Society of Korea Industrial and Systems Engineering Vol. 36, No. 4 : 130-137, December 2013 ISSN : 2005-0461(print)
http://dx.doi.org/10.11627/jkise.2013.36.4.130 ISSN : 2287-7975(online)

Two-Agent Scheduling with Sequence-Dependent Exponential Learning Effects Consideration

Jin Young Choi†

Department of Industrial Engineering, Ajou University

처리순서기반 지수함수 학습효과를 고려한 2-에이전트 스케줄링

최 진 영†

아주대학교 산업공학과

In this paper, we consider a two-agent scheduling with sequence-dependent exponential learning effects consideration, where two agents A and B have to share a single machine for processing their jobs. The objective function for agent A is to minimize the total completion time of jobs for agent A subject to a given upper bound on the objective function of agent B, representing the makespan of jobs for agent B. By assuming that the learning ratios for all jobs are the same, we suggest an enumeration-based backward allocation scheduling for finding an optimal solution and exemplify it by using a small numerical example. This problem has various applications in production systems as well as in operations management.

Keywords : Two-Agent Scheduling, Sequence-Dependent Exponential Learning Effect, Makespan, Total Completion Time, Learning Ratio, Backward Allocation Scheduling

1. 서 론

본 논문에서는 서로 다른 목적함수를 갖는 두 개의 에이전트가 하나의 기계를 공유하면서 각각의 작업들을 처리하고자 할 때, 작업들의 처리 순서를 결정하는 2-에이전트 스케줄링 문제를 다룬다. 일반적으로 이러한 문제는 두 개의 에이전트가 이루고자 하는 목적함수가 서로 다르기 때문에 한 에이전트에 속하는 작업이 다른 에이전트의 목적함수에도 영향을 미칠 수 있다. 따라서 이러한 문제에 대해 기존에 많이 제안되었던 다목적 프로그래밍을 위한 접근 방법을 적용한다면 지역해 밖에 구할 수 없다는 어려움이 발생할 수 있다[7].

이러한 2-에이전트 스케줄링 문제는 다양한 산업 분야에서 찾아볼 수 있다. 하나의 예로서 생산 시스템에 대한 생산 및 정비 계획을 수립하는 두 개 부서의 일정계획 문제를 고려할 수 있다. 먼저 생산 계획을 수립하는 부서에서는 생산 시스템의 유휴 시간을 줄이면서 생산성을 최대화시키기 위한 스케줄을 추구하지만, 정비 계획을 수립하는 부서에서는 생산 시스템의 고장을 예방하기 위해서 빈번한 정비 작업을 수행할 수 있는 스케줄을 선호할 것이다. 따라서 이러한 상황에서 공통의 생산 설비에 대해 생산이 계획된 작업들과 정비를 위한 작업들의 순서를 결정하는 것은 중요한 2-에이전트 스케줄링 문제이다.

2-에이전트 스케줄링 문제는 Agnetis et al.[1]에 의해서 처음으로 제안되었다. Agnetis et al.[1]은 2-에이전트 스케줄링 문제를 제약 최적화와 파레토 최적화 문제로 구분하였으며, 각각에 대해 스케줄링 알고리즘을 제시하고 복잡성(complexity)을 분석하였다. Ng et al.[12]과 Agnetis et al.[2] 등은 두 개의 에이전트가 가질 수 있는 목적함수의 여러 조합에 대한 다양한 스케줄링 알고리즘을 제안

Received 12 November 2013; Finally Revised 15 December 2013; Accepted 23 December 2013
† Corresponding Author : choijy@ajou.ac.kr

하였다. 이러한 2-에이전트 스케줄링 문제는 Cheng et al. [6]에 의해서 멀티-에이전트 스케줄링 문제로 확장되어 다양한 조건들을 고려한 연구들이 진행되고 있다.

한편 2-에이전트 스케줄링 문제에서 작업들의 처리시간이 학습/노화 효과를 갖는 경우를 고려할 수 있다. 작업시간의 학습 효과는 비슷한 제품을 반복해서 조립하는 작업자의 작업 시간 모델에 활용될 수 있으며, 노화 효과는 특정한 도구를 이용하여 처리되는 제품에 대해 사용시간 경과에 따른 마모 등을 고려한 제품 가공 시간을 표현하는데 사용될 수 있다. 지금까지 이러한 조건을 고려한 스케줄링 문제는 주로 단일 에이전트를 고려한 경우에 대해서 연구들이 수행되어 왔으며[3, 4, 5, 10, 11], 최근 들어 2-에이전트에 대한 경우로 확장되고 있다.

최근에 발표된 학습/노화효과를 고려한 2-에이전트 스케줄링 문제에 대한 대표적인 연구 결과는 다음과 같다. Peng et al.[13]은 모든 작업들에 대해서 동일한 학습/노화 비율을 갖는 선형 학습 및 노화 효과를 고려한 2-에이전트 스케줄링 문제에 대해 최적 스케줄링 방법을 제안하였으며, Wan et al.[14]는 추가 비용으로 처리시간을 제어할 수 있는 2-에이전트 스케줄링 문제에 대해 다양한 목적함수를 고려한 알고리즘을 제안하였다. Liu et al.[9]는 작업시간이 선형적으로 시간에 비례해서 증가하는 경우에 대해 다양한 목적함수를 고려한 알고리즘을 제안하였다. 그러나 이러한 연구들은 모두 선형함수에 기반한 학습/노화 효과만을 고려하였으며, 일반적으로 많이 알려진 지수함수학습/노화 효과 모델[15]은 다루어지지 않았다.

이를 계기로 본 논문은 각 작업의 처리시간이 처리순서기반 지수함수 학습효과를 갖는 2-에이전트 스케줄링 문제를 고려하였다. 특히 목적함수가 에이전트 B의 완료시간이 주어진 상한 조건을 만족하면서 에이전트 A 작업들의 완료시간의 합을 최소화하는 문제를 고려하며, 학습 비율이 모든 작업들에 대해 동일한 경우에 대한 스케줄링 알고리즘을 제안하였다. 이를 위해 먼저 처리시간이 처리순서기반 지수함수 학습효과를 갖는 단일 에이전트 스케줄링 문제에 대한 최적 스케줄링의 특성을 분석하고, 이를 활용하여 열거기반 역방향 할당 최적 스케줄링 방법을 설계하였다. 또한 제안된 알고리즘의 적용 예를 수치 예제를 통하여 보였으며, 다양한 경우에 대한 결과를 분석하였다.

본 논문의 구성은 다음과 같다. 먼저 제 2장에서는 본 논문에서 고려된 스케줄링에 대한 기호와 문제 정의를 기술한다. 제 3장에서는 고려된 스케줄링 문제에 대해 최적해를 찾기 위한 열거기반 역방향 할당 스케줄링 방법을 제안하며, 제 4장에서는 수치 예제를 통해 제안된 알고리즘의 적용 방법을 설명한다. 마지막으로 제 5장에서는 논문 요약과 향후 연구 방향을 제시한다.

2. 문제 정의

본 논문에서 고려하고 있는 스케줄링 문제를 정의하기 위해서 필요한 기호는 다음과 같이 같다.

- n_A, n_B : 에이전트 A와 B에 할당된 작업들의 수
- $J^A = \left\{J_1^A, J_2^A, \cdots, J_{n_A}^A\right\}$: 에이전트 A의 작업 집합
- $J^B = \left\{J_1^B, J_2^B, \cdots, J_{n_B}^B\right\}$: 에이전트 B의 작업 집합
- p_i^A, p_j^B : 에이전트 A와 B에 의해 처리되는 작업 i와 j의 정상 처리시간
- $p_i^A(r)$, $p_j^B(v)$: 작업 i가 r번째에 위치하고, 작업 j가 v번째 위치할 때 에이전트 A와 B에 의한 처리시간
- $C_i^A(S), C_j^B(S)$: 스케줄 S를 적용할 때, 에이전트 A와 B에 의한 작업 i와 j의 완료시간
- $\sum_{i=1}^{n_A} C_i^A(S)$: 스케줄 S를 적용할 때, 에이전트 A의 전체 완료 시간의 합
- $f_{\max}^B = \max_{j=1,2,\ldots,n_B} C_j^B(S)$: 스케줄 S를 적용할 때, 에이전트 B의 최대 완료시간
- U : 에이전트 B 최대 완료시간에 대한 상한조건
- $b > 0$: 학습 비율

이를 이용하여 처리순서기반 지수함수 학습효과를 고려한 2-에이전트 스케줄링 문제는 다음과 같이 정의될 수 있다. 먼저 두 개의 에이전트 A와 B는 동일한 한 대의 기계를 이용하여 각각 J^A와 J^B를 처리하고자 하며, 이 작업들은 작업 시작 시점에 모두 가용하다고 가정한다. 각각의 에이전트에 속한 작업들의 처리시간은 처리순서기반 지수함수 학습효과를 가지며, 학습 비율 b를 이용하여 $p_i^A(r) = p_i^A r^{-b}$와 $p_j^B(v) = p_j^B v^{-b}$로 표현될 수 있다.

본 논문에서는 이러한 조건 하에서 에이전트 A의 모든 작업들에 대한 완료시간의 합 $\sum_{i=1}^{n_A} C_i^A$를 최소화하고, 에이전트 B의 최대 완료시간 $f_{\max}^B$가 주어진 상한(upper bound)을 넘지 않는 조건을 만족시킬 수 있는 스케줄을 찾는 문제를 고려하고 있으며, 이러한 문제는 [8]에 의해 제안된 세 항목 표기 $\Psi_1|\Psi_2|\Psi_3$ 방법에 의해서 다음과 같이 표현될 수 있다. 여기서 Ψ_1은 기계와 같은 이용 대상 자원의 수, Ψ_2는 처리시간, 만기시간, 이용할 수 있는 시간 등의 작업 특성, Ψ_3는 목적함수를 나타낸다.

$$1|p_i^A(r) = p_i^A r^{-b}, p_j^B(v) = p_j^B v^{-b}|\sum_{i=1}^{n_A} C_i^A, f_{\max}^B \le U \qquad (1)$$

3. 열거기반 역방향 할당 스케줄링

3.1 단일 에이전트 스케줄링

먼저 전체 m개의 작업들을 처리하는 단일 에이전트 스케줄링 문제에서 각 작업의 처리시간이 처리순서기반 지수함수 학습효과를 갖는 경우를 고려하자. 이 스케줄링 문제는 제 2장에서 정의된 기호에서 에이전트를 나타내는 부분이 생략된 방법으로 정의될 수 있다. 만일 목적함수가 완료시간 $C_{\max}$를 최소화하는 스케줄링 문제인 경우에 이에 대한 최적 스케줄은 다음과 같다.

Lemma 1 $1|p_i(r)=p_i r^{-b}|C_{\max}$ 문제의 최적 스케줄은 p_i가 작은 작업부터 먼저 처리하는 것이다.

(증명) m개의 작업 $\{J_1, J_2, \cdots, J_m\}$이 $p_i \le p_{i+1}$, $(i=1, 2, \cdots, m-1)$조건을 만족한다고 할 때, m개의 작업들을 그 순서대로 나열한 스케줄 S와 J_i와 J_{i+1}의 순서만 다르고 나머지 작업들의 순서는 동일한 스케줄 S'을 고려하자.

$$S: (J_1, J_2, \cdots, J_i, J_{i+1}, \cdots, J_m)$$
$$S': (J_1, J_2, \cdots, J_{i+1}, J_i, \cdots, J_m)$$

이 때 두 스케줄의 마지막 작업의 완료 시간을 비교하면,

$$\begin{aligned} & C_m(S)-C_m(S') \qquad (2)\\ &= \left(p_i i^{-b}+p_{i+1}(i+1)^{-b}\right)-\left(p_{i+1} i^{-b}+p_i(i+1)^{-b}\right)\\ &= \left(p_i-p_{i+1}\right)\left(i^{-b}-(i+1)^{-b}\right) \le 0. \end{aligned}$$

따라서 $C(S) \le C(S')$이며, $1|p_i(r)=p_i r^{-b}|C_{\max}$ 문제는 p_i가 작은 작업부터 먼저 처리함으로서 $C_{\max}$를 최소화할 수 있다. ■

이를 이용하면 임의의 작업 k가 마지막 순서로 처리되는 경우의 완료 시간을 $C_{\max, J_k}$라고 할 때, 이의 최소값 $C^*_{\max, J_k}$를 다음과 같이 계산할 수 있다.

$$\begin{aligned} C^*_{\max, J_k} &= \min \sum_{i \ne k}^{m} \sum_{r=1}^{m-1} p_i(r) x_{ir} + p_k(m) \qquad (3)\\ &= M^* + p_k m^{-b}, \end{aligned}$$

여기서 $x_{ir} \in \{0,1\}$, $\sum_{r=1}^{m} x_{ir}=1, \forall i \ne k$, $\sum_{i \ne k}^{m} x_{ir}=1, \forall r \ne m$ 조건을 만족하며, M^*값은 Lemma 1에 의해서 작업 k를 제외한 나머지 $m-1$개의 작업들을 p_i가 작은 것부터 순서대로 나열한 후 $p_i(r)$ 값을 더해서 구할 수 있다.

한편 동일한 상황에서 목적함수가 $\sum_{i=1}^{m} C_i$ 를 최소화하는 스케줄링 문제에 대한 최적 스케줄은 다음과 같다.

Lemma 2 $1|p_i(r)=p_i r^{-b}|\sum_{i=1}^{m} C_i$ 문제의 최적 스케줄은 p_i가 작은 작업부터 먼저 처리하는 것이다.

(증명) Lemma 1에서 고려한 두 개의 스케줄 S와 S'에 대하여 $\sum_{i=1}^{m} C_i(\bullet)$를 비교하면,

$$\begin{aligned} & \sum_{i=1}^{m} C_i(S)-\sum_{i=1}^{m} C_i(S') \qquad (4)\\ &= \left[(m-i+1)p_i i^{-b}+(m-i)p_{i+1}(i+1)^{-b}\right]\\ &\quad -\left[(m-i+1)p_{i+1} i^{-b}+(m-i)p_i(i+1)^{-b}\right]\\ &= \left[(m-i+1)i^{-b}-(m-i)(i+1)^{-b}\right]\left(p_i-p_{i+1}\right) \le 0 \end{aligned}$$

따라서 $\sum_{i=1}^{m} C_i(S) \le \sum_{i=1}^{m} C_i(S')$이며, p_i가 작은 작업부터 먼저 처리함으로서 $1|p_i(r)=p_i r^{-b}|\sum_{i=1}^{m} C_i$ 문제의 최적해를 구할 수 있다. ■

따라서 두 개의 Lemma를 기반으로 각 작업의 처리시간이 처리순서기반 지수함수 학습효과를 갖는 단일 에이전트 스케줄링 문제에서 목적함수가 $C_{\max}$ 또는 $\sum_{i=1}^{m} C_i$를 최소화하는 경우의 최적해는 정상적인 처리시간 p_i 값이 큰 것을 마지막에 할당하고 나머지 작업들을 p_i 값의 내림차순에 따라 역방향으로 할당하는 것이다.

3.2 2-에이전트 스케줄링

먼저 본 논문에서 고려하고 있는 두 가지 목적함수를 동시에 만족시킬 수 있는 가장 이상적인 방법은 $\sum_{i=1}^{n_A} C_i^A$가 최소가 될 수 있도록 에이전트 A의 작업들을 먼저 처리하고, $f^B_{\max} \le U$의 조건을 만족하도록 에이전트 B의 작업들을 나중에 처리하는 것이다. 그 이유는 $f^B_{\max} \le U$ 조건이 만족될 수 있다면, 에이전트 B의 작업들을 마지막 부분에 할당하는 것이 $\sum_{i=1}^{n_A} C_i^A$ 값을 줄이는데 유리하기 때문이다. 이것은 J^B에 속하는 각각의 작업에 대한 최대 완료시간의 최소값을 계산한 $V_1 = \min_{k \in J^B} \max C_{\max, J_k^B}$이

$V_1 \le U$인 조건을 만족하는 에이전트 B의 작업이 존재하는 경우에 가능하다. 이 때, V_1을 구하기 위한 식에서 $\max C_{\max,J_k^B}$ 값은 Lemma 1에 의해 J_k^B를 마지막 위치에 할당하고 이를 제외한 나머지 작업들을 첫 번째 위치부터 정상 처리시간이 긴 순서대로 스케줄링 한 후 모든 작업들에 대해 처리순서기반 지수함수 학습효과를 고려한 처리시간을 더해서 구할 수 있다.

그러나 많은 경우에 이러한 조건이 만족되지 않으며, 따라서 에이전트 A의 작업을 마지막 순서에 스케줄 하는 것이 필요할 수 있다. 이러한 결정을 하기 위해서 필요한 정보는 에이전트 B의 작업들에 대한 $C^*_{\max,J_k^B}$의 최소값 $V_2 = \min_{k \in J^B} C^*_{\max,J_k^B}$이며, 이것은 식 (3)을 이용하여 구할 수 있다. 만일 $V_2 > U$이라면, 에이전트 B의 작업을 마지막에 처리함으로서 $f^B_{\max} \le U$를 만족시킬 수 있는 방법이 없음을 의미한다.

한편 $V_2 \le U < V_1$인 경우에는 마지막 순서에 정상 처리시간이 가장 긴 에이전트 A의 작업이 할당되거나, 에이전트 B의 임의의 작업이 할당될 수 있다. 이 때 각각의 경우에 대한 최적성(optimality) 또는 가능성(feasibility)은 그 후에 이루어지는 역방향 후속 할당 결과에 따라 판단될 수 있다.

이를 기반으로 하여 본 논문에서는 고려된 스케줄링 문제에 대한 열거기반 역방향 할당 스케줄링 알고리즘을 설계하였다. 먼저 알고리즘의 설명을 위하여 필요한 몇 가지 기호는 다음과 같다. 임의의 시점에서 스케줄 하기 위해 남아있는 에이전트 A와 에이전트 B의 작업들의 집합을 각각 RJ^A와 RJ^B라고 한다. 또한 $V_2 > U$ 또는 $V_2 \le U < V_1$ 경우에서 고려되는 작업들의 할당에 대한 각각의 결과를 서브문제 SP_i로 정의하며, 각각의 SP_i는 그 시점까지의 할당, 남아있는 작업들의 정보, RJ^B에 대한 $f^B_{\max}$의 상한 값 등을 가지고 있다. 이러한 SP_i들의 집합을 USP라고 정의하며, 이는 큐 구조로 동작한다. 임의의 시점까지 찾은 가장 좋은 스케줄을 S_c로 정의하고, 이에 대한 해를 $(S_c, Z_c = \sum_{i=1}^{n_A} C_i^A(S_c))$로 표현한다. 원래의 문제를 SP_0로 정의할 때, 이 기호들을 이용하여 기술된 알고리즘의 세부 절차는 다음과 같다.

[열거기반 역방향 할당 스케줄링 알고리즘]

• 단계 1 : 초기화

SP_0에 대해서 $RJ^A = J^A, RJ^B = J^B, U$를 저장하고, $USP = \{SP_0\}, S_c = \varnothing, Z_c = \infty$라 함.

• 단계 2 : 서브문제 선택

(i) $USP = \varnothing$이면 단계 7로 이동함

(ii) $USP \neq \varnothing$이면, USP에서 하나의 서브문제를 선택하고 이를 USP에서 지운 후 선택된 서브문제를 CP라고 함($|RJ^A| + |RJ^B| = m$이라고 가정함)

• 단계 3 : CP에 대한 V_1과 V_2 계산

(i) $RJ^B \neq \varnothing$이면, V_1과 V_2를 계산한 후 단계 4로 이동함

(ii) $RJ^B = \varnothing$이면, RJ^A에 있는 작업들을 정상 처리시간이 큰 작업 순서로 CP의 할당 가능 위치의 마지막부터 역방향으로 할당함. 현재의 해를 S_c와 비교하여 업데이트 한 후 단계 2로 이동함

• 단계 4 : Case 1

(i) $V_1 \le U$이면, 이를 만족시킨 에이전트 B의 작업 k를 CP의 할당 가능 위치 중의 마지막에 스케줄하고, RJ^B에 있는 작업들부터 역방향으로 랜덤하게 할당한 후, RJ^A에 있는 작업들은 정상 처리시간이 큰 작업부터 역방향으로 할당함. 현재의 해를 S_c와 비교하여 업데이트 한 후 단계 2로 이동함

(ii) $V_1 > U$이면, 단계 5로 이동함

• 단계 5 : Case 2

(i) $V_2 > U$인 경우, $RJ^A = \varnothing$이면, CP는 해가 없으므로 단계 2로 이동함. $RJ^A \neq \varnothing$이면, 에이전트 A의 작업들 중에서 정상 처리시간이 가장 긴 작업 k를 CP의 할당 가능 위치 중의 마지막 위치에 스케줄하고, $RJ^A \leftarrow RJ^A \setminus \{J_k^A\}$로 변경함. 해당 CP문제에서 이미 할당된 에이전트 B의 작업이 있는 경우에만 $U \leftarrow U - p_k^A(m)$로 갱신 후 남아있는 스케줄 문제를 새로운 서브문제로서 USP에 등록한 후 단계 2로 이동함

(ii) $V_2 \le U$인 경우, 단계 6으로 이동함

• 단계 6 : Case 3

$V_2 \le U < V_1$인 경우, CP의 할당 가능 위치 중의 마지막에 정상 처리시간이 가장 긴 에이전트 A의 작업 k를 할당하거나, 에이전트 B의 RJ^B에 있는 작업이 할당할 수 있으며, 이에 대해 $1+|RJ^B|$개의 서브문제를 생성하여 USP에 등록함. 이 때, 에이전트 A의 작업 k가 할당되는 경우에는 해당 CP문제에서 이미 할당된 에이전트 B의 작업이 있는 경우에만 $U \leftarrow U - p_k^A(m)$로 갱신하며, 에이전트 B의 작업 k가 할당되는 경우에는 항상 $U \leftarrow U - p_k^B(m)$로 갱신함. $RJ^A \leftarrow RJ^A \setminus \{J_k^A\}$ 또는 $RJ^B \leftarrow RJ^B \setminus \{J_k^B\}$로 변경 후 단계 2로 이동함

• 단계 7 : $S^* \leftarrow S_c, Z^* \leftarrow Z_c$로 한 후 종료함 ■

본 논문에서 고려된 스케줄링 문제는 에이전트 B의 목적함수에 있는 U 값에 따라 가능해 존재 여부가 결정될 수 있으며, 이는 다음과 같이 정리될 수 있다.

Lemma 3 본 논문에서 고려된 스케줄링 문제 $1|p_i^A(r) = p_i^A r^{-b}, p_j^B(v) = p_j^B v^{-b}|\sum_{i=1}^{n_A} C_i^A, f_{\max}^B \le U$가 해를 갖기 위한 필요충분조건은 $\min C_{\max}^B \le U$이다.

(증명) 만일 $\min C_{\max}^B > U$이면 $f_{\max}^B \le U$ 조건을 만족시킬 수 없다. ■

또한 본 논문에서 제안된 열거기반 역방향 할당 스케줄링 알고리즘은 주어진 문제가 해를 가질 때, 최적해를 찾을 수 있으며, 이는 다음과 같이 정리될 수 있다.

Lemma 4 본 논문에서 제안된 열거기반 역방향 할당 스케줄링 알고리즘은 $1|p_i^A(r) = p_i^A r^{-b}, p_j^B(v) = p_j^B v^{-b}|\sum_{i=1}^{n_A} C_i^A, f_{\max}^B \le U$ 스케줄링 문제에 대한 최적해를 찾는다.

(증명) 앞에서 설명된 열거기반 역방향 할당 스케줄링 알고리즘에서 V_1과 V_2 값을 이용하여 구분된 세 가지 경우들은 발생 가능한 모든 경우를 포함하며, case 2와 case 3에 대해서 생성되는 서브 문제들은 원래 문제와 비교해서 작업의 개수가 한 개 줄어든 새로운 처리순서기반 지수함수 학습효과를 고려한 2-에이전트 스케줄링 문제이다. 또한 case 3에서 임의로 할당된 작업에 대해서는 그 이후에 적용되는 알고리즘의 단계에 의해 에이전트 B에 대한 상한 조건이 만족되는지 확인될 수 있다. 따라서 본 논문에서 제안된 알고리즘은 열거(enumeration) 기반 조합 알고리즘으로 발생 가능한 모든 경우를 고려할 수 있기 때문에 최종적으로 구한 S^*에 대한 최적성을 보장할 수 있다. ■

4. 수치 예제

4.1 수치 예제 설계

본 논문에서 제안된 알고리즘의 적용 예를 설명하기 위해서 수치 예제를 다음과 같이 랜덤하게 설계하였다. 두 에이전트 A와 B는 $J^A = \{J_1^A, J_2^A, J_3^A\}$와 $J^B = \{J_1^B, J_2^B\}$를 단일 기계에서 처리하고자 하며, 각각의 작업은 학습비율이 $b = 0.5$인 처리순서기반 지수함수 학습효과를 가진다. 전체 작업들의 정상 처리시간이 <표 1>과 같을 때, 다음과 같은 스케줄링 문제를 고려한다.

<Table 1> Normal Processing Times for Jobs

Jobs for agent A	p_i^A	Jobs for agent B	p_j^B
J_1^A	4	J_1^B	5
J_2^A	2	J_1^B	1
J_3^A	3		

$$1|p_i^A(r) = p_i^A r^{-b}, p_j^B(v) = p_j^B v^{-b}|\sum_{i=1}^{3} C_i^A, f_{\max}^B \le U \qquad (5)$$

4.2 알고리즘 적용 예

먼저 4.1에서 고려된 스케줄링 문제에 대해 에이전트 B의 최소 및 최대 완료시간을 계산하면 다음과 같다.:

$$\min C_{\max}^B = 4.536$$
$$\max C_{\max,J_1^B} = 10.012,\ \max C_{\max,J_2^B} = 11.008,$$
$$\rightarrow\ V_1 = 10.012,$$
$$C_{\max,J_1^B}^* = 8.382,\ C_{\max,J_2^B}^* = 9.378,$$
$$\rightarrow\ V_2 = 8.382.$$

이를 기반으로 U값의 범위를 다섯 가지 경우로 고려할 수 있으며, 각각의 경우에 대해 다음과 같은 결과를 얻을 수 있다.

1) $U < 4.536$일 때
Lemma 3에 의해 가능해가 존재하지 않음

2) $4.536 \le U < 8.382$일 때
먼저 $U = 8.0$으로 가정하면 3.2에서 제안된 알고리즘은 다음과 같이 적용될 수 있다.

(Iteration 1)
단계 1 : $RJ^A = J^A, RJ^B = J^B, U = 8.0,$
$USP = \{SP_0\}, S_c = \varnothing, Z_c = \infty$
단계 2 : $CP = SP_0,\ m = 5$
단계 3 : $V_1 = 10.012, V_2 = 8.382$
단계 4 : $V_1 > U$이므로 단계 5로 이동
단계 5 : $V_2 > U$이며, $RJ^A \neq \varnothing$이므로, 에이전트 A의 작업들 중에서 정상 처리시간이 가장 긴 작업 J_1^A을 마지막에 스케줄 후 $RJ^A = \{J_2^A, J_3^A\}$, $RJ^B = \{J_1^B, J_2^B\}$로 갱신하고, 이 문제를 SP_1으로 USP에 등록

(Iteration 2)

단계 2 : $CP = SP_1$, $m = 4$

단계 3 : $V_1 = 7.492$, $V_2 = 6.646$

단계 4 : $V_1 \le U$이므로 이를 만족시킨 J_1^B를 CP 문제의 마지막에 할당하고, 남아있는 에이전트 A의 작업들을 $\sum_{i \in RJ^A} C_i^A$가 최소가 되도록 할당. 따라서 해당 스케줄은 $S : J_2^A \rightarrow J_3^A \rightarrow J_2^B \rightarrow J_1^B \rightarrow J_1^A$, 목적식 값은 $Z = \sum_{i=1}^{3} C_i^A = 15.109$이며 $f_{\max}^B = 7.199$이므로 가능해. 기존 S_c보다 좋으므로 새로운 S_c로 등록

(Iteration 3)

단계 2 ; $USP = \varnothing$이므로 단계 7로 이동

단계 7 : $S^* \leftarrow S_c, Z^* \leftarrow Z_c$로 한 후 종료

3) $U = 8.382$일 때

$C^*_{\max, J_1^B} = 8.382$는 J_1^B를 마지막에 위치시키면서 만들 수 있는 최소의 완료시간이며 이것은 나머지 작업들을 정상 처리시간이 작은 것부터 처리하면 가능. 따라서 최적 스케줄은 $S^* : J_2^B \rightarrow J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_1^B$, 목적식 값은 $Z^* = 12.707$

4) $8.382 < U \le 10.012$일 때

먼저 $U = 10.0$으로 가정하면 3.2에서 제안된 알고리즘은 다음과 같이 적용될 수 있다.

(Iteration 1)

단계 1 : $RJ^A = J^A, RJ^B = J^B, U = 10.0$
$USP = \{SP_0\}, S_c = \varnothing, Z_c = \infty$

단계 2 ; $CP = SP_0$, $m = 5$

단계 3 : $V_1 = 10.012$, $V_2 = 8.382$

단계 4 : $V_1 > U$이므로 단계 5로 이동

단계 5 : $V_2 \le U$이므로 단계 6으로 이동

단계 6 : CP의 마지막에 J_1^A, J_1^B 또는 J_2^B를 할당 가능하므로 각각에 대해 RJ^A 또는 RJ^B를 변경. 할당 작업이 J_1^B일 때는 $U = 7.764$, J_2^B일 때는 $U = 9.553$로 변경하고, 각각에 해당하는 서브문제 SP_1, SP_2, SP_3를 USP에 등록 후 단계 2로 이동

(Iteration 2)

단계 2 : $CP = SP_1$(J_1^A를 할당한 경우), $m = 4$, $U = 10.0$

단계 3 : $V_1 = 7.492$, $V_2 = 6.646$

단계 4 : $V_1 \le U$이므로 이를 만족시킨 J_1^B를 CP 문제의 마지막에 할당하고, 남아있는 에이전트 A의 작업들을 $\sum_{i \in RJ^A} C_i^A$가 최소가 되도록 할당. 따라서 해당 스케줄은 $S : J_2^A \rightarrow J_3^A \rightarrow J_2^B \rightarrow J_1^B \rightarrow J_1^A$, 목적식 값은 $Z = 15.109$이며, $f_{\max}^B = 7.199$이므로 가능해. 기존 S_c보다 좋으므로 새로운 S_c로 등록

(Iteration 3)

단계 2 : $CP = SP_2$(J_1^B를 할당한 경우), $m = 4$, $U = 7.764$

단계 3 : $V_1 = 7.776$, $V_2 = 6.931$

단계 4 : $V_1 > U$이므로 단계 5로 이동

단계 5 : $V_2 < U$이므로 단계 6으로 이동

단계 6 : CP의 마지막에 J_1^A, 또는 J_2^B를 할당 가능하므로 각각에 대해 RJ^A 와 RJ^B를 변경. 할당 작업이 J_1^A일 때 $U = 5.764$, J_2^B일 때 $U = 7.264$로 변경하고, 각각에 해당하는 서브문제 SP_4, SP_5를 USP에 등록 후 단계 2로 이동

(Iteration 4)

단계 2 : $CP = SP_3$(J_2^B를 할당한 경우), $m = 4$, $U = 9.553$

단계 3 : $V_1 = 9.776$, $V_2 = 8.931$

단계 4 : $V_1 > U$이므로 단계 5로 이동

단계 5 : $V_2 < U$이므로 단계 6으로 이동

단계 6 : CP의 마지막에 J_1^A, 또는 J_1^B를 할당 가능하므로 각각에 대해 RJ^A와 RJ^B를 변경. 할당 작업이 J_1^A일 때 $U = 7.553$, J_1^B일 때 $U = 7.053$로 변경하고, 각각에 해당하는 서브문제 SP_6, SP_7을 USP에 등록 후 단계 2로 이동

(Iteration 5)

단계 2 : $CP = SP_4$ ($J_1^A \rightarrow J_1^B$를 할당한 경우), $m = 3$, $U = 5.764$

단계 3 : $V_1 = 4.992$, $V_2 = 4.699$

단계 4 : $V_1 \le U$이므로 이를 만족시킨 J_2^B를 CP 문제의 마지막에 할당하고, 남아있는 에이전트 A의 작업들을 $\sum_{i \in RJ^A} C_i^A$가 최소가 되도록 할당. 따라서 해당 스케줄은 $S : J_2^A \rightarrow J_3^A \rightarrow J_2^B \rightarrow J_1^A \rightarrow J_1^B$, 목적식 값은 $Z = 12.820$이며 $f_{\max}^B = 8.935$이므로 가능해. 기존 해보다 좋으므로 새로운 S_c로 등록

(Iteration 6)

단계 2 : $CP = SP_5(J_2^B \rightarrow J_1^B$을 할당한 경우), $m = 3$, $U = 7.264$

단계 3 : $RJ^B = \varnothing$이므로 RJ^A에 있는 작업들을 정상 처리시간이 큰 작업 순서로 CP의 할당 가능 위치의 마지막부터 역방향으로 할당. 따라서 해당 스케줄은 $S : J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_2^B \rightarrow J_1^B$, 목적식 값은 $Z = 12.552$이며, $f_{\max}^B = 9.167$이므로 가능해. 기존 해보다 좋으므로 새로운 S_c로 등록

(Iteration 7)

단계 2 : $CP = SP_6$ ($J_1^A \rightarrow J_2^B$를 할당한 경우), $m = 3$, $U = 7.553$

단계 3 : $V_1 = 7.301$, $V_2 = 7.008$

단계 4 : $V_1 \leq U$이므로 이를 만족시킨 J_1^B를 CP 문제의 마지막에 할당하고, 남아있는 에이전트 A의 작업들을 $\sum_{i \in RJ^A} C_i^A$가 최소가 되도록 할당. 따라서 해당 스케줄은 $S: J_2^A \rightarrow J_3^A \rightarrow J_1^B \rightarrow J_1^A$, $\rightarrow J_2^B$ 목적식 값은 $Z_c = 15.129$이며, $f_{\max}^B = 9.455$이므로 가능해. 현재의 S_c 유지

(Iteration 8)

단계 2 : $CP = SP_7$ ($J_1^B \rightarrow J_2^B$를 할당한 경우), $m = 3$, $U = 7.053$

단계 3 : $RJ^B = \varnothing$이므로 RJ^A에 있는 작업들을 정상 처리시간이 큰 작업 순서로 CP의 할당 가능 위치의 마지막부터 역방향으로 할당. 따라서 해당 스케줄은 $S: J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_1^B \rightarrow J_2^B$, 목적식 값은 $Z = 12.552$이며, $f_{\max}^B = 9.378$이므로 가능해. 현재의 S_c 유지

(Iteration 9)

단계 2 : $USP = \varnothing$이므로 단계 7로 이동

단계 7 : $S^* \leftarrow S_c, Z^* \leftarrow Z_c$로 한 후 종료(최적 스케줄은 $S^*: J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_2^B \rightarrow J_1^B$, $Z^* = 12.552$)

5) $U \geq 10.012$일 때

$U = 11.0$으로 가정하면, $V_1 \leq U$이므로 J_1^B 또는 J_2^B를 마지막에 위치시키면서 에이전트 A의 작업들을 정상 처리시간이 작은 것부터 먼저 처리하는 스케줄이 최적해임. 따라서 $S^*: J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_1^B \rightarrow J_2^B$ 또는 $J_2^A \rightarrow J_3^A \rightarrow J_1^A \rightarrow J_2^B \rightarrow J_1^B$, 목적식 값은 $Z^* = 12.552$.

5. 결 론

본 논문에서는 처리순서기반 지수함수 학습효과를 갖는 2-에이전트 스케줄링 문제를 고려하였다. 특히, 목적함수로서 한 에이전트 작업들의 완료시간이 주어진 상한 조건을 만족하면서 다른 에이전트 작업들의 완료시간의 합을 최소화 시키는 경우를 고려하였으며, 이 때 학습 비율은 모든 작업들에 대해 동일하다고 가정하였다. 본 논문에서는 이를 위해 먼저 처리순서기반 지수함수 학습효과를 갖는 단일 에이전트 스케줄링의 경우에 대한 최적 스케줄링 조건을 제시하고, 이를 활용한 열거기반 역방향 할당 최적 스케줄링 방법을 설계하였다. 또한 제안된 알고리즘의 적용 예를 수치 예제를 통하여 보였으며, 다양한 경우에 대한 결과를 분석하였다.

본 논문에서 제안한 열거기반 역방향 할당 스케줄링은 처리순서기반 지수함수 학습효과를 갖는 2-에이전트 스케줄링 문제에 대하여 처음으로 제안되는 방법이지만, 규모가 큰 스케줄링 문제에 대해서는 U값의 조건에 따라 알고리즘의 복잡도가 증가될 수 있다는 단점이 있다. 따라서 본 논문에 대한 향후 연구 방향으로 동일 문제에 대한 효율적인 휴리스틱 개발과 학습 비율이 작업마다 다른 경우에 대한 효율적인 스케줄링 방법 개발 등이 고려될 수 있다.

Acknowledgement

본 논문은 환경부 글로벌탑 환경기술개발사업 중 폐금속유용자원재활용 기술개발사업의 지원에 의하여 연구되었으며 이에 감사드립니다(과제번호 : GT-12-C-01-320-0).

References

[1] Agnetis, A., Mirchandani, P.B., Pacciarelli, D. and Pacifici, A., Scheduling problems with two agents. *Operations Research*, 2004, Vol. 52, No. 2, p 229-242.

[2] Agnetis, A., Pascale, G., and Pacciarelli, D., A Lagrangian approach to single-machine scheduling problems with two competing agents. *Journal of scheduling*, 2009, Vol. 12, No. 4, p 401-415.

[3] Bachman, A. and Janiak, A., Scheduling jobs with position-dependent processing times. *Journal of the operational research society*, 2004, Vol. 55, No. 3, p. 257-264.

[4] Biskup, D., Single-machine scheduling with learning considerations. *European Journal of Operational Re-*

search, 2000, Vol. 98, No. 1-4, p 273-290.

[5] Biskup, D., A state-of-the-art review on scheduling with learning effects. *European journal of operational research*, 2008, Vol. 188, No. 2, p 315-329.

[6] Cheng, T.C.E., Ng, C.T. and Yuan, J.J., Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical computer science*, 2006, Vol. 362, No. 1-3, p 273-281.

[7] Choi, J.Y. and Kim, D.G., A numerical analysis of solution approaches for multi-agent scheduling problems. Proceeding of the 43rd international conference on computers and industrial engineering, 2013.

[8] Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy, Kan A.H.G., Optimization and approximation in deterministic sequencing and scheduling theory : a survey. *Annals of Discrete Mathematics*, 1979, Vol. 5, p 287-326.

[9] Liu, P., Yi, N., and Zhou, X., Two-agent single-machine scheduling problems under increasing linear deterioration. *Applied Mathematical Modelling*, 2011, Vol. 35, p 2290-2296.

[10] Mosheiov, G., Scheduling problems with a learning effect. *European Journal of Operational Research*, 2001, Vol. 132, No. 3, p 687-693.

[11] Mosheiov, G., A note on scheduling deteriorating jobs. *Mathematical and Computer Modeling*, 2005, Vol. 41, No. 8-9, p 883-886.

[12] Ng, C.T., Cheng, T.C.C.E., and Yuan, J.J., A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, 2006, Vol. 12, No. 4, p 387-394.

[13] Peng, L., Xiaoye, Z. and Lixin, T., Two-agent single-machine scheduling with position-dependent processing times. *International Journal of Advanced Manufacturing Systems*, 2010, Vol. 48, No. 1-4, p 325-331.

[14] Wan, G., Vakati, S.R., Leung, J.Y.T., and Pinedo, M., Scheduling two agents with controllable processing times. *European Journal of Operational Research*, 2010, Vol. 205, No. 3, p 528-539.

[15] Wang, J.B. and Wang, J.J., Single-machine scheduling jobs with exponential learning functions. *Computers and Industrial Engineering*, 2011, Vol. 60, No. 4, p 755-759.