Scheduling Jobs That Arrive Over Time
(Extended Abstract)

Cynthia Phillips*1, Clifford Stein**2 and Joel Wein ***3

1 Sandia National Labs, Albuquerque, NM, USA
2 Department of Computer Science, Dartmouth College, Hanover, NH, USA
3 Department of Computer Science, Polytechnic University, Brooklyn, NY, USA

Abstract. A natural problem in scheduling theory is to provide good average quality of service to a stream of jobs that arrive over time. In this paper we consider the problem of scheduling n jobs that are released over time in order to minimize the average completion time of the set of jobs. In contrast to the problem of minimizing average completion time when all jobs are available at time 0, all the problems that we consider are \textit{NP}-hard, and essentially nothing was known about constructing good approximations in polynomial time.

We give the first constant-factor approximation algorithms for several variants of the single and parallel machine model. Many of the algorithms are based on interesting algorithmic and structural relationships between preemptive and nonpreemptive schedules and linear programming relaxations of both. Many of the algorithms generalize to the minimization of average \textit{weighted} completion time as well.

1 Introduction

Two characteristics of many real-world scheduling problems are (1) tasks arrive over time and (2) the goal is to optimize some function of average performance or satisfaction. In this paper we study several scheduling models that include both of these characteristics; in particular we study minimizing the average (weighted) completion time of a set of jobs with release dates. In most of these models polynomial-time algorithms were known to minimize average completion time when all jobs are available at time 0; the introduction of release dates makes these problems \textit{NP}-hard and little was known about approximation algorithms.

We give the first constant-factor approximation algorithms for the minimization of average completion time in these models. Our performance bounds come

* caphill@cs.sandia.gov. This work was performed under U.S. Department of Energy contract number DE-AC04-76AL85000.
** cliff@cs.dartmouth.edu. Research partly supported by NSF Award CCR-9308701, a Walter Burke Research Initiation Award and a Dartmouth College Research Initiation Award.
*** wein@mem.poly.edu. Research partially supported by NSF Research Initiation Award CCR-9211494 and a grant from the New York State Science and Technology Foundation, through its Center for Advanced Technology in Telecommunications.
from the combination of two different types of results. First, we prove structural theorems about the relative quality of preemptive versus nonpreemptive schedules, and give algorithms to convert from the former to the latter with only a small degradation in schedule quality. Second, we give the first constant-approximation algorithms for preemptively scheduling parallel machines with release dates. When the machines are identical, we give a combinatorial algorithm which yields a two approximation. For unrelated machines, we give a new integer programming formulation. We then solve the linear programming relaxation and give methods to round the fractional solution to valid preemptive schedules. It is possible that these methods will be of use in other settings.

Models: We are given $n$ jobs $J_1, \ldots, J_n$ where job $J_j$ has processing time $p_j$, and release date $r_j$, before which it cannot be processed on any machine. We are also given $m$ machines $M_1, \ldots, M_m$. We focus both on the one machine ($m = 1$) environment and two fundamental variants of parallel machine scheduling. In the identical parallel machine environment, job $J_j$ runs in time $p_j$ on every machine [8]. In the unrelated parallel machine scheduling environment, we are given speeds $s_{ij}$ which characterize how fast job $J_j$ runs on machine $M_i$, and $p_{ij}$, the processing time of job $J_j$ on machine $M_i$, is defined to be $p_{ij} = p_j / s_{ij}$ and thus depends on both the machine and the job [8]. Throughout this paper, unless specified otherwise, all jobs have release dates.

We give algorithms for both preemptive and nonpreemptive scheduling. In nonpreemptive scheduling, once a job begins running on a machine, it must run uninterruptedly to completion, while in preemptive scheduling, a job that is running can be preempted and continued later on any machine. At any time a job may run on at most one machine.

Let $C_j^S$ denote the completion time of job $J_j$ in schedule $S$. We will often drop the superscript when it is clear to which schedule we refer. Our basic optimization criterion is the average completion time of a set of jobs, $\frac{1}{n} \sum_{j} C_j$. At times we will associate with $J_j$ a weight $w_j$ and seek to minimize the average weighted completion time, $\frac{1}{n} \sum_{j} w_j C_j$. These optimality criteria are fundamental in scheduling theory and accordingly have received much attention, e.g. [2, 3, 5, 6, 7, 9, 10, 11, 13].

We distinguish between off-line and on-line algorithms. In an off-line algorithm, all the input data associated with jobs ($r_j$ and $p_j$) is known in advance. In an on-line algorithm, nothing is known about a job $J_j$ until time $r_j$, (at which point $p_j$ is specified as well). The scheduling algorithm must decide what job (if any) to be scheduling at time $t$ with no knowledge of what jobs will arrive in the future. In this paper, unless explicitly stated otherwise, all algorithms are off-line.

A $\rho$-approximation algorithm produces, in polynomial time, a schedule whose value is guaranteed to be at most $\rho$ times the minimum possible value.

New Results: We first focus on nonpreemptively scheduling jobs on one machine to minimize average completion time. This problem is $\mathcal{NP}$-hard [9]. We give a simple 2-approximation algorithm which transforms the optimum preemptive schedule for the problem, which can be found in polynomial time [1].