PREEMPTIVE SCHEDULING OF PRECEDENCE-CONSTRAINED JOBS ON PARALLEL MACHINES

E.L. Lawler
University of California, Berkeley

ABSTRACT

Polynomial time-bounded algorithms are presented for solving three problems involving the preemptive scheduling of precedence-constrained jobs on parallel machines: the "intree problem", the "two-machine problem with equal release dates", and the "general two-machine problem". These problems are preemptive counterparts of problems involving the nonpreemptive scheduling of unit-time jobs previously solved by Brucker, Garey and Johnson and by Garey and Johnson. The algorithms and proofs (and the running times of the algorithms) closely parallel those presented in their papers. These results improve on previous results in preemptive scheduling and also suggest a close relationship between preemptive scheduling problems and problems in nonpreemptive scheduling of unit-time jobs.

1. INTRODUCTION

In this paper we present polynomial time-bounded algorithms for solving three problems involving the preemptive scheduling of precedence-constrained jobs on parallel machines. These three problems, which we call the "intree problem", the "two-machine problem with equal release dates", and the "general two-machine problem", are preemptive counterparts of problems involving the nonpreemptive scheduling of unit-time jobs previously solved by Brucker, Garey and Johnson [1] and by Garey and Johnson [2,3]. The algorithms and proofs we present closely parallel those given in their papers, and the running times of the algorithms are nearly comparable. Problems involving nonpreemptive scheduling of unit-time jobs are sometimes formulated as approximations of preemptive scheduling.
problems. The results presented in this paper suggest that there is an even closer algorithmic relation between preemptive scheduling and nonpreemptive scheduling of unit-time jobs than had previously been appreciated.

2. PROBLEM DEFINITION

We shall define a general scheduling problem and then indicate how each of the three problems dealt with in this paper is a specialization of this problem.

There are \( n \) jobs to be scheduled for processing. For each job \( j \), \( j = 1, 2, \ldots, n \), there are specified a processing requirement \( p_j > 0 \), a release date \( r_j \geq 0 \), prior to which the job is unavailable for processing, and a due date \( d_j \geq 0 \).

The jobs are to be scheduled subject to precedence constraints "\( \rightarrow \)" in the form of a partial order induced by a given acyclic digraph on nodes \( j = 1, 2, \ldots, n \). If \( i \rightarrow j \) then job \( i \) must be completed before the processing of job \( j \) is begun.

The jobs are to be scheduled on \( m \) parallel machines. At least \( m-1 \) of the machines are identical. One machine is permitted to have the same speed or a strictly slower speed than the others. More specifically let \( s_i \) denote the speed of machine \( i \), \( i = 1, 2, \ldots, m \), and assume that \( s_1 = s_2 = \ldots = s_{m-1} = 1 \), \( s_m = s \leq 1 \). The processing capacity of machine \( i \) in a time interval \( [t, t'] \) is equal to \( s_i(t'-t) \). In order for a job to be completed, the job must be allocated sufficient processing capacity to satisfy its processing requirement.

We make the usual assumptions which apply to the scheduling of parallel machines. A machine can process at most one job at a time and a job can be processed by at most one machine at a time. The schedules we consider are preemptive, in that processing of a job can be interrupted at any time and processing resumed at the same time on another machine or at a later time on any machine. There is no penalty for such an interruption or "preemption".

A schedule is feasible if no job is processed prior to its release date, if all jobs are completed, and if precedence constraints are observed. A feasible schedule meets all due dates if each job is completed no later than its due date.

If in a given feasible schedule the completion time of a job \( j \) is \( C_j \), then its lateness with respect to its specified due date \( d_j \) is

\[
L_j = C_j - d_j.
\]

Our objective is to find a schedule which minimizes

\[
L_{\text{max}} = \max_j \{L_j\}.
\]

(Note that there exists a schedule which meets all due dates if