SCHEDULING DAGS TO MINIMIZE TIME AND COMMUNICATION

Foto Afrati[1], **Christos H. Papadimitriou**[2], **and George Papageorgiou**[1]

ABSTRACT: *We study the complexity of a generalization of the unit-execution-time multiprocessor scheduling problem under precedence constraints, in which the number of communication arcs is also minimized. Most versions of the problem are shown NP-complete, and two polynomial algorithms are presented for specialized cases.*

1. INTRODUCTION

Scheduling partially ordered tasks with equal execution times on identical processors to minimize the latest finishing time is one of the best-studied problems in Computer Science (see [Co] for an early review of related results). When the partial order is a tree, [AH] showed that the highest-level-first strategy is optimal. For general graphs, the case in which two processors are available can be solved in polynomial time [CG]; the case in which the number of processors is part of the input is NP-complete; and the cases with 3, 4, etc. processors remains open.

This problem has attracted so much attention mainly because of its obvious relevance to multiprocessor systems. However, in such systems finishing time is only one of the performance criteria, the other being the *communication* required among the processors [PU]. We measure communication as the number of arcs of the precedence relation, the two end points of which are executed by different processors. For example, the tree shown in Figure 1 can be executed by two processors in 3 time units and 2 units of communication (or, of course, in 5 time units and 0 units of communication by a single processor). Can it be executed in both three time units *and* a single unit of communication? The answer is "no."

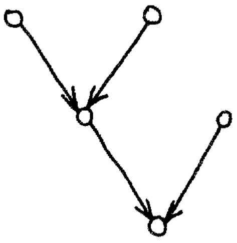

Figure 1: An example.

It turns out that the two-parameter version is a very interesting new twist to this classical problem. In this paper we study the complexity of the following problem: Given a dag (sometimes a tree), a number m of processors (fixed, part of the input, or infinite), and two positive integers t and c, can we schedule the dag on m processors within time t and communication c? Notice

[1] National Technical University of Athens.
[2] University of California at San Diego.

that the case of infinite processors is trivial when communication is not taken into account, as the optimum time is precisely the depth of the dag. It turns out that most of these versions are NP-complete. In particular, for general dags, when the number of processors is fixed (even 2) the problem is NP-complete, and likewise for infinite processors. For trees, the case of infinite processors, or a number of processors that is part of the input, are both NP-complete. Finally, scheduling a tree on two processors (in fact, on any fixed number of processors) is a very interesting problem left open here.

We also present algorithms for two somewhat interesting special cases of the problem. The first is that of *layered dags*, that is, dags for which the number of nodes in any maximal path is the same, and equal to t, the time bound. Using matching techniques, we can solve this problem in time $O(n^{2.5})$. Finally, when the amount of communication c is fixed, we show that the problem for general dags is polynomial. Notice that the original scheduling problem (no communication) is NP-complete even when $t = 3$ [PS], so the two parameters t and c appear to behave very differently in this respect.

2. NP-COMPLETENESS RESULTS

In the *two-parameter scheduling problem* we are given a dag $G = (V, E)$ and three integers m, t, and c. We are asked to find a schedule S, that is, a one-to-one (not necessarily onto) mapping from the nodes of V to $\{1, 2, \ldots, t\} \times \{1, 2, \ldots, m\}$ such that: (1) if $(u, u') \in E$ and $s(u) = (\tau, \mu)$, $s(u') = (\tau', \mu')$, then $\tau < \tau'$ (that is, S respects precedences), and (2) the set of arcs $(u, u') \in E$ such that $s(u) = (\tau, \mu)$, $s(u') = (\tau', \mu')$, and $\mu \neq \mu'$ has cardinality at most c. We shall call the same problem without restriction (2) the *original* scheduling problem.

The problem for general dags is trivially NP-complete, since the NP-complete original scheduling problem (with the communication bound c sufficiently large, say $|E|$) is a special case. We now show that it remains NP-complete even if G is a tree (recall that the original problem is polynomial for trees [AH]).

Theorem 1. The two-parameter scheduling problem for trees is (strongly) NP-complete.

Sketch: We reduce the 3-PARTITION problem [GJ] to it. We are given a set of $3m$ integers $a_1, \ldots, a_{3m}$, all between $\frac{B}{4}$ and $\frac{B}{2}$, and we are asked whether they can be divided into m triples, all adding up to B. Now, for each a_i construct a *tassel* of size a_i (that is, an in-tree with a_i leaves and depth one, see Figure 2), and then connect the roots of all tassels to a common root. We argue that the resulting tree can be scheduled on m processors in time $B + 4$ and communication $3m - 3$ if and only if the 3-PARTITION instance has a solution. □

The problem, for general dags, remains NP-complete even when the number of processors is two (compare with [CG]):

Theorem 2. The two-parameter scheduling problem with two processors is NP-complete.

Sketch: The reduction is from ONE-IN-THREE 3-SAT, in which we are given clauses with three literals each, and asked whether there is an assignment such that exactly one literal in each clause is made true. We employ two "gadgets," shown in Figure 3(a) and 3(b) (M is a sufficiently large number). There is one copy of the dag in Figure 3(a) for each variable, and one copy of the dag in Figure 3(b) for each clause. We think that the three middle nodes of the clause dag (Figure 3(b)) correspond to the three literals of that clause. We connect all these dags in series, identifying the sink of one with the source of the next, with the dags corresponding to varables first. Then we draw an arc from all M nodes in the left (respectively, right) path of a variable dag (Figure 3(a)) to the middle nodes of clause dags that correspond to the positive (respectively, negative) occurrences of the variable. It can now be shown that we can schedule the dag on two processors

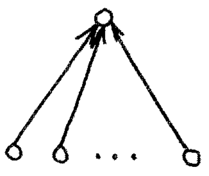

Figure 2: A tassel.

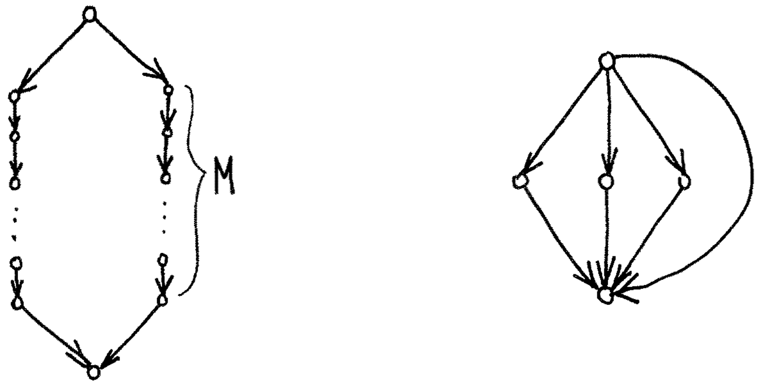

Figure 3: The two gadgets.

within time $n(M+1)+3m+1$ and communication $2n+3m$ (where n is the number of variables and m the number of clauses) if and only if there is a satisfying truth assignment. □

Whether the above result holds when G is a tree is a question that we have been unable to settle (as with any other fixed number of processors). When the number of processors is infinite, the problem is NP-complete even for trees. The reduction is from EXACT COVER BY 3-SETS, and is omitted.

Theorem 3. The two-parameter scheduling problem for trees and an infinite number of processors is NP-complete. □

3. TWO ALGORITHMS

Let us call a dag *layered* if all maximal paths have the same length, called the *depth* of the dag.

Theorem 4. We can solve the two-parameter scheduling problem with infinite processors for layered dags with depth is equal to t in $O(|V|^{2.5})$ time.

Sketch: Minimizing the communication can be shown to be equivalent to determining maximum bipartite matchings between subsequent layers. □

Finally, when the communication bound c is fixed, the problem can also be solved in polynomial time:

Theorem 5. The two-parameter scheduling problem can be solved in time $O(|V|^{4c+2}\log n)$.

Sketch: By exhaustively examining all c-tuples of arcs, all assignments of processors to the weakly connected components formed by deleting each c-tuple, and all possible times for executing the tails of communication arcs, the problem reduces to single-processor scheduling with release times and deadlines [La], solvable in $O(n\log n)$ time. The total time required is $O(|E|^c \min(m, c+1)^{c+1} t^c |V| \log|V|)$, dominated by the stated bound (naturally, we can assume that $t < |V|$). □

Note that, in contrast, fixing the other parameter, t, to any constant greater than 2 does not affect the NP-completeness of the problem (in fact, of the original problem) [PS].

4. RELATED WORK

Our definition of communication (adapted from [PU]) is slightly inaccurate, in that we may overestimate communication by counting twice the communication corresponding to two arcs emanating from the same node v and leading to two different nodes, if these two nodes are executed by the same processor (but different from the processor that executed v). This is of no major consequence for dags with bounded outdegree (such as those studied in [PU], or those used in our NP-completeness constructions). By using variants of our methods as well as more sophisticated techniques, [Pr] has established that our NP-completeness results can actually be extended to the more accurate variant of the problem, in which such arcs are counted only once.

There are two even more accurate versions of the problem. First, the one in which we wish to minimize the weighted sum $t + \tau c$, for some constant τ denoting the number of processor cycles it takes to send a message from a processor to another. This problem is also NP-complete [Pe], essentially under the same assumptions. Even closer to the desired performance measure, consider the scheduling problem in which (1) tasks may be executed more than once in the processors (another inaccuracy of the current formulation), and (2) the endpoints of an arc executed in different processors must be at least τ time units apart (if the tail has many instantiations, we take the least restrictive one, that is, the latest or the one executed on the same processor). We wish to minimize finish time. This problem is shown NP-complete in [PY]. In that paper we also give a very simple linear-time heuristic that approximates the optimum within a ratio of 2. We argue that this can be a useful tool for analysing parallel algorithms in a manner that is independent of the underlying architecture.

REFERENCES

[AH] D. Adolphson and T. C. Hu "Optimal Linear Ordering", *SIAM J. Appl. Math.*, 25, pp. 403-423, (1973).

[CG] E. G. Coffman, R. L. Graham "Optimal Scheduling for Two Processor Systems", *Acta Informatica*, 1, pp. 200-213, (1972).

[Co] E. G. Coffman, ed. *Computer and Jobshop Scheduling Theory*, Wiley, 1978.

[GJ] M. R. Garey, D. S. Johnson *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, 1979.

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

[PS] C. H. Papadimitriou, K. Steiglitz *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1983.

[PU] C. H. Papadimitriou, J. D. Ullman "A Communication-Time Trade-off", *Proc. 1985 STOC Conference*; also, *SIAM J. Comp.*, 1987.

[PY] C. H. Papadimitriou, M. Yannakakis "Towards an Architecture-Independent Analysis of Parallel Algorithms", *Proceedings 1988 STOC*, to appear.

[Pe] E. Petrohilos, Diploma Thesis, National Technical University of Athens, 1986 (in Greek).

[Pr] M. Prastein, manuscript, University of Illinios at Urbana-Champaign, 1987.