JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 17, No. 8 : 1934~1939, August. 2013

추가 머신들을 이용한 동일 길이 작업들의 온라인 마감시간 스케줄링

김재훈*

Online Deadline Scheduling of Equal Length Jobs with More Machines

Jae-hoon Kim*

Department of Computer Engineering, Busan University of Foreign Studies, Busan 608-738, Korea

요 약

본 논문은 마감시간을 가진 작업들의 온라인 스케줄링 문제를 다룬다. 작업들이 시간이 지남에 따라 도착하고 스케줄링 알고리즘은 앞으로 도착할 작업들의 정보를 미리 알지 못한다. 작업들은 동일한 수행시간만큼 실행되고 알고리즘의 목표는 마감 시간 안에 수행을 완료한 작업들의 수를 최대화하는 것이다. 온라인 알고리즘의 성능은 모든 작업 정보를 미리 알고 최적의 답을 줄 수 있는 최적 알고리즘의 성능과 비교하는데 두 알고리즘 성능의 비를 경쟁비라고 한다. 일반적으로 정보의 부재로 인해 온라인 알고리즘은 큰 경쟁비를 가진다. 따라서 온라인 알고리즘에 보다 많은 머신 또는 보다 빠른 머신을 제공했을 때의 경쟁비를 계산하는 자원추가 분석을 수행할 수 있다. 본 논문에서는 온라인 알고리즘이 보다 많은 머신들을 이용할 수 있을 때 최적 알고리즘과 같은 성능을 낼 수 있음을 보일 것이다.

ABSTRACT

In this paper, we consider the online scheduling problem of jobs with deadlines. The jobs arrive over time and the scheduling algorithm has no information about the arriving jobs in advance. The jobs have the processing time of the equal length and the goal of the scheduling algorithm is to maximize the number of jobs completed in their deadlines. The performance of the online algorithm is compared with that of the optimal algorithm which has the full information about all the jobs. The raio of the two performances is called the competitive ratio. In general, the ratio is unbouned. So the case that the online algorithm can have more resources than the optimal algorithm is considered, which is called the resource augmentation analysis. In this paper, the online algorithm have more machines. We show that the online algorithm can have the same performance as the optimal algorithm.

키워드 : 온라인 스케줄링, 마감시간 스케줄링, 경쟁비, 자원추가 분석

Key word : Online scheduling, Deadline scheduling, Competitive ratio, Resource augmentation analysis

접수일자 : 2013. 04. 29 심사완료일자 : 2013. 07. 11 게재확정일자 : 2013. 07. 25

* **Corresponding Author** Jae-hoon Kim(E-mail:jhoon@bufs.ac.kr, Tel:+82-51-640-3421)
Department of Computer Engineering, Busan University of Foreign Studies, Busan 608-738, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2013.17.8.1934 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

본 논문은 마감 시간을 가진 작업들을 스케줄하는 문제를 다룬다. 작업 J_i는 도착시간 r_i, 수행시간 p_i, 마감시간 d_i 그리고 가중치 w_i를 가지고 주어진다. 머신은 작업을 도착시간 r_i 이후에 스케줄할 수 있고 p_i 동안 수행해서 마감시간 d_i 이전에 완료한다면 가중치 w_i를 얻는다. 스케줄링 알고리즘의 목표는 완료된 작업들의 가중치들의 합을 최대로 하는 것이다. 본 논문에서는 작업들의 가중치 w_i가 수행시간 p_i와 같은 경우를 다룰 것이고 모든 작업들의 수행시간이 동일하다고 가정할 것이다. 특별히, $p_i = 1$이라고 가정한다. 따라서, 스케줄러의 목표는 마감시간 전에 완료하는 작업들의 수를 최대로 하는 것과 같다.

작업들이 완료되기 위해서 마지막으로 스케줄될 수 있는 시간을 만료시간(expiration time)이라고 한다. 다시 말해서, 만료시간 x_i는 $d_i - p_i$으로 정의한다. 또한 $s_i = x_i - r_i$를 여유시간(slack time)이라고 한다. 이것은 작업이 완료되기까지 최대로 기다릴 수 있는 시간을 말한다. 여기서, 모든 작업 J_i에 대해서, $s_i \geq \kappa p_i$를 만족하는 상수 κ가 존재하면, κ를 인내도(patience)라고 한다. 이 요소는 [6]에서 처음으로 제안되었다. 특히, $\kappa = 0$인 경우는 $s_i \geq 0$로 임의로 주어질 수 있는 가장 일반적인 경우를 말한다. 본 논문은 $\kappa = 0$인 일반적인 경우를 다룰 것이다.

스케줄링 알고리즘은 온라인(online) 상황을 다룰 것이다. 여기서, 알고리즘은 작업들이 도착하기 전에는 작업들의 정보를 미리 알 수 없고 도착하고 나서 알 수 있다. 따라서 앞으로 도착할 작업들의 정보 없이 현재 도착한 작업들의 정보만을 이용해서 작업들의 스케줄을 결정해야 한다. 이러한 알고리즘을 온라인 알고리즘(online algorithm)이라고 한다. 온라인 알고리즘의 성능은 작업들의 모든 정보를 알고 최고의 성능을 내는 최적 알고리즘과 비교된다.

작업들의 집합 $\mathfrak{I}$에 대해서, 알고리즘 A가 마간시간 안에 완료한 작업들의 집합을 $A(\mathfrak{I})$로 표시한다. 또한 최적 알고리즘 OPT가 완료한 작업들의 집합을 $OPT(\mathfrak{I})$라고 할 때, 아래 식을 만족하면 알고리즘 A는 c-competitive 라고 한다:

$$\forall\, \mathfrak{I},\ |OPT(\mathfrak{I})| \leq c|A(\mathfrak{I})|.$$

$c = 1$이면, 알고리즘 A가 최적 알고리즘 OPT와 같은 성능을 가진다. 이 경우에, A를 최적 온라인 알고리즘이라고 한다.

일반적으로 온라인 알고리즘은 미래의 정보를 알 수 없기 때문에 최적이 되기 어렵다. 따라서 온라인 알고리즘에 최적 알고리즘보다 더 많은 자원을 제공해서 최적이 되는지 알아본다. 이것을 자원추가(resource augmentation) 분석이라고 한다. 본 논문에서는 최적 알고리즘보다 하나 더 많은 머신을 사용할 수 있도록 할 것이다. 다시 말해, 본 논문의 온라인 알고리즘은 두 개의 머신을 사용할 수 있다.

본 논문에서 다루는 스케줄링 알고리즘은 비선점(non-preemptive)이다. 다시 말해서, 한 번 스케줄된 작업은 수행 도중에 중단 되거나 제거될 수 없다. 이와 대조적으로, 작업들이 수행 도중에 중단되었다가 나중에 중단된 지점부터 다시 수행될 수 있는 스케줄링을 선점(preemptive)이라고 한다.

Ⅱ. 관련 연구

마감시간을 가진 작업들의 스케줄링은 스케줄링 분야에서 주요한 문제들 중 하나이다. 따라서 작업들의 정보를 미리 다 알고 있는 오프라인(offline) 상황에 대한 많은 연구들이 있어왔다 [1, 2, 3].

온라인 상황에서 마감시간 스케줄링에 대한 연구는 [4]에서 시작되었다. 저자들은 임의의 수행시간을 가지는 작업을 도착하자마자 스케줄해야 하는 문제를 다루었다. 그들은 $O((\log\Delta)^{1+\epsilon})$-competitive 알고리즘을 제안하였다. 여기서, Δ는 가장 큰 수행시간과 가장 작은 수행시간 간의 비이다.

$\kappa = 0$인 임의의 여유시간을 가지는 작업들에 대해서 작업들의 수행시간이 모두 동일한 경우에는 [5]에서 2-competitive 알고리즘이 제안되었다.

[6]에서는 인내도 κ를 소개하고 $(1 + \frac{1}{\lfloor \kappa \rfloor + 1})$-competitive 알고리즘을 소개하였다. [7]에서는 이 결과를 임의의 개수 $m \geq 1$의 머신들의 경우로 확장하

였다.

이상의 연구들은 온라인 알고리즘과 최적 알고리즘들이 동일한 조건 하에서의 성능 비교를 다룬다. 본 논문에서는 온라인 알고리즘에 보다 많은 자원을 제공하였을 때 최적 알고리즘과의 성능 차이를 비교하는 자원추가 분석을 수행한다. 이 자원추가 분석은 [8]에서 처음 제안되었고 [9]에서 마감시간 선점 스케줄링에 적용해서 2배 빠른 머신에서 최적 온라인 알고리즘을 보였다. 본 논문에서는 최적 알고리즘보다 많은 머신들을 사용하는 경우를 다룰 것이다.

Ⅲ. 알고리즘

이 장에서는 논문에서 제안하는 알고리즘 A_2에 대해서 설명할 것이다. 이 알고리즘은 2개의 머신들을 사용할 수 있다. 반면에 최적 알고리즘OPT는 1개의 머신만을 사용한다.

알고리즘 A_2는 2개의 머신 M_1과 M_2를 사용한다. A_2는 매 시간 다음 조건들을 만족시키면서 동작한다:

1. 머신 M_1가 놀고 있는(idle) 경우에 스케줄 가능한 작업들 중에서 가장 이른 만료시간을 가진 작업을 M_1에 스케줄한다.
2. 현재 M_1에 작업 J가 수행 중 일 때, J의 수행이 완료되는 시간을 t라고 하면, 도착한 작업들 중에 만료시간이 t이전인 작업들이 존재하면 이 작업들 중에서 만료시간이 가장 이른 작업을 머신 M_2에 스케줄한다.

위의 조건 2에서 J의 완료시간 t에 대해서 도착한 작업 중에 만료시간이 t보다 작은 작업들을 급한 작업(urgent job)이라고 부른다. 알고리즘 A_2에서는 급한 작업들만이 머신 M_2에 스케줄 됨을 주목한다.

알고리즘 A_2에 의한 작업들의 스케줄을 생각할 때 머신 M_1과 M_2 중 적어도 하나의 머신에서 작업이 수행 중인 구간을 생각할 수 있고 이 구간을 바쁜 구간(busy period)라고 부른다. 우리는 알고리즘 A_2의 스케줄이 하나의 바쁜 구간을 가진다고 가정할 수 있다. 왜냐하면, 여러 개의 바쁜 구간 T_1, ..., T_m을 가진다고 가정하면, 구간 T_i 안에 도착한 작업들의 집합 $\mathfrak{I}_i$을 생각한다. A_2는 집합 $\mathfrak{I}_i$에 속한 작업들을 구간 T_i 안에 스케줄 할 것이다. 따라서 $A_2(\mathfrak{I}) = \sum_{i=1}^{m} A_2(\mathfrak{I}_i)$ 이고, 각 i에 대해서, $OPT(\mathfrak{I}_i) \le cA_2(\mathfrak{I}_i)$이면, $OPT(\mathfrak{I}) \le \sum_{i=1}^{m} OPT(\mathfrak{I}_i) \le c\sum_{i=1}^{m} A_2(\mathfrak{I}_i) = cA_2(\mathfrak{I})$을 보일 수 있다. 따라서 일반성의 손실 없이, A_2의 스케줄이 하나의 바쁜 구간으로 구성되고 작업들은 이 구간 안에 도착한다고 가정할 수 있다.

Ⅳ. 분 석

알고리즘 A_2의 경쟁비를 구하기 전에 주어지는 사례(instance)를 제한할 것이다. A_2가 바쁜구간 $[0, \alpha]$ 동안 수행될 때, 다음 조건을 만족하는 사례 I를 생각할 것이다.

1. A_2는 시간 0에서 M_1에 작업 J를 수행한다.
2. $I - \{J\}$의 각 작업은 α보다 작은 만료시간을 가진다.

이 사례 I를 기본사례(fundamental instance)라고 부를 것이다. 이 사례는 1개 머신의 온라인 마감시간 스케줄링 문제에 대한 연구[6]에서 정의되었고, 우리는 여기서 2개 머신들의 경우로 확장할 것이다. [6]에서 저자는 기본사례에 대해서 욕심쟁이(greedy) 알고리즘이 경쟁비 c를 가지면, 모든 사례들에 대해서 경쟁비 c를 가짐을 보였다. 알고리즘 A_2도 위와 같은 성질을 가짐을 비슷하게 보일 수 있다.

보조 정리 4.1 알고리즘 A_2가 기본사례에 대해서 c-competitive 이면, A_2는 모든 사례에 대해서 c-competitive 이다.

위의 보조정리에 의해서 앞으로 기본사례만을 생각할 것이고 기본사례 I에 대해서 A_2가 적어도 OPT가 스케줄하는 작업들의 수 이상을 스케줄함을 보일 것이다. 다시 말해서, 경쟁비 1을 가짐을 보일 것이다.

기본사례 I에 대해서, A_2에 의해서 시간 0에 스케줄되는 작업 J가 α보다 작은 만료시간을 가지면 모든 작업들이 α보다 작은 만료시간을 가지므로 OPT는 많아야 $\lceil \alpha \rceil$ 개의 작업들만을 스케줄할 수 있다. 하지만 A_2는 이 구간동안 바쁘기 때문에 적어도 $\lceil \alpha \rceil$ 개 이상의 작업들을 스케줄할 수 있다. 따라서 $|A_2(I)| \geq |OPT(I)|$.

앞으로는 작업 J가 α 이상의 만료시간을 가진다고 가정할 것이다. OPT는 구간 $[0, \alpha]$에 많아야 $\lceil \alpha \rceil$ 개의 작업과 작업 J를 스케줄 할 수 있다.

우리는 OPT가 구간 $[0, \alpha]$에 스케줄하는 작업들의 집합 $OPT[0, \alpha]$에서 A_2가 스케줄하는 작업들의 집합 $A_2(I)$으로의 매핑 f를 정의할 것이다. 작업 $H \in OPT[0, \alpha]$에 대해서, H가 스케줄된 시간을 s라고 하면, A_2에서 시간 s에 수행되고 있는 작업을 K라고 하면 $f(H) = K$로 정의한다.

여기서, 시간 s에 수행되고 있는 작업이 2개이면, 머신 M_1에 수행되고 있는 작업을 K로 한다. 매핑 f가 일대일(one-to-one) 매핑이라는 것은 쉽게 증명할 수 있다.

매핑 f의 치역 $f(I)$에 작업 J가 속하지 않는다면, OPT가 구간 $[0, \alpha]$에 많아야 $\lceil \alpha \rceil - 1$ 개의 작업들을 스케줄함을 알 수 있고, 따라서, $|OPT(I)| \leq \lceil \alpha \rceil \leq |A_2(I)|$.

앞으로 우리는 작업 J가 매핑 f의 치역 $f(I)$에 속하는 경우만을 생각할 수 있다. 우리는 A_2가 작업 J를 제외하고도 적어도 $\lceil \alpha \rceil$ 개 이상의 작업을 스케줄 할 수 있음을 증명할 것이다.

적어도 하나의 작업 $H \in OPT[0, \alpha]$에 대해서, H가 스케줄된 시간 s에서 A_2가 2개의 작업을 수행하였다고 가정하자. 다시 말해서, 머신 M_1과 M_2 모두에서 작업을 수행하고 있다. 이 경우는 아래의 그림 1과 그림 2와 같다.

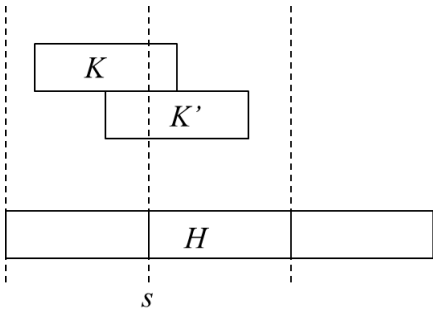

그림 1. 작업 H가 매핑 f에 의해 작업 K에 매핑되는 첫 번째 경우
Fig. 1 The first case : job H is mapped into job K by a mapping f

이 두 경우 모두 작업 H는 매핑 f에 의해서 머신 M_1에 수행되는 작업 K에 매핑된다. 그러면 시간 s에 머신 M_2에 수행되고 있는 작업 K'에 대해서, f에 의해 K'에 매핑되는 작업이 존재하지 않음을 그림에서처럼 쉽게 알 수 있다. 다시 말해서, 작업 H의 앞, 뒤에 스케줄된 작업들의 수행 시작 시간은 K'의 수행 구간 안에 속할 수 없다.

결과적으로 $|A_2(I)| \geq |OPT[0, \alpha]| + 1$ 임을 보인 것이다. 따라서, $|A_2(I)| \geq |OPT(I)|$.

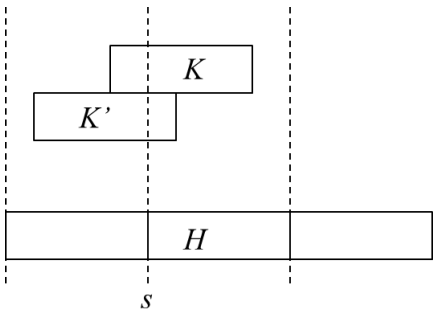

그림 2. 작업 H가 매핑 f에 의해 작업 K에 매핑되는 두 번째 경우
Fig. 2 The second case : job H is mapped into job K by a mapping f

우리는 남아있는 한 가지 경우를 생각한다. 모든 작업 $H \in OPT[0, \alpha]$가 스케줄된 시간 s에서 A_2가 1개의 작업만을 수행하고 있는 경우이다. 모든 작업

$H \in OPT[0,\alpha]$가 A_2에 의해서 스케줄되면 증명은 끝난 것이다.

따라서, 적어도 하나의 작업 $H \in OPT[0,\alpha]$가 A_2에 의해서 스케줄되지 않는다고 가정한다. 그런 작업 중에서 가장 먼저 스케줄된 작업 $H \in OPT[0,\alpha]$를 생각한다. H가 스케줄된 시간을 s라고 하자. $f(H)$가 머신M_2에서 수행되는 작업이면, A_2에서 시간 s에 머신 M_1 은 놀고 있다. 이것은 모순이다. 왜냐하면 작업 H가 시간 s에 머신 M_1에 스케줄 될 수 있기 때문이다. $f(H)=K$가 머신 M_1에서 수행되는 작업이고 K의 수행 완료시간이 h라고 하자. 작업 H의 만료시간이 h보다 작다면, A_2의 정의에 의해서 시간 s에 머신 M_2에 스케줄 될 수 있었기 때문에 모순이다. 따라서 작업 H의 만료시간은 h 이상이다. 작업 H의 만료시간을 e라고 하자.

시간 e에 머신 M_1에서 어떤 작업 O가 수행 중 이여야 한다. 아니면, H가 M_1에 스케줄 될 수 있다. 작업 O의 수행 시작시간을 l이라고 하자. 머신 M_1에서 시간 l에서 어떤 작업이 끝나고 작업O가 시작되어야 한다. 아니면, H가 시간 l에 M_1에 스케줄 될 수 있다. 또한 시간 e에 머신 M_2에 어떤 작업 P를 수행하고 있어야 한다. 따라서 작업 P의 수행시간 동안 머신 M_1과 M_2가 동시에 수행되고 있어야 한다.

모든 작업 $H \in OPT[0,\alpha]$가 스케줄된 시간 s에서 A_2가 1개의 작업만을 수행하고 있어야 하기 때문에 매핑 f는 작업 P에 매핑될 수 없다. 결과적으로 매핑 f에 의해 매핑되지 않은 적어도 하나의 작업을 찾을 수 있고 $|A_2(I)| \geq |OPT[0,\alpha]|+1$임을 보였다.

지금까지 알고리즘 A_2가 최적 알고리즘이 스케줄하는 작업들의 수 이상을 스케줄할 수 있음을 보였다. 결과적으로 다음의 정리를 얻는다.

정리 4.2 임의의 작업 사례 $\mathfrak{I}$에 대해서,

$$OPT(\mathfrak{I}) \leq A_2(\mathfrak{I}).$$

Ⅴ. 결 론

본 논문에서는 마감시간을 가진 동일 길이 작업들의 스케줄링 문제를 다루었다. 스케줄링 알고리즘의 목표는 완료된 작업들의 수를 최대화하는 것이다. 우리는 두 개의 머신들을 사용하면 한 개의 머신을 사용하는 최적알고리즘과 같은 성능을 가지는 알고리즘이 존재함을 보였다.

향후 연구로는 본 연구를 선점(preemptive) 스케줄링의 경우로 확장하여 적용해 보는 것이고, 또한 서로 다른 임의의 길이를 가진 작업들의 마감시간 온라인 스케줄링을 다룰 수 있을 것이다.

감사의 글

본 연구는 2013년도 부산외국어대학교 학술연구조성비에 의해 연구되었으므로, 관계부처에 감사 드립니다.

REFERENCES

[1] J. M. Moore, “A n jobs, one machine sequencing algorithm for minimizing the number of late jobs”, *Management Science*, vol. 15, pp. 102-109, Sep. 1968.

[2] Philippe Baptiste, “An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs”, *Operational Research Letters*, vol. 24, pp. 175-180, May 1999.

[3] Philippe Baptiste, “Polynomial time algorithms for minimizing the weighted number of jobs on a single machine with equal processing times”, *Journal of Scheduling*, vol. 2, pp. 245-252, Nov. 1999.

[4] R. Lipton and A. Tomkins, “Online interval scheduling”, in *Proceeding of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, Arlington, Virginia, pp. 302-311, 1994.

[5] S. Goldman, J. Parwatikar, and S. Suri, “On-line scheduling with hard deadlines”, in *Proceeding of the 5th International Workshop on Algorithms and Data Structure*, Halifax,

Canada, pp. 258-271, 1997.

[6] M. H. Goldwasser, “Patience is a virtue: The effect of slack on competitiveness for admission control”, in *Proceeding of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, pp. 396-405, 1999.

[7] Jae-Hoon Kim and Kyung-Yong Chwa, “Online deadline scheduling with multiple resources”, in *Proceeding of the 7th Annual Int. Computing and Combinatorics Conference*, Guilin, China, pp. 443-452, 2001.

[8] C. Phillips, C. Stein, E. Torng, and J. Wein, “Optimal time-critical scheduling via resource augmentation”, in *Proceeding of the 29th Annual ACM-SIAM Symposium on Theory of Computing*, El Paso, Texas, pp. 140-149, 1997.

[9] T. W. Lam and K. K. To, “Performance guarantee for online deadline scheduling in the presence of overload”, in *Proceeding of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, pp. 755-764, 2001.

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