가단성 태스크들의 마감시간 스케줄링의 자원추가 분석

김재훈*

Resource Augmentation Analysis on Deadline Scheduling with Malleable Tasks

Jae-Hoon Kim*

요 약

본 논문은 마감시간을 가지는 병렬 태스크들을 스케줄하는 문제를 다룬다. 특히, 가단성 태스크, 다시 말해서, 수행시간이 수행 머신들의 개수의 함수로 주어지는 태스크를 다룬다. 스케줄링 알고리즘의 목표는 마감시간 안에 수행을 끝마친 태스크들의 작업량의 합을 최대화하는 것이다. 이 문제는 NP-hard 문제로 알려져 있다. 따라서, 근사 알고리즘을 찾으려하고, 알고리즘의 성능은 최적 알고리즘 성능과의 비, 다시 말해서, 근사비를 구해서 분석한다. 특히, 본 논문에서는 알고리즘이 최적 알고리즘보다 많은 자원, 즉, 보다 많은 머신들을 가지는 경우에 근사비를 구할 것이다. 이것은 자원추가분석이라고 불린다. 본 논문은 최적 알고리즘보다 1.5배의 머신들을 사용해서 3.67의 근사비를 보장하는 스케줄링 알고리즘을 제안한다.

ABSTRACT

In this paper, we deal with the problem of scheduling parallel tasks with deadlines. Parallel tasks can be simultaneously executed on various machines and specially, we consider the malleable tasks, that is, the tasks whose execution time is given by a function of the number of machines on which they are executed. The goal of the problem is to maximize the throughput of tasks completed within their deadlines. This problem is well-known as NP-hard problem. Thus we will find an approximation algorithm, and its performance is compared with that of the optimal algorithm and analyzed by finding the approximation ratio. In particular, the algorithm has more resources, that is, more machines, than the optimal algorithm. This is called the resource augmentation analysis. We propose an algorithm to guarantee the approximation ratio of 3.67 using 1.5 times machines.

키워드

병렬 태스크, 마감시간, 근사 알고리즘, 근사비, 자원추가 분석

Key word

Parallel task, Deadline, Approximaiton algorithm, Approximation ratio, Resource augmentation analysis

* 정회원 : 부산외국어대학교 컴퓨터공학과(교신저자, jhoon@bufs.ac.kr)

접수일자 : 2012. 05. 22
심사완료일자 : 2012. 06. 19

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.10.2303

Ⅰ. 서 론

본 논문에서는 태스크 T_i들이 작업량 w_i와 마감시간 d_i을 가지고 주어질 때, 작업량만큼 마감시간 안에 수행된 태스크들의 작업량 총합을 최대로 하는 스케줄을 찾는다.

태스크 T_i는 병렬 태스크로서 임의의 시간에 여러 머신들에서 동시에 수행될 수 있다. 본 논문에서는 병렬 태스크 중에 가단성 태스크만을 다룬다. 가단성 태스크란, 임의의 수의 머신들에서 수행될 수 있지만, 수행시간이 수행한 머신들 수의 함수로 주어지는 태스크를 말한다.

정확히 말해서, 태스크 T_i는 최대병렬도 p_i가 주어지고, t_i를 T_i를 p_i개의 머신들에 수행했을 때 작업량 w_i를 완료할 수 있는 시간, 다시 말해서, $w_i = p_i t_i$, 이라고 하면, x개의 머신들에 수행했을 때의 시간 $\tau_i(x)$는 다음과 같이 주어진다:

$$\tau_i(x) = \begin{cases} \dfrac{t_i p_i}{x} & \text{if } x \le p_i \\ t_i & \text{if } x > p_i. \end{cases}$$

이것에 대비되는 태스크가 비가단성태스크이다. 이것은 수행될 머신들의 개수가 고정적으로 주어지고 수행될 수행시간이 주어진다. 따라서 정확히 이 개수의 머신들에 스케줄 되어야 한다.

이 문제의 목표는 작업을 완료한 태스크들의 작업량의 합을 최대화하는 것이다. 하지만, 이 문제는 NP-hard로 알려져 있다. 따라서, 우리는 본 논문에서 근사 알고리즘을 생각할 것이다. 근사 알고리즘의 성능은 최적 알고리즘 성능과의 비, 다시 말해, 근사비로 평가된다. 형식적으로, T를 태스크들의 집합이라 하고, $A(T)$ ($OPT(T)$)를 T에 대해서 알고리즘 A(최적 알고리즘 OPT)가 마감시간 안에 수행한 태스크들의 작업량들의 합이라고 하자. 그러면,

$$\forall\ T,\ OPT(T) \le \rho A(T),$$

를 만족하는 ρ를 근사비라고 한다.

특히, 본 논문에서는 최적 알고리즘 OPT보다 알고리즘 A가 보다 많은 머신을 사용할 수 있는 경우의 근사비를 구할 것 이다. 이런 분석 방법을 자원추가 분석이라고 부른다[1]. 우리는 1.5배의 머신들을 사용해서 3.67의 근사비를 보장하는 근사 알고리즘을 제안한다. 이 알고리즘은 [2]에서 제안되었고, 본 논문에서는 새롭게 자원추가 분석을 사용해서 근사비를 구하였다.

Ⅱ. 관련 연구

스케줄링 분야에서 병렬 태스크들의 스케줄링에 대한 연구는 활발히 이루어져 왔다. 대부분의 문제들이 NP-hard로 알려져 있어서, 근사 알고리즘들을 찾고 분석하는 연구들이 이루어졌다. 특히 마감시간을 가지지 않은 태스크들에 대해서, 가장 마지막에 수행 완료된 태스크의 완료시간, 다시 말해, makespan을 최소화하는 문제 또는 모든 태스크들의 완료시간의 합을 최소화하는 문제들이 많이 연구되었다[3].

본 논문에서는 마감시간을 가진 태스크들을 스케줄하는 문제를 다룬다. 이 문제는 지금까지 많이 연구되어지지 않았다. 이 문제를 다룬 논문으로 [2]가 있는데, 이 논문에서는 비가단성과 가단성 태스크들에 대해서 각각의 근사 알고리즘을 제안하였고, 가단성 태스크의 경우에 4.5의 근사비를 증명하였다. 본 논문은 [2]에서 제안된 알고리즘에 자원추가 분석을 적용해서 근사비를 구할 것이다. 자원추가 분석이란, 최적 알고리즘보다 많은 머신을 사용하거나 더 빠른 머신을 사용해서 성능을 분석하는 방법이다[1].

Ⅲ. 알고리즘 설명

이 장에서는 논문에서 제안하는 알고리즘 A에 대해서 설명할 것이다. 알고리즘 A는 크게 두 부분으로 구성된다. 알고리즘 A가 사용하는 머신들을 $M_i, i = 1, ..., \frac{3}{2}m$, 으로 나타내면, 최적알고리즘 OPT는 $M_i, i = 1, ..., m$, 을 사용한다. 알고리즘 A는 최적알고리즘보다 $\frac{1}{2}m$개의 머신들을 더 사용할 수

있다.

입력 (가단성) 태스크들을 $T_i, i=1,...,n$, 라 하면, 각 태스크 T_i는 작업량 w_i와 마감시간 d_i를 가진다. 또한 그것은 최대 병렬도 p_i와 대응되는 수행시간 t_i, 다시 말해서, 태스크를 p_i개의 머신들에 수행할 때의 작업량 w_i을 수행하는데 수행시간 t_i이 걸린다. 따라서 태스크 T_i는 대응되는 수행시간이 마감시간 d_i를 넘지 않으면, 작업량을 보전하면서 p_i개 보다 적은 머신들에서 수행될 수 있다.

우리는 태크스 T_i들을 두 그룹으로 나눌 것이다. 먼저, $t_i \le \frac{d_i}{2}$이고 $t_i p_i \le \frac{3md_i}{8}$ 이면, T_i를 *작은(small) 태스크*라고 부르고 T_i^s라고 나타낸다. 작은 태스크 외의 나머지 태스크들을 *큰(large) 태스크*라고 하고 T_i^l라고 나타낸다. 그러면, 큰 태스크 T_i^l, $i=1,...,n_l$, 의 최대 병렬도, 수행시간, 마감시간을 각각 p_i, t_i, d_i로 나타내고, 작은 태스크 T_j^s, $j=1,...,n_s$, 의 경우에는 $\bar{p}_j, \bar{t}_j, \bar{d}_j$로 나타낸다.

우리는 큰 태스크들을 마감시간의 내림차순으로 정렬한다. 다시 말해, $d_1 \ge d_2 \ge \cdots \ge d_{n_l}$. 또한 작은 태스크들은 마감시간의 오름차순으로 정렬한다. 다시 말해, $\bar{d}_1 \le \bar{d}_2 \le \cdots \le \bar{d}_{n_s}$.

이제 본 논문에서 분석할 알고리즘 A를 설명할 것이다. 알고리즘 A는 크게 두 부분으로 나뉜다. 우선 큰 태스크 T_i^l들을 정렬된 순서(마감시간의 내림차순)로 시간 0에 스케줄하는 것이다. 큰 태스크들의 스케줄이 완료되면 작은 태스크 T_i^s들을 이른마감시간(Earliest Deadline First(EDF)) 스케줄링 알고리즘으로 스케줄한다. 이른마감시간 스케줄링은 태스크들을 마감시간의 오름차순으로 스케줄하고 태스크 T_i^s의 차례에 스케줄할 수 있는 쉬는(idle) 머신들이 존재하면 스케줄하는 욕심쟁이(greedy) 알고리즘이다. 특히, 작은 태스크 T_i^s에 대해서, 병렬도 $\bar{p}_i \le \frac{3}{4}m$와 대응되는 수행시간 $\bar{t}_i \le \frac{d_i}{2}$가 만족하도록 스케줄 할 수 있다.

이상 설명한 것이 알고리즘의 A의 큰 골격이고 다음 장에서는 몇 가지 경우들을 나누어서 분석하고, 각 경우에 A의 동작에 약간의 차이가 있고 자세한 사항은 다음 장에서 설명할 것이다.

Ⅳ. 성능 분석

알고리즘 A에서 큰 태스크 T_i^l들을 시간 0에 p_i개의 머신들에 스케줄하면, 어떤 양의 정수 K가 존재해서, $\sum_{i=1}^{K-1} p_i \le \frac{3}{2}m$이고 $\sum_{i=1}^{K} p_i > \frac{3}{2}m$를 만족한다. 그러면, 큰 태스크 T_i^l, $i=1,...,K-1$, 들은 알고리즘 A에 의해서 스케줄 되어 질 수 있다.

알고리즘 A에서 큰 태스크 T_i^l, $i=1,...,K-1$, 들을 스케줄 한 후, 작은 태스크 T_i^s들을 스케줄 할 때, 알고리즘 A에 의해 마지막으로 거절된 태스크를 생각한다. 그리고 이 태스크의 마감시간을 ℓ이라고 하자. 그러면 ℓ보다 큰 마감시간을 가지는 작은 태스크들의 집합 Λ를 고려한다. 그러면 Λ에 속하는 모든 작은 태스크들은 알고리즘 A에 의해 스케줄 됨을 알 수 있다.

각 머신 $M_i, i=1,...,\frac{3}{2}m$, 에 대해서, 함수 ℓ_i의 값은 머신 M_i에 스케줄 된 큰 태스크 T_j^l의 마감시간과 ℓ 중의 최대값으로 정의한다.

다시 말해서, $\sum_{j=1}^{k-1} p_j < i \le \sum_{j=1}^{k} p_j$이면, $h_i = d_k$ 라고 하자. 그러면, $\ell_i = \max(h_i, \ell)$.

우리는 [4]에서 비가단성 태스크들에 대해서 증명했던 것과 같은 방식으로 다음 보조정리를 증명할 수 있다.

보조정리 4.1 최적 알고리즘 OPT에 대해서,

$$OPT(T) \le \sum_{i=1}^{m} \ell_i + \sum_{T_i^s \in \Lambda} \bar{p}_i \bar{t}_i.$$

알고리즘 A는 Λ에 속하는 모든 작은 태스크들을 스

케줄 할 수 있기 때문에, 앞으로 우리는 몇 가지 경우들을 나누어서 각각의 경우에 $\sum_{j=1}^{m}\ell_i$와 $A(T-\Lambda)$를 비교할 것이다.

경우 1) $p_i \geq \frac{3}{4}m$을 만족하는 어떤 큰 태스크 T_i^l, $1 \leq i \leq K-1$, 가 존재한다.

이 경우에 알고리즘 A는 큰 태스크들의 경우에 T_j^l, $1 \leq j \leq i$, 만을 시간 0에 스케줄 한다. 그러면, T_i^l는 큰 태스크이기 때문에 $t_i > \frac{d_i}{2}$ 또는 $t_i p_i > \frac{3md_i}{8}$을 만족해야 하고, 두 경우 모두 $t_i p_i > \frac{3md_i}{8}$이다. 또한 $p_i \geq \frac{3}{4}m$이고 $\sum_{j=1}^{i} p_j \leq \frac{3}{2}m$이기 때문에, $p_j \leq \frac{3}{4}m$, $1 \leq j \leq i-1$. 그러면, $t_j > \frac{d_i}{2}$, $1 \leq j \leq i-1$.

우선, $\ell \leq d_i$라고 가정하자. 그러면,

$$\sum_{j=1}^{m}\ell_i \leq \sum_{j=1}^{i-1} p_j d_j + d_i(m - \sum_{j=1}^{i-1} p_j)$$
$$\leq \sum_{j=1}^{i-1} 2p_j t_j + d_i m \leq \sum_{j=1}^{i-1} 2p_j t_j + \frac{8}{3}p_i t_i$$
$$\leq \frac{8}{3}A(T-\Lambda).$$

다음으로, $\ell > d_i$ 라고 가정하자. 그러면, $h_j > \ell (1 \leq j \leq i-1)$를 만족하는 머신 M_j에 대해서는, 어떤 큰 태스크 T_k^l가 시간 0에 스케줄되고, $t_k > \frac{1}{2}d_k = \frac{1}{2}\ell_k$. 또한 $p_i \geq \frac{3}{4}m$이라는 사실과 작은 태스크 T_k^s에 대해서, $t_k \leq \frac{1}{2}d_k$와 EDF의 욕심쟁이 성질에 의해서, 적어도 $\frac{1}{2}\ell$이상의 시간에서 $\frac{3}{4}m$개 이상의 머신들이 쉬는 시간 없이 실행되고 있음을 알 수 있다. 따라서 $\sum_{j=1}^{m}\ell_i \leq \frac{8}{3}A(T-\Lambda)$.

위의 경우에 모두 근사비 $8/3$을 얻었다. 따라서 앞으로는 큰 태스크 T_i^l, $1 \leq i \leq K-1$, 에 대해서, $p_i \leq \frac{3}{4}m$라고 가정할 수 있다. 또한 $t_i > \frac{1}{2}d_i$를 만족한다.

다음 경우를 설명하기 전에, 태스크 T_i의 최소 병렬도 $\hat{p}_i$를 정의한다. 이것은 태스크 T_i가 대응되는 수행시간 $\hat{t}_i$이 마감시간을 넘지 않으면서 최소로 할당될 수 있는 머신의 수를 말한다.

경우 2) $\sum_{i=1}^{K-1} p_i t_i < \frac{3}{8}\sum_{i=1}^{m} h_i$ 이고,
$\sum_{i=1}^{K-1} p_i + \hat{p}_K > \frac{3}{2}m$

이 경우에, 우선 $\sum_{i=1}^{K-1} p_i < \frac{3}{4}m$를 만족하고, $p_K > \frac{3}{4}m$. $\delta = (m - \sum_{i=1}^{K-1} p_i)$라고 하면, 두 번째 부등식에 의해, $\hat{p}_K \hat{t}_K > (\delta + \frac{1}{2}m)d_K \geq \frac{3}{2}\delta d_K$. 그런데, $\delta d_K = \sum_{i=1}^{m} h_i - \sum_{i=1}^{K-1} p_i d_i > \sum_{i=1}^{m} h_i - 2\sum_{i=1}^{K-1} p_i t_i$ 이고 $\delta d_K > \sum_{i=1}^{m} h_i - \frac{3}{4}\sum_{i=1}^{m} h_i = \frac{1}{4}\sum_{i=1}^{m} h_i$이 성립한다. 결과적으로, $\hat{p}_K \hat{t}_K > \frac{3}{8}\sum_{i=1}^{m} h_i$.

여기서, 큰 태스크들 $T_1^l, T_2^l, ..., T_{K-1}^l$을 두 집합으로 나눈다. 먼저 $d_i - t_i \geq d_K$을 만족하는 태스크 T_i^l 들의 집합 T'과 나머지 태스크들의 집합 T''이다. 인덱스를 새로 조정해서 $T' = \{T_1^l, ..., T_h^l\}$과 $T'' = \{T_{h+1}^l, ..., T_{K-1}^l\}$라고 하자. 여기서 각 집합 안에서 태스크들은 마감시간의 내림차순으로 정렬한다. 인덱스가 새롭게 조정됨으로서 함수 h_i와 ℓ_i도 맞춰서 조정된다. 그러나 합 $\sum_{j=1}^{m}\ell_i$은 변하지 않는다.

이 경우에 알고리즘 A는 T''의 모든 태스크들을 거절하고 T'의 태스크들과 태스크 T_K^l를 스케줄한다. 태스크 T_K^l를 시간 0에 머신 1부터 p_K까지 스케줄하고 T'의 태스크들은 시간 t_K에 태스크 T_K^l 다음에 스케줄한다. $\sum_{i=1}^{h} p_i < p_K$이고 $d_i - t_i \geq d_K \geq t_K$, $1 \leq i \leq h$이기 때문에, 이런 스케줄은 항상 가능하다.

그럼 이 스케줄을 분석해 보자. 먼저, $d_K \geq \ell$인 경우를 생각해 보자. 이 경우에는 $\sum_{i=1}^{m} \ell_i = \sum_{i=1}^{m} h_i \leq \frac{8}{3} \hat{p}_K \hat{t}_K$을 만족해서 근사비 $8/3$을 보인다.

다음으로 $d_K < \ell$을 가정하자. 먼저 새로운 함수 ℓ'_i을 정의한다. $\sum_{i=1}^{h} p_i < i \leq \sum_{i=1}^{K-1} p_i$이면, $\ell'_i = \ell$이고 나머지 i에 대해서는 $\ell'_i = \ell_i$로 정의한다. 그러면, $\ell'_i > \ell$인 머신 M_i에는 $t_j > \frac{1}{2} d_j$인 태스크 T_j^l가 스케줄된다. 또한 EDF의 성질과 작은 태스크의 정의로부터 $t \leq \frac{1}{2}\ell$인 임의의 시간 t에서 적어도 $\frac{3}{4}m$개 이상의 머신들이 실행된다. 따라서 $\sum_{j=1}^{m} \ell'_i \leq \frac{8}{3} A(T-\Lambda)$.

그리고 $\sum_{i=1}^{m} \ell_i - \sum_{i=1}^{m} \ell'_i$의 상한을 다음과 같이 구할 수 있다:

$$\sum_{i=1}^{m} \ell_i - \sum_{i=1}^{m} \ell'_i = \sum_{i=h+1}^{K-1} \ell_i - \ell'_i = \sum_{i=h+1}^{K-1} p_i (d_i - \ell)$$
$$\leq \sum_{i=h+1}^{K-1} p_i (d_i - d_K) \leq \sum_{i=h+1}^{K-1} p_i t_i$$
$$\leq \frac{3}{8} \sum_{i=1}^{m} h_i \leq \hat{p}_K \hat{t}_K \leq A(T-\Lambda).$$

따라서, 위의 두 식으로부터 다음을 얻는다.

$$\sum_{i=1}^{m} \ell_i \leq (\frac{8}{3}+1) A(T-\Lambda).$$

경우 3) $\sum_{i=1}^{K-1} p_i t_i \geq \frac{3}{8} \sum_{i=1}^{m} h_i$

이 경우에, 알고리즘 A는 큰 태스크 $T_1^l, \ldots, T_{K-1}^l$을 시간 0에 스케줄 한다. 그러면, $d_K \geq \ell$이면, $\sum_{i=1}^{m} \ell_i = \sum_{i=1}^{m} h_i \leq \frac{8}{3} \sum_{i=1}^{K-1} p_i t_i$. $d_K < \ell$이면, $t_i > \frac{1}{2} d_i$, $1 \leq i \leq K-1$, 라는 사실과 EDF의 성질과 작은 태스크의 정의로부터 $\sum_{j=1}^{m} \ell_i \leq \frac{8}{3} A(T-\Lambda)$임을 알 수 있다.

경우 4) $\sum_{i=1}^{K-1} p_i + \hat{p}_K \leq \frac{3}{2} m$

이 경우에는 큰 태스크 $T_1^l, \ldots, T_K^l$을 시간 0에 스케줄 한다. 여기서, T_K^l은 남은 $\frac{3}{2}m - \sum_{i=1}^{K-1} p_i$개의 머신에 스케줄 할 수 있다. 그러면, $t_i > \frac{1}{2} d_i$, $1 \leq i \leq K-1$와 $t_K > \frac{3}{8} d_K$ 그리고 EDF의 성질과 작은 태스크의 정의로부터, 역시 $\sum_{j=1}^{m} \ell_i \leq \frac{8}{3} A(T-\Lambda)$임을 알 수 있다.

지금까지 몇 가지 경우를 나누어서 각각의 경우에 알고리즘 A의 성능을 분석하였다. 결과적으로 우리는 다음의 정리를 얻을 수 있다.

정리 4.2 임의의 태스크 집합 T에 대해서,

$$OPT(T) \leq \frac{11}{3} A(T).$$

V. 결 론

본 논문에서는 마감시간을 가진 가단성 태스크들의 스케줄링 문제를 다루었다. 특별히, 자원 추가 분석을 통해서 최적 알고리즘보다 1.5배의 머신을 사용해서 3.67

의 근사비를 보장하는 알고리즘을 제안하였다. 향후에도 자원 추가 분석을 이용해서 다른 스케줄링 문제에도 적용해 볼 수 있을 것이다.

참고문헌

[1] C. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation", *In Proc. of the 29th Ann. ACM Symp. on Theory of Computing,* pp. 140-149, 1997.

[2] Oh-Heum Kwon and Kyung-Yong Chwa, "Scheduling parallel tasks with individual deadlines", *Theoretical Computer Science,* vol. 215, no.1-2, pp. 209-223, 1999.

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

[4] Jae-Hoon Kim, "Approximation algorithms for scheduling parallel jobs with more machines", vol. 9, no. 4, pp. 471-474, 2011.

저자소개

김재훈(Jae-Hoon Kim)

1994년 서강대학교 수학과 이학사
1996년 KAIST 수학과 이학석사
2003년 KAIST 전산과 공학박사
2003년 ~ 현재 부산외국어대학교 컴퓨터공학과 부교수
※관심분야 : 알고리즘, 최적화, 스케줄링