가중치 작업들의 온라인 비선점 마감시한 스케줄링
(Online Non-preemptive Deadline Scheduling for Weighted Jobs)

김 재 훈 † 장 정 환 ††
(Jae-Hoon Kim) (Jung-Hwan Chang)

요 약 마감시한 스케줄링에서 작업들은 수행을 완료해야 하는 마감시한을 가진다. 스케줄링 알고리즘은 각 시간에 어떠한 작업이 수행되어야 하는 지 결정한다. 수행이 마감시한 안에 완료된 작업들만이 알고리즘의 처리량 또는 이익에 포함된다. 작업들은 임의의 가중치를 가지고 알고리즘의 이익은 완료된 작업들의 가중치의 합으로 주어진다. 스케줄링 알고리즘의 목표는 이익을 최대화하는 것이다.

본 논문에서 우리는 온라인 비선점 스케줄링을 생각한다. 여기서 작업들은 온라인으로 도착하고, 스케줄링 알고리즘은 앞으로 도착할 작업들의 정보를 미리 알 수 없다. 작업들은 스케줄 되고 나서 수행 도중에 멈출 수 없다. 이 문제에 대해서, 우리는 임의의 온라인 알고리즘의 성능의 하한을 구한다. 또한 이 하한과 일치하는 성능을 가지는 최적의 온라인 알고리즘을 제안하다.

키워드 : 온라인 알고리즘, 스케줄링, 마감시한, 경쟁분석

Abstract In deadline scheduling, jobs have deadlines by which they are completed. The scheduling algorithm determines which jobs are executed at each time. Then only the completed jobs contribute to the throughput or gain of the algorithm. The jobs have arbitrary weights and the gain of the algorithm is given as the sum of weights of the completed jobs. The goal of the scheduling algorithm is to maximize its gain.

In this paper, we consider online non-preemptive scheduling, where jobs arrive online and the scheduling algorithm has no information about jobs arriving ahead. Also the jobs cannot be preempted or rejected while they are executed. For this problem, we obtain lower bounds for any online algorithms and also we propose an optimal online algorithm meeting the lower bounds.

Key words : online algorithm, scheduling, deadline, competitive analysis

1. 서 론

스케줄링 문제는 알고리즘 분야의 가장 기본이 되는 문제들 중 하나이다. 특히, 이 논문에서는 마감시한을 가진 작업들을 스케줄하는 문제를 다룬다. 작업들은 시간이 지나면서 하나씩 도착한다. 각 작업은 마감시한과 가중치를 가지고 있다. 이 작업들을 수행할 기계가 존재하고, 스케줄링 알고리즘은 작업들을 적당한 시간에 기계에 스케줄해서 수행한다. 각 작업은 마감시한 안에 수행을 완료해야 하며, 완료하지 못한 작업은 스케줄링 알고리즘의 *처리량(throughput)* 또는 *이익(gain)*에 포함되지 못한다. 그래서 알고리즘의 이익은 마감시한 안에 수행을 마친 작업들의 가중치의 합이 된다. 스케줄링 알고리즘의 목표는 이 가중치의 합을 최대화하는 것이다.

스케줄링 알고리즘은 도착한 작업들만을 고려할 수 있고 앞으로 도착할 작업들의 정보를 가지고 있지 않다. 이러한 상황을 *온라인(on-line)*이라고 한다. 그래서 알고리즘은 기계가 놀고 있을 때마다 현재 도착한 작업들 중에 하나를 기계에 스케줄한다. 스케줄링 알고리즘이 작업을 스케줄하고 나면, 이 작업은 수행도중 멈출 수 없고 끝까지 수행된다. 이런 스케줄링을 *비선점(non-preemptive)*이라고 한다. 온라인 알고리즘의 성능은 *경쟁분석(competitive analysis)*을 통해서 분석된다. 주어질 전체 작업들을 미리 알고 최적의 답을 주는 *오프라인(off-line) 최적(optimal)* 알고리즘을 생각할 수 있고, 온라인 스케줄링 알고리즘의 이익을 오프라인 최적 스케줄링 알고리즘의 이익과 비교한다.

논문을 통해서 *OPT*는 오프라인 최적 스케줄링 알고리즘을 나타내고 동시에 최적 알고리즘에 의해서 스케줄된 작업들의 집합을 나타낸다. 또한 임의의 온라인 알

† 비 회 원 : 부산외국어대학교 컴퓨터공학부 교수
jhoon@pufs.ac.kr
†† 종신회원 : 부산외국어대학교 디지털정보공학부 교수
jhchang@pufs.ac.kr
논문접수 : 2004년 6월 25일
심사완료 : 2004년 11월 10일

고리즘 A에 대해서 A는 알고리즘에 의해서 스케줄된 작업들의 집합을 나타낸다. 그리고, 어떤 집합 S에 대해서 $||S||$는 집합 S에 속한 작업들의 가중치의 합을 나타낸다. 따라서 $||A||, ||OPT||$는 각각 온라인 알고리즘 A, 오프라인 최적 알고리즘 OPT의 이익을 나타낸다. 그래서 임의의 작업들의 집합에 대해서 $||OPT|| \le c\,||A||$를 만족하면, 온라인 알고리즘 A는 c *competitive*라고 한다. 특히, 위 부등식을 만족하는 c의 *최대하계(infimum)*를 A의 *경쟁비율(competitive ratio)*이라고 한다. 온라인 알고리즘 A가 상수의 경쟁비율을 가질 때, A는 *경쟁적(competitive)*이라고 한다.

많은 온라인 문제에서 반대자(adversary)의 힘이 너무 강해서 온라인 알고리즘은 경쟁적이지 않다. 따라서 온라인 알고리즘에 OPT에 대항해서 보다 많은 자원, 예를 들어, 빠른 속도의 기계 또는 보다 많은 기계들이 주어지는 모델이 제시되었고, 이를 *자원증대(resource augmentation)*라고 부른다. 본 논문에서는 또한 이러한 모델에서의 온라인 알고리즘의 성능을 분석할 것이다. 특히, 온라인 알고리즘이 s-speed 기계를 사용할 때, 1 speed 기계를 가진 OPT와 성능을 비교할 것이다.

임의의 작업 J_i에 대해서, a_i는 도착시간, p_i는 수행시간, d_i는 마감시한, w_i는 가중치라고 하자. 우리는 시간 $x_i = d_i - p_i$를 생각할 수 있고, 이 시간은 작업 J_i가 마감시한 안에 수행 되도록 스케줄 될 수 있는 마지막 시간이고, *만료시간(expiration time)*이라고 부른다. 또한 시간차 $s_i = x_i - a_i$를 *여유시간(slack time)*이라고 부른다. 이 때, 모든 작업 J_i에 대해서 $s_i \ge \varkappa p_i$를 만족하는 상수 $\varkappa$가 존재하면, $\varkappa$를 *인내도(patience)*라고 한다. 특히, $\varkappa = 0$인 경우는 도착하는 작업 중에 도착하자마자 스케줄 되지 않으면 마감시한 안에 마칠 수 없는 작업들(다시 말해, $s_i = 0$)이 포함되는 가장 일반적인 경우에 해당한다. 그리고 $\lambda_i = \frac{w_i}{p_i}$를 작업 J_i의 *가중치량(value density)*이라고 하며, 최대 가중치량과 최소 가중치량의 비를 *중요도비율(importance ratio)*이라고 하고, λ로 나타낸다. 또한 주어진 작업 수행시간 중 최대 수행시간을 Δ로 나타낸다.

1.1 관련연구

각각의 작업들의 가중치가 작업 수행시간으로 주어지는 경우, 다시 말해, $w_i = p_i, \forall i$ 이고 $\lambda = 1$, 알고리즘의 이익은 수행 완료된 작업들의 수행시간들의 합으로 주어지는데, 이 문제에 대한 연구는 선점[1,2,5], 비선점[7,8,13] 스케줄링 모두 많이 존재한다. 선점 스케줄링의 경우에, 일반적으로($\varkappa = 0$의 경우에) 임의의 온라인 알고리즘은 4-competitive 보다 좋을 수 없다는 것이 알려져 있고[2], Garay et.al.[5]는 $\min\{5.828,\ 2+\frac{1}{\varkappa}\}$-competitive 알고리즘을 제안하였다. 비선점 스케줄링의 경우에, Lipton과 Tomkins[13]는 $O((\log \Delta)^{1+\varepsilon})$-competitive 알고리즘을 제안했고, 임의의 온라인 알고리즘이 $O(\log \Delta)$-competitive 보다 좋을 수 없음을 증명하였다. Goldwasser[8]는 $\varkappa > 0$인 경우에, 기계가 노는 시간마다 도착된 작업들 중에서 만료시간이 제일 작은 작업을 스케줄하는 알고리즘이 $(2+\frac{1}{\varkappa})$-competitive 임을 보였고, 임의의 온라인 알고리즘이 이 보다 적은 경쟁비율을 가질 수 없음을 증명함으로서 이것이 최적의 온라인 알고리즘임을 증명하였다.

본 논문에서는 각각의 작업들이 임의의 가중치 값을 가질 수 있는 일반적인 경우를 생각한다. 이 문제의 어려움은 주어지는 모든 작업들의 정보를 미리 모두 알고 스케줄하는 오프라인 문제에 대해서, 작업의 가중치들이 모두 동일한 경우에도 strongly NP-hard 임이 증명되었다[6]. 온라인 문제에 대해서, 선점 스케줄링의 경우에 $(1+\sqrt{\lambda})^2$-competitive 알고리즘이 존재한다[10]. 그리고 임의의 온라인 알고리즘은 $(1+\sqrt{\lambda})^2$ 보다 작은 경쟁비율을 가질 수 없음이 증명되었다[1,2]. 그러나 비선점 스케줄링의 경우에는 알려진 논문이 존재하지 않는다. 본 논문에서는 비선점 스케줄링에서 온라인 알고리즘의 성능에 관해서 알아본다.

자원증대 모델은 Kalyanasundaram과 Pruhs[9]에 의해서 처음으로 제안되었다. [9]에서는 본 논문에서처럼 임의의 가중치 값을 가진 작업들의 스케줄링을 다루고, 임의의 상수 $\varepsilon > 0$에 대해서 $(1+2\varepsilon)$-speed $(1+\varepsilon^{-1})(1+\varepsilon^{-1/2})(1+\varepsilon^{-1/2}+\varepsilon^{-1})$-competitive 알고리즘이 존재함을 보였다. 이어서 Lam과 To[12]는 이 경쟁비율을 $1+2\varepsilon^{-1}+4\varepsilon^{-2}$으로 향상시켰다. 이 모델은 그 밖의 여러 다른 온라인 스케줄링 문제에 적용되었다[3,4,14].

2. 임의의 가중치를 가진 작업 스케줄링

본 절에서는 작업의 가중치들이 임의의 값으로 주어질 때, 우선 임의의 온라인 알고리즘이 가질 수 있는 경쟁비율의 하한에 관해서 생각하고, 그 하한과 일치하는 경쟁비율을 가지는 최적의 온라인 알고리즘을 제안한다.

2.1 온라인 알고리즘의 성능 하한

다음에서 우리는 임의의 온라인 알고리즘이 가질 수 있는 경쟁비율의 두 가지 하한을 얻는다. 하나는 $\varkappa = 0$인 일반적인 입력 작업이 주어질 때, λ, Δ에 의해서 주어지고, 다른 하나는 $\varkappa > 0$인 경우에 $\lambda, \varkappa$에 의해서 주

어진다.

정리 1. $x=0$ 인 일반적인 작업이 입력으로 주어질 때, 중요도비율 λ, 최대 수행시간 Δ에 대해서, 임의의 온라인 알고리즘은 $\lambda(\Delta+1)+1$ 보다 적은 경쟁비율을 가질 수 없다.

증명. 시간 0 에 $1+2\varepsilon$ $(\varepsilon>0)$의 수행시간, 무한대의 마감시한, 가중치 $1+2\varepsilon$을 가지는 작업 J_1이 도착한다. 임의의 온라인 알고리즘 A는 적당한 시간 t에 작업 J_1을 스케줄해야 한다. 왜냐하면, A가 어떠한 시간에도 J_1을 스케줄하지 않는다면 반대자는 J_1이외에 더 이상 아무런 작업도 도착시키지 않는다. 그러면 OPT는 J_1을 스케줄하고 $1+2\varepsilon$의 이익을 얻고, A는 무한대의 경쟁비율을 가진다. 시간 $t+\varepsilon$에 수행시간 1, 마감시한 $t+1+\varepsilon$, 가중치 λ를 가지는 작업 J_2가 도착한다. 이 작업은 도착하자마자 스케줄되지 않으면 마감시한 안에 수행을 마칠 수 없다. 하지만, 알고리즘 A는 작업 J_1이 수행 중이므로 작업 J_2를 수행하지 못한다. 그러면 시간 $t+1+\varepsilon$에 수행 시간 Δ, 마감시한 $t+1+\Delta+\varepsilon$, 가중치 $\Delta\lambda$를 가지는 또 다른 작업 J_3가 도착한다. 작업 J_3 역시 도착하자마자 스케줄 되어야 하지만 A는 또한 수행중인 작업 J_1 때문에 작업 J_3를 수행하지 못한다. 그러나 최적 오프라인 알고리즘은 작업 J_1의 스케줄을 뒤로 미루고, 시간 $t+\varepsilon$에 작업 J_2, 시간 $t+1+\varepsilon$에 작업 J_3, 마지막에 작업 J_1을 수행함으로써 모든 작업들을 수행할 수 있다. 그래서 임의의 온라인 알고리즘 A의 경쟁비율은 $\frac{\lambda(\Delta+1)+1+2\varepsilon}{1+2\varepsilon}$ 이다. 충분히 작은 ε을 생각하면, $\lambda(\Delta+1)+1$을 얻는다. □

다음으로, $x>0$ 인 경우, 다시 말해, 각각의 작업이 0이 아닌 여유시간을 가지는 경우에 임의의 온라인 알고리즘의 경쟁비율의 하한을 구한다.

정리 2. $x>0$ 인 경우에, 중요도비율 λ, 인내도 $x>0$에 대해서, 임의의 온라인 알고리즘은 $\lambda(1+\frac{1}{x})+1$ 보다 적은 경쟁비율을 가질 수 없다.

증명. 시간 0 에 수행시간 $\lceil x\rceil+1+2\varepsilon$ $(\varepsilon>0)$, 무한대 마감시한, 가중치 $\lceil x\rceil+1+2\varepsilon$을 가지는 작업 J_1이 도착한다. 정리 1의 증명에서처럼 임의의 온라인 알고리즘 A는 무한대의 경쟁비율을 갖지 않기 위해서 작업 J_1을 적당한 시간 t에 스케줄한다. 그러면 시간 $t+\varepsilon$에 수행시간 1, 마감시한 $t+\lceil x\rceil+1+\varepsilon$, 가중치 λ를 가지는 $\lceil x\rceil+1$ 개의 작업들이 도착한다. 같은 시간에 수행시간 $\frac{\lceil x\rceil+1}{x}$, 가중치 $(\frac{\lceil x\rceil+1}{x})\lambda$를 가지는 또 하나의 작업 J_2가 도착한다. 또한 J_2의 마감시한은 $t+(\lceil x\rceil+1)(1+\frac{1}{x})+\varepsilon$ 으로 주어진다(이상 주어진 작업들은 인내도 x를 정의하는 부등식을 만족함에 주목한다). 그러면 온라인 알고리즘 A는 작업 J_1의 수행 때문에 시간 $t+\varepsilon$에 주어진 모든 작업들을 스케줄하지 못한다. 왜냐하면, J_1은 시간 $t+\lceil x\rceil+1+2\varepsilon$까지 수행되는데, $\lceil x\rceil+1$ 개의 작업들은 만료시간이 $t+\lceil x\rceil+\varepsilon$이고 J_2의 만료시간은 $t+\lceil x\rceil+1+\varepsilon$이다. 하지만 OPT는 작업 J_1을 뒤로 미루고 시간 $t+\varepsilon$부터 수행시간 1의 $\lceil x\rceil+1$ 개의 작업들을 차례로 스케줄한 후 시간 $t+\lceil x\rceil+1+\varepsilon$에 작업 J_2를 스케줄해서 주어진 모든 작업들을 스케줄할 수 있다. 그리하여 A의 경쟁비율은 $\frac{\lambda(\lceil x\rceil+1)}{\lceil x\rceil+1+2\varepsilon}(1+\frac{1}{x})+1$ 이다. ε를 충분히 작게 선택하면, 경쟁비율은 $\lambda(1+\frac{1}{x})+1$ 이다. □

다음으로 온라인 알고리즘이 OPT보다 빠른 기계를 가지고 있을 때, 알고리즘이 가질 수 있는 경쟁비율의 하한을 얻는다. 선점 스케줄링의 경우에는 [12]에서 상수 속도의 기계를 가지고 상수 경쟁비율을 가지는 알고리즘이 존재한다. 예를 들어, 3-speed 7-competitive 알고리즘이 존재한다. 하지만 비선점 스케줄링의 경우에는, λ보다 작은 속도를 가지는 기계를 사용한다면, 온라인 알고리즘은 상수의 경쟁비율을 가질 수 없음을 증명할 것이다.

정리 3. 중요도비율 λ와 임의의 실수 $\varepsilon>0$에 대해서, $\lambda^{1-\varepsilon}$-speed 기계를 사용하는 임의의 온라인 알고리즘은 $1+\lambda^{\varepsilon}$보다 적은 경쟁비율을 가질 수 없다.

증명. 기계의 속도 $s=\lambda^{1-\varepsilon}$가 주어지고, 충분히 큰 상수 Δ에 대해서, 시간 0 에 수행시간 $p_1=\Delta$과 무한 마감시한을 가진 작업 J_1이 도착한다. 작업 J_1은 최소의 가중치량 $v_{\min}$을 가진다. 다시 말해, 작업 J_1의 가중치 $w_1=v_{\min}\Delta$. 임의의 온라인 알고리즘 A는 무한의 경쟁비율을 가지지 않기 위해서 작업 J_1을 적당한 시간 t에 스케줄하고 그것을 구간 $[t, t+p_1')$ 동안 수행한다. 여기서, $p_1'=\frac{\Delta}{s}$ 이다. 그러면 시간 t가 지나고 바로, 정확히 말해서 시간 $t+\delta$(충분히 작은 $\delta>0$)에 수행시간 1을 가지고 마감시한 $t+\frac{\lceil\Delta\rceil}{s}+\delta$을 가진 $\lceil\Delta\rceil$ 개의 작업들이 도착한다. δ를 충분히 작게 잡으면, A는 작업 J_1의 수행으로 주어진 $\lceil\Delta\rceil$ 개의 작업을 모두

스케줄 할 수 없다. 또한 그 작업들은 최대 가중치량 v_{max} 를 가진다. 그러나 OPT는 작업 J_1의 스케줄을 뒤로 미루고 $\lceil \Delta \rceil$ 개의 작업 중 $\frac{\Delta}{s}$ 개를 스케줄할 수 있다. 여기서 Δ는 $\frac{\Delta}{s}$ 가 정수 값이 되도록 주어졌다. 따라서 경쟁비율은 $1+\frac{v_{\max}\Delta/s}{v_{\min}\Delta} \geq 1+\lambda^{\varepsilon}$ 이다. □

2.2 최적 온라인 알고리즘

여기서 우리는 이전 2.1절에서 주어진 임의의 온라인 알고리즘의 두 하한을 경쟁비율로 성취하는 최적의 온라인 알고리즘을 얻는다. 우리는 기계가 노는 시간에 현재 도착된 작업들 중에서 최대의 가중치를 가지는 작업을 스케줄하는 알고리즘을 생각하고, 이를 *Greedy* 라고 부른다. *Greedy* 는 온라인 알고리즘이고, 이 알고리즘의 경쟁비율이 임의의 온라인 알고리즘의 하한과 일치함을 증명함으로서, *Greedy* 가 최적의 온라인 알고리즘임을 보일 것이다.

분석을 하기 전에, 일반성의 손실 없이 우리는 *Greedy* 알고리즘이 스케줄한 결과가 단 하나의 *바쁜 구간(busy period)*을 이룬다고 가정할 수 있다. 여기서 바쁜 구간이란, 이 구간 동안 기계가 노는 시간 없이 계속 작업을 수행한 시간 구간을 의미한다. 만약 *Greedy* 가 1개 이상의 바쁜 구간 T_i, $i=1,\cdots,k$,를 생성하였다고 가정하자. 우리는 전체 작업들의 집합 $\mathcal{J}$ 를 각각 구간 T_i에 도착한 작업들의 집합인 $\mathcal{J}_i$ 들로 나눈다. 여기서 구간 T_i 들 외의 시간에 작업들은 도착하지 않는다. 왜냐하면, 도착하였다면, *Greedy* 가 그 작업을 스케줄하였고, 구간 T_i 들이 다시 정의되어야 한다. 그러면 임의의 집합 $\mathcal{J}_i$에 대해서 각각 독립적으로 *Greedy* 의 성능을 분석할 수 있다. 다시 말해, 임의의 i에 대해서, $OPT(\mathcal{J}_i)$, $Greedy(\mathcal{J}_i)$ 를 집합 $\mathcal{J}_i$에 속하는 작업들 중 각각 OPT와 *Greedy* 에 의해서 스케줄된 작업들의 집합이라고 하자. 우리가 $||OPT(\mathcal{J}_i)|| \leq c\,||Greedy(\mathcal{J}_i)||$ 임을 증명한다면,

$$||OPT|| \leq \sum_i ||OPT(\mathcal{J}_i)|| \leq c \sum_i Greedy(\mathcal{J}_i) = c\,||Greedy||$$

이므로, 전체 작업 집합에 대한 성능을 증명할 수 있다. 따라서 우리는 *Greedy* 가 하나의 바쁜 구간을 생성한다고 가정하고, 모든 작업들은 이 구간 안에 도착한다고 가정한다.

Greedy 의 성능 분석에 앞서서 다음 보조정리를 통해서 *Greedy* 스케줄의 특징 중 하나를 제시한다. 앞으로 *Greedy* 는 구간 $[0, \ell]$ 동안 작업들을 수행하고, 모두 작업들은 이 구간동안 도착한다. 또한 작업의 최소 수행시간이 1이상이라고 가정하고, 따라서 $\ell \geq 1$. 이때, 만료시간이 ℓ 보다 큰 작업들을 *늦은 작업(late job)*이라고 하고, 만료시간이 ℓ 이하인 작업들을 *이른 작업 (early job)*이라고 한다.

보조정리 4. 모든 늦은 작업들은 *Greedy* 에 의해서 스케줄된다.

증명. L 을 늦은 작업들의 집합이라고 하고, 작업 J 를 L에 속하는 임의의 작업이라고 하자. 작업 J가 구간 $[0, \ell]$ 동안 *Greedy* 의해서 스케줄 되지 않았다고 하면, 시간 ℓ 에 *Greedy* 는 J의 만료시간이 ℓ 보다 크므로 J를 스케줄할 수 있다. 이것은 *Greedy* 가 구간 $[0, \ell]$ 동안 작업들을 수행한다는 가정에 모순된다. □

다음에서 위의 보조정리 4를 이용해서 온라인 알고리즘 *Greedy* 가 최적임을 보인다.

정리 5. 중요도비율 λ, 최대 수행시간 Δ, 인내도 x 에 대해서, $x=0$ 인 경우에 *Greedy* 는 $(\lambda(\Delta+1)+1)$ -competitive이고, $x>0$ 인 경우에, $(\lambda(1+\frac{1}{x})+1)$ -competitive이다.

증명. 우선 이른 작업들에 대한 OPT의 스케줄을 생각한다. 이른 작업들 중 OPT에 의해서 스케줄된 작업들을 O_i, $i=1,\cdots,k$,라고 하고, 이들의 가중치량을 v_i^o, $i=1,\cdots,k$, 수행시간을 p_i^o, $i=1,\cdots,k$ 라고 하자. 이른 작업들에 대한 OPT의 이익은 $v_1^o p_1^o+\cdots+v_k^o p_k^o \leq v_{\max}(p_1^o+\cdots+p_k^o) \leq v_{\max}(\ell+\Delta)$ 을 만족한다. 여기서 $v_{\max}$ 는 최대 가중치량이고, 이른 작업들은 ℓ 이전에 스케줄되어야 하기 때문에 OPT에 의해서 스케줄된 이른 작업들의 수행시간의 합은 $\ell+\Delta$ 을 넘지 않음을 알 수 있다.

Greedy 에 의해서 스케줄된 전체 작업들을 I_j, $j=1,\cdots,h$,라고 하고, 이들의 가중치량을 v_j, $j=1,\cdots,h$, 수행시간을 p_j, $j=1,\cdots,h$, 라고 하자. *Greedy* 의 전체 이익은 $v_1p_1+\cdots+v_hp_h \geq v_{min}(p_1+\cdots+p_h) = v_{min}\ell$ 을 만족한다. 여기서 $v_{\min}$ 는 최소 가중치량을 나타낸다. 따라서 이른 작업들의 집합 E에 대해서, 집합 E에 속한 작업들 중 OPT에 의해서 스케줄된 작업들의 집합을 $OPT(E)$ 라고 하면, 다음 부등식이 만족된다.

$$\frac{||OPT(E)||}{||Greedy||} \leq \frac{v_{\max}}{v_{\min}}(1+\frac{\Delta}{\ell}) \leq \lambda(1+\Delta).$$

결과적으로 위의 보조정리 4에 의해서 *Greedy* 는 늦은 작업들을 모두 스케줄한다. 늦은 작업들의 집합을 L이라고 하면, *Greedy* 는 다음 부등식을 만족함을 알 수 있다.

$$\frac{||OPT||}{||Greedy||} \leq \frac{||OPT(E)||+||L||}{||Greedy||} \leq \lambda(1+\Delta)+1.$$

$x>0$ 인 경우에 우리는 또 하나의 상한을 얻는다. 위에서처럼 이른 작업들 중 *OPT*에 의해서 스케줄된 작업들 O_i, $i=1,\cdots,k$ 를 생각하고, 이중 제일 마지막에 스케줄된 작업 O_k의 수행시간 p_k^o는 $xp_k^o \le \ell$ 를 만족한다. 왜냐하면, 작업 O_k는 이른 작업이므로 작업의 여유시간은 ℓ 이하이다. 따라서 이른 작업들에 대한 *OPT*의 이익은 $v_1^o p_1^o + \cdots + v_{k-1}^o p_{k-1}^o + v_k^o p_k^o \le v_{\max}(\ell + \frac{\ell}{x})$ 을 만족하고, 위에서처럼 *Greedy* 전체 이익의 하한과 보조정리 4로부터 *Greedy*는 $(\lambda(1+\frac{1}{x})+1)$ -competitive 이다. □

3. 동일 수행시간 작업 스케줄링

본 절에서 작업들의 수행시간이 모두 동일한 경우를 생각한다. 우리는 동일 수행시간의 길이가 1이라고 가정한다. 그러면 모든 작업들은 길이가 1인 구간동안에 수행된다. 다음에서 우리는 길이가 1인 구간들이 겹치는 (overlap) 관계를 나타낼 것이다. 어떠한 구간 $[a, a+1]$ 에 대하여 구간 $[x, x+1]$와 $[y, y+1]$을 생각한다. $x < a \le x+1$ 만족하면 구간 $[a, a+1]$는 구간 $[x, x+1]$과 *f-overlap* 이라고 하고, $a \le y < a+1$ 만족하면 구간 $[a, a+1]$는 구간 $[y, y+1]$과 *b-overlap* 이라고 한다.

다음에서 인내도 x에 대한 두 가지 경우 $0 \le x < 1$과 $x \ge 1$를 나누어 생각한다. $x \ge 1$의 경우에 *Greedy*는 상수의 경쟁비율을 가진다.

3.1 $0 \le x < 1$ 경우

정리 6. 중요도비율 λ에 대해서, *Greedy*는 $(\lambda+1)$ -competitive 이다.

증명. 이른 작업들은 자신의 마감시한 안에 수행되기 위해서는 $[0, \ell]$ 안에 스케줄되어져야 한다. 이른 작업들 중 *OPT*에 의해서 스케줄된 작업들을 O_i, $i=1,\cdots,k$, 라고 하고, 이 작업들 각각은 어떠한 구간 $[a_i, a_i+1]$ 동안 수행된다. 그러면 각각의 작업 O_i 에 대해서, 구간 $[a_i, a_i+1]$과 f-overlap하는 구간 $[j, j+1]$가 반드시 하나 존재하고, 이 구간동안 *Greedy*가 수행한 작업 I_i 가 존재한다. 작업 O_i의 가중치 w_i^o와 작업 I_i의 가중치 w_i의 비는 $\frac{w_i^o}{w_i} \le \lambda$을 만족하고, 보조정리 4에 의해서 모든 늦은 작업들은 *Greedy*에 의해서 스케줄되므로, *Greedy*의 경쟁비율은 $\lambda+1$ 보다 클 수 없다. □

3.2 $x \ge 1$ 경우

정리 7. *Greedy*는 *3*-competitive 이다.

증명. E를 이른작업들의 집합이라고 하자. 우리는 E에 속하는 작업들에 대한 *OPT*의 스케줄을 생각한다. 특별히 우리는 E의 작업들 중에서 *Greedy*에 위해서 거절되어진 작업들의 집합 $\hat{E}$ 만을 생각해도 충분하다. 그러면 우리는 집합 $\hat{E}$에서 집합 *Greedy*로의 함수 m을 정의한다. 집합 $\hat{E}$에 속하는 작업들은 시간 ℓ 안에 스케줄되어야 하고, 각각의 작업들은 *Greedy*에 의해 스케줄된 작업들과 겹치는 관계를 이룬다. 작업 J가 집합 $\hat{E}$의 임의의 원소이고, 작업 J의 *OPT*에 의한 실행구간을 $[a, a+1]$이라고 하자. 그러면 *Greedy*의 스케줄에서 구간 $[a, a+1]$과 각각 f-overlap, b-overlap 하는 구간 $[x, x+1]$, $[y, y+1]$이 존재한다. 사실, $x+1=y$. 구간 $[x, x+1]$, $[y, y+1]$에서 *Greedy*에 의해서 수행된 작업들을 각각 J_1, J_2라고 한다면, 작업 J의 도착시간 a에 따라서 두 가지 경우가 존재한다. $a \le x$ 라고 하면, 시간 x에서 *Greedy*는 작업 J를 고려했고, 작업 J_1을 선택했으므로 J_1의 가중치가 J의 가중치 보다 큼을 알 수 있다. 그러면 함수 m에 의한 작업 J의 함수값 $m(J)$를 작업 J_1으로 정의한다. $a > x$라고 하면, $x \ge 1$ 이므로 시간 y에서 *Greedy*는 작업 J를 고려했고, 작업 J_2를 선택했으므로 J_2의 가중치가 J의 가중치보다 크고, 함수 m에 의한 작업 J의 함수값은 작업 J_2로 정의한다. 그러면 어떤 경우에도 집합 $\hat{E}$에 속하는 임의의 작업 J의 함수값 $m(J)$의 가중치는 J의 가중치보다 큼을 알 수 있다. 그런데 우리는 *Greedy*에 의해서 스케줄된 임의의 작업 I에 대해서 많아야 두 개의 작업이 함수값으로 I를 가짐을 알 수 있다. 다시 말해서, 많아야 두개의 작업 I_1, I_2가 존재해서, $m(I_1)=I=m(I_2)$. 여기서 두 개의 작업 I_1, I_2가 모두 존재했다면, I_1, I_2는 I와 각각 b-overlap, f-overlap한다.

결과적으로 위의 함수 m과 보조정리 4로부터 다음을 얻을 수 있다.

$$\begin{aligned} ||OPT|| &\le ||\hat{E}|| + ||E \cap Greedy|| + ||L|| \\ &\le ||\{m(J) | J \in \hat{E}\}|| + ||Greedy|| \\ &\le 2||Greedy|| + ||Greedy|| = 3||Greedy||. \end{aligned}$$ □

3.3 자원증대(Resource augmentation)

여기에서는 *Greedy*가 보다 빠른 기계를 가지는 경우의 성능에 관해서 알아본다. 기계의 속도가 커질수록 경쟁비율이 *1*에 접근함을 보인다.

정리 8. $\varepsilon \ge 1$ 일 때, $(1+\varepsilon)$ -speed 기계를 사용하는 *Greedy*는 $(1+\frac{1}{\varepsilon})$ -competitive이다.

증명. *Greedy*가 구간 $[0, \ell]$ 동안 작업들을 수행했다면, 각 작업은 어떠한 구간 $[x, x+\frac{1}{1+\varepsilon}]$ 동안 수행된다. 정리 7에서처럼 이른작업들 중에서 OPT에 의해서 수행되지만, *Greedy*에 의해서 거절된 작업들의 집합 $\hat{E}$를 생각한다. 임의의 작업 $J \in \hat{E}$에 대해서, OPT는 J를 구간 $[a, a+1]$에 수행한다고 하자. 그러면 어떠한 연속된 구간 $[y, y+\frac{1}{1+\varepsilon}], \cdots, [y+\frac{\varepsilon}{1+\varepsilon}, y+1]$이 존재해서, *Greedy*는 각 구간에서 작업을 수행하고 $y < a \leq y+\frac{1}{1+\varepsilon}$. 여기서 우리는 편의상 $\varepsilon \geq 1$ 이 정수라고 가정한다. 정수가 아닌 경우에도 비슷하게 증명할 수 있다. 우리는 작업 J의 *Greedy* 스케줄에서의 만료시간이 $a+\frac{\varepsilon}{1+\varepsilon}$ 이상임을 알 수 있다. 왜냐하면, 작업 J의 마감시한은 $a+1$ 이상이고, J가 시간 a에 도착되었다 하더라도 *Greedy*는 시간 $a+\frac{\varepsilon}{1+\varepsilon}$ 에 스케줄해도 마감시한 안에 마칠 수 있다. 그런데, $y+\frac{\varepsilon}{1+\varepsilon} \leq a+\frac{\varepsilon}{1+\varepsilon}$. 작업 J는 *Greedy*에 의해 거절되었으므로, 연속된 시간 $y+\frac{1}{1+\varepsilon}, \cdots, y+\frac{\varepsilon}{1+\varepsilon}$ 에 *Greedy*에 의해서 고려되었고, 각각의 시간에 스케줄된 작업들보다 가중치가 적음을 알 수 있다. 결과적으로 우리는 $\|\hat{E}\| \leq \frac{1}{\varepsilon}\|Greedy\|$임을 보였다. 따라서 보조정리 4로부터 결론을 얻을 수 있다. □

4. 결 론

본 논문에서는 마감시한을 가진 작업들의 온라인 스케줄링을 다루었다. 특히 각각의 작업들은 자신만의 가중치를 가지고, 스케줄링 알고리즘은 가중치의 합이 최대가 되도록 스케줄할 작업들을 결정한다. 이전 연구들은 각 작업의 가중치가 작업의 수행시간으로 주어지는 경우에 많이 이루어졌다[1,2,5,7,8,13].

임의의 가중치를 가진 작업들에 대해서 선점 스케줄링의 경우에 이전 연구들이 있었다[9,10 12]. 하지만 비선점 스케줄링의 경우에는 알려진 결과가 없었다. 본 논문에서는 이 비선점 스케줄링의 경우를 다루었다. 임의의 온라인 알고리즘이 가질 수 있는 경쟁비율의 하한을 주었고, *Greedy*가 하한을 성취하는 최적의 온라인 알고리즘임을 증명하였다. 또한 작업들의 수행시간이 동일한 경우에 *Greedy*가 상수 경쟁비율을 가질 수 있음을 보였다 ($x \geq 1$ 경우).

본 논문은 하나의 기계에 대한 스케줄링을 다루었다. 다음 연구로서 여러 기계가 있는 경우의 스케줄링을 제안한다. 선점 스케줄링의 경우에는 m개의 기계에 대해서, $1+m(\lambda^{1/\phi}-1)$ -competitive, 여기서 $\phi = m\ln\lambda/2(\ln\lambda+1)$, 알고리즘이 존재함이 증명되었다[11].

참 고 문 헌

[1] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha. "On-line scheduling in the presence of overload," In Proc. of IEEE Foundations of Computer Science, 101-110, 1991.

[2] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wand. "On the competitiveness of on-line task real-time task scheduling," Journal of Real-Time Systems, 4(2):124-144, 1992.

[3] J. Garay, J. Naor, B. Yener, and P. Zhao. "On-line admission control and packet scheduling with interleaving," In Proc. of IEEE INFOCOM, 2002.

[4] S. Goldman, J. Parwatikar, and S. Suri. "On-line scheduling with hard deadlines," In Proc. of the Workshop on Algorithms and Data Structures, 258-271, 1997.

[5] M.H. Goldwasser. "Patience is a virtue; The effect of slack on competitiveness for admission control," In Proc. of 10th ACM Sym. of Discrete Algorithms, 396-405, 1999.

[6] R. Lipton and A. Tomkins. "Online interval scheduling," In Proc. of 5th ACM Sym. of Discrete Algorithms, 302-311, 1994.

[7] M.R. Garay and D. S. Johnson. "Computers and Intractability, A Guide to the Theory of NP-Completeness," Freeman, 1979.

[8] G. Koren and D. Shasha. "Dover : An optimal on-line scheduling algorithm for overloaded real-time systems," SIAM Journal of Computing, 24(2):318-339, 1995.

[9] B. Kalyanasundaram and K.R. Pruhs. "Speed is as powerful as clairvoyance," J. of ACM, 47(4):617-643, 2000.

[10] T. Lam and K. To. "Performance guarantee for online deadline scheduling in the presence of overload," In Proc. of 12th ACM Sym. of Discrete Algorithms, 755-764, 2001.

[11] M. Brehob, E. Torng, and P. Uthaisombut. "Applying extra-resource analysis to load balancing," In Proc. of 11th ACM Sym. of Discrete Algorithms, 560-561, 2000.

[12] L. Epstein and R. van Stee. "Minimizing the maximum starting time on-line," In Proc. of the 10th European Symposium on Algorithms, 449-460, 2002.

[13] C.A. Phillips, C. Stein, E. Torng, and J. Wein. "Optimal time-critical scheduling via resource augmentation," In Proc. of 29th ACM Sym. on Theory of Computing, 140-149, 1997.

[14] G. Koren, D. Shasha, and S. C. Huang. "MOCA : A multiprocessor on-line competitive algorithm for real-time system scheduling," In Proc. of 14th Real-Time Systems Symposium, 172-181, 1993.

김 재 훈

1990년 3월~1994년 2월 서강대학교 수학과(학사). 1994년 3월~1996년 2월 KAIST 수학과(석사). 1996년 9월~2003년 2월 KAIST 전산학과(박사). 2003년 3월~현재 부산외국어대학교 컴퓨터공학부 조교수. 관심분야는 알고리즘 및 스케줄링, 암호학

장 정 환

1979년 3월~1983년 2월 경북대학교 전자공학과(학사). 1983년 3월~1985년 2월 KAIST 전산학과(석사). 1993년 3월~1998년 8월 KAIST 전산학과(박사). 1985년 4월~2000년 8월 KT 선임연구원. 2000년 9월~현재 부산외국어대학교 디지털정보공학부 조교수. 관심분야는 상호연결망 및 그래프이론 응용, 통신망보안