한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003년 5월 16일-17일 한동대학교(포항)

가용시간이 서로 다른 병렬기계 스케줄링 문제에 대한 PTAS

황학진
조선대학교 산업공학과
광주광역시 동구 서석동 375, 501-759

A PTAS for nonsimultaneous parallel machine scheduling

Hark-Chin Hwang
Department of Industrial Engineering, Chosun University,
Seosuk-Dong, Dong-Gu 501-709 Gwangju

Abstract

The parallel machine scheduling problem of assigning n jobs on m identical machines with the objective of minimizing makespan is considered. In this note, we apply the PTAS (Polynomial Time Approximation Scheme) of Hochbaum and Shmoys to our problem and show that it is still a PTAS for our problem.

1. Introduction

We consider the problem of parallel machine scheduling of n independent jobs on m identical machines where machines are not available at time zero. Formally, we are given a problem instance $I=(J,\pi)$, where J is a set of jobs and π is a set of machines $\{1,2,\ldots,m\}$. Each job $x \in J$ has its processing time $p(x)$ and each machine i has its starting time $s_i \geq 0$ for $i \in \pi$. An *assignment* $S_i \subseteq J$ is a set of jobs placed on the machine i so that the load on the machine i is $s_i + t(S_i)$, where $t(X) = \sum_{x \in X} p(x)$ if X is not empty, otherwise zero for any subset $X \subseteq J$. Then a schedule $S = < S_1, \cdots, S_m >$ is a partition of J into m disjoint assignments and its *makespan* is defined to be $\max_{i \in \pi}\{s_i + t(S_i)\}$. In our problem, the objective function is to find a schedule with the minimum makespan. Given an instance I, we denote $OPT(I)$ as the minimum makespan.

This problem is NP-Complete [3] and hence it is unlikely that there exists any polynomial-time algorithm generating a schedule with the minimum makespan. To attack such an NP-Complete problem, it is reasonable to consider some approximation algorithm which

produces a schedule with a near optimal makespan. Given a constant $\epsilon > 0$, an algorithm is called $(1+\epsilon)$-*approximation algorithm* if it always generates a schedule with makespan at most $(1+\epsilon)$ times optimal, i.e., $(1+\epsilon)$ $OPT(I)$, in polynomial time of the parameters n and m for every instance I. Then, the family of $(1+\epsilon)$-approximation algorithms which run in polynomial
time of the parameters n, m and $1/\epsilon$ is called *polynomial time approximation scheme (PTAS).*

For the classical parallel machine scheduling problem where all the machines are available at time zero, Graham [4] presented the algorithm LPT and showed it always generates a schedule with makespan at most 4/3-1/3m times optimal. Another algorithm using a binary search procedure, MULTIFIT, is developed by Coffman et al. [1] whose performance is proved to be 13/11 approximate [8]. Similar heuristic approaches have been taken to deal with our problem where machines are not available at the start. The two algorithms LPT and MLPT are proposed by Lee [7], which are shown to be 3/2-1/2m and 4/3 approximate, respectively. Chang and Hwang applied the algorithm MULTIFIT to our problem and they proved that it always yields a schedule with makespan no more than 9/7 times
optimal.

In order to achieve better approximation, a new approach has been developed. In [5], Hochbaum and Shmoys proposed a *dual* approximation approach which uses the primal-dual relationship in the classical parallel machine scheduling problem (primal) and the bin-packing problem (dual). For the classical parallel machine scheduling problem, they found a PTAS which always generates a schedule with makespan at most $1+\epsilon$ times optimal with the running time of $O((n/\epsilon)^{\log\frac{1/\epsilon}{\epsilon}})$ for every constant $\epsilon > 0$ [6].

In this paper, we apply the dual approximation approach and develop a PTAS for our problem. In section 2, we present the way dual approximation approach operates and then we give a PTAS in section 3.

2. Dual Approximation Approach

Given a time deadline d and a constant $\epsilon > 0$, we classify jobs in J into two classes: *big* and *small* jobs. Job x is called big job with respect to d if $p(x) > \frac{\epsilon}{1+\epsilon}d$. Otherwise it is called small. Let's first observe the fundamental property of the list scheduling algorithm, which will be a part of our algorithm. The list scheduling algorithm is a greedy algorithm which runs as follows: given a predefined schedule S and a set of unassigned jobs in J, it chooses any job x from the set of remaining jobs and assigns it to the machine with the least load, that is, the minimum value of $s_i + t(S_i) + P(x)$ for $1 \le i \le m$.

Lemma 1

Given an instance $I=(J,\pi)$ and a time deadline $d \geq (1+\epsilon)OPT(I)$, $\epsilon > 0$, let $J = J_1 \cup J_2$ where J_1 (J_2) is the set of big (small) jobs with respect to d, respectively. If a schedule S for the set J_1 has makespan no greater than d, the schedule obtained by applying the list scheduling algorithm to S and the set J_2 has also makespan no greater than d.

Proof.

Suppose the lemma does not hold. Then let x be the first small job which could not be assigned within time d and S' be the latest schedule made from S at the time right before x is tried by the list scheduling algorithm. From the nature of the list scheduling algorithm we have $s_i + t(S_i') + p(x) > d$ for all $1 \leq i \leq m$. This implies $s_i + t(S_i') > \frac{1}{1+\epsilon} d \geq OPT(I)$ since $p(x) \leq \frac{\epsilon}{1+\epsilon} d$ and thus we have $\sum_{i=1}^{m}(s_i + t(S_i')) > mOPT(I)$, which is a contradiction.

$DUAL_\epsilon(I, d)$represents any function that operates for the subset of big jobs with respect to d and always returns *true* with a schedule whose makespan is at most d for every instance I whenever$d \geq (1+\epsilon)OPT(I)$. However, if $d < (1+\epsilon)OPT(I)$ then $DUAL_\epsilon$ might return *false* in which case the function could not assign all the big jobs within time d. Given any schedule successfully obtained by $DUAL_\epsilon$, it is trivial to construct the complete schedule for all the jobs with the same makespan of the $DUAL_\epsilon$ schedule (Lemma 1) using the list scheduling algorithm.

Now we consider our $(1+\epsilon)$-approximation algorithm $MDUAL_\epsilon$ (Multi DUAL). $MDUAL_\epsilon(I,k)$ is a binary search procedure that starts with its initial lower and upper bounds $CL(I)$ and $CU(I)$ and gradually converges to a desired time deadline by iteratively calling a function $DUAL_\epsilon$ k times. The two bounds $CL(I)$ and $CU(I)$ are defined as follows:

$$CL(I) = \max\{s_{max}, p_{max}, (\sum_{i=1}^{m} s_i + t(J))/m\},$$

$$CL(I) = \max\{s_{max} + p_{max}, 2(\sum_{i=1}^{m} s_i + t(J))/m\}.$$

where s_{max} denotes the largest starting time and p_{max} the greatest size. It has been proven that $CL(I) \leq OPT(I)$ and $CU(I) \leq 2\,OPT(I)$ in [2]. Now we formally describe the algorithm $MDUAL_\epsilon$ as shown in Fig. 1.

Procedure $MDUAL_\epsilon(I,k)$

begin

$lower := CL(I)$;

$upper := CU(I)$;

for $i := 1$ **to** k **do begin**

$d := (lower + upper)/2$;

if $DUAL_\epsilon(I,d) = true$

then $upper := d$;
else $lower := d$;
end
end

Fig. 1. Algorithm $MDUAL_\epsilon$

Let S be the latest schedule obtained from $DUAL_\epsilon(I, d)$ with the returned value of *true* during the procedure $MDIAL_\epsilon(I, k)$. The schedule of $MDUAL_\epsilon$ is then defined as that constructed from S after applying the list scheduling algorithm for the remaining small jobs. Similarly to the arguments in [1], we have the following result on the performance of $MDUAL_\epsilon$.

Theorem 2
The $MDIAL_\epsilon(I, k)$ always generates a schedule with makespan at most $1+\epsilon+2^{-k}$ times optimal.

Proof.
Suppose the theorem does not hold. Then, there exists a problem instance I for which

$$\frac{z_{MDUAL}}{OPT(I)} > 1+\epsilon+2^{-k} \qquad (1)$$

where z_{MDUAL} is the makespan of the final schedule of $MDIAL_\epsilon(I, k)$. From Lemma 1 and the definition of $DUAL_\epsilon$, z_{MDUAL} cannot be greater than the makespan of the schedule generated by $DUAL_\epsilon(I, d)$ for the big jobs whose sizes are greater than $\frac{\epsilon}{1+\epsilon}d$ where $d=(1+\epsilon+2^{-k})OPT(I)$. Let $upper_k$ ($lower_k$) be the last upper (lower) bound values right after the kth trial of $DUAL_\epsilon(I, d)$. Then by (1), we know the last upper bound value is greater than $(1+\epsilon+2^{-k})OPT(I)$. Thus we have

$$upper_k > (1+\epsilon+2^{-k})OPT(I). \qquad (2)$$

By the nature of a binary search, we have

$$upper_k - lower_k = 2^{-k} \cdot (CU(I) - CL(I))2^{-k} \cdot OPT(I), \qquad (3)$$

since $CU(I) \le 2\,CL(I)$ and $CL(I) \le OPT(I)$. Then by (2) and (3)

$$lwer_k > (1+\epsilon)OPT(I). \qquad (4)$$

Thus we know that $DUAL_\epsilon$ must have been tried with its time deadline $lower_k$ and it must have returned *false*, that is, $DUAL_\epsilon(I, lower_k) = false$ at some point during the binary search. However, this is impossible by the definition of $DUAL_\epsilon$ and the fact $lower_k > (1+\epsilon)OPT(I)$.

In the next section, we present a $DUAL_\epsilon$ which is a modification of the dual algorithm by Hochbaum and Shmoys (1997). Then, we have a PTAS for our problem, which the family of the

$DUAL_\epsilon$ algorithms for any $\epsilon > 0$.

3. A Polynomial Time Approximation Scheme

To simplify our arguments, we assume without loss of generality that the optimal makespan of an instance $I = (J, \pi)$ is one, i.e., $OPT(I) = 1$ so that all the job sizes and machine starting times are at most one.

We partition the interval $(\epsilon, 1]$ into subintervals $(u_1, v_1], (u_2, v_2], \ldots, (u_k, v_k]$ where $u_1 = \epsilon$, $v_k = 1$, and $u_j = v_{j-1}(1+\epsilon)$. Note that $k \le \frac{\log(1/\epsilon)}{\epsilon}$. A *configuration* of an assignment S_i can be presented by an k-tuple $(x_1, \ldots, x_k)$, where each x_j denotes the number of jobs with size in the interval $(u_j, v_j]$. Then, a configuration $(x_1, \ldots, x_k)$ is called *feasible* for machine i if $s_i + \sum_{j=1}^{k} x_j u_j \le 1$. We can show that any assignment S_i with feasible configuration $(x_1, \ldots, x_k)$ has load at most $(1+\epsilon)$.

Let $f_i(n_1, \ldots, n_k)$ be a function which returns *true* if it can find a feasible configurations for machines i to m otherwise *false*, where n_j denotes the number jobs with size in $(u_j, v_j]$.

We can find a schedule with makespan at most $1+\epsilon$, using dynamic programming which runs in time $O((n/\epsilon)^{\log\frac{1/\epsilon}{\epsilon}})$:

$f_i(n_1, \ldots, n_k) = true$ if there exists a feasible configuration $(x_1, \ldots, x_k)$ for machine i such that $f_{i+1}(n_1 - x_1, \ldots, n_k - x_k) = true$.

Note that for any time deadline $d \ge (1+\epsilon)OPT(I)$, the dynamic programming finds a schedule with makespan at most $(1+\epsilon)OPT(I)$ for big jobs. Then, from the schedule we can make the final schedule for all the jobs with makespan $(1+\epsilon)OPT(I)$ by Lemma 1.

References

[1] E.G. Coffman Jr, M.R. Garey, D.S. Johnson, An application of bin-packing to multiprocessor scheduling, SIAM J. Comput. 7 (1978) 1-17.

[2] S.Y. Chang, H.-C. Hwang, The worst-case analysis of the MULTIFIT algorithm for scheduling nonsimultaneous parallel machines, Discrete Appl. Math. 92 (1999), 135-147.

[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the theory of NP-Completeness (Freeman, San Francisco, 1979.

[4] R.L. Graham, Bounds on multiprocessor timing anomalies, SIAM J. Comput. 17 (1969) 416-429.

[5] D.S. Hochbaum and D. Shmoys, Using dual approximation algorithms for scheduling problems: Theoretical and practical results, J.

ACM 34 (1987), 144-162.

[6] D.S. Hochbaum, Approximation Algorithms for NP-Hard Problems (PWS PUBLISHING COMPANY, Boston, 1997, 370-371.

[7] C.Y. Lee, Parallel machine scheduling with nonsimultaneous machine available time, Discrete Appl. Math. 30 (1991), 53-61.

[8] M. Yue, On the exact upper bound for the MULTIFIT processor scheduling algorithm, in {\em Operations Research in China}, M. Yue (ed.), Vol. 24 of Annals of Operations Research, Baltzer, Basel, Switzerland, (1990) 233-259.