Approximation Algorithms for Scheduling Parallel Jobs with More Machines

Jae-Hoon Kim, *Member, KIMICS*

***Abstract*— In parallel job scheduling, each job can be executed simultaneously on multiple machines at a time. Thus in the input instance, a job J_i requires the number m_i of machines on which it shall be processed. The algorithm should determine not only the execution order of jobs but also the machines on which the jobs are executed.**

In this paper, when the jobs have deadlines, the problem is to maximize the total work of jobs which is completed by their deadlines. The problem is known to be strongly NP-hard [5] and we investigate the approximation algorithms for the problem.

We consider a model in which the algorithm can have more machines than the adversary. With this advantage, the problem is how good solution the algorithm can produce against the optimal algorithm.

***Index Terms*— Parallel job scheduling, approximation algorithms, deadlines, adversary.**

I. INTRODUCTION

IN the classical scheduling, a job is executed by only one machine at a time. Due to the rapid development of parallel computer systems, new approaches are required to model scheduling problems on parallel architecture. See [1], [2], [3] for an overview about scheduling parallel jobs.

We are given m identical machines and n independent parallel jobs to be scheduled on the machines. Each job J_i has a processing time p_i, a deadline d_i and a number of machines needed for execution m_i, $1 \leq m_i \leq m$. A machine can process at most one job at a time. During p_i time units from a starting time, exactly m_i machines are required to run the job J_i.

We consider non-preemptive model, where once a job is scheduled, it may not be preempted and continued at a later time. The jobs should be completed before or at their deadlines, if not, it cannot contribute to the objective function of the problem. The goal is to maximize the sum of works, *i.e.*, $p_i \times m_i$, of the jobs J_i completed before or at their deadlines d_i, that is, $\sum_i p_i \cdot m_i \cdot \overline{U}_i$, where $\overline{U}_i = 1$, if J_i is completed before or at time d_i, and $\overline{U}_i = 0$ otherwise.

This problem is denoted as $P \mid m_i \mid \sum_i p_i \cdot m_i \cdot \overline{U}_i$ in the well-known notation of the scheduling literature. It was shown in [4] that the problem is strongly NP-hard. Thus, we will investigate the approximation algorithms for the problem. An ρ-approximation algorithm for a maximization problem is a polynomial-time algorithm that constructs for any instance a solution of value at least $\frac{1}{\rho}$ times the optimal value. Designing a good approximation algorithm may be viewed as minimizing the performance guarantee ρ against the adversary trying to give the worst-case instances.

In this paper, we consider a model in which the algorithm may have more machines than the adversary, called resource augmentation. An algorithm is said to be a σ-machine ρ-approximation algorithm if using the machines σ times more than the adversary, its solution is at least $\frac{1}{\rho}$ times the optimal value for any instance. In this paper, we shall provide a 2-machine 2-approximation algorithm.

Manuscript received June 16, 2011; revised July 13, 2011; accepted July 29, 2011.

Jae-Hoon Kim is with the Division of Computer Engineering, Pusan University of Foreign Studies, Busan, 604-168, Korea (Email: jhoon@pufs.ac.kr)

II. RELATED WORK

For the parallel job scheduling, most previous research is concentrated on minimizing the makespan $C_{max} = max_i C_i$ and the sum of completion times $\sum_i C_i$.

In the makespan minimization, that is, $P \mid m_i \mid C_{max}$, the maximum completion time of jobs is minimized. It is strongly NP-hard [5] and unless P = NP, there is no approximation algorithm with a performance ratio better than 1.5 [6]. The best known algorithm for this problem is the list-based algorithm with approximation ratio 2 given in [7]. As variants of the problem, for the case that the number of available machines is a constant, that is, $Pm \mid m_i \mid C_{max}$, there is a polynomial time approximation scheme (PTAS) [8], [9].

Also this problem is closely related to the strip-packing problem, where rectangles should be packed into a strip with unit width and unbounded height so as to minimize the total height of the packing. In contrast to the model considered in this paper, machines assigned to a job need to be contiguous in a solution to the strip-packing problem. Therefore a solution to the strip-packing problem is also the solution to our problem, but the converse is not valid. For the strip-packing, the currently best known result is the 2-approximation algorithm [10], [11].

For minimizing the sum of completion times, that is, $P \mid m_i \mid \sum_i C_i$, there is an 8-approximation algorithm [12]. Also, the authors gave an 10.45-approximation algorithm for the problem $P \mid m_i \mid \sum_i w_i C_i$.

For the problem of scheduling the parallel jobs with deadlines, there is few known work. To our knowledge, [13] is only the known result. In [13], the author provided a $(5+\varepsilon)$-approximation algorithm. In this paper, we will develop this work applying the resource augmentation analysis.

III. ALGORITHMS

There are M machines and a set J of jobs J_i with a processing time p_i, a number of required machines m_i, and a deadline d_i. We will denote each of the machines as M_i, $i = 1, \dots, M$.

A schedule S for J is a subset of J, where each job $J_i \in S$ is associated with a starting time s_i, $s_i + p_i \le d_i$, and a subset σ_i of machines with $|\sigma_i| = m_i$. It is said that a job $J_i \in S$ is accepted in S, while a job $J_i \notin S$ is rejected in S. We will denote as $\|S\|$, the sum of the works of jobs that are accepted in S, that is, $\|S\| = \sum_{J_i \in S} p_i \cdot m_i$.

We will first introduce the heuristic algorithm $EDF(Earliest\ Deadline\ First)$. To describe EDF algorithm, we define idle and free machines. For any schedule S, a machine m is said to be idle at time t if m is executing no job at time t in S. Also, a machine m is said to be free at time t if for any time $t' \ge t$, m is idle at time t' in S.

In EDF, the jobs are scheduled in non-decreasing order of deadlines. When a job J_i is considered, EDF finds the earliest time t, $t \le d_i - p_i$, when m_i or more machines are free. If it exists, J_i is scheduled at the time t on those machines. Otherwise, it is rejected. In this paper, EDF is used in the next approximation algorithm as a sub-procedure.

Now, we will describe an approximation algorithm $\mathcal{A}$ with more machines than the optimal algorithm, say, M additional machines. In $\mathcal{A}$, the jobs $J_i \in J$ are classified into two groups B and T. Each job J_i in B is called a big job and satisfies $p_i > \frac{1}{2}d_i$, while J_i in T is called a tiny job and satisfies $p_i \le \frac{1}{2}d_i$.

In $\mathcal{A}$, the jobs are scheduled separately from the groups B and T. For the jobs in B, we consider them in the non-increasing order of their deadlines. Specifically, we assume that $d_1 \ge d_2 \ge \dots \ge d_{|B|}$, for the jobs $J_i \in B$ with their deadlines d_i, $i = 1, \dots, |B|$. The job J_1 is scheduled on the first m_1 machines, say, $M_1, \dots, M_{m_1}$. The job J_2 is scheduled on the next m_2 machines. Continuously, there is a job J_K such that $\sum_{1 \le i \le K-1} m_i \le 2M$ and $\sum_{1 \le i \le K} m_i > 2M$, if it exists. Otherwise, the job $J_{|B|}$ satisfies $\sum_{1 \le i \le |B|} m_i \le 2M$. For the former case, the jobs J_i, $i = 1, \dots, K-1$, are scheduled at time 0 and for the latter case, J_i, $i = 1, \dots, |B|$, are scheduled at time 0.

And then the jobs in T are scheduled in EDF over the jobs already scheduled from B. Note that EDF is well-defined under the situation where the jobs in B are already scheduled on the machines.

This algorithm was provided in [13] and the authors proved that it is $(5+\varepsilon)$-approximation algorithm. In this paper, we will analyze the algorithm with additional machines.

IV. ANALYSIS

In this section, we shall show that the algorithm $\mathcal{A}$ is a 2-machine 2-approximation algorithm.

First, we consider the case that $\mathcal{A}$ accepts all the jobs in T. Then it is sufficient to consider only the jobs of B. If $\mathcal{A}$ accepts all the jobs of B, then we get 1-approximation algorithm. Otherwise, there is a job of B rejected by $\mathcal{A}$. Then from the definition of $\mathcal{A}$, there is a rejected job J_K such that $\sum_{1 \le i \le K-1} m_i \le 2M$ and $\sum_{1 \le i \le K} m_i > 2M$. Since $m_K \le M$, the machines more than or equal to M should execute the scheduled jobs $J_1, \dots, J_{K-1}$. Also, since the jobs J_i, $i = 1, \dots, K-1$, are big jobs, that is, $p_i > \frac{1}{2}d_i$, the machines execute the jobs during at least half of the intervals $[0, d_i]$, $i = 1, \dots, K-1$. But in the optimal algorithm, the M machines may execute the big jobs during at most the intervals $[0, d_i]$, because in $\mathcal{A}$, the big jobs are sorted in non-increasing order. Therefore $\mathcal{A}$ is a 2-approximation algorithm.

From now on, we assume that there is a job of T rejected by $\mathcal{A}$. Let ℓ be the deadline of the job of T which is rejected by $\mathcal{A}$ in the last. Then ℓ is well-defined. Note that the jobs of T is scheduled by EDF in $\mathcal{A}$. Then, the jobs of T with deadlines larger than ℓ is called the last jobs. Denote the set of the last jobs by $\mathcal{L}$.

For $i = 1, \dots, 2M$, we define a positive integer k_i to be the deadline of the big job which is scheduled on machine M_i in $\mathcal{A}$, if it exits. Otherwise, $k_i = 0$. Thus, we define a height h_i of machine M_i to be the maximum of ℓ and k_i for each i. Figure 1 represents the heights h_i of machines. Let $\mathcal{R}$ be the region bounded by the heights. Note that the area of $\mathcal{R}$ is equal to $\sum_{i=1}^{2M} h_i$. Then we shall obtain an upper bound of the optimal algorithm in the following lemma.

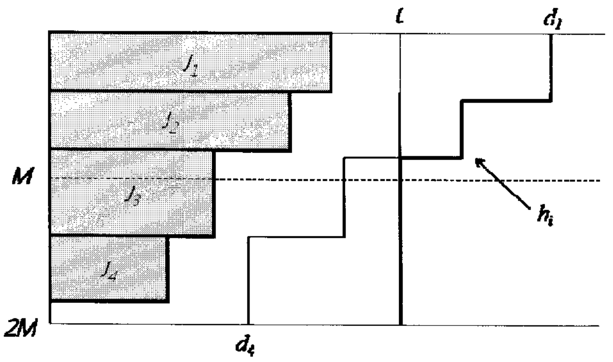


Fig. 1. The heights h_i of machines.

Lemma 1 The optimal algorithm OPT satisfies the following inequality:

$$\|OPT\| \leq \sum_{J_i \in \mathcal{L}} p_i \cdot m_i + \sum_{i=1}^{M} h_i$$

Proof: First, we consider the big jobs scheduled by OPT. Let g_i to be the time when the last big job scheduled on machine M_i in OPT is completed, if it exits. Otherwise, $g_i = 0$. Since all the machines are identical, we may assume, without loss of generality, that $g_1 \geq \cdots \geq g_M$. Then from the definition of the algorithm $\mathcal{A}$, we can see that $g_i \leq k_i \leq h_i$, for $i = 1, \ldots, M$. Thus all the big jobs scheduled by OPT should be completed within the region $\mathcal{R}$.

We consider the tiny jobs which do not belong to $\mathcal{L}$. They have the deadline smaller than or equal to ℓ. Thus they also should be completed within the region $\mathcal{R}$ in OPT. This completes the proof.

Next, we shall obtain an upper bound of $\mathcal{A}$. It shows that $\mathcal{A}$ schedules all the last jobs and the total work of the other jobs scheduled by $\mathcal{A}$ is at least half of the region $\mathcal{R}$.

Lemma 2 The algorithm $\mathcal{A}$ satisfies the following inequality:

$$\|\mathcal{A}\| \geq \sum_{J_i \in \mathcal{L}} p_i \cdot m_i + \frac{1}{2}\sum_{i=1}^{M} h_i$$

Proof: First, from the definition of the time ℓ, we can see that all the last jobs belonging to $\mathcal{L}$ are scheduled by $\mathcal{A}$.

We consider only the big jobs and the tiny jobs with the deadline smaller than or equal to ℓ.

For the machines M_i satisfying that $k_i > \ell$, they are busy during the interval $[0, \frac{1}{2}k_i]$, because of the definition of the big job, that is, $p_i > \frac{1}{2}k_i$.

If the number of the machines satisfying that $k_i > \ell$ is more than or equal to M, then it completes the proof.

Now, we assume that the number of the machines satisfying that $k_i > \ell$ is less than M. In other words, for some $H < M$, $k_i > \ell$, $i = 1, \ldots, H$. Note that this is the case that all the big jobs are scheduled by $\mathcal{A}$. Then for the machines M_i, $i = H+1, \ldots, I$, satisfying that $k_i \leq \ell$, we shall show that the machines are busy during the interval $[0, \frac{1}{2}\ell]$ such that $I > M$.

We consider the time t when a tiny job J_k with deadline ℓ is rejected by $\mathcal{A}$ in the last. At time t, the machines more than M are busy, because J_k requires at most M machines. Also since J_k is a tiny job, that is, $p_i \leq \frac{1}{2}\ell$, the machines should be busy at least until $\frac{1}{2}\ell$, because if not, the job J_k can be scheduled by ℓ. Therefore $I > M$ and the machines M_i, $i = H+1, \ldots, I$, satisfying that $k_i \leq \ell$, are busy during the interval $[0, \frac{1}{2}\ell]$. Figure 2 represents the schedule of $\mathcal{A}$ for the tiny jobs. This completes the proof.

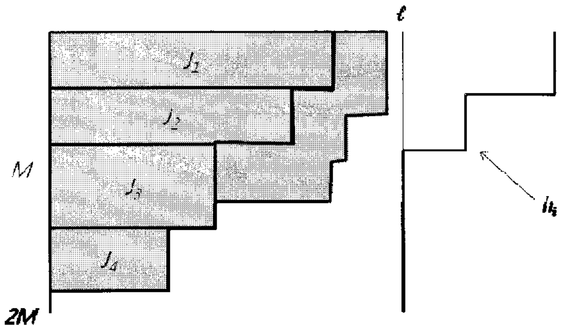


Fig. 2. The schedule of tiny jobs in $\mathcal{A}$.

Having the above lemmas, we can obtain the approximation ratio of $\mathcal{A}$.

Theorem 1 The algorithm $\mathcal{A}$ is a 2-machine 2-approximation algorithm.

Proof: From Lemma 1 and Lemma 2, we obtain the following inequalities:

$$\begin{aligned}\|\mathcal{A}\| &\geq \sum_{J_i \in \mathcal{L}} p_i \cdot m_i + \frac{1}{2}\sum_{i=1}^{M} h_i \\ &\geq \frac{1}{2}\|OPT\|.\end{aligned}$$

This completes the proof.

V. CONCLUSIONS

In this paper, we deal with the problem to schedule the parallel jobs with deadlines. In particular, the algorithms are allowed to have more machines than the adversary. We provide a 2-approximation algorithm with the machines two times more than the optimal algorithm.

As the further work, we may consider the problem in which the jobs can be executed on any number of machines but their execution times depend on the number of machines allocated to it.

REFERENCES

[1] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt and J. Weglarz, *Handbook on Scheduling: From Theory to Applications*. Springer, Heidelberg, 2007.

[2] M. Drozdowski, "Scheduling multiprocessor tasks – an overview", *European Journal of Operational Research*, vol. 94(2), pp. 215-230, 1996.

[3] J. Y. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004..

[4] A. V. Fishkin and G. Zhang, "On maximizing the throughput of multi-processor tasks", *Theoretical Computer Science*, vol. 302, pp. 319-335, 2003.

[5] J. Du and J. Y. Leung, "Complexity of scheduling parallel task systems", *SIAM J. Disc. Math.*, vol. 2(4), pp. 473-487, 1989.

[6] B. Johannes, "Scheduling parallel jobs to minimize the makespan", *Journal of Scheduling*, vol. 9(5), pp. 433-452, 2006.

[7] M. R. Garey and R. L. Graham, "Complexity results for multiprocessor scheduling under resource constraints", *SIAM Journal on Computing*, vol. 4(4), pp. 397-411, 1975.

[8] A. K. Amoura, E. Bampis, C. Kenyon, and Y. Manoussakis, "Scheduling independent multiprocessor tasks", *Algorithmica*, vol. 32(2), pp. 247-261, 2007.

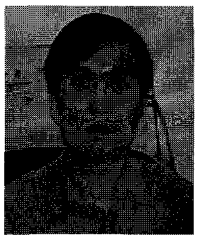
[9] K. Jansen and L. Porkolab, "Linear-time approximation schemes for scheduling malleable parallel tasks", *Algorithmica*, vol. 32(3), pp. 507-520, 2002.

[10] I. Schiermeyer, "Reverse-fit: A 2-optimal algorithm for rectangle packing", *Proc. 2th Annual European Symposium on Algorithms*, pp. 290-299, 1994.

[11] A. Steinberg, "A strip-packing algorithm with absolute performance bound 2", *SIAM Journal on Computing*, vol. 26(2), pp. 401-409, 1997.

[12] U. Schwiegelshohn, W. Ludwig, J. L. Wolf, J. J. Turek, and P. S. Yu, "Smart SMART bounds for weighted reponse time scheduling", *SIAM Journal on Computing*, vol. 28(1), pp. 237-253, 1998.

[13] Oh-Heum Kwon and Kyung-Yong Chwa, "Scheduling parallel tasks with individual deadlines", Theoretical Computer Science, vol. 215(1-2), pp. 209-223, 1999.



Jae-Hoon Kim
Received his B.S. degree in Mathematics from Sogang University in 1994, his M.S. degree in Mathematics from KAIST in 1996, and his Ph.D. degree in Computer Science from KAIST in 2003. He is currently an associate professor at Division of Computer Engineering, Pusan University of Foreign Studies. His research interests include on-line algorithms, scheduling, and computational geometry