한국경영과학회지
제38권 제2호
2013년 6월
http://dx.doi.org/10.7737/JKORMS.2013.38.2.141

자원 제약이 없는 환경에서 부분 우선순위를 고려한 Earliness-Tardiness 최적 일정계획 알고리즘*

하 병 현**†

An Algorithm for Resource-Unconstrained Earliness-Tardiness Problem with Partial Precedences

Byung-Hyun Ha**

Abstract

In this paper, we consider the minimization of the total weighted earliness-tardiness penalty of jobs, regarding the partial precedences between jobs. We present an optimal scheduling algorithm in $O(n(n+m\log m))$ where n is the number of jobs and m is the number of partial precedences. In the algorithm, the optimal schedule is constructed iteratively by considering each group of contiguous jobs as a block that is represented by a tree.

Keyword : Scheduling, Earliness-Tardiness Penalty, Partial Precedences, Optimization Algorithm

1. 서 론

본 연구는 일부 작업들에 대한 처리 순서가 주어졌을 때 작업의 납기 시간과 완료 시간의 차이에 대한 가중합을 최소로 하는 일정계획 문제를 대상으로 한다. 가용 자원에는 한계가 없으며, 따라서 처리 순서 제약이 없는 작업들은 개수에 제한 없이 동시에 가공될 수 있다.

이 문제는 earliness-tardiness(E-T) 비용이 고려되는 JIT(just-in-time) 환경에서 프로젝트 일정

논문접수일 : 2013년 05월 06일 논문게재확정일 : 2013년 05월 09일
논문수정일(1차 : 2013년 05월 09일)
* 이 논문은 2013년 정부(교육부)의 지원을 받아 수행된 연구임(산학협력 선도대학 (PNU-LINC) 육성사업).
** 부산대학교 산업공학과
† 교신저자, bhha@pusan.ac.kr

계획 또는 부분 우선순위(partial precedence)를 가진 작업들에 대한 병렬 기계(parallel machines) 일정계획 문제로 파악할 수 있다. 만일 가용 자원 또는 기계 대수에 제한이 있다면 일반적으로 NP-hard임이 알려져 있다[3, 8]. 하지만 본 연구에서와 같이 자원 제약을 고려하지 않을 경우 선형계획 문제로 형식화된다. 따라서 다항 시간 알고리즘이 존재하는 것을 알 수 있다.

본 논문에서는 이 문제에 특화된 다항 시간(polynomial-time) 알고리즘을 제시한다. 제시된 알고리즘은 본문제(main problem)를 해결하기 위한 부분문제(subproblem)의 최적해를 구하는데 효과적으로 이용될 수 있다. 예를 들어, E-T 일정계획 문제[4]를 해결하는 어떤 branch-and-bound 알고리즘에서, 만일 중간 노드(internal node)에 대응하는 부분 계획이 작업들 간의 부분 우선순위로 주어진다면, 본 연구에서 제안된 알고리즘을 사용하여 하한(lower bound)을 효율적으로 구할 수 있다.

이와 같은 부분문제 해결을 목적으로, 작업 순서가 주어졌을 때(즉, 모든 작업들 간의 우선순위가 완전히 정해졌을 때) 최적 일정을 도출하는 특화된 알고리즘이 존재한다[2, 6]. 하지만 본 연구가 대상으로 하는 부분 우선순위를 고려하는 연구는 찾아보기 어렵다. 따라서 일정 자체가 부분 우선순위의 계획을 필요로 하는 프로젝트, 병렬 기계, 잡숍(job shop) 등의 문제[1, 5]와, 배치계획 등 다차원을 고려하는 문제에서 효과적으로 사용될 수 있다. 예를 들어, 이 문제는 시설 배치[10] 또는 컨테이너 터미널의 선석계획(berth planning)[7]에서 해를 평가하기 위해 사용되었으나, 특화된 알고리즘을 사용하는 대신 네트워크 흐름 문제 또는 발견적 기법으로 접근하였다.

본 논문의 나머지 부분의 구성은 다음과 같다. 제 2장에서 문제를 정의하고 알고리즘 개발을 위한 주요 개념을 제시한다. 제 3장에서 주어진 일정의 최적 여부를 판단하는 방법을 개발한다. 제 4장에서 최적 일정을 수립하는 알고리즘을 제안하고, 알고리즘의 정확성과 시간 복잡도를 논한다. 마지막으로 제 5장에서 결론을 제시한다.

2. 문제 정의

계획 대상 *작업*(job)의 집합은 $J=\{1, \cdots, n\}$로 주어진다. 작업 $i\in J$에 필요한 *작업 시간*(processing time)은 p_i이며, 납기 시간(due date)을 준수하기 위한 *목표 시작 시간*(target starting time)은 s_i이다. 즉, 납기 시간은 (s_i+p_i)이다. 작업은 비선점형(nonpreemptive)으로 처리된다. 작업 i는 *가중치*(weight) $w_i(>0)$를 가지며, 작업의 *시작 시간*(start time) x_i가 정해지면, 그 작업에 대한 비용은 $w_i|x_i-s_i|$로 주어진다. 일정(schedule)은 $\sigma=(x_1, \cdots, x_n)$로 표현한다.

작업들 간의 처리 순서는 *직접 우선순위*(direct precedence)의 집합 A로 주어진다. 각 $(i, j)\in A$에 대하여, 모든 가능 일정 $\sigma=(x_1, \cdots, x_n)$는 $x_i+p_i \le x_j$를 만족해야 한다. 작업 i를 j의 *직접 선임*(direct predecessor)이라 하고, 작업 j를 i의 *직접 후임*(direct successor)이라 한다.

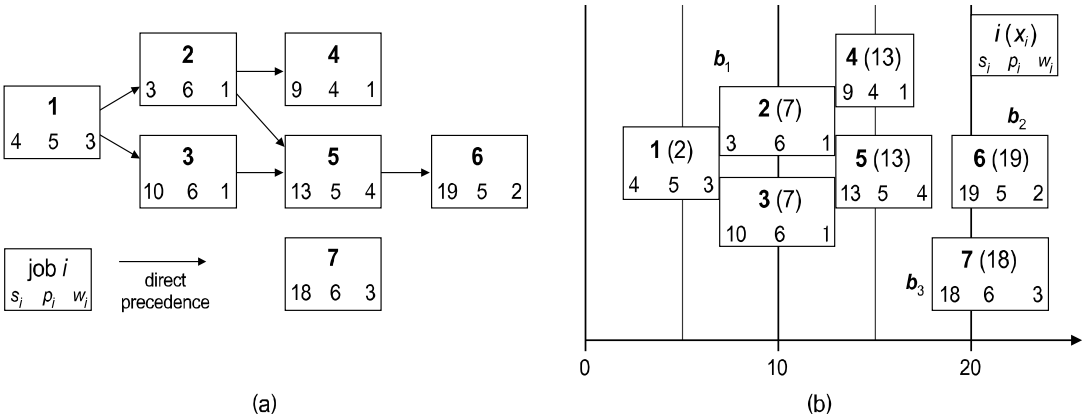
일정계획의 목적은 모든 작업들에 대한 총비용의 최소화이다. 구체적으로 수리계획 모형으로 나타내면 다음과 같다:

$$\text{min.} \quad \sum_{i\in J} w_i|x_i-s_i| \tag{1}$$

$$s.t. \quad x_i+p_i \le x_j \;\forall (i, j)\in A \tag{2}$$

$$x_i \ge 0 \qquad \forall i \in J \tag{3}$$

어떤 일정 $\sigma=(x_1, \cdots, x_n)$가 주어졌을 때, 완료시간과 시작 시간이 동일한 직접 우선순위 관계의 두 작업을 *인접 작업*(contiguous jobs)이라 하자. 즉, $(i, j) \in A$이고 $x_i+s_i=x_j$이면 i와 j는 인접 작업이다. 본 연구에서는 최적 일정을 도출하기 위해 인접 관계를 통해 *연결된*(connected) 작업들로 구성되는 *블록*(block)을 고려한다. 연결된 모든 작업들은 하나의 블록을 구성하며, 그렇지 않은 작업들은 서로 다른 블록에 속한다. 다른 어떤 작업과도 인접하는 않는 작업은 그 작업만으로 블록을 이룬다.



[그림 1] An Example of 7-Job Problem, (a) Job Descriptors, (b) The Optimal Schedule.

어떤 블록 b를 구성하는 모든 작업들의 집합을 $\mathcal{J}(b)$로 정의한다.

주어진 일정 $\sigma=(x_1, \cdots, x_n)$로 정해진 어떤 블록 b를 고려하자. 만일 다른 어떤 가능 일정 $\sigma'=(x'_1, \cdots, x_n')$에 대해서도 $\sum_{i\in\mathcal{J}(b)} w_i|x_i-s_i| \le \sum_{i\in\mathcal{J}(b)} w_i|x_i'-s_i|$라면, 그 블록을 *B-최적*(B-optimal)이라 하자.

보조정리 1 : 어떤 일정으로 정해진 모든 블록이 B-최적이면, 그 일정은 최적이다.

증명. 다음과 같은 어떤 일정 $\sigma=(x_1, \cdots, x_n)$가 존재한다고 하자. 그 일정으로 정해진 모든 블록 $b_1, \cdots, b_m$은 B-최적이지만 그 일정은 최적이 아니다. 일정 σ가 최적이 아니므로 더 작은 총비용을 가지는 어떤 다른 일정 $\sigma'=(x'_1,\cdots, x'_n)$이 존재한다. 즉,

$$\sum_{k=1}^{m}\sum_{i\in\mathcal{J}(b_k)} w_i|x'_i-s_i| < \sum_{k=1}^{m}\sum_{i\in\mathcal{J}(b_k)} w_i|x_i-s_i| \quad (4)$$

부등호가 성립하기 위해서는 적어도 어떤 하나의 k에 대하여 $\sum_{i\in\mathcal{J}(b_k)} w_i|x'_i-s_i| < \sum_{i\in\mathcal{J}(b_k)} w_i|x_i-s_i|$가 만족한다. 이는 곧 블록 b_k가 B-최적이 아니라는 것을 의미하며, 모순이다. □

[그림 1]의 (a)는 일정계획 문제의 예를 보여준다. 직접 우선순위는 화살표로 표시되어 있다. 즉, $A=\{(1, 2), (1, 3), (2, 4), (2, 5), (3, 5), (5, 6)\}$이다.

총비용이 17인 하나의 가능 일정 $(x_1, \cdots, x_7)=(2, 7, 7, 13, 13, 19, 18)$이 (b)에 나타나있다. 주어진 일정으로 세 블록 b_1, b_2, b_3가 정해진다. 블록 b_1을 구성하는 작업들의 집합은 $\mathcal{J}(b_1)=\{1, 2, 3, 4, 5\}$이다. 블록 b_1이 B-최적인 것은 다음 절에서 확인할 수 있으며, 나머지 두 블록이 B-최적인 것은 자명하다. 따라서 주어진 일정은 최적이다.

본 연구에서는 가상 작업 0의 존재를 가정한다. 작업 0은 $p_0=0$, $s_0=0$, $w_0=M$이며(M은 충분히 큰 수), 모든 주어진 작업들보다 우선한다. 즉, 주어진 작업들 중 직접 선임이 없는 모든 작업들에 대해 작업 0을 직접 선임으로 지정한다. 모든 최적 일정에서 가상 작업의 시작 시간이 0임은 자명하므로, 이와 같은 가상 작업을 도입하여도 최적 일정과 최소 총비용은 바뀌지 않는다.

3. 최적 조건

일정 $\sigma=(x_1, \cdots, x_n)$가 주어졌을 때, 어떤 작업 i에 대하여 $x_i=s_i$이면 i를 *정시 작업*(on-time job)이라 하자. 가상 작업 0을 가정하면, 아래의 성질이 존재한다. 따라서 고려 대상 일정을 한정할 수 있다.

보조정리 2 : 정해지는 각 블록이 모두 정시 작업을 가지는 최적 일정이 존재한다.

증명. 그렇지 않다고 하자. 어떤 최적 일정 σ를 고려하면, 그 일정으로 정해지는 정시 작업이 없는 어떤 블록 b가 존재한다. $E=\{i\in J(b)\mid x_i<s_i\}$이고, $L=\{i\in J(b)\mid x_i>s_i\}$이라 하자. 그러면 $\sum_{i\in E}w_i=\sum_{i\in L}w_i$가 성립한다. 아니라면 $J(b)$에 속한 작업들의 시작 시간을 모두 동일하게 변화시켜 더 나은 일정을 얻을 수 있기 때문이다.

총비용을 증가시키지 않고 동시에 $J(b)$에 속하지 않는 작업들의 시작 시간을 바꾸지 않는 범위에서, $J(b)$에 속한 작업들의 시작 시간을 모두 동일하게 변화시키자. 그러면, i) 어떤 작업이 정시 작업이 되거나, ii) 블록 b와 다른 블록이 하나로 합쳐지거나, iii) 두 경우가 모두 발생한다. 모든 경우 총비용이 동일하면서 σ에 비해 정시 작업이 없는 블록이 하나 이상 줄어든 새로운 일정을 얻을 수 있다.

동일한 방식으로 반복해 수정해나가면, 총비용이 σ와 동일하지만 모든 블록들이 정시 작업을 가지는 일정 σ'을 항상 생성할 수 있다. 따라서 모순이다. □

본 연구에서는 블록을 루트를 가진 트리(rooted tree)로 표현한다. 트리와 관련된 정의는 West[9]를 따른다. 즉, 트리에서 두 작업 i와 j를 연결하는 유일한 *경로*(path)를 i, j-*경로*라 한다. 루트를 f라 할 때, 작업 i의 *부모*(parent)는 i, j-경로 상에 있는 i의 인접 작업이며, 부모가 아닌 인접 작업을 *자식*(child)이라 한다. 어떤 작업 i에 대하여 i, f-경로가 작업 j를 포함하면, 작업 i를 j의 *자손*(descendant)이라 한다. 참고로 정의에 의하여 모든 작업은 자기의 자손이다.

어떤 일정 $\sigma=(x_1, \cdots, x_n)$로 정해지고 정확히 하나의 정시 작업이 있는 특정 블록 b가 주어졌다고 하자. 정시 작업을 f라 하자. $J(b)$에 속한 각 작업을 노드로 하고 인접 작업들을 아크로 이은 그래프(graph)를 고려하자. 이 그래프에서 도출되고 f를 루트로 하는 어떤 하나의 신장 트리(spanning tree)를 T라 할 때, T를 바탕으로 각 작업 $i\in J(b)$의 *변경 비용* $\bar{c}_i$를 다음과 같이 정의 한다.

$$\bar{c}_i=\sum_{j\in T(i)}\hat{w}_j \tag{5}$$

여기서 $T(i)$는 T에서 작업 i의 모든 자손들의 집합이며, $\hat{w}_j$는 일정 σ에 따른 작업 j의 *수정 가중치*로 다음과 같이 주어진다.

$$\hat{w}_j=\begin{cases} w_j & \text{if } x_j\le s_j \\ -w_j & \text{otherwise}\end{cases} \tag{6}$$

보조정리 3 : 만일 모든 작업 $i\in J(b)$가 다음의 *최적 조건*을 만족시키면 블록 b는 B-최적이다.

$$\begin{cases} 0\le\bar{c}_i\le 2w_i & \text{if } i=f & (7a)\\ \bar{c}_i\ge 0 & \text{if}(i,\pi(i))\in A & (7b)\\ \bar{c}_i\le 0 & \text{otherwise}((\pi(i),i)\in A) & (7c)\end{cases}$$

여기서 $\pi(i)$는 T에서 i의 부모이다.

증명. $A_b=\{(i,\ j)\in A\mid i,\ j\in J(b)\}$라 하면, $J(b)$에 속한 작업들만 고려하여 최적 일정을 수립하는 선형계획 모형은 아래와 같다. 이 모형은 제 2장에서 제시된 모형을 보조 변수 y_i^+와 y_i^-, e_{ij}를 도입하여 표준형(standard form)으로 변환한 것이다.

$$\min.\ \sum_{i\in J(b)}w_i(y_i^++y_i^-) \tag{8}$$

$$\text{s.t. } -x_i+x_j-e_{ij}=p_i\ \forall\ (i,\ j)\in A_b \tag{9}$$

$$x_i-y_i^++y_i^-=s_i\ \forall i\in J(b) \tag{10}$$

$$x_i,\ y_i^+,\ y_i^-,\ e_{ij}\ge 0 \tag{11}$$

$E=\{i\in J(b)\mid x_i<s_i\}$이고 $L=\{i\in J(b)\mid x_i>s_i\}$라 하자. 참고로 정시 작업은 f 하나이므로 $J(b)=E\cup L\cup\{f\}$이다. A_b'를 신장 트리 T의 아크에 대응하는 모든 직접

우선순위의 집합이라 하자. 즉, $|A_b'| = |J(b)| - 1$. 다음을 모든 기저 변수(basic variable)들의 집합이라 하면,

$$\{x_i \mid i \in J(b)\} \cup \{y_i^+ \mid i \in L\} \cup \{y_i^- \mid i \in E\} \cup \{e_{ij} \mid (i,\ j) \in A_b \setminus A_b'\} \quad (12)$$

주어진 시작 시간 x_i와 제약식 (9)~(11)로 결정된 다른 변수들의 값은 기저가능해(basic feasible solution)를 이룬다. 아래에서 최적 조건 (7)을 만족하면 위의 가능해가 최적임을 보인다.

기저 행렬(basis matrix)과 비기저 행렬을 각각 $\mathbf{B}$와 $\mathbf{N}$이라 하고, 기저 변수와 비기저 변수에 대응하는 비용 벡터(cost vector)를 각각 c_B와 c_N이라 하자. 각 $(i,\ j) \in A_b$에 대하여 제약식 (9)에 대응하는 쌍대 변수(dual variable)를 α_{ij}라고 하고, 각 $i \in J(b)$에 대하여 제약식 (10)에 대응하는 쌍대 변수를 β_i라 하자. 쌍대 변수의 벡터를 μ라 하면 다음을 만족한다.

$$\mu \mathbf{B} = c_B \quad (13)$$

위의 기저가능해가 다음의 쌍대 가능 조건을 만족하면, 그 해는 최적이다.

$$c_N - c_B \mathbf{B}^{-1}\mathbf{N} = c_N - \mu \mathbf{N} \geq 0, \quad \text{i.e.,} \quad \mu \mathbf{N} \leq c_N \quad (14)$$

식 (13)으로부터 쌍대해에 대한 다음의 연립방정식을 얻을 수 있다.

$$\sum_{j:(j,i) \in A_b} \alpha_{ji} - \sum_{j:(i,j) \in A_b} \alpha_{ij} + \beta_i = 0 \quad \forall i \in J(b) \quad (15)$$

$$\alpha_{ij} = 0 \quad \forall (i,\ j) \in A_b \setminus A_b' \quad (16)$$

$$\beta_i = \hat{w}_i \quad \forall i \in E \cup L \quad (17)$$

그리고 쌍대 최적 조건인 식 (14)는 다음과 같이 구체화 된다.

$$\alpha_{ij} \geq 0 \quad \forall (i,\ j) \in A_b' \quad (18)$$

$$\beta_i \leq w_i \quad \forall i \in E \cup \{f\} \quad (19)$$

$$\beta_i \geq -w_i \quad \forall i \in L \cup \{f\} \quad (20)$$

식 (16)과 식 (17)에 의해 결정되지 않는 α_{ij}와 β_i는 식 (15)로 구해진다. 식 (15)는, 식 (16)에 의해 A_b 대신 A_b'에 대해서만 고려하면 충분하므로, $i \in J(b)$에 대하여 다음과 같이 다시 쓸 수 있다.

$$\begin{cases} \beta_i = -\sum_{j:(j,i) \in A_b} \alpha_{ji} + \sum_{j:(i,j) \in A_b} \alpha_{ij} & \text{if } i = f \\ \alpha_{i\pi(i)} = \sum_{j:(j,i) \in A_b} \alpha_{ji} - \sum_{\substack{j:(i,j) \in A_b \\ j \neq \pi(i)}} \alpha_{ij} + \hat{w}_i & \text{if } (i,\ \pi(i)) \in A \\ \alpha_{\pi(i)i} = -\sum_{\substack{j:(j,i) \in A_b \\ j \neq \pi(i)}} \alpha_{ji} + \sum_{j:(i,j) \in A_b} \alpha_{ij} - \hat{w}_i & \text{otherwise} \end{cases} \quad (21)$$

트리 T에서 $C(i)$를 i의 모든 자식들의 집합이라 하면, 변경 비용의 정의인 식 (5)는 다음과 같이 쓸 수 있으므로,

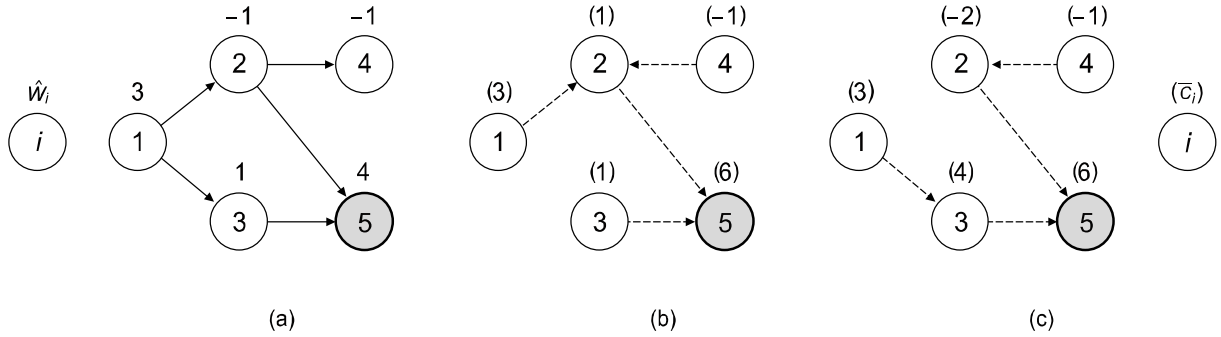
$$\bar{c}_i = \sum_{j \in T(i)} \hat{w}_j = \sum_{j \in C(i)} \bar{c}_j + \hat{w}_i \quad (22)$$

작업 $i \in J(b)$에 대하여 아래에 주어진 관계 식 (23)을 사용하면, 변경 비용 정의와 식 (15)는 동치임을 알 수 있다.

$$\bar{c}_i = \begin{cases} w_i - \beta_i & \text{if } i = f \\ \alpha_{i\pi(i)} & \text{if } (i,\ \pi(i)) \in A \\ -\alpha_{\pi(i)i} & \text{otherwise} \end{cases} \quad (23)$$

각 $i \in E \cup L$에 대하여 식 (17)에 의해 결정된 β_i는 위의 쌍대 최적 조건 식 (19)와 식 (20)을 항상 만족함을 알 수 있다. 따라서 $(i,\ j) \in A_b'$에 대하여 α_{ij}가 조건 식 (18)을 만족하고, β_f가 조건 식 (19)와 식 (20)을 만족하면 주어진 기저가능해는 최적이다. 이는 최적 조건 식 (7)과 동일하다. □

어떤 작업의 변경 비용은 자손들의 시작 시간을 모두 한 단위씩 앞당기는 경우 발생하는 비용 증가분의 합으로 정의된다. 따라서 최적 조건 식 (7)이 의미하는 바는 다음과 같다. i) 주어진 블록에 속한 모든 작업들의 시작 시간을 동일하게 앞당기거나



[그림 2] Spanning Trees Corresponding to Block b_1, (a) Graph of the Block, (b) A Spanning Tree Showing the B-Optimality of the Block, (c) Another Spanning Tree

늦출 때 총비용이 줄어들지 않으며, ii) 정시 작업이 아닌 모든 작업에 대하여, 트리에서 부모가 직접 후임(직접 선임)이면, 그 작업의 모든 자손들의 시작 시간을 동일하게 앞당길(늦출) 때 총비용이 줄어들지 않는다.

[그림 2]는 [그림 1]의 일정으로 정해진 블록 b_1에 대한 트리를 보여준다. (a)는 b_1을 그래프로 표현한 것이다. 화살표의 방향은 우선순위를 의미하며, 각 작업에 수정 가중치가 표시되어 있다. (b)는 정시 작업 5를 루트로 하는 하나의 신장 트리를 보여준다. 화살표는 자식에서 부모로 향하며, 각 작업에 변경 비용이 표시되어 있다. 예를 들어 작업 2의 모든 자손들의 집합은 $T(2)=\{1, 2, 4\}$이며, 변경 비용은 $\bar{c}_2=\hat{w}_1+\hat{w}_2+\hat{w}_4=1$이다. 부모가 $\pi(2)=5$이고, $(2,5)\in A$이므로, 최적 조건 (7b)를 만족함을 알 수 있다. 다른 작업들도 모두 최적 조건을 만족하여 b_1은 B-최적이다. (c)는 같은 블록에 대한 다른 가능한 신장 트리를 보여준다. 작업 2의 변경 비용 $\bar{c}_2=-2<0$으로 최적 조건을 위배함을 알 수 있다. 이 예는 보조정리 3의 최적 조건이 B-최적을 위한 필요 조건이 아님을 보여준다.

4. 최적 일정 수립

본 절에서는 먼저 다음의 *유일 정시 작업 가정* 하에 최적 일정을 도출하는 알고리즘을 제시한다. *알고리즘 수행 도중 어떤 블록도 둘 이상의 정시 작업을 가지는 경우가 발생하지 않는다.* 그리고 제시한 알고리즘의 정확성과 시간 복잡도를 논한다. 제 4.2절의 마지막에서 위의 가정을 해소하는 방안을 제시한다.

4.1 알고리즘

일정계획 대상 작업 집합은 $J=\{1, \cdots, n\}$이다. 편의상 작업 번호는 우선순위를 거스르지 않는다고 한다. 즉, 모든 $(i, j)\in A$에 대하여 $i<j$이다. 알고리즘에서 블록은 트리로 표현되며, 사용되는 기호는 다음과 같다.

$\rho(i)$: 작업 i가 속한 트리의 루트 작업
$C(i)$: 작업 i의 모든 자식들의 집합
$\pi(i)$: 작업 i의 부모
$T(i)$: 작업 i의 모든 자손들의 집합
$P(i, j)$: i, j-경로 상의 모든 작업들의 집합

그리고 $Q(i, j)=\{k\in P(i, j)\,|\,k=\rho(i) \text{ or } (k, \pi(k))\in A\}$이다.

알고리즘은 먼저 모든 작업 i에 대해 $C(i)\leftarrow\varnothing$으로 초기화하고, 가상 작업 0으로 블록을 만든다. 그리고 1부터 n까지의 단계(stage)를 거친다. 각 단계 i에서는 이전 단계에서 도출된 작업 $(i-1)$까지에 대한 최적 일정에 작업 i를 추가하여 새로운 최적 일정을 수립한다. 단계 i의 절차는 다음과 같다.

Step 1. $x_i \leftarrow \max(s_i, \max_{j:(j,i)\in A}\{x_j + p_j\})$.
If $x_i = s_i$ then $\rho(i) \leftarrow i$ and stop.

Step 2. Let j be the job such that $(j, i)\in A$ and $x_j + p_j = x_i$ (break tie arbitrarily).
$\pi(i) \leftarrow j$, $C(j) \leftarrow C(j) \cup \{i\}$, $\rho(i) \leftarrow \rho(j)$.

Step 3. $\bar{c}_{\min} \leftarrow \min_{k\in Q(i,\rho(i))}\{\bar{c}_k\}$.
If $\bar{c}_{\min} \geq 0$ then stop.

Step 4. Let $q \in Q(i, \rho(i))$ be the job such that $\bar{c}_q = \bar{c}_{\min}$ (break tie arbitrarily).
If $q \neq \rho(q)$ then $C(\pi(q)) \leftarrow C(\pi(q)) \setminus \{q\}$.

Step 5. $\Delta_1 \leftarrow \min_{\substack{k\in T(q):\\ x_k > s_k}}\{x_k - s_k\}$,
$\Delta_2 \leftarrow \min_{\substack{(k,l)\in A:\\ k\notin T(q),\, l\in T(q)}}\{x_l - x_k - p_k\}$.
For each $k \in T(q)$, $x_k \leftarrow x_k - \min(\Delta_1, \Delta_2)$.
If $\Delta_1 > \Delta_2$ then go to Step 7.

Step 6. Let $f \in T(q)$ be the job such that $x_f = s_f$.
ReversePath(f, q).
For each $k \in T(f)$, $\rho(k) \leftarrow f$.
Go to Step 3.

Step 7. Let $(e, f)\in A$ be the pair of jobs such that $e \notin T(q)$, $f \in T(q)$, and $x_e + p_e = x_f$ (break tie arbitrarily).
ReversePath(f, q).
$\pi(f) \leftarrow e$, $C(e) \leftarrow C(e) \cup \{f\}$.
For each $k \in T(f)$, $\rho(k) \leftarrow \rho(e)$.
Go to Step 3.

여기서 ReversePath(f, q)는 f, q-경로 상의 작업들의 부모-자식 관계를 바꾼다. 알고리즘의 전반적인 흐름은 다음과 같다.

- 스텝 1에서 새로 추가되는 작업 i가 목표 시간에 시작할 수 있는지 확인한다.
- 작업 i가 늦어지는 경우, 스텝 2에서 i를 기존 트리에 추가한다.
- 스텝 3에서 최적 조건 식 (7)을 사용하여 최적 여부를 확인한다.
- 최적이 아니면, 스텝 4에서 시작 시간을 앞당길 작업들을 한정한다.
- 작업들의 시작 시간을 앞당겨(스텝 5) 정시 작업을 바꾸거나(스텝 6) 다른 트리에 포함시킨다(스텝 7).
- 최적 일정이 수립될 때까지 스텝 3부터 반복한다.

수치 예제 : 다음 문제를 풀이하는 단계 8을 고려하자.

job i	1	2	3	4	5	6	7	8	9
s_i	8	9	12	15	14	17	10	23	27
p_i	4	6	6	4	7	5	7	8	7
w_i	2	3	2	1	3	2	3	3	7

A = {(1,2),(2,6),(3,4),(3,5),(4,6),(4,7),(5,7),(6,8),(7,8),(8,9)}

단계 8을 시작하기 전 작업 7까지를 고려한 최적 일정은 [그림 3]의 (a)에 주어져 있다. 루트는 굵은 선으로 표시되어 있으며, 화살표로 트리 구조가 나타나 있다. 그때의 구체적인 자료는 다음과 같다.

job i	1	2	3	4	5	6	7
x_i	5	9	8	14	14	18	21
$\pi(i)$	2	-	5	3	-	4	5
$C(i)$	{}	{1}	{4}	{6}	{3, 7}	{}	{}
$\rho(i)$	2	2	5	5	5	5	5

단계 8에서의 진행은 다음과 같다.

Step 1. $i = 8$, $x_8 = 28$.

Step 2. $j = 7$, $\pi(8) = 7$, $C(7) = \{8\}$. $\rho(8) = 5$.

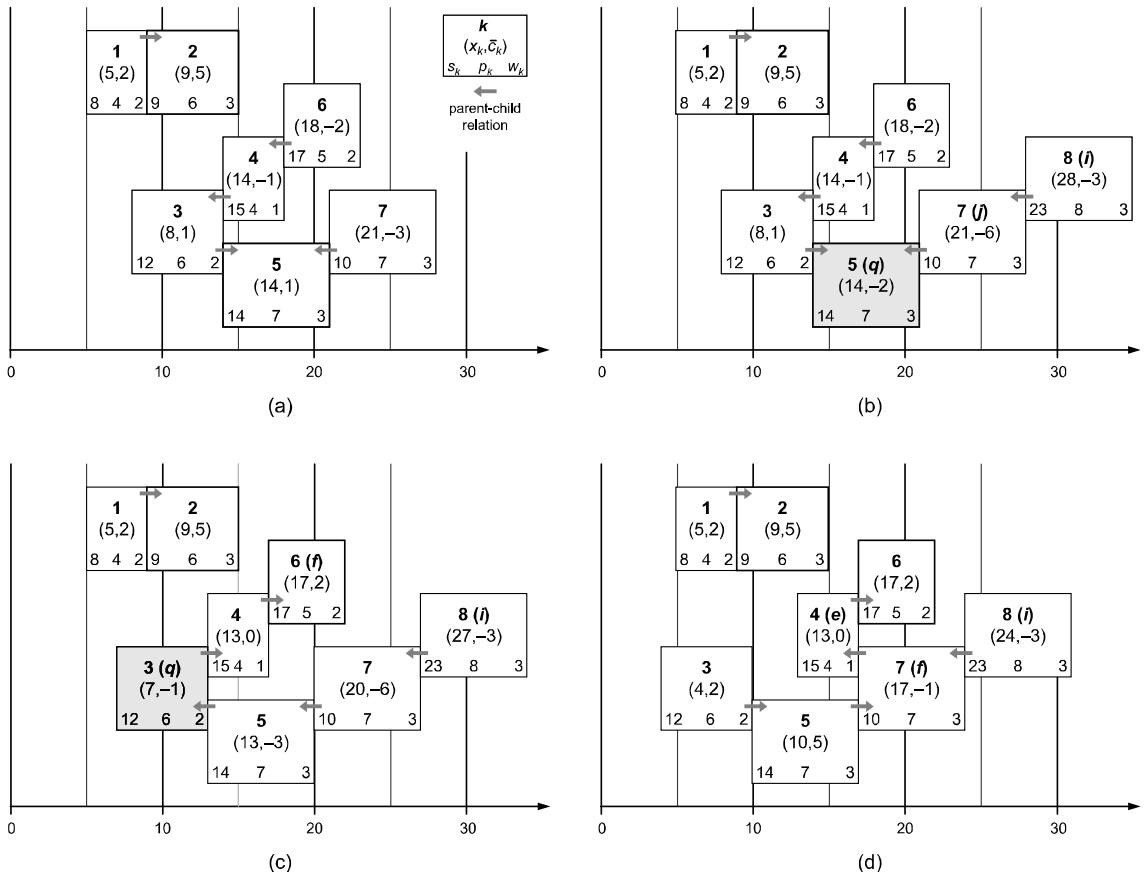
Step 3. $\bar{c}_{\min} = -2 < 0$.

Step 4. $q = 5$, $T(5) = \{3, 4, 5, 6, 7, 8\}$ [그림 3b].

Step 5, 6. $\Delta_1 = 1 < \Delta_2 = 3$, $f = 6$ [그림 3c].
Go to Step 3.

job i	3	4	5	6	7	8
x_i	7	13	13	17	20	27
$\pi(i)$	4	6	3			
$C(i)$	{5}	{3}	{7}	{4}		
$\rho(i)$	6	6	6	6	6	6

(갱신된 항목만 표시)



[그림 3] Procedure of Determining the Optimal Schedule, (a) Result of Stage 7, (b) Schedule after Step 2 in Stage 8, (c) Schedule after Step 6, (d) Result of Stage 8

Step 3. $\bar{c}_{\min} = -1 < 0$.

Step 4. $q = 3$, $T(3) = \{3, 5, 7, 8\}$.

Step 5, 7. $\Delta_1 = 4 > \Delta_2 = 3$, $e = 4$, $f = 7$ [그림 3d].

$job\ i$	3	4	5	6	7	8
x_i	4		10		17	24
$\pi(i)$	5		7		4	
$C(i)$	{}	{7}	{3}		{5, 8}	
$\rho(i)$	6		6		6	6

14

Go to Step 3.

Step 3. $\bar{c}_{\min} = 0 \geq 0$. Stop.

최적 일정의 총비용은 87이다. 각 단계 i를 마친 후 각각의 수립된 일정과 그때의 총비용(z)은 다음과 같다.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	z
Stage 1	8									0
Stage 2	5	9								6
Stage 3	5	9	12							6
Stage 4	5	9	12	18						9
Stage 5	5	9	8	15	14					14
Stage 6	5	9	8	14	14	18				17
Stage 7	5	9	8	14	14	18	21			50
Stage 8	5	9	4	13	10	17	17	24		60
Stage 9	5	9	2	11	8	17	15	22	30	87

4.2 분석

성질 4 : 알고리즘에 의해 수립되는 일정은 작업들 간의 우선순위를 고려할 때 가능 일정이다.

스텝 1을 제외하면, 일정의 변화는 스텝 5에서만

일어난다. 작업들의 시작 시간은 모든 직접 선임을 고려하여 변경되므로 위의 성질이 성립한다.

성질 5 : 알고리즘으로 수립되는 일정에 의해 연결되는 작업들은 모두 동일한 트리에 속하며, 그렇지 않은 작업들은 서로 다른 트리에 속한다.

이는 인접해진 두 작업이 이어지는 스텝 2와 7, 인접 관계가 없어져 두 작업이 *분리*되는 스텝 4를 검토하면 파악할 수 있다. 먼저, 이어질 수 있는 직접 선임이 하나인 경우와 분리된 후 시작 시간의 감소 정도가 0보다 큰 경우는 자명하다. 다음으로, 유일 정시 작업 가정을 고려하므로, 스텝 2와 7에서 동률을 발생시키는 직접 선임들은 모두 같은 트리에 속한 작업들이다. 따라서 어떤 직접 선임과 이어지더라도 연결된 작업들이 모두 하나의 트리에 속하게 된다. 마지막으로, 만일 일정에 맞추어 정확히 트리가 구성되어 왔다고 가정하면, 분리된 후 시작 시간이 변하지 않는 경우는 분리된 트리가 다시 합쳐지는 경우이다. 따라서 위의 성질이 역시 유지된다. 결과적으로 알고리즘 수행 도중 트리는 블록을 정확하게 표현한다.

일정계획 문제가 n개의 작업과 m개의 직접 우선순위로 이루어졌다고 하자.

보조정리 6 : 알고리즘의 시간 복잡도(time complexity)는 $O(n(n+m\log m))$이다.

증명. 총 n단계가 수행되며, 각 단계는 스텝 3의 종료 조건($\bar{c}_{\min} \geq 0$)이 만족될 때까지 스텝 3부터 6까지 또는 스텝 3부터 7까지 반복한다. 작업 0이 존재하므로 종료 조건은 결국 만족하게 된다. 스텝 6과 7의 실행 가능 회수를 산정하여 시간 복잡도를 계산한다. 알고리즘 실행 도중 모든 작업들의 시작 시간이 늦춰지지 않으므로 스텝 6의 실행 회수는 $O(n)$이다. 그리고 스텝 7이 실행될 때 정확히 한 쌍의 작업이 인접하게 되므로, 스텝 7의 실행 회수는 직접 선후임 관계의 두 작업이 인접해지는 최대 회수보다 커질 수 없다. 아래에서는 최대 가능 인접 회수를 산정하여 스텝 7 실행 회수의 상한(upper bound)을 구한다.

만일 어떤 작업의 직접 후임이 하나라면, 그 두 작업은 최대 한 번 인접해질 수 있다. 반면 직접 후임이 둘 이상이고 각각의 직접 후임의 시작 시간이 교대로 앞당겨진다면, 그 작업과 각각의 직접 후임 사이에 인접과 분리가 반복될 수 있다. 따라서, 직접 후임들이 앞당겨지는 회수만큼 인접해질 수 있다. 하지만 교대로 직접 후임의 시작 시간이 앞당겨져야 하므로 가장 많이 앞당겨지는 직접 후임에 의해 최대 인접 가능 회수가 제한된다. 구체적으로, 직접 후임을 q개 가지는 어떤 작업을 고려하자. 그 작업의 j번째 직접 후임은 최대 $l_j (l_j \geq 1)$번 앞당겨진다고 하면, 직접 후임들과 인접해질 수 있는 회수의 상한 $h(l_1, \cdots, l_q)$는 다음과 같다.

$$h(l_1, \cdots, l_q) = \min\left(2 \cdot \left(k - \max_{j=1, \cdots, q} l_j\right) + 1, k\right) \quad (24)$$

여기서 $k = \sum_{j=1}^{q} l_j$이다. 단계 i에서는 작업 i와 그 작업보다 우선순위가 높은 작업만의 시작 시간이 앞당겨질 수 있으며, 한 번 인접해진 두 작업은 단계가 종료될 때까지 분리되지 않는다. 따라서, l_j를 j번째 직접 후임의 모든 후임들의 개수에 1을 더한 값으로 놓을 수 있다. 이 경우 k는 해당 작업의 모든 후임들의 개수가 된다.

어떤 작업과 그 작업의 모든 후임들이 $k(\geq 1)$개의 직접 우선순위로 연결되어있다고 하자. 그러면 그 작업은 최대 k개의 후임을 가질 수 있다. 따라서 $H(k)$를 그 모든 작업들 사이의 최대 인접 가능 회수라 하면, 다음과 같이 계산된다.

$$H(k) = \max_{q=1, \cdots, k} \left\{ \max_{\substack{l_1, \cdots, l_q \geq 1: \\ l_1 + \cdots + l_q = k}} \left\{ h(l_1, \cdots, l_q) + \sum_{j=1}^{q} H(l_j - 1) \right\} \right\} \quad (25)$$

$k \geq 2$일 때, $q=2$이고 $|l_1 - l_2| \leq 1$이면 $H(k)$는 최대값을 가진다. 따라서,

$$H(k) = \begin{cases} k & \text{if } k = 0 \text{ or } 1 \\ k + H(\lfloor k/2 \rfloor - 1) + H(\lceil k/2 \rceil - 1) & \text{if } k \geq 2 \end{cases} \quad (26)$$

여기서 $\lfloor a \rfloor$와 $\lceil a \rceil$는 각각 a보다 크지 않은 최대 정수와 작지 않은 최소 정수이다. 따라서 m개의

직접 우선순위에 대하여 모든 작업들 간의 총 연결 가능 회수는 $O(m\log m)$이다.

식 (22)를 참고하면, 각 작업의 변경 비용을 순차적으로 계산하는 것이 가능함을 알 수 있다. 따라서 각 스텝은 모두 $O(n)$의 시간에 수행된다. 그러므로 시간 복잡도는 $O(n(n+m\log m))$이다. □

보조정리 7 : 단계 $(i-1)$에서 결정된 일정과 모든 트리들에 대하여 작업 1, …, $(i-1)$이 최적 조건 (7)을 만족하면, 단계 i의 수행 후 결정된 일정과 모든 트리들에 대하여 작업 1, …, i가 최적 조건 (7)을 만족한다.

위 보조정리에 대한 증명은 부록에 실었다.

정리 8 : 본 논문의 알고리즘은 $O(n(n+m\log m))$의 시간에 최적 일정을 수립한다.

증명. 보조정리 7에 의해 이전 단계의 결과가 최적 조건을 만족하면 다음 단계에서 반드시 최적 조건을 만족하는 결과를 산출한다. 그리고 가상 작업 0으로 구성된 일정과 트리는 최적 조건을 만족하며 단계 1의 입력으로 사용되므로, 귀납에 의해 알고리즘은 항상 최적 조건을 만족하는 결과를 도출한다. 보조정리 1과 3에 의해 그 결과는 최적 일정이다. □

제 4.1절에서 제시한 유일 정시 작업의 가정을 해소하기 위하여 작업들의 목표 시작 시간에 작은 변화(perturbation)를 도입한다. 구체적으로, 작업 i의 목표 시작 시간을 $(s_i+\delta_i)$로 변경하고 알고리즘을 진행한다($\delta_1<\delta_2<\cdots<\delta_n \ll \min\{p_i\}$). 알고리즘 진행 도중 각 δ_i는 누적되지 않으므로 수치계산 시 안정함(numerically stable)을 알 수 있다.

5. 결 론

본 연구에서는 E-T 일정계획 문제에서 작업들 간의 부분 우선순위가 주어졌을 때 최적 일정을 도출하는 알고리즘을 제시하였다. 이 알고리즘은 인접 관계에 있는 작업들의 블록을 기반으로 일정을 생성하며, 알고리즘에서 블록은 트리 구조로 표현된다. 주어진 작업의 개수 n과 직접 우선순위의 개수 m에 대하여 본 알고리즘의 시간 복잡도는 $O(n(n+m\log m))$이다. 본 연구의 알고리즘은 늦은 작업과 이른 작업에 대한 비용 가중치가 다른 비대칭(asymmetric) 경우로도 쉽게 확장 가능하다.

작업 순서가 모두 정해졌을 때 E-T 일정계획 문제는 $O(n\log n)$으로 해결할 수 있다[2]. 반면 본 논문에서 제시된 알고리즘은 $O(n^2)$이 소요된다. 본 연구와 Garey et al.[2]의 결과를 비교하면, 순서가 정해진 경우 인접하게 될 직접 선임이 정해져 있으며 한 번 인접해진 작업들은 분리되지 않는 성질을 가진다. 따라서 특별한 자료 구조를 사용하여 보다 효율적으로 최적 일정을 수립할 수 있다. 하지만 부분 우선순위가 주어진 경우 그러한 성질을 사용하기 힘든 한계를 가진다.

참 고 문 헌

[1] Cho, S.A., C.H. Cho, D.H. Lee, and C.B. Kim, “Single machine scheduling with maximum allowable tardiness in ET model,” *Journal of the Korean Operations Research and Management Science Society*, Vol.23, No.1(1998), pp.29-41.

[2] Garey, M.R., R.E. Tarjan, and G.T. Wilfong, “One-processor scheduling with symmetric earliness and tardiness penalties,” *Mathematics of Operations Research*, Vol.13, No.2 (1988), pp.330-348.

[3] Kedad-Sidhoum, S., Y.R. Solis, and F. Sourd, “Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates,” *European Journal of Operational Research*, Vol.189, No.3(2008), pp.1305-1316.

[4] Kim, S.J. and C.Y. Lee, “Application of ge-

netic algorithm to a job scheduling problem," *Journal of the Korean Operations Research and Management Science Society*, Vol.17, No.3(1992), pp.1-12.

[5] Kim, S.Y. and R.C. Leachman, "A new approach for resource allocation in project scheduling with variable-duration activities," *Journal of the Korean Operations Research and Management Science Society*, Vol.19, No.3(1994), pp.139-149.

[6] Kim, Y.D. and C.A. Yano, "Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates," *Naval Research Logistics*, Vol.41, No.7(1994), pp.913-933.

[7] Park, K.T. and K.H. Kim, "Berth scheduling for container terminals by using a sub-gradient optimization technique," *Journal of the Operational Research Society*, Vol.53, No.9 (2002), pp.1054-1062.

[8] Vanhoucke, M., E. Demeulemeester, and W. Herroelen, "An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem," *Annals of Operations Research*, Vol.102, No.1-4(2001), pp.330-348.

[9] West, D.B., *Introduction to Graph Theory*, Prentice-Hall, Inc., Upper Saddle River, NJ, second edition, 2001.

[10] Xie, W. and N.V. Sahinidis, "A branch-and-bound algorithm for the continuous facility layout problem," *Journal of the Operational Research Society*, Vol.32, No.4-5(2008), pp. 1016-1028.

<부 록>

본 부록에서는 보조정리 7을 증명한다.

증명. 특정 단계 i에서 스텝 3이 실행될 때, 그때까지 결정된 일정과 모든 트리들을 고려하면 항상 다음의 *최적후보 조건*이 성립함을 가정하자.

(A1) $\bar{c}_{\rho(i)} \le \min(0, \bar{c}_{\min}) + 2w_{\rho(i)}$.

(A2) 각 작업 $k \in P(i, \rho(i)) \setminus \{\rho(i)\}$에 대하여, 만일 $(\pi(i), i) \in A$이면 $\bar{c}_k \le \min(0, \bar{c}_{\min})$.

(A3) 각 작업 $k \in \{1, ..., i\} \setminus P(i, \rho(i))$는 최적 조건 (7)을 만족한다.

단계 i는 스텝 3에서 다음의 *종료 조건*을 만족하여 끝난다.

$$\bar{c}_{\min} = \min_{k \in Q(i,\ \rho(i))} \{\bar{c}_k\} \ge 0 \quad \Leftrightarrow \quad \bar{c}_k \ge 0 \quad \forall k \in Q(i, \rho(i)) \tag{27}$$

정의에 의하여, $\rho(i) \in Q(i,\ \rho(i))$이며, 다음이 성립한다.:

$$\begin{cases} k \in Q(i,\ \rho(i)) \Leftrightarrow k, \pi(k)) \in A \\ k \notin Q(i,\ \rho(i)) \Leftrightarrow (\pi(k), k) \in A \end{cases} \quad \forall k \in P(i, \rho(i)) \setminus \{\rho(i)\} \tag{28}$$

따라서 단계가 끝나면, 모든 작업들에 대해, 종료 조건 (27)과 A1이 만족하므로 최적 조건 (7a)가 만족되고, 종료 조건과 A3가 만족하므로 (7b)가, A2와 A3가 만족하므로 (7c)가 각각 만족되는 것을 알 수 있다. 그러므로 최적후보 조건의 가정이 유효하면 이 보조정리가 성립하다.

스텝 3은, 작업 i를 스텝 2에서 결정된 작업 j의 자식으로 추가한 후(C1) 처음으로 한 번 실행되고, 일부 작업들의 시작 시간을 앞당겨 정시 작업을 바꾼 후(C2) 또는 그 작업들을 다른 블록에 포함시킨 후(C3) 재차 실행된다. 아래에서는 각각의 경우에 대하여 이전에 최적후보 조건이 만족하였으면 다음에도 최적후보 조건이 만족함을 보인다.

C1. 처음으로 실행되는 경우

스텝 3이 실행될 때의 일정과 모든 트리들을 고려하자. 단계 시작 시 작업 $k \in \{1, \cdots, i\}$의 변경 비용을 $\bar{c}_k^0$라 하고, 스텝 3이 실행될 때 갱신된 변경 비용을 $\bar{c}_k$라 하면, 변경 비용의 변화는 다음과 같다($\bar{c}_i^0 = 0$으로 둠).

$$\bar{c}_k = \begin{cases} \bar{c}_k^0 - w_i & \text{if } k \in P(i, \rho(i)) \\ \bar{c}_k^0 & \text{otherwise} \end{cases} \tag{29}$$

(1) $k = \rho(i)$의 경우 : $\bar{c}_{\rho(i)} = \bar{c}_{\rho(i)}^0 - w_i$.

단계 i를 시작할 때 최적 조건이 만족됨을 가정하였으므로, (7a)에 의해 $0 \le \bar{c}^0_{\rho(i)} \le 2w_{\rho(i)}$이다. 또한 모든 $l \in Q(i, \rho(i)) \setminus \{\rho(i)\}$에 대하여, 식 (28)에 의해 $(l, \pi(l)) \in A$이므로, (7b)에 의해 $\bar{c}^0_l \ge 0$이다. 따라서,

$$\bar{c}_{\min} = \min_{l \in Q(i,\ \rho(i))} \{\bar{c}_l\} = \min_{l \in Q(i,\ \rho(i))} \{\bar{c}^0_l - w_i\} \ge -w_i \tag{30}$$

$w_i > 0$이므로, 다음과 같이 A1을 만족한다.

$$\bar{c}_{\rho(i)} = \bar{c}^0_{\rho(i)} - w_i \le \min(0,\ \bar{c}_{\min}) + 2w_{\rho(i)} \tag{31}$$

(2) $k \in P(i, \rho(i)) \setminus \{\rho(i)\})$의 경우 : $\bar{c}_k = \bar{c}^0_k - w_i$.

만일 k가 $(\pi(k), k) \in A$이면 최적 조건 (7c)에 의해 $\bar{c}^0_k \le 0$이다. 식 (30)을 고려하면, $\bar{c}_k = \bar{c}^0_k - w_i \le \min(0,\ \bar{c}_{\min})$가 되어 A2가 만족된다.

(3) $k \in \{1, \cdots, i\} \setminus P(i,\ \rho(i))$의 경우 : $\bar{c}_k = \bar{c}^0_k$.

변경 비용과 k와 관련된 트리 구조가 변하지 않았으므로 A3를 만족한다.

따라서 처음 스텝 3을 실행할 때 최적후보 조건이 성립한다.

다음으로 두 번 이상 스텝 3을 실행하는 경우를 검토한다. 단계 i에서 스텝 3을 $r(\ge 1)$ 번째 실행할 때를 고려하자. 만일 종료 조건 (27)을 만족하지 못하면 스텝 6 또는 스텝 7을 실행한 후 다시 스텝 3을 $(r+1)$ 번째 실행하게 된다. 다음을 정의하자.

$\sigma,\ \sigma'$: 각각 r 번째와 $(r+1)$ 번째 실행 때의 일정
$\tau,\ \tau'$: 각각 r 번째와 $(r+1)$ 번째 실행 때의 작업 i를 포함하는 트리
$\bar{c}_k, \bar{c}_k'$: 각각 r 번째와 $(r+1)$ 번째 실행 때의 작업 $k \in \{1, \cdots, i\}$의 변경 비용
$\bar{c}_{\min},\ \bar{c}_{\min'}$: 각각 r 번째와 $(r+1)$ 번째 실행 때의 스텝 3에서 계산되는 $\bar{c}_{\min}$
$\hat{w}_k$: 일정 σ에 따른 작업 $k \in \{1, \cdots, i\}$의 수정 가중치
$\pi'(k)$: 트리 T'에서의 k의 부모

어떤 작업 $k,\ l$에 대하여 $\rho(k),\ \pi(k),\ P(k, l),\ Q(k, l)$는 모두 r 번째 실행 때의 트리들에서 정의된다. 그리고 스텝 3을 r 번째 실행할 때 *만족되었다고 가정하는* 최적후보 조건과 $(r+1)$ 번째 수행할 때 *만족을 확인해야 하는* 최적후보 조건을 구분하기 위하여, 후자를 A1′, A2′, A3′이라 하자. 즉, A1−A3가 만족한다는 가정 하에 A1′−A3′이 만족함을 보인다.

C2. 스텝 6을 실행하는 경우

스텝 4와 6을 실행할 때 결정되는 작업 q와 f를 고려한다. T'의 루트는 f가 된다. 그리고 $\bar{c}_{\min} = \bar{c}_q$임을 고려하면 변경 비용의 변화는 다음과 같다.

$$\bar{c}_k' = \begin{cases} \bar{c}_{\min} + 2w_f & \text{if } k = f \quad (32a) \\ \bar{c}_{\min} - \bar{c}_{\pi'(k)} & \text{if } k \in P(f, q) \setminus \{f\} \quad (32b) \\ \bar{c}_k - \bar{c}_{\min} & \text{if } q \neq \rho(i) \text{ and } k \in P(\pi(q), \rho(i)) \quad (32c) \\ \bar{c}_k & \text{otherwise} \quad (32d) \end{cases}$$

- T'은 $T(q)$에 속한 작업들로 구성된다. 따라서 $\bar{c}_f'$는 일정 σ'에서 $T(q)$에 속한 작업들에 대한 수정 가중치의 합이다. $\sum_{l \in T(q)} \hat{w}_l = \bar{c}_q = \bar{c}_{\min}$이며, f는 σ'에서 정시 작업으로 변해 수정 가중치가 $2w_f$만큼 증가한다. 따라서 식 (32a)가 성립한다.
- 작업 $k \in P(f, q) \setminus \{f\}$의 경우, T와 T'에서 k와 $\pi'(k)$의 부모-자식 관계가 반대로 된다. 따라서 T'에서 k의 모든 자손들의 집합은 $T(q) \setminus T(\pi'(k))$가 된다. 그리고 그 작업들의 수정 가중치는 변하지 않으므로, $\bar{c}_k' = \sum_{l \in T(q) \setminus T(\pi'(k))} \hat{w}_l$이다. 이는 식 (32b)를 의미한다.
- 만일 $q \neq \rho(i)$이면 q를 부모로부터 분리한다. 따라서 식 (32c)와 같이 $P(\pi(q), \rho(i))$에 속한 작업들의 변경 비용이 바뀐다.
- 나머지 작업들의 변경 비용에는 변화가 없다.

다음을 고려하자.

- 종료 조건을 만족하지 못했으므로, $\bar{c}_{\min} < 0$이며 다음이 성립한다.

$$\bar{c}_{\min} \leq \bar{c}_k \quad \forall k \in Q(i, \rho(i)) \quad (33)$$

- 작업 $k \in P(f, q)$에 대하여, T와 T'에서 k와 $\pi'(k)$의 부모-자식 관계가 반대로 되므로,

$$\begin{cases} (k, \pi'(k)) \in A \Leftrightarrow \pi'(k) \notin Q(f, q) \\ (\pi'(k), k) \in A \Leftrightarrow \pi'(k) \in Q(f, q) \end{cases} \quad \forall k \in P(f, q) \setminus \{f\} \quad (34)$$

- $\bar{c}_{\min}'$은 $P(i, f)$에 속한 작업들을 대상으로 구해진다. T'의 구조를 고려하면,

$$\bar{c}_{\min}' = \min\left(\min_{k \in V_1}\{\bar{c}_k'\},\ \min_{k \in V_2}\{\bar{c}_k'\},\ \bar{c}_f'\right) \quad (35)$$

여기서 $V_1 = \{Q(i, q) \setminus P(f, q)\}$, $V_2 = \{k \in P(i, f) \cap P(f, q) \setminus \{f\} \mid (k, \pi'(k)) \in A\}$이다. 먼저 V_1에 속한 작업들의 경우, 식 (32d)에 의해 $\bar{c}_k' = \bar{c}_k$이고, 식 (33)을 고려하면,

$$\min_{k \in V_1}\{\bar{c}_k'\} = \min_{k \in V_1}\{\bar{c}_k\} \geq \bar{c}_{\min} \quad (36)$$

그리고, V_2에 속한 작업들의 경우 식 (32b)에 의해 $\bar{c}'_k = \bar{c}_{\min} - \bar{c}_{\pi'(k)}$이고, A3와 식 (34), 식 (7c)를 고려하면,

$$\min_{k\in V_2}\{\bar{c}'_k\} = \min_{k\in V_2}\{\bar{c}_{\min} - \bar{c}_{\pi'(k)}\} \geq \bar{c}_{\min} \tag{37}$$

마지막으로, 식 (32a)에 의해, $\bar{c}'_f = \bar{c}_{\min} + 2w_f \geq \bar{c}_{\min}$이다. 따라서, $\bar{c}_{\min} < 0$을 고려하면, 다음을 얻을 수 있다.

$$\bar{c}_{\min} \leq \min(0,\ \bar{c}'_{\min}) \tag{38}$$

각각의 A1′−A3′을 검토하면 다음과 같다.

(1) $k=f$의 경우 : 식 (32a)에 의해, $\bar{c}'_f = \bar{c}_{\min} + 2w_f$.
식 (38)에 의해 다음과 같이 A1′을 만족한다.

$$\bar{c}'_f = \bar{c}_{\min} + 2w_f \leq \min(0,\ \bar{c}'_{\min}) + 2w_f \tag{39}$$

(2) $k \in P(i,\ f)\setminus\{f\}$의 경우

i) $k \in P(i, f)\cap P(f, q)\setminus\{f\}$의 경우 : 식 (32b)에 의해, $\bar{c}'_k = \bar{c}_{\min} - \bar{c}_{\pi'(k)}$.
만일 $(\pi'(k), k)\in A$이면, 식 (34)에 의해 $\pi'(k)\in Q(f, q)$이며, 따라서 A3에 의해 $\bar{c}_{\pi'(k)} \geq 0$이다. 그리고 식 (38)을 고려하면 다음과 같이 A2′을 만족한다.

$$\bar{c}'_k = \bar{c}_{\min} - \bar{c}_{\pi'(k)} \leq \min(0,\ \bar{c}'_{\min}) \tag{40}$$

ii) $k \in P(i, f)\setminus P(f, q)$의 경우 : 식 (32d)에 의해, $\bar{c}'_k = \bar{c}_k$.
$k \notin P(f, q)$의 경우 부모-자식 관계가 바뀌지 않으므로, 만일 $(\pi'(k), k)\in A$이면, A3와 식 (38)에 의해 다음과 같이 A2′을 만족한다.

$$\bar{c}'_k = \bar{c}_k \leq \min(0,\ \bar{c}_{\min}) \leq \bar{c}_{\min} \leq \min(0,\ \bar{c}'_{\min}) \tag{41}$$

(3) $k \in \{1,\ \cdots,\ i\}\setminus P(i, f)$의 경우

i) $k\in P(i, q)\setminus P(i, f)$의 경우 : 식 (32b)에 의해, $\bar{c}_k{}' = \bar{c}_{\min} - \bar{c}_{\pi'(k)}$.
만일 $(k, \pi'(k))\in A$이면, 식 (34)에 의해 $\pi'(k) \notin Q(i, q)$이며, 따라서 A2에 의해 $\bar{c}_{\pi'(k)} \leq \bar{c}_{\min}$이므로 다음이 성립하여 최적 조건 (7b)를 만족하고,

$$\bar{c}'_k = \bar{c}_{\min} - \bar{c}_{\pi'(k)} \geq 0 \tag{42}$$

그렇지 않으면, 즉, $(\pi'(k), k)\in A$이면, 식 (34)에 의해 $\pi'(k)\in Q(i, q)$이며, 따라서 식 (33)에 의해 다음이 성

립하여 최적 조건 (7c)를 만족한다.

$$\bar{c}'_k = \bar{c}_{\min} - \bar{c}_{\pi'(k)} \le 0 \tag{43}$$

그러므로 A3′을 만족한다.

ii) $q \neq \rho(i)$이고 $k \in P(\pi(q),\ \rho(i)) \setminus \{\rho(i)\}$인 경우 : 식 (32c)에 의해, $\bar{c}'_k = \bar{c}_k - \bar{c}_{\min}$.

부모-자식 관계가 바뀌지 않는다. 만일 $(k, \pi'(k)) \in A$이면, $(k, \pi(k)) \in A$이므로, 식 (33)에 의해 다음과 같이 최적 조건 (7b)를 만족하며,

$$\bar{c}'_k = \bar{c}_k - \bar{c}_{\min} \ge 0 \tag{44}$$

만일 $(\pi'(k), k) \in A$면 $(\pi(k), k) \in A$이므로, A2에 의해 다음과 같이 최적 조건 (7c)를 만족한다.

$$\bar{c}'_k = \bar{c}_k - \bar{c}_{\min} \le 0 \tag{45}$$

그러므로 A3′을 만족한다.

iii) $q \neq \rho(i)$이고 $k = \rho(i)$인 경우 : 식 (32c)에 의해, $\bar{c}'_k = \bar{c}_k - \bar{c}_{\min}$.

위의 ii)의 경우와 같은 논리로 A3′을 만족한다.

iv) $k \in \{1, \cdots, i\} \setminus (P(i, \rho(i)) \cup P(f, q))$의 경우 : 식 (32d)에 의해, $\bar{c}'_k = \bar{c}_k$.

변경 비용과 k와 관련된 트리 구조가 변하지 않았으므로 A3′을 만족한다.

따라서 스텝 6을 마치고 다시 스텝 3을 실행할 때 최적후보 조건이 성립한다.

C3. 스텝 7을 실행하는 경우

스텝 4와 7까지 실행할 때 결정되는 작업 q, f, e를 고려한다. 작업 f가 e의 자식이 되고, T'의 루트는 $\rho(e)$가 된다. 변경 비용의 변화는 다음과 같다.

$$\bar{c}'_k = \begin{cases} \bar{c}_k + \bar{c}_{\min} & \text{if } k \in P(e,\ \rho(e)) & (46a) \\ \bar{c}_{\min} & \text{if } k = f & (46b) \\ \bar{c}_{\min} - \bar{c}_{\pi'(k)} & \text{if } k \in P(f,\ q) \setminus \{f\} & (46c) \\ \bar{c}_k - \bar{c}_{\min} & \text{if } q \neq \rho(i) \ \text{ and } \ k \in P(\pi(q),\ \rho(i)) & (46d) \\ \bar{c}_k & \text{otherwise} & (46e) \end{cases}$$

이 경우 $\bar{c}'_{\min}$은 $P(i,\ f)$와 $P(e,\ \rho(e))$의 작업들을 대상으로 구해진다. 즉,

$$\bar{c}'_{\min} = \min\left(\min_{k\in V_1}\{\bar{c}'_k\},\ \min_{k\in V_2}\{\bar{c}'_k\},\ \min_{k\in V_3}\{\bar{c}'_k\}\right) \tag{47}$$

여기서 V_1과 V_2는 C2의 경우와 동일하며, $V_3 = Q(e,\ \rho(e))$이다. A3에 의해 $\min_{k\in V_3}\{\bar{c}'_k\} \geq \bar{c}_{\min}$임을 보일 수 있다. 따라서 이때도 다음이 성립한다.

$$\bar{c}_{\min} \leq \min(0,\ \bar{c}'_{\min}) \tag{48}$$

다음과 같이 스텝 7을 마치고 다시 스텝 3을 실행할 때 최적후보 조건이 성립한다.

- 각 $k \in \{f\} \cup P(e,\ \rho(e))$를 제외하면 C2의 경우와 동일하게 최적후보 조건을 만족함을 보일 수 있다.
- $k = f$의 경우는 C2의 (2)-ii)의 경우와 동일하게 보일 수 있다.
- $k \in P(e,\ \rho(e))$인 경우는 C1의 (1)과 (2)의 경우와 동일하게 보일 수 있다.

따라서 귀납에 의해 이 보조정리가 성립한다. □