MATHEMATICS OF OPERATIONS RESEARCH
Vol. 4, No. 3, August 1979
Printed in U.S.A.

SEQUENCING WITH SERIES-PARALLEL PRECEDENCE CONSTRAINTS*

CLYDE L. MONMA†§ AND JEFFREY B. SIDNEY‡

One of the most important ideas in the theory of sequencing and scheduling is the method of adjacent pairwise job interchange. This method compares the costs of two sequences which differ only by interchanging a pair of adjacent jobs. In 1956, W. E. Smith defined a class of problems for which a total preference ordering of the jobs exists with the property that in any sequence, whenever two adjacent jobs are *not* in preference order, they may be interchanged with no resultant cost increase. In such a case the unconstrained sequencing problem is easily solved by sequencing the jobs in preference order.

In this paper, a natural subclass of these problems is considered for which such a total preference ordering exists for all subsequences of jobs. The main result is an efficient general algorithm for these sequencing problems with series-parallel precedence constraints. These problems include the least cost fault detection problem, the one-machine total weighted completion time problem, the two-machine maximum completion time flow-shop problem and the maximum cumulative cost problem.

Introduction. In this paper we introduce a class of problems based on easily verifiable properties. This class includes many widely-studied problems; for example, Johnson's [10] classical two-machine maximum completion time flow-shop problem, the one-machine total weighted completion time problem, the maximum cumulative cost problem and the least cost fault detection problem. An efficient general algorithm is derived for solving these problems with series-parallel precedence constraints. This unifies the recent results by Sidney [30], Lawler [16] and Abdel-Wahab and Kameda [2], [3], and extends a result by Garey [7].

1. Basic definitions and notation.

A *job* is the basic unit of work to be sequenced. Each job is characterized by certain parameters; for example, in the total weighted completion time problem a job is specified by a weighting factor and a nonnegative processing time. We assume given a set $J = \{1, 2, \dots, n\}$ of jobs to be sequenced. A *sequence* s of k jobs is a function from $\{1, 2, \dots, k\}$ to J and is represented by $(s(1), s(2), \dots, s(k))$, where $s(i)$ is the ith job in the sequence s. A job may appear several times, or not at all, in a sequence. Multiple occurrences of the same job are considered to be duplicate copies of that job, i.e., they have the same parameter values. Sequences are denoted by the lower case letters s, t, u, v and w. A *permutation* of J is denoted by Π. A *sequencing* or *cost function* assigns a real value or cost to each sequence.

A *sequencing problem*, on a set of jobs J with cost function f, is to find a permutation of J contained in a set of feasible permutations F which minimizes f, i.e.,

$$\underset{\Pi \in F}{\text{minimize}}\, f(\Pi).$$

The problem is called *unconstrained* if all permutations are feasible.

The first type of constraint considered is called a *precedence constraint*. A job i is said to have *precedence* over job j, written $i \rightarrow j$, if job i must occur before job j in

* Received September 6, 1977; revised September 18, 1978.
AMS 1970 subject classification. Primary 90B35.
IAOR 1973 subject classification. Main: Scheduling.
Key words. Scheduling, sequencing, series-parallel precedence constraints, unified algorithm, one and two machine cases, least-cost fault detection.

† Cornell University.

§ Monma's work was partially supported by NSF Grant ENG 76-09936.

‡ University of Ottawa.

0364-765X/79/0403/0215$01.25

every feasible permutation. As an example, it is usually considered good practice to perform the job of recording customers' payments before the job of sending out the next month's bills in order to prevent duplicate billing (see [8, chapter 13]). These precedence constraints will be represented by a directed *precedence graph* $G = (J, R)$, where the nodes of G are given by the jobset J and an arc directed from i toward j, denoted by $\langle i, j \rangle$, is contained in the set R if and only if $i \to j$.

A *subgraph* $G' = (J', R')$ of $G = (J, R)$ *induced* by $J' \subseteq J$ has $R' = \{\langle i, j\rangle \in R : i, j \in J'\}$. $J' \subseteq J$ is a *job module* of $G = (J, R)$ if for every job $k \in J - J'$ either

(a) $k \to i$ for all $i \in J'$, or

(b) $i \to k$ for all $i \in J'$, or

(c) $k \not\to i$ and $i \not\to k$ for all $i \in J'$.

The term job module will also be used to refer to the subgraph induced by J'. Precedence graphs $G_1 = (J_1, R_1)$ and $G_2 = (J_2, R_2)$ with $J_1 \cap J_2 = \emptyset$ are in *series* if $i \to j$ for every $i \in J_1$ and $j \in J_2$, and are in *parallel* if $i \not\to j$ and $j \not\to i$ for all $i \in J_1$ and $j \in J_2$. A precedence graph is called a *chain* if exactly one permutation is feasible. The term chain will also be used for the set of jobs making up the graph and the unique feasible permutation of the jobs. A *parallel-chains* precedence graph is either a chain, or a chain in parallel with a parallel-chains precedence graph. A precedence graph is called *series-parallel* if it can be reduced to a chain by the recursive replacement of any parallel-chains job module by a single (feasible) chain of its jobs. A detailed study of series-parallel precedence graphs is contained in [18].

The second type of constraint considered is called a *contiguity constraint*. $J' \subseteq J$ is called a *compound job* if the jobs in J' must process consecutively in every feasible permutation. This type of constraint might occur, for example, if a set of jobs must be performed at a location a considerable distance from the site at which the remaining jobs are to be performed. If, in addition, a compound job must process in a prespecified order in every feasible permutation then we refer to it as a *string*. When strings are present the problem can be thought of as sequencing strings, rather than the jobs in the strings, since they must process consecutively and in a prespecified order in every feasible permutation. Strings are denoted by $\vec{s}, \vec{t}, \vec{u}, \vec{v}$ and $\vec{w}$. Job related notation extends to strings in the natural way, e.g., $\vec{s} \to \vec{t}$ means that string $\vec{s}$ must precede string $\vec{t}$ in every feasible permutation. There is also a natural correspondence between strings and sequences: A string $\vec{s}$ defines a "sequence" on the jobs in $\vec{s}$ and a sequence s which processes consecutively defines a "string" on the jobs in s.

We conclude this section by defining the least cost fault detection problem which is used for motivation throughout this paper. In this problem a system consisting of n components is to be inspected by sequentially applying tests to each component until one fails (i.e., the system is "defective") or all components pass their test (i.e., the system successfully completes inspection). Associated with each component j is a testing cost c_j and a probability q_j of passing its test, $0 \leqslant q_j \leqslant 1$. The tests are assumed to be statistically independent and so for any sequence s, $Q_i^s = q_{s(1)} q_{s(2)} \cdots q_{s(i-1)}$ is the probability that the ith component in s will be tested (by convention $Q_1^s = 1$). The expected testing cost for a sequence s of length k is given by $\sum_{i=1}^{k} Q_i^s c_{s(i)}$. The problem, denoted by $\sum Q_i c_i$, is to find a feasible permutation which minimizes the expected testing cost, i.e.,

$$\underset{\Pi \in F}{\text{minimize}} \sum_{i=1}^{n} Q_i^{\Pi} c_{\Pi(i)}.$$

A recursive definition of the cost function f for $\sum Q_i c_i$ is defined on all sequences by

$$f(j) = c_j \quad \text{for job } j,$$

and

$$f(s, t) = f(s) + q(s)f(t) \quad \text{for sequences } s \text{ and } t, \tag{1}$$

where $q(s) = q_{s(1)}q_{s(2)} \cdots q_{s(k)}$ for a sequence s of length k.

2. Unconstrained sequencing. One of the most important ideas in the theory of sequencing and scheduling is the method of adjacent pairwise job interchange. This method compares the costs of two sequences which differ only by interchanging a pair of adjacent jobs. In 1956, W. E. Smith [31] defined a class of problems for which a total preference ordering of the jobs exists with the property that in any sequence, wherever two adjacent jobs are *not* in preference order, they may be interchanged with no resultant cost increase. As an example, define job i to be *preferable* to job j, denoted by $i \underline{\lessdot} j$, for $\sum Q_i c_i$ when $c_i/(1 - q_i) \leqslant c_j/(1 - q_j)$; that is, jobs with small costs and large probabilities of failing their tests are most preferable. (We define $a/0$ to be $+\infty$ when $a > 0$ and define it to be $-\infty$ when $a \leqslant 0$. By convention, we say that $a/0 = b/0$ for all $a > 0$ and $b > 0$ and for all $a \leqslant 0$ and $b \leqslant 0$.) The binary preference relation $\underline{\lessdot}$ is *transitive*, i.e., $i \underline{\lessdot} j \underline{\lessdot} k$ implies that $i \underline{\lessdot} k$, and *complete*, i.e., $i \underline{\lessdot} j$, $j \underline{\lessdot} i$ or both. We apply the method of adjacent pairwise job interchange by considering an arbitrary sequence (u, i, j, v); let (u, j, i, v) be obtained by interchanging jobs i and j, where $i \underline{\lessdot} j$. Using (1) we obtain that

$$\begin{aligned} f(u, i, j, v) - f(u, j, i, v) &= q(u)(c_i + q_i c_j) - q(u)(c_j + q_j c_i) \\ &= q(u)(c_i(1 - q_j) - c_j(1 - q_i)) \leqslant 0. \end{aligned}$$

The inequality follows since $q(u) \geqslant 0$, $q_i \leqslant 1$, $q_j \leqslant 1$ and $i \underline{\lessdot} j$. An optimal permutation has the jobs in preference order.

We say that a sequencing function f satisfies the *adjacent pairwise interchange* (*API*) *property* if there is a transitive and complete binary preference relation $\underline{\lessdot}$ defined on jobs satisfying the following condition:

For all jobs i and j, $i \underline{\lessdot} j$ implies that

$$f(u, i, j, v) \leqslant f(u, j, i, v) \quad \text{for all sequences } u \text{ and } v. \tag{2}$$

We say that job i is *strictly preferable* to job j, denoted by $i \lessdot j$, if $i \underline{\lessdot} j$ but $j \underline{\lessdot} i$ does not hold. A job i is *minimal* (*maximal*) with respect to $\underline{\lessdot}$ in a set T if $i \in T$ and $i \underline{\lessdot} j$ ($j \underline{\lessdot} i$) for all $j \in T$. As the notation suggests, the relation $\underline{\lessdot}$ on jobs shares properties with the relation $\leqslant$ on real numbers. For example, $i \lessdot j \underline{\lessdot} k$ or $i \underline{\lessdot} j \lessdot k$ implies that $i \lessdot k$.

We now present a theorem, due to Smith [31], which represents a fundamental result in unconstrained sequencing.

THEOREM 1 (SMITH [31]). *Let f satisfy the API property. Then any permutation satisfying the following property is optimal: $i \lessdot j$ implies that i precedes j in the permutation.*

PROOF. Let Π be a permutation of the form specified in the theorem. Such a permutation exists since the relation $\lessdot$ is acyclic. It suffices to show that any permutation Π' can be transformed into Π by a series of adjacent pairwise job interchanges without increasing the cost. Consider a permutation $\Pi' \neq \Pi$. It must be true that $\Pi' = (u, j, i, v)$, where job j follows job i in Π. (If all adjacent pairs are in the same order then $\Pi' = \Pi$.) By the choice of Π, $i \underline{\lessdot} j$. Therefore, the permutation obtained from Π' by interchanging i and j has no greater cost than Π' and has one fewer disagreements of job order with Π. Repeated application of this procedure completes the proof. ∎

Given an optimal permutation Π for J of the form specified in the theorem, $\Pi \mid J'$ is an optimal permutation for $J' \subseteq J$, where $\Pi \mid J'$ is the permutation of J' with the jobs ordered as in Π. Another consequence of the theorem is that the *Unconstrained Algorithm*, which orders the jobs from the most preferred to the least preferred, produces only optimal permutations. This algorithm requires $O(n \log n)$ comparisons of the form "Is $i \leq j$?" which can be achieved by using a "heap" or a comparable data structure (see [12]) to sort the jobs according to the preference relation.

Actually, Smith's original hypothesis of Theorem 1 required only a complete binary preference relation $\leq$ which satisfies (2). However, for the desired permutation of the theorem to exist, $<$ must be acyclic. Furthermore, if $<$ is acyclic then the desired permutation Π is obtained by repeatedly sequencing next a minimal, unsequenced job; the proof remains unchanged. A transitive and complete binary preference relation $\leq'$ which satisfies (2) is defined by $i \leq' j$ if and only if i precedes j in Π. Therefore the class of problems considered by Smith is *exactly* those which satisfy the API property.

3. Parallel-chains precedence contrained sequencing. We now consider sequencing problems with parallel-chains precedence constraints and strings of jobs. J will be taken to be the set of strings, rather than jobs, to be sequenced since the jobs in a string must process consecutively and in a prespecified order in every feasible permutation. The results of the previous section would apply in a natural way to the problem of sequencing a set of strings without precedence constraints if a total preference ordering existed on all sequences; simply order the strings from the most to the least prefered. With this in mind, we say that a sequencing function f satisfies the *adjacent sequence interchange* (*ASI*) *property* if there exists a transitive and complete binary preference relation $\leq$ defined on sequences satisfying the following property:

For all sequences s and t, $s \leq t$ implies that

$$f(u, s, t, v) \leqslant f(u, t, s, v) \quad \text{for all sequences } u \text{ and } v.$$

As an example, for $\sum Q_i c_i$; we define $s \leq t$ if and only if $f(s)/(1 - q(s)) \leqslant f(t)/(1 - q(t))$. We apply the method of adjacent sequence interchange by considering an arbitrary sequence (u, s, t, v); let (u, t, s, v) be obtained by interchanging s and t, where $s \leq t$. Using (1),

$$\begin{aligned} &f(u, s, t, v) - f(u, t, s, v) \\ &\quad = q(u)[f(s) + q(s)f(t)] - q(u)[f(t) + q(t)f(s)] \\ &\quad = q(u)[f(s)(1 - q(t)) - f(t)(1 - q(s))] \leqslant 0. \end{aligned}$$

The inequality follows since $q(u) \geqslant 0$, $q(s) \leqslant 1$, $q(t) \leqslant 1$ and $s \leq t$. Thus $\sum Q_i c_i$ satisfies the ASI property.

The ASI property clearly implies the API property. One difficulty of sequencing with precedence constraints is that a conflict exists between ordering jobs according to the preference relation and preserving feasibility according to the precedence constraints. The following theorem provides a means for resolving this conflict for parallel-chains problems.

THEOREM 2. *Let f satisfy the ASI property. Consider a job module $\{\vec{s}, \vec{t}\}$ in a general precedence graph, where $\vec{s} \to \vec{t}$ and $\vec{t} \leq \vec{s}$. Then there is an optimal permutation with $\vec{s}$ immediately preceding $\vec{t}$.*

PROOF. Every optimal permutation must be of the form $(u, \vec{s}, v, \vec{t}, w)$. If v is an empty sequence then the theorem holds. Suppose that v is a nonempty sequence. If $v \leq \vec{s}$ then interchanging $\vec{s}$ and v does not increase the cost. On the other hand, if $\vec{s} < v$ then $\vec{t} < v$ by transitivity; therefore, interchanging $\vec{t}$ and v does not increase the cost. Both interchanges are feasible, since $\{\vec{s}, \vec{t}\}$ is a job module, and result in permutations of the desired form. ∎

The strings $\vec{s}$ and $\vec{t}$ of the theorem whose precedence and preference orderings conflict can be replaced by the larger string $(\vec{s}, \vec{t})$. This replacement resolves the conflict and preserves an optimal permutation. This idea forms the basis for the following algorithm for parallel-chains precedence constrainted sequencing.

Parallel-Chains Algorithm

Step 1. If $\vec{s} \to \vec{t}$ implies that $\vec{s} \lessdot \vec{t}$ for all strings $\vec{s}$ and $\vec{t}$ (i.e., $\lessdot$ is consistent with $\to$) then go to Step 2. If not, there is a job module $\{\vec{s}, \vec{t}\}$, where $\vec{s} \to \vec{t}$ and $\vec{t} \lessdot \vec{s}$. Replace $\vec{s}$ and $\vec{t}$ by the string $(\vec{s}, \vec{t})$ and repeat Step 1.

Step 2. Sort the strings according to the preference relation.

Step 1 of the Parallel-Chains Algorithm is justified by Theorem 2. Any preference ordering produced by Step 2 is optimal to the unconstrained problem defined on the strings produced in Step 1; it is also feasible to the precedence constraints and therefore is optimal for the parallel-chains precedence constrained problem.

We say that an algorithm for a parallel-chains precedence constrained sequencing problem is *associative* when the following condition holds for all parallel-chains precedence graphs $G_1 = (J_1, R_1)$ and $G_2 = (J_2, R_2)$: If the algorithm produces optimal permutations Π_1 and Π_2 for G_1 and G_2, respectively, then there is an optimal permutation Π for G_1 in parallel with G_2 satisfying $\Pi_1 = \Pi \mid J_1$ and $\Pi_2 = \Pi \mid J_2$.

THEOREM 3. *The Parallel-Chains Algorithm is associative.*

PROOF. Let $G_1 = (J_1, R_1)$ and $G_2 = (J_2, R_2)$ be parallel-chains precedence graphs. Step 1 of the Parallel-Chains Algorithm is applied to each chain independently and produces the same result when applied to G_1 in parallel with G_2 as when it is applied to each separately. A minimal job in Step 2 for G_1 in parallel with G_2 is either minimal in G_1 or minimal in G_2. Hence, the optimal permutations Π_1 and Π_2 produced by the Parallel-Chains Algorithm for G_1 and G_2, respectively, can be merged into an optimal permutation for G, which is of the desired form. ∎

COROLLARY 4. *Let $G_1 = (J_1, R_1)$ and $G_2 = (J_2, R_2)$ be parallel-chains precedence graphs. If Π is an optimal permutation for G_1 in parallel with G_2 produced by the Parallel-Chains Algorithm then $\Pi \mid J_1$ and $\Pi \mid J_2$ are optimal permutations for G_1 and G_2, respectively.*

4. Series-parallel precedence constrained sequencing.

We now consider the sequencing problem with series-parallel precedence constraints and compound jobs. We say that a sequencing function f satisfies the *series-network decomposition* (*SND*) *property* if the following condition holds for all permutations s and t of the same jobset:

If $f(s) \leqslant f(t)$ then $f(u, s, v) \leqslant f(u, t, v)$ for all sequences u and v.

The SND property implies that if Π_1 and Π_2 are optimal permutations for $G_1 = (J_1, R_1)$ and $G_2 = (J_2, R_2)$ then (Π_1, Π_2) is an optimal permutation for G_1 in series with G_2. That is, a precedence graph can be decomposed into its series components, each of which can be independently solved.

As an example, $\sum Q_i c_i$ satisfies the SND property. To see this, consider s and t to be permutations of the same set (note that $q(s) = q(t)$) and consider u and v to be arbitrary sequences. Using (1), if $f(s) \leqslant f(t)$ then

$$f(u, s, v) - f(u, t, v) = q(u)(f(s) - f(t)) \leqslant 0.$$

The inequality follows, since $q(u) \geqslant 0$.

THEOREM 5. *Let f satisfy the SND property. Consider a compound job $T \subseteq J$ in an arbitrary precedence graph $G = (J, R)$, where t is an optimal permutation for T. Then there is an optimal permutation Π for J satisfying $t = \Pi \mid T$.*

PROOF. Any optimal permutation for J must be of the form (u, s, v), where s is a feasible permutation of T. Now $f(t) \leqslant f(s)$ implies by the SND property that (u, t, v) is an optimal permutation of the desired form. ∎

The previous theorem implies that a compound job can be replaced by a string corresponding to any optimal permutation for it when the SND property holds. Henceforth, we assume that compound jobs have been "preprocessed" and replaced by strings. We now use the SND property to efficiently extend an associative parallel-chains algorithm to solve the series-parallel precedence constrained problem.

THEOREM 6. *Let f satisfy the SND property and suppose we are given an associative parallal-chains algorithm for f. Consider $T \subseteq J$ which forms a parallel-chains job module in a general precedence graph $G = (J, R)$, where t is an optimal permutation produced by the algorithm for T. Then there is an optimal permutation Π for J satisfying $t = \Pi \mid T$.*

PROOF. An optimal permutation for J must be of the form

$$\Pi = (\Pi \mid U_1, \Pi \mid T_1, \Pi \mid U_2, \Pi \mid T_2, \ldots, \Pi \mid T_r, \Pi \mid U_{r+1}),$$

where

(1) $T_1, T_2, \ldots, T_r$ partitions T, and

(2) $U_1, U_2, \ldots, U_{r+1}$ partitions $U = J - T$.

Using a proof technique developed in [28], [30] we consider the precedence graph $G' = (J, R')$, where R' contains all of the constraints in R plus additional ones so that:

(1′) the subgraph induced by U_1 is in series with the subgraph induced by $(J - U_1 - U_{r+1})$ which, in turn, is in series with the subgraph induced by U_{r+1}; and

(2′) the subgraph of G' induced by U is the chain corresponding to the permutation $\Pi \mid U$.

G is a relaxation of G' and Π is feasible to G'; therefore Π is an optimal permutation for G'.

Using the SND property we obtain a new optimal permutation by replacing $\Pi \mid (J - U_1 - U_{r+1})$ by any optimal permutation for the subgraph of G' induced by $(J - U_1 - U_{r+1})$.

By (1′) and (2′), the subgraph of G' induced by $(J - U_1 - U_{r+1})$ is a parallel-chains precedence graph consisting of T in parallel with the chain $\Pi \mid (U - U_1 - U_{r+1})$. Since the parallel-chains algorithm is associative, T may be replaced by any optimal permutation generated by the algorithm for T. This yields the desired optimal permutation. ∎

The previous theorem implies that a parallel-chains job module in an arbitrary precedence graph can be replaced by a single feasible chain produced by an associative parallel-chains algorithm; this replacement preserves an optimal permutation. A series-parallel precedence graph, by definition, can be reduced to a single chain by repeating such a process. We call the algorithm to accomplish this reduction the *Series-Parallel Algorithm*.

The Parallel-Chains Algorithm is associative by Theorem 3 and produces optimal permutations for sequencing functions which satisfy the ASI property by Theorem 2. Therefore, a corollary of Theorem 6 is that the Series-Parallel Algorithm using the Parallel-Chains Algorithm produces optimal permutations for the series-parallel precedence constrained problem when the sequencing function satisfies the ASI and SND properties. An implementation of this algorithm requires $O(n \log n)$ comparisons of the form "Is $s \leq t$?". This time bound assumes that the series-parallel precedence graph is specified by a "decomposition tree" as defined in [15]. The algorithm is a modification of one presented in [16] for the one-machine total weighted completion time problem with series-parallel precedence constraints. See [23] for the details of this implementation.

5. Example problems. In this section we present several widely studied problems which satisfy the ASI and SND properties.

(1) In the one-machine total weighted completion time problem n jobs are to be sequenced on a single machine; each job j requires $p_j \geqslant 0$ time units of processing. For any sequence s let $C_i^s = \sum_{j=1}^{i} p_{s(j)}$ denote the completion time of the ith job in the sequence s. Each job j also has a weighting factor w_j and incurs a linear cost of w_j times its completion time, i.e., the cost of a sequence s of length k is $\sum_{i=1}^{k} w_{s(i)} C_i^s$. The problem, denoted by $\sum w_i C_i$, is to find a feasible permutation to minimize the total weighted completion time, i.e.,

$$\underset{\Pi \in F}{\text{minimize}} \sum_{i=1}^{n} w_{\Pi(i)} C_i^{\Pi}.$$

A recursive definition of the cost function f for $\sum w_i C_i$ is defined on all sequences by

$$f(j) = p_j w_j \text{ for job } j, \text{ and}$$

$$f(s, t) = f(s) + f(t) + p(s) w(t) \quad \text{for sequences } s \text{ and } t, \tag{3}$$

where $p(s) = \sum_{i=1}^{k} p_{s(i)}$ and $w(t) = \sum_{i=1}^{l} w_{t(i)}$ for sequences s and t of lengths k and l, respectively. Using (3) it is easy to verify the ASI and SND properties, where $s \lesssim t$ if and only if $w(s)/p(s) \geqslant w(t)/p(t)$.

(2) For the maximum cumulative excess cost problem consider a single item inventory system. A set of n transactions are to be posted against the inventory. Each transaction performs a fixed sequence of deposits and withdrawals. The relevant data for transaction j is summarized in its net change in inventory level, c_j, its maximum net increase in inventory level, m_j, and its "target" level, e_j. The processing of a typical job j is illustrated in Figure 1. For any sequence s, define $CC_i^s = \sum_{j=1}^{i-1} c_{s(j)} + m_{s(i)}$ to be the maximum level attained by the ith job in the sequence s (by convention $CC_1^s = m_{s(1)}$). The cost for a sequence s of length k is $\text{maximum}_{1 \leqslant i \leqslant k} \{CC_i^s - e_{s(i)}\}$. The problem, denoted by $CE_{\max}$, is to find a feasible permutation to minimize the maximum cumulative excess cost of a transaction above its target level, i.e.,

$$\underset{\Pi \in F}{\text{minimize}} \underset{1 \leqslant i \leqslant n}{\text{maximum}} \{ CC_i^{\Pi} - e_{\Pi(i)} \}.$$

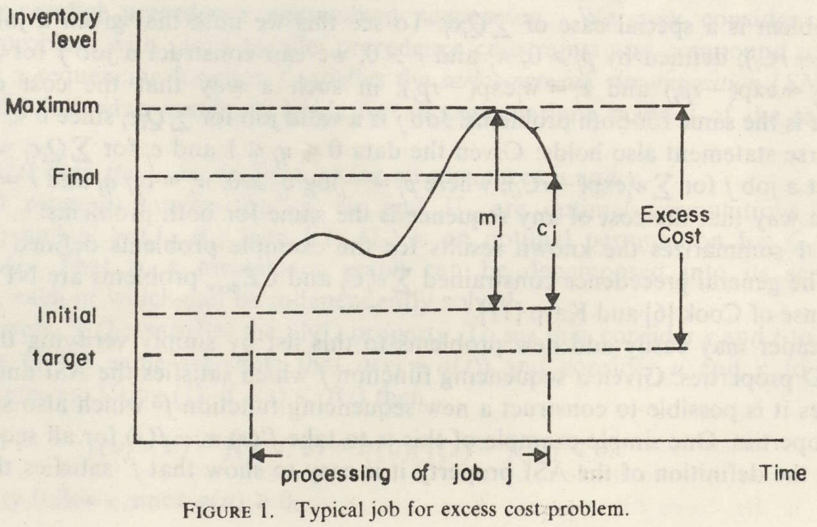

FIGURE 1. Typical job for excess cost problem.

A recursive definition of the cost function f for $CE_{\max}$ is defined on all sequences by

$$f(j) = m_j - e_j \quad \text{for job } j \quad \text{and}$$

$$f(s, t) = \max(f(s), c(s) + f(t)) \quad \text{for sequences } s \text{ and } t, \tag{4}$$

where $c(s) = \sum_{i=1}^{k} c_{s(i)}$ for a sequence s of length k.

Using (4) it is easy to verify the ASI and SND properties, where $s \leq t$ if and only if
(a) $c(s) \leqslant 0$ and $c(t) \geqslant 0$; or
(b) $c(s) < 0$, $c(t) < 0$ and $f(s) \leqslant f(t)$; or
(c) $c(s) > 0$, $c(t) > 0$ and $c(s) - f(s) \leqslant c(t) - f(t)$.
Johnson's [10] two-machine maximum completion time flow-shop problem ($C_{\max}$) and the maximum cumulative cost problem ($CC_{\max}$) [2], [3] are shown to be special cases of $CE_{\max}$ in [22].

(3) The maximum cost fault detection problem is similar to $\sum Q_i c_i$: the *sum* cost function is replaced by the *maximum* cost function. That is, the cost of a sequence s of length k is $\text{maximum}_{1 \leqslant i \leqslant k} \{Q_i^s c_{s(i)}\}$. The problem, denoted by $Q_{\max}$, is to find a feasible permutation which minimizes the maximum expected component testing cost, i.e.,

$$\underset{\Pi \in F}{\text{minimize}} \ \underset{1 \leqslant i \leqslant n}{\text{maximum}} \ Q_i^{\Pi} c_{\Pi(i)}.$$

A recursive definition of the cost function f for $Q_{\max}$ is defined on all sequences by

$$f(j) = c_j \quad \text{for job } j, \quad \text{and}$$

$$f(s, t) = \max(f(s), q(s) f(t)) \quad \text{for sequences } s \text{ and } t. \tag{5}$$

Using (5) it is easy to verify the ASI and SND properties, where $s \leq t$ if and only if
(a) $f(s) \leqslant 0$ and $f(t) \geqslant 0$, or
(b) $f(s) < 0$, $f(t) < 0$ and $f(s)/q(s) \geqslant f(t)/q(t)$, or
(c) $f(s) > 0$, $f(t) > 0$ and $f(s) \leqslant f(t)$.

(4) The total weighted exponential completion time problem is similar to $\sum w_i C_i$: cost accumulates exponentially rather than linearly. The cost of a sequence s of length k is $\sum_{i=1}^{k} w_{s(i)} \exp(-rC_i^s)$, where $r > 0$ is a constant. The problem, denoted by $\sum w_i \exp(-rC_i)$, is to find a feasible permutation which minimizes the total weighted exponential completion time, i.e.,

$$\underset{\Pi \in F}{\text{minimize}} \sum_{i=1}^{n} w_{\Pi(i)} \exp(-rC_i^{\Pi}).$$

This problem is a special case of $\sum Q_i c_i$. To see this we note that given a job j for $\sum w_i \exp(-rC_i)$, defined by $p_j \geqslant 0$, w_j and $r > 0$, we can construct a job j for $\sum Q_i c_i$, where $q_j = \exp(-rp_i)$ and $c_j = w_i \exp(-rp_i)$, in such a way that the cost of any sequence is the same for both problems. Job j is a valid job for $\sum Q_i c_i$ since $0 < q_j \leqslant 1$. A converse statement also holds: Given the data $0 < q_j \leqslant 1$ and c_j for $\sum Q_i c_i$ we may construct a job j for $\sum w_i \exp(-rC_i)$, where $p_j = -\log q_j \geqslant 0$, $w_j = c_j / q_j$ and $r = 1 > 0$, in such a way that the cost of any sequence is the same for both problems.

Table 1 summarizes the known results for the example problems defined in this paper. The general precedence constrained $\sum w_i C_i$ and $CE_{\max}$ problems are NP Hard, in the sense of Cook [6] and Karp [11].

The reader may easily add new problems to this list by simply verifying the ASI and SND properties. Given a sequencing function f which satisfies the ASI and SND properties it is possible to construct a new sequencing function f' which also satisfies these properties. One simple example of this is to take $f'(s) = -f(s)$ for all sequences s. Using the definition of the ASI property it is easy to show that f' satisfies the ASI property using $s \leq' t$ if and only if $t \leq s$, where $\leq$ is a binary preference relation for f. This implies that the problem of finding a *maximum* cost permutation satisfies the ASI property. Variations of the examples cited can also be easily shown to satisfy the ASI and SND properties. Examples are the sequencing function for $\sum w_i C_i$ given by (3), where $w_j \geqslant 0$ and p_j is unconstrained for all jobs j, rather than $p_j \geqslant 0$ and w_j unconstrained for all jobs j; and, the sequencing functions for $\sum Q_i c_i$ and $Q_{\max}$ given by (1) and (5), where $c_j \geqslant 0$ and $q_j \geqslant 0$ for all jobs j, rather than $0 \leqslant q_j \leqslant 1$ and c_j unconstrained.

TABLE 1

Summary of computational complexity

f	Unconstrained	Chains/Tree	Series-Parallel	General Precedence
$\sum Q_i c_i$	[20]	[7]		Open Problem
$\sum w_i \exp(-rC_i)$	[26]	—	[17]	Open Problem
$\sum w_i C_i$	[31]	[4], [5], [9], [27], [28]	[13] [16]	NP Hard [16] [19]
$CE_{\max}$	$C_{\max}$ [10]	$C_{\max}$ [14]	$CC_{\max}$ [2] [3] $C_{\max}$ [21], [30]	$CC_{\max}$ NP Hard [1] $C_{\max}$ NP Hard [22]

5. Concluding remarks. We defined a class of problems based on easily verifiable properties. These were solved with series-parallel precedence constraints by an efficient general algorithm. This algorithm uses information about the cost function only through the preference order defined on the elements to be sequenced. That is, only *ordinal* data is necessary to solve these problems. *Cardinal* data values are superfluous. This class of problems is studied with general precedence constraints in [23] and [24].

The ASI property is one natural extension of the API property. Other extensions, for example, considering pairwise (not necessarily adjacent) job interchanges and the insertion of a job into another position in a sequence, are examined in [23]. These lead to efficient algorithms for sequencing with general precedence constraints and secondary criteria problems.

The reader may easily verify that even though the Series-Parallel Algorithm produces only optimal sequences, it can not produce *all* optimal sequences in general. In [23] stronger versions of the ASI and SND properties are defined which lead to a version of the Series-Parallel Algorithm which can produce a sequence if and only if the sequence is optimal. These stronger properties are satisfied by $\sum w_i C_i$ where $p_i > 0$, and by $\sum Q_i c_i$ where $0 < q_i < 1$.

References

[1] Abdel-Wahab, H. M. (1976). Scheduling with Applications to Register Allocation and Deadlock Problems. Ph.D. thesis, University of Waterloo, Waterloo, Canada.

[2] ——— and Kameda, T. (1978). Scheduling to Minimize Maximum Cumulative Cost Subject to Series-Parallel Precedence Constraints. *Operations Res.* **26** 141–158.

[3] ——— and ———. (1978). On the *C*-Optimal Scheduling Problem. Department of Electrical Engineering, University of Waterloo, Waterloo, Canada.

[4] Adolphson, D. L. and Hu, T. C. (1973). Optimal Linear Ordering. *SIAM J. Appl. Math.* **25** 403–423.

[5] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967). *Theory of Scheduling.* Addison-Wesley, Reading, Massachusetts.

[6] Cook, S. A. (1971). The Complexity of Theorem-Proving Procedures. In *Proc. Third Annual ACM Symp. on the Theory of Computing*, 151–158, Shaker Heights, Ohio.

[7] Garey, M. R. (1973). Optimal Task Sequencing with Precedence Constraints. *Discrete Math.* **4** 37–56.

[8] Herriot, J. (1972). *All Creatures Great and Small.* Bantam Books, Inc., New York.

[9] Horn, W. A. (1972). Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties. *SIAM J. Appl. Math.* **23** 189–202.

[10] Johnson, S. M. (1954). Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Naval Res. Logist. Quart.* **1** 61–68.

[11] Karp, R. M. (1975). On the Computational Complexity of Combinatorial Problems. *Networks.* **5** 54–68.

[12] Knuth, D. E. (1973). *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*. Addison-Wesley, Reading, Massachusetts.

[13] ———. (1973). Private communication to T. C. Hu, July 23, 1973.

[14] Kurisu, T. (1976). Two-Machine Scheduling under Required Precedence among Jobs. *J. Operations Res. Soc. Japan.* **19** 1–13.

[15] Lawler, E. L. (1976). Graphical Algorithms and Their Complexity. Math. Centre Tracts 81. University of California at Berkeley, 3–32.

[16] ———. (1978). Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. *Ann. Discrete Math.* **11** 75–90.

[17] ——— and Sivaslian, B. D. (1978). Minimization of Time Varying Costs in Single Machine Sequencing. *Operations Res.* **26** 563–569.

[18] ———, Tarjan, R. E. and Valdez, J. (to appear). Analysis and Isomorphism of Series-Parallel Diagraphs.

[19] Lenstra, J. K., Rinnooy Kan, A. H. G. and Brucker, P. (1977). Complexity of Machine Scheduling Problems. *Ann Discrete Math.* **1** 343–362.

[20] Mitten, L. G. (1960). An Analytic Solution to the Least Cost Testing Sequence Problem. *J. Industrial Engineering.* **11** 17.

[21] Monma, C. L. (to appear). Two-Machine Flow-Shop Problem with Series-Parallel Precedence Relations: An Algorithm and Extensions. *Operations Res.*

[22] ———. (1978). Sequencing to Minimize the Maximum Job Cost. Submitted for publication.

[23] ———. (1978). Properties and Efficient Algorithms for Certain Classes of Sequencing Problems. Ph.D. thesis, School of OR/IE, Cornell University, Ithaca, New York.

[24] ———. (1979). Sequencing with General Precedence Constraints. Submitted for publication.

[25] ——— and Sidney, J. B. (1977). A General Algorithm for Optimal Job Sequencing with Series-Parallel Precedence Constraints. Report 347. School of OR/IE, Cornell University, Ithaca, New York.

[26] Rothkopf, M. E. (1966). Scheduling Independent Tasks on Parallel Processors. *Management Sci.* **12** 437–447.

[27] Sidney, J. B. (1970). Single-Machine Deterministic Job-Shop Sequencing with Precedence Relations and Deferral Costs. Ph.D. thesis, University of Michigan.

[28] ———. (1975). Decomposition Algorithms for Single Machine Sequencing with Precedence Relations and Deferral Costs. *Operations Res.* **23** 283–298.

[29] ———. (1976). A General Algorithm for Optimal Job Sequencing with Parallel-Chains Precedence Constraints. University of Ottawa, Ottawa, Canada.

[30] ———. (1979). The Two-Machine Flow Time Problem with Series-Parallel Precedence Constraints. *Operations Res.* **27**.

[31] Smith, W. E. (1956). Various Optimizers for Single-Stage Production. *Naval Res. Logist. Quart.* **3** 59–66.

MONMA: BELL LABORATORIES, HOLMDEL, NJ 07733

SIDNEY: FACULTY OF ADMINISTRATION, UNIVERSITY OF OTTAWA, OTTAWA, ONTARIO, CANADA K1N 6N5

Note added in proof. It has recently been pointed out to us by E. L. Lawler that the Series-Parallel Algorithm using the Parallel-Chains Algorithm works when only the ASI property holds. That is, the SND property is not needed.