stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 54/75 NOVEMBER

E.L. LAWLER

OPTIMAL SEQUENCING OF JOBS SUBJECT TO SERIES PARALLEL PRECEDENCE CONSTRAINTS

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 90B35

OPTIMAL SEQUENCING OF JOBS SUBJECT TO SERIES PARALLEL PRECEDENCE CONSTRAINTS

E.L. LAWLER

ABSTRACT

Suppose that n jobs are to be performed by a single machine subject to precedence constraints that can be characterized as "series parallel". Each job j has associated with it a known processing time p_j and weight w_j. The objective is sequence the jobs so as to minimize total weighted completion time. It is shown that this problem can be solved by an algorithm with worst case running time of O(n log n). This result is implicit in results of Sidney, and algorithms of Baker, Horn, and Adolphson and Hu, which dealt with precedence constraints in the form of rooted trees.

KEY WORDS & PHRASES: *machine sequencing, precedence constraints, series parallel, polynomial algorithm.*
AUTHOR'S ADDRESS: Department of Electrical Engineering and Computer Sciences, The University of California, Berkeley, California 94720, USA.

1. INTRODUCTION

Suppose n jobs are to be performed by a single machine. Each job j has associated with it a known processing time $p_j > 0$ and weight w_j, j = 1,2,...,n. For any given sequence of jobs, $\sigma = (\sigma(1),\sigma(2),\ldots,\sigma(n))$, the completion time of job $\sigma(i)$ is

$$C_{\sigma(i)} = \sum_{k=1}^{i} p_{\sigma(k)},$$

and the total weighted completion time of σ is

$$\sum_{i=1}^{n} w_{\sigma(i)} C_{\sigma(i)}.$$

In the absence of further constraints, the problem of finding a sequence which minimizes total weighted completion time is solved by a simple rule due to W.E. SMITH [11]: For each job j, form the ratio

$$\rho_j = \frac{w_j}{p_j},$$

and then place the jobs in monotone nonincreasing ratio order. Clearly, this task can be accomplished in O(n log n) running time.

Now suppose that precedence constraints are specified, in the form of an (irreflexive) partial order "$\prec$". Jobs i must precede job j if $i \prec j$.

For arbitrary precedence constraints, the problem is quite difficult and may be NP-complete in the sense of COOK and KARP [6], even if all job weights are equal. At the time this is written, NP-completeness remains an open question [9]. (Note that this is in constrast to the problem of minimizing the *maximum* weighted completion time, subject to arbitrary precedence constraints, for which there is an efficient algorithm [7].)

However, if the precedence constraints are suitably restricted, the

problem is much easier. CONWAY, et al, [4] dealt with precedence constraints in the form of "parallel chains". BAKER [3] and HORN [5] solved the problem for more general precedence constraints in the form of rooted trees (or "arborecences"), either rooted to a point or from a point. ADOLPHSON and HU [1], working in the context of the "optimal linear ordering" problem (cf. Section 6), showed that Horn's algorithm can be implemented in $O(n \log n)$ running time. SIDNEY [10] has obtained a number of theorems concerning arbitrary precedence constraints, but his results do not suggest a polynomial bounded algorithm for the general case.

In this paper, we make use of some of Sidney's results to obtain an algorithm for solving the sequencing problem in the case that the precedence constraints can be characterized as "series parallel". We also show that this algorithm can be implemented with worst case running time of $O(n \log n)$. Since rooted trees are special cases of series parallel constraints, this result provides a proper generalization of previous results.

2. SERIES PARALLEL DIGRAPHS

We represent a given set of precedence constraints by a digraph $G = (N,A)$ in the obvious way. That is, each node $j \in N$ corresponds to a job and each arc $(i,j) \in A$ corresponds to a pair of jobs in the relation, i.e. $i \prec j$. We now proceed to define series parallel precedence constraints in terms of digraphs.

The class of *transitive series parallel* digraphs is defined recursively as follows:

(2.1) A digraph consisting of a single node, e.g. $G = (\{i\},\emptyset)$, is transitive series parallel.

(2.2) If $G_1 = (N_1,A_1)$ and $G_2 = (N_2.A_2)$, where $N_1 \cap N_2 = \emptyset$, are transitive series parallel, then

$$G = G_1 \times G_2 = (N_1 \cup N_2, A_1 \cup A_1 \cup N_1 \times N_2)$$

is also transitive series parallel. G is said to be the *series composition* of G_1 and G_2.

(2.3) If $G_1 = (N_1,A_1)$ and $G_2 = (N_2,A_2)$, where $N_1 \cap N_2 = \emptyset$, are transitive series parallel, then

$$G = G_1 \cup G_2 = (N_1 \cup N_2, A_1 \cup A_2)$$

is also transitive series parallel. G is said to be the *parallel composition* of G_1 and G_2.

(2.4) Only those digraphs which can be formed by a finite number of applications of rules (2.1) - (2.3) are transitive series parallel.

A digraph G is said to be series parallel if and only if its transitive closure is transitive series parallel. Some examples of series parallel digraphs are shown in Figure 1. Note that every series parallel digraph is acyclic. The simplest acyclic digraph which is not series parallel is shown in Figure 2.

Given a series parallel digraph G, it is possible to repeatedly decompose G into series and parallel components, so as to show how G can be obtained by application of rules (2.1) - (2.3). The result is a rooted binary tree we call a *decomposition tree*. Each leaf of the decomposition tree is identified with a node of G. Each internal node marked "S" indicates the series composition of the subgraphs identified with its sons, with the convention that the left son precedes the right son. Each inter-

nal node marked "P" indicates the parallel composition of the subgraphs identified with its sons. (Here the left-right ordering of sons is unimportant.) Decomposition trees T_1,T_2,T_3 for the digraphs G_1,G_2,G_3 shown in Figure 1 are given in Figure 3. The "S's" and "P's" in the internal nodes of T_3 are given subscripts to facilitate reference in the next section.

In [8] it is shown how to test a given digraph to determine if it is series parallel and, if it is, to obtain a decomposition tree. This task can be accomplished on $O(n^2)$ running time. In the sequel, we shall assume that a decomposition tree is already known for any given precedence constraints. Hence, when the claim of O(n log n) running time is made for the algorithm presented in this paper, this claim must be qualified to apply to a problem for which a decomposition tree is given.

3. A RECURSIVE PROCEDURE

Suppose we are able to construct optimal sequences for the series and parallel compositions of subsets of jobs N_1,N_2, given optimal sequences for N_1 and N_2. Then we have all that is necessary for a recursive procedure for solving sequencing problems with series parallel precedence constraints. Rules for these constructions are as follows.

Series Composition

Let σ_1,σ_2 be optimal sequences for N_1,N_2. Then an optimal sequence for the series composition of N_1 and N_2 is simply $\sigma = \sigma_1\sigma_2$, the concatenation of σ_1 and σ_2.

The proof of this assertion is trivial and is omitted.

Parallel Composition

Let σ_1,σ_2 be optimal sequences for N_1,N_2, where

$$\sigma_i = (\sigma_i(1),\sigma_i(2),\ldots,\sigma_i(n_i)), \qquad i = 1,2,$$

and job $\sigma_i(k)$ has weight w_{ik} and processing time p_{ik}. An optimal sequence for the parallel composition of N_1 and N_2 is obtained by repeatedly finding "maximum-ratio prefixes" of σ_1,σ_2 and placing there in successive positions of the solution sequence σ. This technique is implemented by the following algorithm.

Step 0. Let $\sigma = \lambda$, the empty sequence. Set $m_i = 1$, $i = 1,2$.

Step 1. If $m_i = n_i + 1$, for $i = 1,2$, stop; σ is an optimal sequence for the parallel composition of N_1,N_2. Otherwise, continue to Step 2.

Step 2. Compute

$$\rho_{ir} = \frac{\sum_{j=m_i}^{r} w_{ij}}{\sum_{j=m_i}^{r} p_{ij}}, \qquad i = 1,2;\ r = m_i,m_i+1,\ldots,n_i.$$

Find

$$\rho_i^* = \max_{m_i \le r \le n_i} \{\rho_{ir}\}, \qquad i = 1,2,$$

and values k,m such that

$$\rho_{km} = \max\{\rho_1^*,\rho_2^*\}.$$

Let

$$\sigma_k^* = (\sigma_k(m_k),\sigma_k(m_k+1),\ldots,\sigma_k(m)).$$

Set $\sigma = \sigma\sigma_k^*$, the concatenation of σ and σ_k^*, and set $m_k = m + 1$.

Return to Step 1.

This algorithm is essentially the same as Sidney's "parallel chain" algorithm, with some changes in notation. The validity of the algorithm in this context follows immediately from Sidney's theorem 21. We refer the reader to his paper [10] for a proof.

An Example

With the above rules for series and parallel composition it is clearly possible to carry out a recursive computation, starting at the bottom of the decomposition tree and working upwards until an optimal sequence is found for the complete set of jobs. As an example, consider the problem with precedence constraints given by digraph G_3 in Figure 1 and with weights and processing times as follows:

j	1	2	3	4	5	6	7	8	9	10	11	12	13
w_j	1	1	3	2	7	2	10	3	1	3	4	9	1
p_j	1	4	1	3	2	8	1	5	10	7	8	2	6

Optimal solutions for the various subproblems, defined by the decomposition tree T_3 in Figure 3, are as follows:

$P_1 : \sigma_1 = (7,8)$

$S_2 : \sigma_2 = (7,8,10)$

$S_3 : \sigma_3 = (4,7,8,10)$

$P_4 : \sigma_4 = (3,2)$

$S_5 : \sigma_5 = (5,9)$

$P_6 : \sigma_6 = (5,6,9)$

$S_7 : \sigma_7 = (5,6,9,11)$

$S_8 : \sigma_8 = (3,2,5,6,9,11)$

$P_9 : \sigma_9 = (4,7,3,2,5,8,10,6,9,11)$

$P_{10}: \sigma_{10} = (12,13)$

$S_{11}: \sigma_{11} = (4,7,3,2,5,8,10,6,9,11,12,13)$

$S_{12}: \sigma_{12} = (1,4,7,3,2,5,8,10,6,9,11,12,13)$

As an example of parallel composition, consider in detail the formation of the sequence σ_9 from the sequences σ_3 and σ_8. At successive executions of Step 2 of the algorithm for parallel computation, data are as follows:

σ_3^*	ρ_3^*	σ_8^*	ρ_8^*	σ_9
(4,7)	12/4	(3)	3/1	λ
(8)	3/5	(3)	3/1	(4,7)
(8)	3/5	(2,5)	8/6	(4,7,3)
(8)	3/5	(6,9,11)	6/26	(4,7,3,2,5)
(10)	3/7	(6,9,11)	6/26	(4,7,3,2,5)
λ	$-\infty$	(6,9,11)	6/26	(4,7,3,2,5,10)
				(4,7,3,2,5,10,6,9,11)

4. MODIFICATION OF RECURSIVE PROCEDURE

We now present a number of refinements and modifications of the basic recursive procedure described in the previous section. The principal objective of these modifications is to enable us to represent the solutions to subproblems in the recursion by *sets* rather than *sequences*. It is believed

that the validity of assertions we make in this section is intuitively obvious, and that proofs can safely be left to the reader.

Composite Jobs

Suppose that the parallel composition of sequences σ_1, σ_2 is carried out as part of the recursive computation, and that σ_i^* is a maximum-ratio sequence found in Step 2 of the subalgorithm for parallel composition. We assert that there exists an optimal sequence for the complete set of jobs in which the jobs in σ_i^* are consecutive. Accordingly, it is possible to treat the subsequence σ_i^* as a single *composite job*, with processing time and weight set equal to the sums of the processing times and weights of the jobs contained within it.

Moreover, we assert that there is no change in the ultimate outcome of the computation if maximum-ratio subsequences are found, and composite jobs formed, at the time of series, rather than parallel, composition. As a consequence of this modification, the solution to each subproblem can be represented by a set of jobs, some or all of which are composite. An optimal sequence for a subproblem can be constructed by simply placing these jobs in monotonically nonincreasing order of w_j/p_j ratios.

The rules for series and for parallel composition can now be restated as follows.

Series Composition

Suppose that solutions to two subproblems are given by sets N_1 and N_2 of elementary and composite jobs. The series composition of N_1 and N_2 can result in the formation of at most one new composite job. Moreover, this composite job must contain a minimum ratio job from N_1 and a maximum

ratio job from N_2. The following algorithm is now used for series composition.

Step 0. Find

$$\rho_1^* = \min_{j \in N_i'} \left\{\frac{w_j}{p_j}\right\},$$

$$\rho_2^* = \max_{j \in N_2'} \left\{\frac{w_j}{p_j}\right\},$$

and $j(1), j(2)$ such that

$$\frac{w_{j(i)}}{p_{j(i)}} = \rho_i^*, \qquad i = 1,2.$$

If $\rho_1^* \geq \rho_2^*$, then stop; the solution to the series composition of N_1, N_2 is given by $N_1' \cup N_2'$ (no composite job is formed). If $\rho_1^* < \rho_2^*$, then form a composite job $k = (j(1), j(2))$, with $w_k = w_{j(1)} + w_{j(2)}$, $p_k = p_{j(1)} + p_{j(2)}$, set $N_i' = N_i' - \{j(i)\}$, $i = 1,2$, and proceed to Step 1.

Step 1. Find ρ_1^*, ρ_2^*, $j(1)$, $j(2)$, as in Step 0. If $\rho_1^* \geq w_k/p_k \geq \rho_2^*$, then stop; the solution to the series composition of N_1, N_2 is given by $N_1' \cup N_2' \cup \{k\}$. Otherwise proceed to Step 2.

Step 2. If $\rho_1^* < w_k/p_k$, then replace k by $j(1),k$ (the concatenation of $j(1)$ and k), set $w_k = w_k + w_{j(1)}$, $p_k = p_k + p_{j(1)}$, and $N_1' = N_1' - \{j(1)\}$. If $\rho_1^* \geq w_k/p_k$ and $w_k/p_k < \rho_2^*$, then replace k by $k,j(2)$, set $w_k = w_k + w_{j(2)}$, $p_k = p_k + p_{j(2)}$, and $N_2' = N_2' - \{j(2)\}$. Return to Step 1.

Parallel Composition

Suppose that solutions to two subproblems are given by sets N_1, N_2 of

elementary and composite jobs. A solution for parallel composition of N_1 and N_2 is given simply by $N_1 \cup N_2$.

An Example

We now rework the example of the previous section with these new procedures. We denote composite jobs by sequences, e.g. N_8 contains the elementary job 3 and composite jobs (2,5) and (6,9,11). The reader should trace through the subalgorithm for series composition to verify the formation of composite jobs.

$P_1 : N_1 = \{7,8\}$

$S_2 : N_2 = \{7,8,10\}$

$S_3 : N_3 = \{(4,7),8,10\}$

$P_4 : N_4 = \{3,2\}$

$S_5 : N_5 = \{5,9\}$

$P_6 : N_6 = \{5,6,9\}$

$S_7 : N_7 = \{5,(6,9,11)\}$

$S_8 : N_8 = \{3,(2,5),(6,9,11)\}$

$P_9 : N_9 = \{(4,7),3,(2,5),8,10,(6,9,11)\}$

$P_{10} : N_{10} = \{12,13\}$

$S_{11} : N_{11} = \{(4,7),3,(2,5),(8,10,6,9,11,12),13\}$

$S_{12} : N_{12} = \{(1,4,7,3),(2,5),(8,10,6,9,11,12),13\}$

5. IMPLEMENTATION IN O(n log n) TIME

We have observed that optimal solutions to subproblems in the recursive computation can be represented as sets. In order to solve subproblems

with sets as inputs and outputs, we need to be able to perform the following operations efficiently: (i) identify a maximal or minimal element of a set, (ii) delete (a maximal or minimal) element from a set, (iii) form the union of two sets. Each of these operations is performed as many as O(n) times. It follows that if we are to attain our goal of O(n log n) running time overall, we must be able to perform each of these operations in O(log n) steps.

There are data structures which facilitate these set operations known as "mergeable heaps". A complete exposition of these data structures can be found in the book of AHO, HOPCROFT and ULLMAN [2, p.152], with a demonstration that the desired running times can be attained.

6. RELATIONSHIP TO OPTIMAL LINEAR ORDERING

A problem related to sequencing is that of "optimal linear ordering", studied by ADOLPHSON and HU [1]. Let G = (N,A) be an acyclic digraph and let p'_i be a positive weight associated with node i and q_{ij} a positive weight associated with arc(i,j).

The number p'_i can be thought of as the "width" of node i. The number q_{ij} can be thought of as the "number of wires" from i to j, where each wire leaves the "right" side of i and enters the "left" side of j. The object of the problem is to arrange the nodes adjacently in a one-dimensional array, in such a way that all wires are directed from left to right, and the sum of the wire lengths is minimized.

The sequencing problem and the optimal linear ordering problems are equivalent, under the relations

$$p_i = p_i'$$

$$w_i = \sum_j q_{ji} - \sum_j q_{ij}.$$

Thus, the algorithm presented in this paper can also be employed to solve the optimal linear ordering problem with series parallel precedence constraints.

REFERENCES

[1] ADOLPHSON, D. & T.C. HU, "*Optimal Linear Ordering*", SIAM J. Appl. Math., 25 (1973), p. 403-423.

[2] AHO, A.V., J.E. HOPCROFT & J.D. ULLMAN, *The Design of Computer Algorithms*, Addison Wesley, Reading, Mass., 1974.

[3] BAKER, K.R., "*Single Machine Sequencing with Weighting Factors and Precedence Constraints*", unpublished paper, 1971.

[4] CONWAY, R.W., W.L. MAXWELL & L.W. MILLER, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.

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

[6] KARP, R.M., "*On the Computational Complexity of Combinatorial Problems*", Networks 5 (1975), p. 45-68.

[7] LAWLER, E.L., "*Optimal Sequencing of a Single Machine Subject to Precedence Constraints*", Management Science 19 (1973), p. 544-546.

[8] LAWLER, E.L. & R.E. TARJAN, "*Analysis and Isomorphism of Series Parallel Digraphs*", to appear.

[9] LENSTRA, J.K., A.H.G. RINNOOY KAN & P. BRUCKER, "*Complexity of Machine Scheduling Problems*", to appear in Annals of Discrete Math., North Holland, 1975.

[10] SIDNEY, J.B., "*Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs*", Operations Research 22 (1975) p.283-298.

[11] SMITH, W.E., "*Various Optimizers for Single-Stage Production*", Naval Res. Log. Quart. 3 (1956) p.59-66.

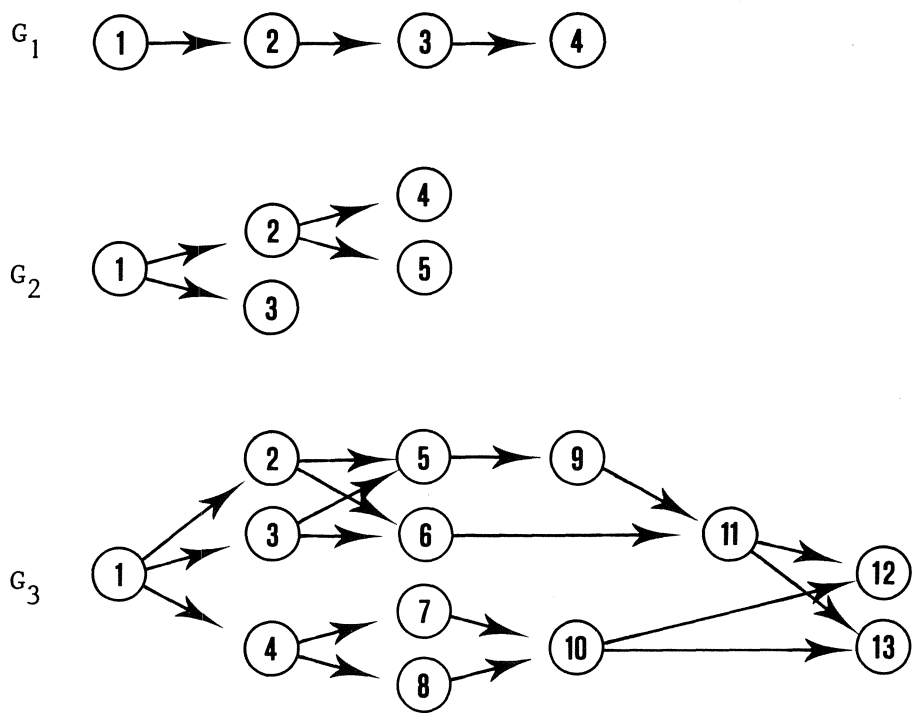

Figure 1 Series Parallel Digraphs.

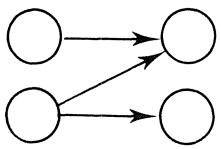

Figure 2 A Nonseries Parallel Digraph.

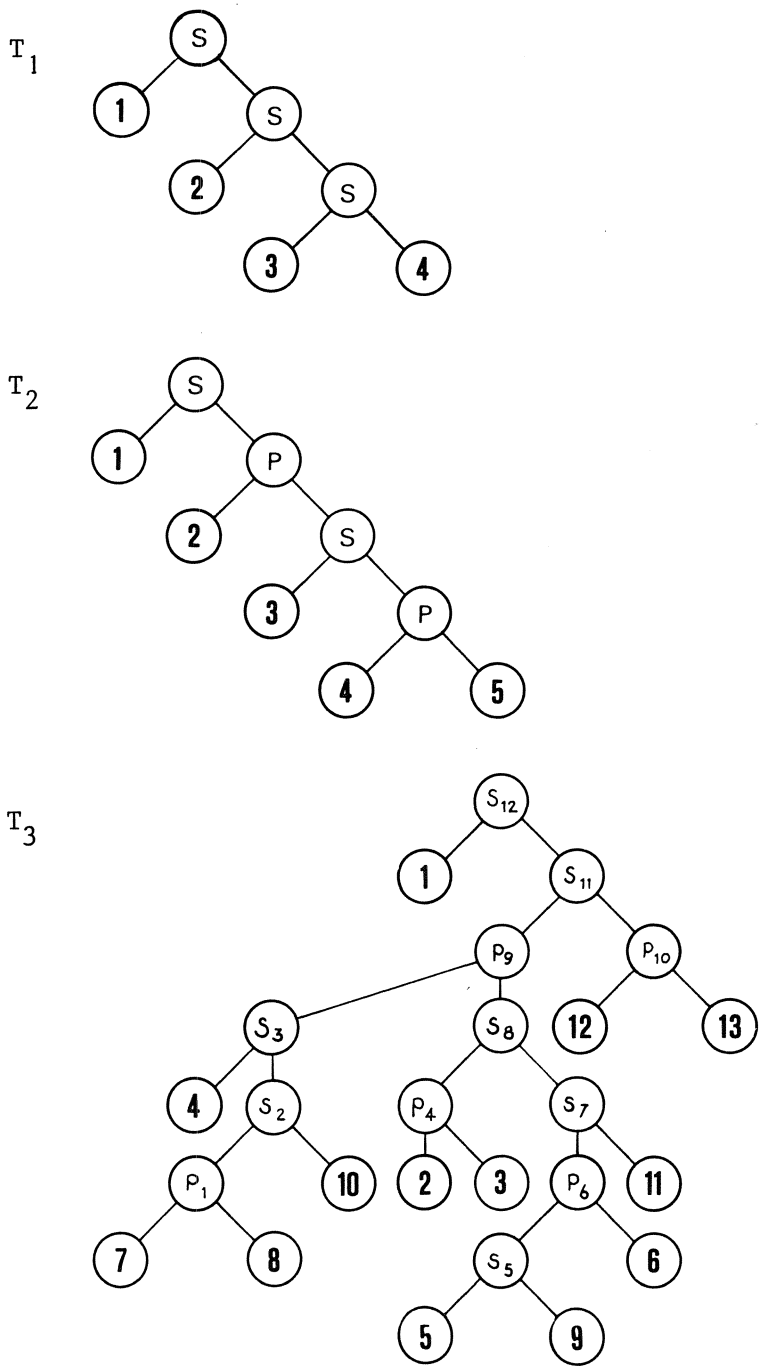

Figure 3 Decomposition Trees.