed out by Horn (1974), Labetoulle et al. (1979) and Martel (1982a, b), to solve this problem, we find the smallest value L, such that adding L to all due dates results in a feasible scheduling problem. This smallest value L can be found in two stages. First note that there are at most $n(n-1)$ critical values of L such that $d_i + L = r_j$ for some release time and due date. For values of L which lie between a pair of consecutive critical values, the same network topology describes the feasibility problem. Performing a binary search, first, over the $O(n^2)$ critical values of L, thus narrows the search to an interval $[L_0, L_1]$ such that the same network topology applies to the feasibility problem for all lateness values $L \in [L_0, L_1]$. This stage requires at most $O(\log n)$ calls to the feasibility routine. From this point on, several alternatives prevail.

Continued binary search over the interval $[L_0, L_1]$ achieves any ϵ-optimal solution in at most $O(|\log \epsilon|)$ calls to the feasibility routine. An exact optimal schedule can in fact be generated in at most $O(\log s_1 + \log n + \log p_{\max})$ calls to the feasibility routine, as can be shown following an argument in Labetoulle et al. (1984) for the case of identical machines. Consider the network prevailing when the lateness $L \in [L_0, L_1]$. A cut in this network has capacity $C_f + (L - L_0)C_v$, where C_f and C_v are integers. From Figure 1 we find by adding the capacities of all arcs to and from the machine-period nodes that $|C_v| = O(n^2 s_1 + n\sum m_i s_i) \leqslant O(n^2 s_1 + nms_1) = O(n^2 s_1)$, since $m < n$. Since the minimum lateness has the form $L^* = L_0 + (P - C_f)/v$, where $P = \sum p_j$ is the total processing requirement, L^* can be found by conducting a bisection search until the remaining interval of uncertainty (L', L'') is no longer than $(n^4 s_1^2)^{-1}$ which is an upperbound on the difference of fractions of the form $(P - C_f)/C_v$, when $|C_v| \leqslant n^2 s_1$. If $C_f' + (L' - L_0)C_v'$ is the capacity of a minimal cut for lateness L^*, then $L = L_0 + (P - C_f')/C_v'$. The bisection search can of course be performed in $O(\log(Ps_1^2 n^4)) = O(\log s_1 + \log n + \log p_{\max})$ time. This substantially improves Martel's bound of $O(n^2 + n\log s_1 + \log(d_{\max} + P))$ for the number of calls to the feasibility routine.

As an alternative for the second stage, note that a lateness value $L \in [L_0, L_1]$ affects the induced network flow problem only through the capacities on some of the arcs and the dependence is linear. Thus, an alternative procedure for determining the maximum lateness is to solve a linear program of the form:

$$\begin{aligned}
&\min L \\
&\text{s.t.} \quad \sum_{\{l \in E:\ 0 \text{ tail of } l\}} x_l = \sum_{j=1}^{n} p_j, \qquad \text{(P)} \\
&\qquad \sum_{\{l \in E:\ \iota \text{ head of } l\}} x_l - \sum_{\{l \in E:\ \iota \text{ tail of } l\}} x_l = 0, \qquad \iota \in N \setminus \{0\}, \\
&\qquad x_l \leqslant \alpha_l + \beta_l L, \qquad L_0 \leqslant L \leqslant L_1 .
\end{aligned}$$

Note that (P) is a minimum cost network flow model with one extra variable. Simple network simplex codes solve this problem as efficiently as pure network problems using elementary basis partitioning techniques, see e.g. Glover and Klingman (1981).

The above approaches may again be applied to the case where jobs (machines) have a list of time-intervals within which they are available. For the case of unit processing times our procedure has an $O(t(n^3/m)(\log n + \log s_1))$ complexity bound. (The procedure in Simons and Sipser, applicable for identical machines only, requires $O(n^4)$ operations.)

4. Alternative Criteria

In this section we show how various performance measures may be optimized by path augmenting algorithms applied to an appropriate modification of the network in Figure 1. Consider a network $(\overline{N}, \overline{E})$, with a single source s, let $T \subset \overline{N}$ and let $(z_j)_{j \in T}$ be the vector of net inflows into the nodes of T. The following algorithm maximizes any separable concave objective function in $(z_j)_{j \in T}$, see Federgruen and Groenevelt (1984a). (The same algorithm determines an optimal solution for certain classes of "concave" *nonseparable* functions as well, see ibid.)

This procedure is a generalization of the well-known augmenting path algorithms for the maximal flow problem. In each iteration, labels are given to nodes of the form ι^+ or ι^- where $\iota \in \overline{N}$. (Only the source s has a special label $-$.) A label ι^+ $[\iota^-]$ indicates that there exists a unit size augmenting path from the source to node η in question and that (ι, η) $[(\eta, \iota)]$ is the last arc in this path. Let e^i be the ith unit basis vector in R^T. For any given feasible vector $z = (z_j)_{j \in T}$ of supplies to the 0-level nodes T, $z + e^j$ is feasible if and only if the labeling procedure succeeds in labeling node j, see Federgruen and Groenevelt (1984a). For all $l \in \overline{E}$, let x_l represent the flow on this arc and let u_l be the arc's capacity:

```
CAPA: (Concave Objective Augmenting Path Algorithm)
Step 1. For $j = 1, \dots, n$ do $z_j := 0$; for $l \in \overline{E}$ do $x_l := 0$;
Step 2. Give the source a special label $-$.
Step 3. If all labeled nodes have been scanned, go to Step 5.
Step 4. Fix a labeled but unscanned node $\iota$ and scan it as follows:
    if $l = (\iota, \eta) \in \overline{E}$, $x_l < u_l$ and $\eta$ unlabeled, give $\eta$ the label $\iota^+$;
    if $l = (\eta, \iota) \in \overline{E}$, $x_l > 0$ and $\eta$ unlabeled, give $\eta$ the label $\iota^-$;
    go to Step 3.
Step 5. Find a node $j$ with $j \in T$ such that $j$ is labeled and $f(z + e^j) \geqslant f(z + e^k)$
    for all labeled nodes $k$ $(1 \leqslant k \leqslant n)$.
Step 6. If (no such node $j$ exists) or $f(z + e^j) < f(z)$ then stop.
Step 7. Starting at node $j$, backtrack an augmenting path; for a node $\eta$ on this path
    with label $\iota^+$ $(\iota^-)$ increase (decrease) $x_{\iota\eta}$ $(x_{\eta\iota})$ by one; set $z_j := z_j + 1$; erase all labels
    and return to Step 2.
```

Completion Criteria

Completion criteria are defined with respect to the vector $\{z_j, j = 1, \dots, n\}$ where z_j represents the amount of work performed on job j prior to its due date. Consider the following modification of the network (N, E). At the zeroth level of the network, eliminate the source and its outgoing arcs and add a node j' for every $j = 1, \dots, n$. Connect node j' with job node j by an arc with capacity p_j, see Figure 4. Reverse the direction of all arcs and let $(\overline{N}, \overline{E})$ be the resulting expanded network. This network has node O' as its unique source and the set of nodes $T = \{j' : j = 1, \dots, n\}$ as the set of sinks. Note that z_j represents the inflow into sink j'. Note that CAPA requires no more than $\sum_j p_j \leqslant np_{\max}$ iterations through Steps 2–7 and each iteration requires scanning of $|\overline{E}| = O(tn^2)$ arcs; hence the complexity bound $O(tn^3 p_{\max})$.

The function $g(z) = \min_j z_j / p_j$ (the minimum completion criterion discussed in the introduction) is not separable but an optimal solution may be obtained by optimizing the more selective criterion of lexicographic maximization of $T(z)$, the vector of

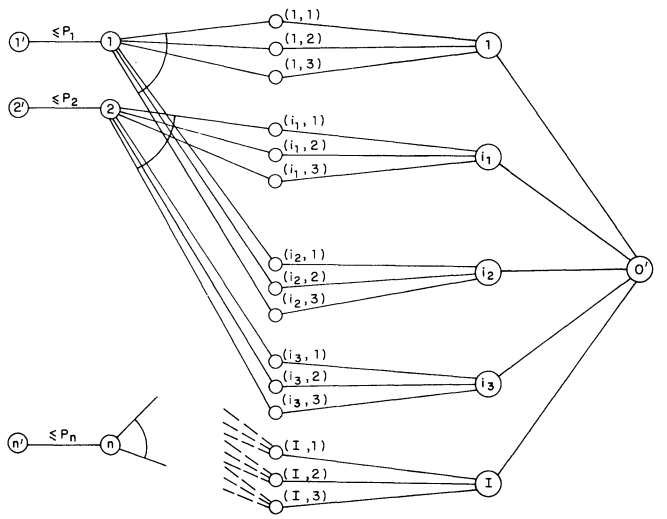

FIGURE 4. Expanded Network $(\bar{N}, \bar{E})$ for Completion Criteria: $(t = 3)$.

completion rates $\{z_j/p_j\}$ ranked in ascending order. Fujishige (1980) has shown that an optimal solution with respect to the latter criterion may be obtained by maximizing $f(z) = \sum_j z_j - \frac{1}{2} z_j^2/p_j$ which, of course, is separable and concave.

CAPA applied to the network $(\bar{N}, \bar{E})$ in Figure 4 thus results in a schedule which maximizes the minimum completion criterion. Similarly for given positive weights $\{w_j, j = 1, \ldots, n\}$ the minimum weighted completion criterion, $g_w(z) = \min_j w_j z_j / p_j$ may be maximized by applying CAPA with the objective function

$$f_w(z) = \sum_j \left\{ \left(\sum_k w_k \right) z_j - \tfrac{1}{2} w_j z_j^2 / p_j \right\}.$$

An algorithm in Fujishige (1980) solves the problem in $O(t^3 n^4)$.

Utilization Criteria

Let $\{I_1, \ldots, I_K\}$ be a collection of disjoint time intervals and let y_l be the amount of work scheduled for I_l. Utilization criteria are defined with respect to the vector $\{y_l, l = 1, \ldots, K\}$. Consider the following modification of the network (N, E) in Figure 1. First, insert the starting and ending times of the intervals $\{I_l, l = 1, \ldots, K\}$ in the sorted list of release and due dates and determine the (at most $2n + 2K - 1$) periods between consecutive milestones. Introduce a period node for each such period and use the list of period nodes and the corresponding list of machine-period nodes in levels 2 and 3 of the network (N, E), see Figure 1. Also replace level 4 by a string of interval nodes (one for each I_l). A period node i is connected with interval node l (by an infinite capacity arc) if the period is part of the interval.

The function $g(y) = \max_l y_l$ (the maximum utilization criterion discussed in the introduction) is not separable but its minimum may be obtained by lexicographic maximization of $T(y)$, the vector $\{y_l, l = 1, \ldots, K\}$ ranked in ascending order, see Ichimori et al. (1982). As above, it follows in view of Fujishige (1980) that the maximum utilization criterion may be optimized by applying CAPA with $f(y) = \sum_l \{(\sum_k p_k) y_l - \frac{1}{2} y_l^2\}$. (More generally, for given positive weights $\{w_l, l = 1, \ldots, K\}$, the function $g(y) = \max_l w_l y_l$ may be minimized by applying CAPA with the function $f_w(y) = \sum_l \{(\sum_k p_k)(\sum_j w_j) y_l - \frac{1}{2} w_l y_l^2\}$.)

References

BRUNO, J. AND T. GONZALEZ, "Scheduling Independent Tasks with Release Dates and Due Dates in Parallel Machines," Technical Report 213, Computer Science Dept., Pennsylvania State University, 1976.

FEDERGRUEN, A. AND H. GROENEVELT, "Optimal Flows in Networks with Multiple Sources and Sinks: Applications to Oil and Gas Lease Investment Programs," *Oper. Res.* (to appear).

——— AND ———, "Optimal Polymatroidal Network Flows with Multiple Sources and Sinks," (1984b) (in preparation).

FUJISHIGE, S., "Lexicographically Optimal Base of a Polymatroid with Respect to a Weight Vector," *Math. Oper. Res.*, 5 (1980), 186–196.

GLOVER, F. AND D. KLINGMAN, "The Simplex SON Algorithm," *Math. Programming Stud.*, 15 (1981), 148–176.

———, ———, J. MOTE AND D. WHITMAN, "A Primal Simplex Variant for the Maximum Flow Problem," *Naval Res. Logist. Quart.*, 31 (1984a), 41–63.

———, ———, M. MEAD AND J. MOTE, "A Note on Specialized Versus Unspecialized Methods for Maximum-Flow Problems," *Naval Res. Logist. Quart.*, 31 (1984b), 63–67.

GONZALEZ, T. AND D. JOHNSON, "A New Algorithm for Preemptive Scheduling of Trees," *J. Assoc. Comput. Mach.*, 27 (1980), 287–312.

——— AND S. SAHNI, "Preemptive Scheduling of Uniform Processor Systems," *J. Assoc. Comput. Mach.*, 25 (1978), 92–101.

GUSFIELD, D., C. MARTEL AND D. FERNANDEZ-BACA, "Fast Algorithms for Bipartite Network Flow," Yale University, Department of Computer Science Working Paper, 1985.

HORN, W., "Some Simple Scheduling Algorithms," *Naval Res. Logist. Quart.*, 21 (1974), 177–185.

HORVATH, E., S. LAM AND R. SETHI, "A Level Algorithm for Preemptive Scheduling," *J. Assoc. Comput. Mach.*, 25 (1977), 32–43.

ICHIMORI, T., H. ISHII AND T. NISHIDA, "Optimal Sharing," *Math. Programming*, 23 (1982), 341–348.

LABETOULLE, J., E. LAWLER, J. K. LENSTRA AND A. RINNOOY KAN, "Preemptive Scheduling of Uniform Machines Subject to Release Dates," in Proc. Silver Jubilee Conf. on Combinatorics, University of Waterloo, Waterloo, Canada, 1984.

LAWLER, E., *Combinatorial Optimization: Networks and Algorithms*, Holt, Rinehart and Winston, New York, 1976.

———, J. K. LENSTRA AND A. RINNOOY KAN, "Recent Developments in Deterministic Sequencing and Scheduling: A Survey," in M. Dempster et al. (Eds.), *Deterministic and Stochastic Scheduling*, D. Reidel, Dordrecht, 1982, 35–73.

——— AND C. MARTEL, "Computing Maximal Polymatroidal Network Flows," *Math. Oper. Res.*, 7 (1982), 334–347.

MALHOTRA, V., M. KUMAR AND S. MAHESHIVARI, "An $O(|V|^3)$ Algorithm for Finding Maximum Flows in Networks," *Inform. Process. Lett.*, 7 (1978), 277–278.

MARTEL, C., "Scheduling Uniform Machines with Release Times, Deadlines and Due Times," in M. Dempster et al. (Eds.), *Deterministic and Stochastic Scheduling*, D. Reidel, Dordrecht, 1982a, 89–99.

———, "Scheduling Uniform Machines with Release Times, Deadlines and Due Times," *J. Assoc. Comput. Mach.*, 29 (1982b), 812–829.

MEGIDDO, N., "Optimal Flows in Networks with Multiple Sources and Sinks," *Math. Programming*, 7 (1974), 97–107.

SAHNI, S. AND Y. CHO, "Nearly on Line Scheduling of a Uniform Processor System with Release Times," *SIAM J. Comput.*, 8 (1979), 275–285.

——— AND ———, "Scheduling Independent Tasks with Due Times on a Uniform Processor System," *J. Assoc. Comput. Mach.*, 27 (1980), 550–563.

SIMONS, B. AND M. SIPSER, "On Scheduling Unit-Length Jobs with Multiple Release Time/Deadline Intervals," *Oper. Res.*, 32 (1984), 80–89.