Discrete Optimization 17 (2015) 14–24

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Scheduling on uniform processors with at most one downtime on each machine

Liliana Grigoriu [a,*], Donald K. Friesen [b]

[a] Department of Control and Computers, Politehnica University Bucharest, Splaiul Independentei 313, Bucharest, 060042, Romania

[b] Department of Computer Science, Texas A&M University, College Station, TX, 77843-3112, USA

ARTICLE INFO

Article history:
Received 21 February 2012
Received in revised form 18 August 2014
Accepted 30 October 2014
Available online 13 April 2015

Keywords:
Multiprocessor scheduling
Uniform processors
MULTIFIT
Fixed jobs
Worst-case bounds
Makespan

ABSTRACT

We consider the problem of scheduling a given set of tasks on uniform processors with predefined periods of unavailability, with the aim of minimizing the maximum completion time.

We give a simple polynomial MULTIFIT-based algorithm, the schedules of which finish within 1.5 times the maximum between the latest end of a downtime and the end of the optimal schedule, when there is at most one downtime on each machine. Even when all processors have the same processing speed, it is NP-hard to obtain schedules that obey better bounds for this problem.

1. Introduction

Nonpreemptive scheduling of a set of tasks on multiple resources is a widely encountered problem. The jobs are usually assumed to be given as an integer number of suitably chosen units such as the time needed by a clock cycle on the slowest processor, in order to characterize the time needed to process them.

The multiprocessor scheduling problem, which asks whether it is possible to nonpreemptively schedule a set of independent tasks on m same-speed processors to meet a given deadline (with m considered to be an input parameter) is strongly NP-hard [1]. Therefore, the study of this problem and its generalizations has mainly been concentrating on approximation algorithms: the *largest processing time first* (LPT) algorithm was first proposed in [2] and shown to have a makespan within $4/3-1/(3m)$ times the optimal makespan, and later the MULTIFIT algorithm was considered [3], and was shown to generate schedules which end within 13/11 times the optimal makespan in [4]. A review on deterministic scheduling was given by Chen, Potts, and Woeginger in [5].

Due to maintenance or failures, machines might exhibit periods of unavailability. Reviews focusing on scheduling with availability constraints were given by Lee, Lei, and Pinedo in [6] and by Sanlaville and Schmidt in [7].

We focus on the static variant of the problem, when downtimes are known in advance. A dynamic variant can also be conceived, when downtimes can occur unexpectedly.

A special case for scheduling on multiple processors in the presence of machine downtimes is the case when all downtimes are at the beginning of the schedule, that is, when the processors start processing at different times. For same-speed

* Corresponding author.
E-mail addresses: liliana.grigoriu@cs.pub.ro (L. Grigoriu), friesen@cs.tamu.edu (D.K. Friesen).

http://dx.doi.org/10.1016/j.disopt.2014.10.001
1572-5286/

processors, Lee [8] and Chang and Hwang [9] give worst-case analyses of the multiprocessor scheduling problem for scheduling on parallel machines that do not start simultaneously, when using LPT and MULTIFIT respectively.

Given that all downtimes could be infinite, the strong NP-hardness of multiprocessor scheduling results in the NP-hardness of the problem of finding an approximation algorithm that ends within a multiple of the time needed by the optimal schedule, unless assumptions about the downtimes are made.

For the case when there is at most one downtime on each machine, the authors in [10] make the assumption that no more than half the machines are unavailable at any time. They show that for this situation the LPT algorithm ends within twice the time needed by the optimal schedule when all processors have the same speed. In [11], the result is generalized to the case when an arbitrary number of machines, $\lambda \in \{1, .., m-1\}$, may be unavailable at the same time. In that case the makespan generated by the LPT schedule is not worse than the tight worst-case bound of $1 + \frac{1}{2}\lceil m/(m-\lambda)\rceil$ times the optimal makespan.

In [12] Scharbrodt et al. consider the problem of scheduling on multiple same-speed processors with "fixed" jobs, which are jobs that have to be executed at certain predefined times, in order to minimize the makespan of the schedule of all jobs. In their setting, the number of processors is not considered a part of the input, and there can be more than one fixed job on one machine. They give a polynomial-time approximation scheme for this problem, and then a generalization thereof for the uniform processor case. They also show that no polynomial-time approximation scheme exists for the case where the number of machines is not constant, and that no fully polynomial time approximation scheme exists for the case where the number of machines is constant.

Scheduling on same-speed processors with at most one downtime on each machine was considered in [13], where an LPT-based algorithm which finishes within 3/2 times the optimal (smallest possible) maximum completion time (of a schedule) or 3/2 times the latest end of a downtime was given. There it is also shown that it is NP-hard to obtain a better bound, which follows from a proof presented in [12], which was made for the case when there may be multiple fixed jobs on one machine.

In this paper we consider scheduling on uniform processors with at most one downtime on each machine. Scheduling on uniform processors has also been studied in the past. In [14], the authors show that for nonpreemptive scheduling a variant of MULTIFIT finishes within 1.4 times the optimal maximum completion time. This bound was improved to 1.382 in [15]. In [16], a polynomial-time approximation scheme for scheduling on uniform processors is given. For two uniform processors, the authors of [17] derive a tight worst-case bound of $\sqrt{6}/2 + (1/2)^k$ for scheduling using MULTIFIT with k calls of the *first fit decreasing* (FFD) algorithm within MULTIFIT. When MULTIFIT is combined with LPT as an incumbent algorithm, they show that the worst case bound decreases to $(\sqrt{2}+1)/2 + (1/2)^k$.

In [18], the performance of LPT for scheduling on uniform processors with nonsimultaneous machine available times is studied, and it is shown that LPT finishes within 5/3 times the optimal maximum completion time, and that the bound is better when there are only 2 machines or when the speed ratio is small. The paper also presents a polynomial algorithm that finishes within 6/5 times the optimal maximum completion time if there are only 2 processors in the system.

In this work, we consider the more general case where each machine may have a downtime which does not necessarily start at the beginning of the scheduling period. Such a situation may occur when multiple uniform machines become available for processing at the same time, for example at the start of the work day, when one maintenance activity must be performed for each machine at a predefined time during the scheduling period, i.e., the day, while the purpose is to finish as soon as possible. We give a simple polynomial MULTIFIT-based algorithm which finishes within 3/2 times the optimal schedule's maximum completion time or 3/2 times the latest end of a downtime. This result does not depend on how big or small the ratios between the speeds of the different processors are or on the number of processors in the system. This implies that, when our algorithm's schedule finishes after 3/2 times the latest end of a downtime, or when the downtimes represent fixed jobs, it also finishes within 3/2 times the optimal maximum completion time. Also, if the optimal schedule ends after the latest end of downtime, for example if the total available time for processing before the latest end of downtime is not enough to process all tasks, then our algorithm's schedule finishes within 3/2 times the maximum completion time of the optimal schedule.

The classic MULTIFIT algorithm first assigns upper and lower bounds for the maximum completion time of the schedule. Then, it uses binary search while assigning schedule deadlines to find new upper and lower bounds for the maximum completion time of the schedule. Once a deadline is assigned, MULTIFIT uses the *first fit decreasing* (FFD) algorithm to assign tasks to the time slots formed between the start of the schedule and the deadline. Given an ordered list of time slots and a set of tasks, the FFD algorithm orders the tasks in nonincreasing order of their processing time on the slowest processor and then assigns each task to the first time slot encountered in which it fits. If a feasible schedule is found the deadline is decreased, and otherwise it is increased, until a desired accuracy is achieved. Within the MULTIFIT loop, our algorithm uses the following scheduling policy. It orders the time slots formed between the start of the schedule and the downtimes, and those which start at the end of a downtime and end at the MULTIFIT assigned deadline in nondecreasing order of their duration multiplied by the speed factor of the processor they are on. Then it assigns tasks to time slots using the FFD algorithm. The 3/2 bound achieved by our MULTIFIT variant is the best that can be achieved by a polynomial algorithm assuming that $P \neq NP$.

Unlike in [10], we do not have any restriction on the times when the machines shut down. The problem in [12] is similar to our problem in that the machine downtimes are equivalent to the fixed jobs. The difference is that they can have more than one fixed job on a machine, and that the maximum completion time of the optimal schedule with fixed jobs cannot be less than the maximum completion time of a fixed job, which corresponds to the latest end of a downtime in our setting.