Discrete Optimization 20 (2016) 11–22

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Approximation algorithms for scheduling on multi-core processor with shared speedup resources

Xufeng Chen[a], Deshi Ye[b,*]

[a] Department of Mathematics, School of Science, Hangzhou Dianzi University, Hangzhou 310018, China
[b] College of Computer Science, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Article history:
Received 9 September 2013
Received in revised form 5 February 2016
Accepted 9 February 2016

Keywords:
Multi-core scheduling
Approximation algorithm
Resource allocation

ABSTRACT

We consider a joint resource partition and scheduling problem. We are given m identical cores and discrete resources of total size k. We need to partition the resources among these cores. A set of jobs must be processed non-preemptively on these cores after the resource partition. The processing time of a job on a core depends on the size of resources allocated to that corresponding core. The resource allocation scheme is static, i.e., we cannot change the amount of resources that was allocated to a core during the whole scheduling. Hassidim et al. (2013) investigated this problem with a general processing time function, i.e., the processing time of a job is an arbitrary function of the level of resources allocated to that core. They provided an algorithm with approximation ratio of 36. In this paper, we improve the approximation ratio to 8 by presenting a new resource partition scheme. Next, we consider a special model where the core's speed is proportional to its allocated resource, then we present two algorithms with improved approximation ratios.

1. Introduction

In this paper, we consider the problem of scheduling n jobs $J = \{t_1, \ldots, t_n\}$ to m identical cores on a multi-core processor with shared resources of size k. The shared resources will be partitioned among cores. Each job t_j is associated with a non-increasing processing time function $T_j(x)$ on a core of resource size x $(0 \leq x \leq k)$. For each core i, we need to assign resource of size $p(i) \geq 0$, such that the total size of resources used by all cores is bounded by k, i.e., $\sum_{i=1}^{m} p(i) \leq k$. Moreover, the partition of resources among the cores is fixed for all times. In other words, the partition is static. One solution is to define a resource partition p, assigning $p(i)$ resources to core i, and a job assignment S, mapping each job j to a core $S(j)$.

* Corresponding author.
E-mail addresses: chenxf@hdu.edu.cn (X. Chen), yedeshi@zju.edu.cn (D. Ye).

http://dx.doi.org/10.1016/j.disopt.2016.02.002
1572-5286/

The goal is to find a resource partition and a job assignment such that the longest completion time of a job is minimized, i.e., the *makespan* is minimized.

This problem generalizes many machine scheduling problems. Given a resource partition, our problem becomes the related machine scheduling [1] if $T_j(x) = w_j/x$, where x can be regarded as the speed of a machine, and w_j is the processing time of job t_j. Even for a very special case that $T_j(x) = w_j$, the processing time of a job t_j is independent of the size of allocated resources, our problem becomes the parallel identical machine scheduling, which is NP-hard [2]. Given a partition, our problem will be a special case of the unrelated machine scheduling problem. A well known 2-approximation algorithm for the unrelated machine scheduling was given by Lenstra et al. [3]. Later, Shchepin and Vakhania [4] improved the approximation ratio to $2 - 1/m$, where m is the number of machines. However, our problem is difficult than a scheduling problem, because different partitions of the resource will affect a job's scheduling.

This problem has many applications, and most obvious is multi-core scheduling, in which the resource is a shared cache. Besides, the shared cache of size k can be regarded as k additional discrete resources (e.g. budget). For example, one would like to rent virtual machines in a cloud like Amazon's Elastic Cloud, in which virtual machines have different computing ability due to different prices one will pay. One will rent a total m virtual machines but with budget k and the goal is to minimize the longest completion time among these machines. Another example arises in human resource management. Suppose that we have m teams to execute n projects, and have k new employees, the manager needs to assign projects to these m teams and allocate the new employees to some of the teams. It is reasonable to assume that each employee only joins one team since employee change from one group to another may bring additional costs or are not happy to experience frequent changes of their working environment.

Liu et al. [5,6] were the first ones to study this problem by providing experimental heuristics. Their results suggested that jointly solving for the cache partition among the cores and for the job assignment to cores would lead to significant improvements over combining separate algorithms for the two problems. Recently, Hassidim et al. [7] considered this problem theoretically, gave an algorithm with approximation ratio of at most 36, in which the main idea is to provide a 18-approximate algorithm that uses cache at most $(1 + \frac{5}{2}\epsilon)k$ for any $0 < \epsilon < 1/2$, and then extends this to the general case with at most k resources. However, the disadvantage is that the approximation ratio doubles.

Our contribution. In this work, we present an improved algorithm with an approximation ratio of at most 8 for the general non-increasing processing time function $T_j(x)$ for job t_j with x units of resource. The key idea is to provide a new resource partition scheme. We then consider a special case in which the processing time of a job is linear with respect to the amount of resources. The approximation ratio is further improved to $\min\{4 + \epsilon, 2 + 2\alpha\}$, where α is the maximum speed ratio (a detailed definition is given in Section 3) and ϵ is any given positive number.

Related work. Hassidim et al. [7] investigated the joint cache partition and job assignment problem that allows for dynamic cache partitions and dynamic job assignments, in which the size of the cache on a core can be changed over time and the job can be processed on different cores. This means that the scheduler is allowed to interrupt the processing of a job at any point in any time. Regarding the optimal schedule, it was shown that the dynamic variant will improve the makespan when we compare it to the static variant.

A closely related problem is called scheduling with an additional renewable speedup resource, in which the processing time of a job depends on resources required. In contrast to our problem, this problem allows a dynamic partition of resources, but keeps the job assignment static and non-preemptive. Grigoriev et al. [8] considered this problem where the processing time depends both on the size of the shared resources and also on the cores, and presented a 6.83-approximation algorithm. Later, this ratio was improved to 4 by Kumar et al. [9,10], and the current best ratio is 3.75 [11]. In [12] the approximation ratio was improved to $3 + \varepsilon$ for any small number ε for parallel machine scheduling. Kellerer [13] studied a linear model of the above