SIAM J. COMPUT.
Vol. 14, No. 3, August 1985

010

SCHEDULING FLAT GRAPHS*

DANNY DOLEV† AND MANFRED WARMUTH‡

Abstract. The problem of scheduling a partially ordered set of unit length tasks on m identical processors is known to be NP-complete. There are efficient algorithms for only a few special cases of this problem. In this paper we analyze the effect of the structure of the precedence graph and the availability of the processors on the construction of optimal schedules. We prove that to find an optimal schedule it suffices to consider at each step only initial tasks which belong to the $m-1$ highest components of the precedence graph. This result reduces the number of cases we have to check during the construction of an optimal schedule. Our method leads to polynomial algorithms if the number of processors is fixed and the precedence graph has a certain form. In particular, if the precedence graph contains only intrees and outtrees, this result leads to linear algorithms for finding an optimal schedule on two or three processors.

Key words. identical processors, profile, optimal schedule, intree and outtree

1. Introduction. The goal of deterministic scheduling is to obtain efficient algorithms under the assumption that all the information about the tasks to be scheduled is known in advance. One of the fundamental problems in deterministic scheduling is to schedule a set of unit length tasks, subjected to precedence constraints, on a system of identical processors. The precedence constraints between tasks are represented by a *precedence graph*, which is a directed acyclic graph. As in [GJ81] we allow the number of identical processors to vary with time. A *profile* is a sequence of natural numbers specifying how many processors are available at each time slot. A schedule for a given profile is a partitioning of all the tasks into a sequence of sets which does not violate the precedence graph. The ith set of the sequence is scheduled in the ith time slot (i.e. interval $[i-1, i)$). Thus, the cardinality of the ith set cannot exceed the number of processors which are available in the ith time slot of the profile. A profile is *straight* if it has the same number of processors available at each time slot. The *breadth* of a profile is the maximum number of processors available at any time slot.

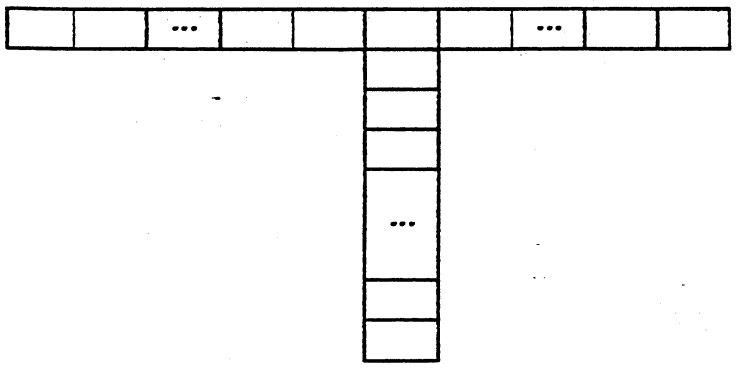
Various aspects of scheduling theory have been studied extensively in recent years [GL79] and many scheduling problems are known to be NP-complete [Ul75], [GJ79], [LR78], [Wa81], [GJ81], [Ma81]. The first NP-completeness result on scheduling with precedence constraints was published by Ullman [Ul75]. He showed that the existence of a schedule of a given length on a straight profile for a collection of unit length tasks subjected to some given precedence constraints is NP-complete, if the number of available processors is a variable of the problem. Notice that the breadth of the profile is not bounded by a constant. The problem remains NP-complete even for certain classes of precedence graphs [GJ81], [Ma81], [Wa81]. To support the idea that the breadth of the profile is the main source of NP-completeness we prove in [DW82b] that scheduling unit length tasks is NP-complete even if the precedence graph has height one and the profile has one processor available in each slot except for one slot that has an arbitrary number (see Table 1.1). Polynomial algorithms have been developed for only a few special cases. The first polynomial algorithm was developed

* Received by the editors December 23,1980, and in final revised form March 23, 1984. This paper is a revision of IBM Research Report RJ3398, 1982.

† Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel. Part of this work was done while this author visited IBM Research, San Jose, California.

‡ Computer Science Department, University of California, Santa Cruz, California 95064. Part of this work was done while this author visited the Hebrew University, Jerusalem. The research was supported by the Fulbright Commission of West Germany, grants from Univac Corporation and Storage Technology Corporation, and by the United States-Israel Binational Science Foundation under grant 2439/82.

TABLE 1.1
The question of existence of a schedule for a precedence graph of height one and a profile of the below form is NP*-complete.*



by Hu [Hu61]. It produces an optimal schedule for a straight profile of arbitrary breadth if the precedence graph is an inforest. Hu's algorithm produces a schedule according to the *Highest Level First* (HLF) strategy, meaning tasks of higher level are chosen over tasks of lower level and tasks of the same level are chosen arbitrarily. HLF also produces an optimal schedule for outforests and straight profiles of arbitrary breadth [Br81]. A restricted version of HLF provides an optimal schedule when the precedence graph is an interval order [PY79], [Ga82], or if the number of available processors is two [Ga81]. Recently, polynomial algorithms have been published [GJ81], [Wa81], [DW82c] for scheduling certain classes of precedence graphs on profiles of fixed breadth. In [Wa81], [DW82a] it was also shown that scheduling an arbitrary graph on a profile of fixed breadth is polynomial, if the height of the graph is bounded by a constant.

The major scheduling problem remaining open is whether the scheduling of an arbitrary graph of unbounded height is NP-complete or polynomial for a fixed number ($m \geqq 3$) of processors.

Let m be the breadth of the profile. The median (see § 3) of the precedence graph is defined to be one plus the height of the mth highest component of the precedence graph (see Fig. 3.1) and if the precedence graph contains less than m components, then the median is zero. A task is *initial* if it does not have any predecessors. The *Elite* of the precedence graph is the set of all initial tasks that belong to components that are higher than the median.

Our main result, the Elite theorem (§ 4), states that it is enough to choose tasks from the Elite of the precedence graph for the first slot of an optimal schedule. If we do not have enough tasks in the Elite, then we choose tasks according to highest height from the set of initial tasks that are not in the Elite. After filling the first slot, the Elite theorem can be applied to the remaining precedence graph and the next slot, and so on. The theorem restricts the number of cases that need to be considered for constructing an optimal schedule. Variations of the Elite theorem are the basis for the algorithms in [Wa81], [DW82c].

In § 5, we generalize results of [Hu61], [Br81], and [GJ81] by applying the Elite theorem. We show that HLF produces an optimal schedule if the precedence graph is either an inforest or an outforest and the profile is of a certain type. In § 6 we prove some properties of graphs containing only inforest and outforest components (opposing forests). In § 7 we use these properties to develop a linear algorithm for scheduling an opposing forest on a straight profile of breadth three improving the $O(n \log n)$ time

bound of Garey et al. [GJ81].[1] Furthermore, we give an $O(n \log n)$ algorithm for scheduling an opposing forest on a profile that has two or three processors available at any time slot. The algorithm is essentially the one described in [Do80].

2. Basic definitions and properties. A precedence graph G is denoted by a tuple (V, E), where V is the set of n *tasks* and E the set of *edges* of G. A (*directed*) *path* π of length r in G is a sequence of tasks $x_0, \cdots, x_r$ such that the edge (x_i, x_{i+1}), for $0 \leqq i \leqq r-1$, is in E. We assume that if a task x has to be executed before a task y, then there exists a (directed) path from x to y in G. Note that G is acyclic.

If there exists a path from x to y, then x is a *predecessor* of y, and y is a *successor* of x. In the case where the longest path from a task x to a task y is the edge (x, y), we call x the *immediate predecessor* of y and y the *immediate successor* of x.

By $h(G)$ we mean the *height* of G, which is the length of the longest path in G. For a task $x \in G$ (i.e., $x \in V$) we denote by $h(x)$ the length of the longest path that starts at x. Note that a task with no successors has zero height. Tasks with identical height are said to be at the same *level*.

The graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$, denoted by $G' \subseteq G$, if $V' \subseteq V$ and for all x and y in G', x is a predecessor of y in G' if and only if x is a predecessor of y in G. A subgraph G' of G is called a *closed subgraph* if every task in G' has the same successors in G' as it has in G. For two graphs $G = (V, E)$ and $G' = (V', E')$, $G \cup G'$ denotes the graph $(V \cup V', E \cup E')$. The graph $G = (V, E)$ is composed of $\{G_1, \cdots, G_r\}$ if these subgraphs (called *components* of G) are a decomposition of G into its connected components, that is, each subgraph is a nonempty connected graph and there are no edges between tasks of different components; therefore, $G = \cup_i G_i$. A task of G is *initial* if it has no predecessors. Note that an initial task of G is not necessarily of maximum height in G. A set of k *highest initial tasks* is a subset of the set of initial tasks consisting of some k highest ones; when there are less than k initial tasks, it contains all of them. Let R be a set of initial tasks of a precedence graph G. Then $G - R$ is the subgraph of G obtained by removing the tasks of R.

We partition the time scale into time slots of length one. The time interval $[i-1, i)$ for $i \geqq 1$ is the ith time slot. A *profile*, M, is a sequence of positive integers, $(m_1, m_2, \cdots, m_d)$, specifying the number of identical processors, m_i, that are available in each time slot i, for $i \leqq i \leqq d$ (see Table 2.1); d is the *length* of the profile M. The *breadth* of profile M is the maximum number of processors that is available at any time slot of M. Throughout the paper we denote the breadth of the given profile with the letter m. The profile of Table 2.1 has breadth three. We call a profile M *straight* if $m_i = m$, for all $1 \leqq i \leqq d$.

A *schedule* S for a precedence graph G is a sequence of sets $(S)_1 | \cdots | (S)_k$ such that:

i) the sets $(S)_i$, for $1 \leqq i \leqq k$, partition the tasks of G;

ii) if $x \in (S)_i$ and $y \in (S)_j$, for $1 \leqq i \leqq j \leqq k$, then there is no path from y to x.

The *length* of a schedule S, denoted by $\lambda(S)$, is the index of the last nonempty set in the sequence. A minimum length schedule is called *optimal*. The schedule S *fits* the profile M if the length of S is not greater than the length of the profile and the cardinality of $(S)_i$ is not greater than m_i. The set of tasks $(S)_i$ gets executed in the ith time slot, that is $|(S)_i|$ of the m_i processors of slot i are executing the tasks of $(S)_i$ during the time interval $[i-1, i)$. Note that the length of a task equals the length of a time slot. We call the schedule S an *M-schedule* for G.

[1] In a revised version of [GJ81] Garey et al. also obtain a linear algorithm.

As an example assume we have a set of twelve tasks subject to the precedence aph G presented in Fig. 2.1. We name the tasks by numbers. Throughout the paper e always assume that the edges of the graphs are directed downwards. We look for schedule for G that fits the profile $M = (2, 3, 3, 1, 3, 2)$. The following sequence S is valid schedule:

$$\{1, 2\}|\{3, 4, 5\}|\{6, 7\}|\{8\}|\{9, 10, 11\}|\{12\}.$$

he M-schedule S can be shown as in Table 2.1. Notice that $\lambda(S) = 6$. In Table 2.2 a chedule S' of length 5 is given for the same precedence graph, which is optimal.

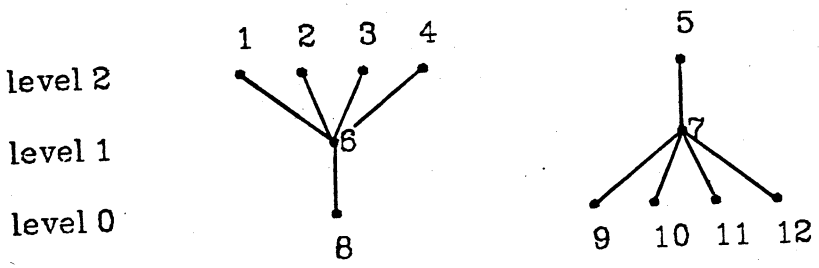


FIG. 2.1. *The precedence graph G.*

TABLE 2.1
The schedule S for the precedence graph G of Fig. 2.1 and the profile M = (2, 3, 3, 1, 3. 2).

slot	1	2	3	4	5	6
P_1	1	3	6	8	9	12
P_2	2	4	7		10	
P_3		5			11	
m_i	2	3	3	1	3	2

TABLE 2.2
The schedule S' for G and M.

slot	1	2	3	4	5	6
P_1	1	2	4	6	8	
P_2	5	3	9		11	
P_3		7	10		12	
m_i	2	3	3	1	3	2

The ith slot of a schedule S, $1 \leqq i \leqq \lambda(S)$, has $m_i - |(S)_i|$ *idle periods.* Such an idle period corresponds to a processor being idle during time slot i of S.

A schedule S is an HLF-*schedule* for G and M if $(S)_i$, $1 \leqq i \leqq \lambda(S)$, is a set of m_i highest initial tasks of the subgraph of G induced by all tasks scheduled in slot i of S or later. Note that in the above example S is a HLF-schedule, whereas S' is not. HLF-schedules have the following property. Assume task x is scheduled in slot i and y is scheduled in slot j. If $h(x) > h(y)$, then either $i \leqq j$ or there is a predecessor of x in the jth slot. We say that HLF *produces an optimal schedule* if any HLF-schedule is

optimal; that is, if an optimal schedule can be constructed by choosing higher initial tasks before lower ones and choosing arbtrarily among initial tasks of the same height.

A schedule for G is greedy if whenever there is an idle period in some slot i then this slot contains all initial tasks of the subgraph of G induced by the tasks that appear in slot i or later. It is easy to see that any schedule can be made into a greedy one without increasing its length; thus there exists greedy schedules which are optimal.

3. The median. In this paper we study graphs that have more than m components ("flat" graphs), where m is the breadth of the profile. We use the notion of the median to characterize this property of a precedence graph.

DEFINITION. The *median* of a precedence graph G with respect to a given breadth m, denoted by $\mu(G)$, is one plus the height of the mth highest component of the precedence graph.

Thus the graph of Fig. 3.1 has median 3 with respect to $m=3$. If the graph has fewer than m components the median is 0. For example, in the graph described by Fig. 2.1 the median with respect to $m=3$ is 0.

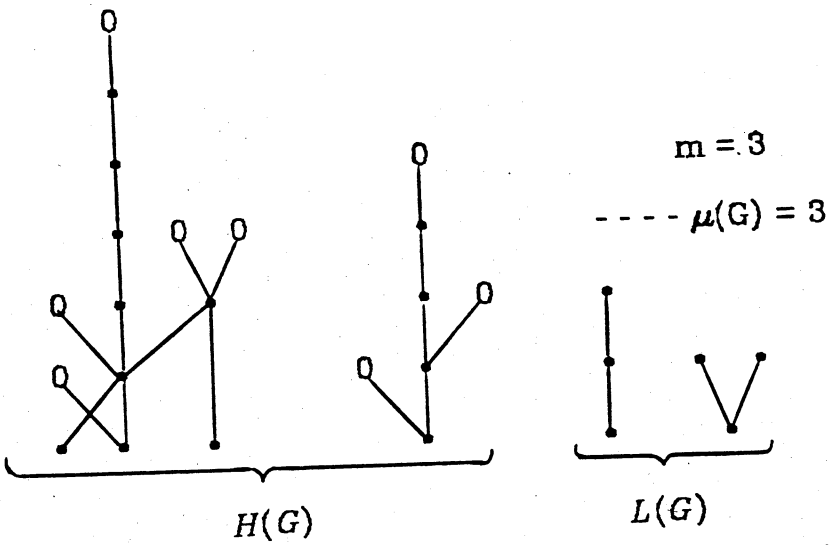


FIG. 3.1. *The decomposition of a graph G into H(G) and L(G); 0 denotes tasks of E(G).*

We use the median to partition the components of G into two sets, $H(G)$ and $L(G)$ (see Fig. 3.1).

DEFINITION. The closed subgraph $H(G)$ consists of all components of height higher than the median, and $L(G)$ contains all the components that are at most as high as the median.[2] The set of all initial tasks of $H(G)$ is called the *Elite* of G, denoted by $E(G)$.

During the construction of a schedule, the median is a dynamic line. When a set of initial tasks is removed, the median might increase, because some components of the graph might split into several components. On the other hand, the median can drop at most by one. If it drops by one, then some initial tasks of $L(G)$ were removed. If only tasks of $H(G)$ are removed, then the median does not drop. This leads to the following properties which are used in the current paper and in [DW82c].

[2] All the results of this paper will still hold if we define $H(G)$ to be the closed subgraph that contains all initial tasks of height higher than the median plus all their successors, and $L(G)$ to be the remaining subgraph. See also [DW82b].

Properties of the

M1: There are

M2: If $\mu(G)>$ least $\mu(G)-1$.

M3: If G has a

M4: If G has a

M5: Let R be

M6: Let R be $R)\subseteq H(G)-R$ and

M7: Let T be a $L(G-T)\geq L(G)-$

M8: Let R be a

$H(G-($

The proofs of the median. The proof remaining properti

CLAIM 3.1. *Le*

Proof. The cla set R contains only one shorter than th $h(I-R)$, which co

Proof of M5.

M5. Let R be

M5 is clearly only have to show $\mu(G)-2$. To do th components of hei component I of C $G-R$ has at least m components of

Proof of M6.

M6. Let R be $R)\subseteq H(G)-R$ an

The second pa Readily this inequ $\mu(G-R)\geq\mu(G)$ height $\mu(G)-1$ (in $L(G)$ and there $\mu(G)$. This comp components I, sa sponding subgrap $\mu(G)-1$.

Proof of M7

M7: Let T b $L(G-T)\geq L(G$

Assume that by M5, $\mu(G-R$