数理解析研究所講究録
第427巻 1981年 165-174

Scheduling in File Processing and Related Problems

T. Kameda
Dept. of Elec. Eng.
Univ. of Waterloo, Canada

1. Introduction

A stage of file processing may require input files stored in a certain number (n_1) of magnetic tapes and may produce output files in a certain number (n_2) of magnetic tapes. We represent this stage by a node labeled by (n_2-n_1) in a graph. (See Fig. 1) Edges of the graph represent the precedence constraints among the stages, which are in general a partial order. A totally ordered sequence of the nodes consistent with the given precedence constraints will be called a schedule. We consider the following two problems:

(1) Find an M-optimal schedule, which minimizes the maximum number of tapes required during processing.

(2) Associate with each node the processing time t_i. Find an A-optimal schedule, which minimizes the total tape·hours.

Solution to the problem (1) is relevant to the feasibility of processing with the available number of tapes, and (2) is relevant to the minimization of resource (i.e., tape) cost over time.

Consider the example of Fig. 1, where the nodes represent processing stages and the edges represent precedence constraints.

For the stage C, for example, two more input tapes are used than output tapes. We call this 2 the cost of stage C. Suppose stages are executed sequentially according to a certain schedule. For a partially executed schedule, the cumulative cost (CC) at this point is the sum of the costs of the stages that have already been executed. For example, Fig. 2 shows the change of the CC for the schedule σ_1 = ABCDEF for the example of Fig. 2. Note that it reaches the maximum value of 6 when the stage B is being executed. We call this the M-value of σ_1 and express it by $M(\sigma_1) = 6$. Consider another schedule σ_2 = ACBEDF of the same example. As one can easily verify, $M(\sigma_2) = 5$. It turns out that σ_2 is an M-optimal schedule for this problem.

Now suppose that the processing time for node i is t_i. Then during the execution of stage i, a certain number of tapes, which equals the CC at this point, are used for the duration t_i. We define the A-value of a schedule to be the sum of cumulative costs weighted by the processing times. More precisely,

$$A(\sigma) = c_1t_1 + (c_1+c_2)t_2 +\ldots+ (c_1+\ldots+c_n)t_n$$

where $\sigma = S_1S_2\ldots S_n$ and c_i and t_i are cost and processing time, respectively, of the stage S_i. If $t_A = t_B = t_C = t_E = t_F = 1$ and $t_D = 4$ in the example of Fig. 1, then $A(\sigma_1) = 35$ and $A(\sigma_2) = 36$, as the reader can easily verify. From the above examples, it is seen that an M-optimal schedule is not necessarily an A-optimal schedule.

2. Related Problems

1) The register allocation problem [9] is encountered in

the evaluation of an arithmetic expression using a computer. Since the hardware registers are scarce resource, it is desirable to use as few of them as possible. The expression A*B + C may be represented by Fig. 3(a). Using our convention the multiplication node is labeled by -1 (Fig. 3(b)), since the operands are stored in two registers and the result appears in one. Nodes A, B, and C are labeled by 1, since these operands must first be placed in registers. This expression can be evaluated using only two registers, since M(ABDCE) = 2. In the above example, the precedence constraints are represented by a tree. In a more complex case, where the precedence constraints are represented by a general acyclic digraph, the problem of finding an M-optimal schedule is NP-hard [9]. Such an example is shown in Fig. 4. Note that we introduced a dummy node labeled by -1 in Fig. 4(b).

2) In the minimum weighted completion time problem, we are given a number of jobs, S_1, S_2, ..., S_n. Associated with each job S_i is its processing time t_i (>0) and a weight w_i, which may be negative. Also given is a precedence relation among the jobs. The weighted completion time (WCT) for a schedule $\sigma = S_1 S_2 \ldots S_n$ is defined by

$$WCT(\sigma) = t_1 w_1 + (t_1+t_2)w_2 + \ldots + (t_1+\ldots+t_n)w_n$$

This is superficially similar to the formula for $A(\sigma)$, if we replace t_i by c_i and w_i by t_i. However, such a replacement is not allowed, because $t_i > 0$, and c_i and w_i can be negative. If we rewrite $WCT(\sigma)$ as

$$WCT(\sigma) = w_n t_n + (w_n+w_{n-1})t_{n-1} +\ldots+ (w_n+\ldots+w_1)t_1$$

and replace w_{n-i+1} by c_i and t_{n-i+1} by t_i, then we see that the two problems are equivalent.

3. Difficulty Results

1) M-optimal scheduling problem is NP-complete.

This follows from a result of Sethi [9] that the register allocation problem is NP-complete, since the register allocation problem is polynomially tranformable to our problem (cf. Fig. 4).

2) A-optimal scheduling problem is NP-complete.

This follows from a result of Lawler [6] that the minimum weighted completion time problem is NP-complete and the equivalence of the two problems as shown in Section 2.

Because of the above results, we consider in Section 5 a special case where precedence constraints are series-parallel and develop an efficient algorithm for this special case.

4. Adjacent Sequences Interchange

We shall introduce the concept of adjacent sequences interchange (or ASI) property [7] in terms of the A-optimal scheduling problem. For the sake of simplicity consider first scheduling of only two stages, $S_1 = (c_1, t_1)$ and $S_2 = (c_2, t_2)$, where c_i and t_i denote the cost and processing time of stage i, respectively. Suppose there is no precedence constraint between S_1 and S_2. By definition, we have

$$A(S_1S_2) = c_1t_1 + (c_1+c_2)t_2, \text{ and}$$

$$A(S_2S_1) = c_2t_2 + (c_2+c_1)t_1$$

and therefore,

$$A(S_1S_2) - A(S_2S_1) = t_1t_2(c_1/t_1-c_2/t_2).$$

If we define the rank of S_i by $r(S_i) = c_i/t_i$, then $A(S_1S_2)-A(S_2S_1) = t_1t_2(r(S_1)-r(S_2))$, and thus $A(S_1S_2) \leq A(S_2S_1)$ iff $r(S_1) \leq r(S_2)$.

We now generalize the concept of the rank and define it for arbitrary sequences. Let $C(\sigma)$ be the sum of all costs, c_i, such that S_i belongs to the sequence σ, and $T(\sigma)$ be the sum of all processing times, t_i, such that S_i belongs to the sequence σ. Then $A(\sigma)$ can be defined recursively as follows.

$$\begin{cases} A(S_i) = c_it_i \\ A(\sigma_1\sigma_2) = A(\sigma_1) + A(\sigma_2) + C(\sigma_1)T(\sigma_2) \text{ for sequences } \sigma_1 \text{ and } \sigma_2. \end{cases} \quad (1)$$

We thus have

$$A(\sigma_1\sigma_2) - A(\sigma_2\sigma_1) = T(\sigma_1)T(\sigma_2)[C(\sigma_1)/T(\sigma_1)-C(\sigma_2)/T(\sigma_2)]$$

and so $A(\sigma_1\sigma_2) \leq A(\sigma_2\sigma_1)$ iff $r(\sigma_1) \leq r(\sigma_2)$, where rank $r(\sigma)$ of σ is defnied by

$$r(\sigma) = C(\sigma)/T(\sigma). \quad (2)$$

Lemma 1:

For arbitrary sequences α, β, σ_1, and σ_2, such that $\alpha\sigma_1\sigma_2\beta$ and $\alpha\sigma_2\sigma_1\beta$ are both legal schedules, we have $A(\alpha\sigma_1\sigma_2\beta) \leq A(\alpha\sigma_2\sigma_1\beta)$ iff $r(\sigma_1) \leq r(\sigma_2)$.

Proof: By the recursive definition of A, we have

$$\begin{aligned} A(\alpha\sigma_1\sigma_2\beta) &= A(\alpha\sigma_1) + A(\sigma_2\beta) + C(\alpha\sigma_1)T(\sigma_2\beta) \\ &= A(\alpha)+A(\sigma_1)+A(\sigma_2)+A(\beta)+C(\alpha)[T(\sigma_1)+T(\sigma_2)+T(\beta)] \end{aligned}$$

$$+ C(\sigma_2)T(\beta) + C(\sigma_1)[T(\sigma_2) + T(\beta)].$$

We thus have

$$A(\alpha\sigma_1\sigma_2\beta) - A(\alpha\sigma_2\sigma_1\beta) = T(\sigma_1)T(\sigma_2)[r(\sigma_1)-r(\sigma_2)].$$

The lemma follows directly from this equation. Q.E.D.

A cost function f is said to satisfy the ASI property [7] if there exists a rank r_f such that $f(\alpha\sigma_1\sigma_2\beta) \leq f(\alpha\sigma_2\sigma_1\beta)$ for all α, σ_1, σ_2, and β iff $r_f(\sigma_1) \leq r_f(\sigma_2)$. Lemma 1 shows that the cost function A satisfies the ASI property (let $r_A(\sigma) = C(\sigma)/T(\sigma)$). The ASI property plays an important role in what follows, i.e., efficient algorithms for the case where precedence constranis are series-parallel.

A subset S of jobs (or stages) is said to be a module if each job not in S is constrained by the precedence relation

a) to precede all jobs in S, or

b) to succeed all jobs in S, or

c) to be before or after any job in S, even between jobs of S (i.e., no constraint).

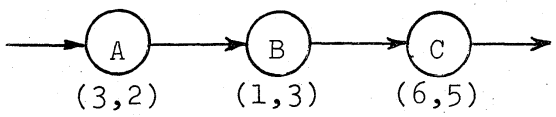
The following lemma is valid for general precedence constraints.

Lemma 2 [7]:

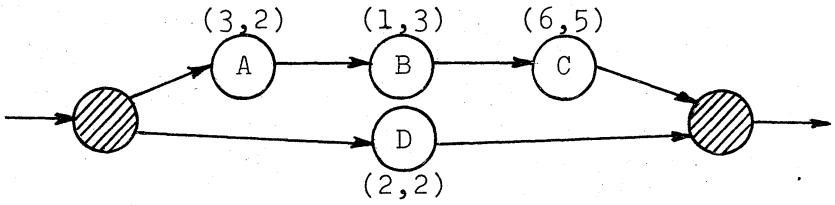
Let f satisfy the ASI property and let $\alpha\sigma_1\sigma_2\beta$ be a schedule such that σ_1 and σ_2 are chains (in the precedence graph), $\sigma_1\sigma_2$ is a module, $\sigma_2\sigma_1$ is not allowed , and $r_f(\sigma_1) \geq r_f(\sigma_2)$. Then there is an optimal schedule in which σ_1 immediately precedes σ_2.

5. Series-Parallel Precedence

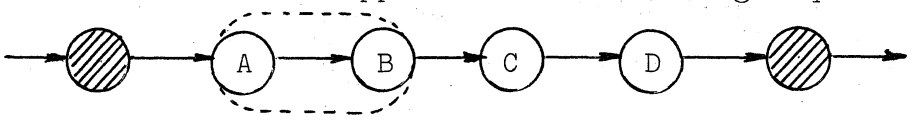
Suppose there is a chain that is also a module. The figure



is such an example, where the node lables are (c, t) pairs. By computing ranks we find $r(A) = 3/2 > r(B) = 1/3$. According to Lemma 2, we may always assume A immediately precedes B in the search for an A-optimal schedule. For the chain AB, we have $r(AB) = (3+1)/(2+3) < r(C) = 6/5$, and therefore we can not assume C immediately succeeds AB. Consider now the following module. Such a module always exists in a series-parallel graph.



We have already seen that A and B can be considered as a single node AB. Since $r(AB) < r(D) < r(C)$, there is an optimal schedule in which these nodes appear in the following sequence. Start-



ing with a series-parallel graph, if a loop is replaced by a chain in this way repeatedly, we end up with a single chain, i.e., an optimal schedule. A straightforward implementation of the above procedure requires $O(n^2)$ time [1].

In the above we have obtained an A-optimal schedule when the precedence constraints were series-parallel. The same algorithm can be used to obtain an f-optimal schedule, provided f has the ASI property. By using a clever data structures, the

time complexity of the algorithm can be improved.

Theorem 1 [7]:

If a cost function f satisfies the ASI property, an f-optimal schedule can be obtained in O(nlogn) time in the case where the precedence constraints are series-parallel.

The algorithm is a generalization of the one given by Lawler [6].

6. M-optimal Schedules

In this section we shall show that the M-function has the ASI property, so that the algorithm of the previous section can be used to obtain an M-optimal schedule. For an arbitrary string σ, we define $M(\sigma)$ to be the maximum cumulative cost, i.e.,

$$M(\sigma) = \text{Max}\{C(\alpha) \mid \alpha \text{ is a prefix of } \sigma\}.$$ (See Fig. 2.)

We introduce a rank function for M by

$$r_M(\sigma) = \begin{cases} -1/M(\sigma) & \text{if } C(\sigma) < 0 \\ 0 & \text{if } C(\sigma) = 0 \\ 1/[M(\sigma)-C(\sigma)] & \text{if } C(\sigma) > 0 \end{cases}$$

Lemma 3:

The function M satisfies the ASI property.

Proof: It can be shown that $M(\alpha\sigma_1\sigma_2\beta) \leq M(\alpha\sigma_2\sigma_1\beta)$ iff $r_M(\sigma_1) \leq r_M(\sigma_2)$. See [2] for details. Q.E.D.

Monma has recently shown that the M-optimal scheduling problem is equivalent to Johnson's two-machine maximum completion time problem [3,5,8].

References

[1] Abdel-Wahab, H.M. and Kameda, T., Scheduling to minimize the maximum cumulative cost subject to series-parallel precedence constraints, Operations Research 26 (1977), 141-159.

[2] _____ and _____, On strictly optimal schedules for the cumulative cost-optimal scheduling problem, Computing 24 (1980), 61-86.

[3] Baker, K.R., Introduction to sequencing and scheduling, John Wiley and Sons, 1974.

[4] Housel, B.C., Pipelining: a technique for implementing data restructures, ACM Trans. on Database Systems 4 (1979), 470-494.

[5] Johnson, S.M., Optimal two- and three-stage production schedules with setup times included, Naval Res. Logist. Quart. 1 (1954), 61-68.

[6] Lawler, E.L., Sequencing jobs to minimize total weighted completion time subject to precedence constraints, Ann. of Disc. Math. 2 (1978), 75-90.

[7] Monma, C.L. and Sidney, J.B., Sequencing with series-parallel precedence constraints, Math. of Opns. Res. 4 (1979), 215-224.

[8] Monma, C.L., Sequencing to minimize the maximum job cost, Submitted for publication.

[9] Sethi, R., Complete register allocation problems, SIAM J. Computing, 4 (1975), 226-248.

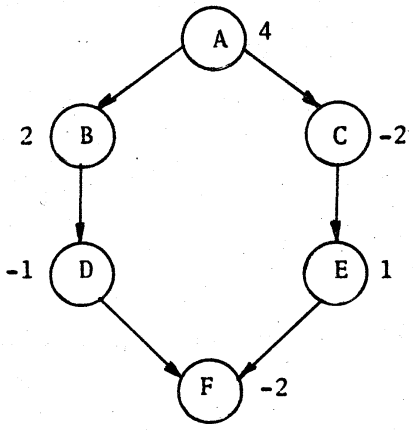


Fig. 1. An example

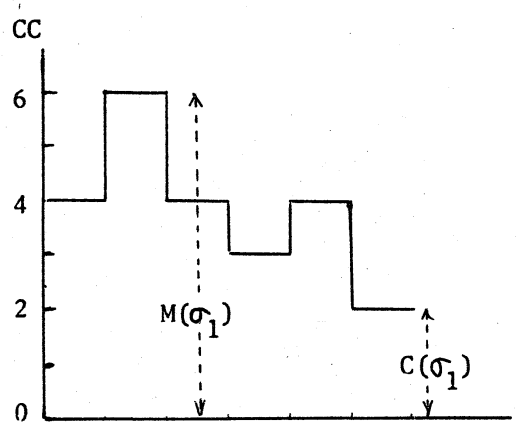
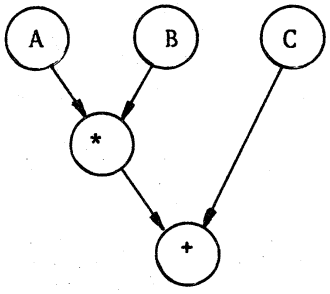
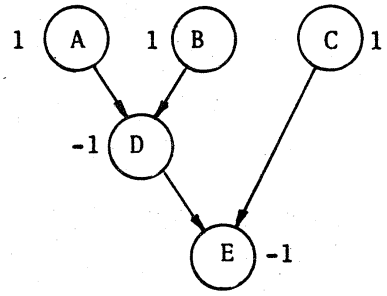


Fig. 2. CC for σ_1 = ABCDEF

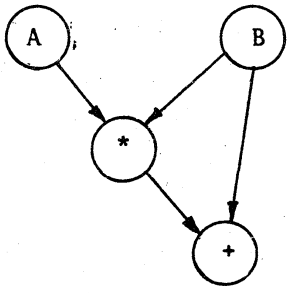


(a) Representation for A*B+C

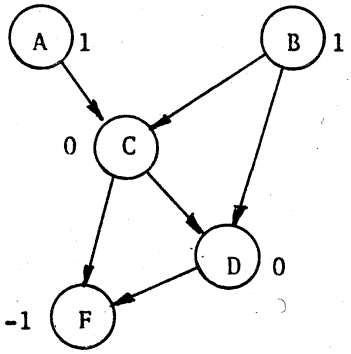


(b) Formulated in our model

Fig. 3. Register allocation for A*B+C



(a) Representation for A*B+B



(b) Formulated in our model

Fig. 4. Example of a general precedence