Journal of Korean Society of Industrial and Systems Engineering
https://doi.org/10.11627/jksie.2021.44.4.032

Vol. 44, No. 4 : 32-42, December 2021
ISSN : 2005-0461 / EISSN : 2287-7975

Scheduling of Printing Process in which Ink Color Changes Exist

Jae Kyeong Moon · Hyun Seop Uhm · Hyun Chul Tae†

Department of Digital Healthcare Research Korea Institute of Industrial Technology

잉크 색상 변화가 존재하는 인쇄 공정의 스케줄링

문재경 · 엄현섭 · 태현철†

한국생산기술연구원 디지털헬스케어연구부문

The printing process can have to print various colors with a limited capacity of printing facility such as ink containers that are needed cleaning to change color. In each container, cleaning time exists to assign corresponding inks, and it is considered as the setup cost required to reduce the increasing productivity. The existing manual method, which is based on the worker's experience or intuition, is difficult to respond to the diversification of color requirements, mathematical modeling and algorithms are suggested for efficient scheduling. In this study, we propose a new type of scheduling problem for the printing process. First, we suggest a mathematical model that optimizes the color assignment and scheduling. Although the suggested model guarantees global optimality, it needs a lot of computational time to solve. Thus, we decompose the original problem into sequencing orders and allocating ink problems. An approximate function is used to compute the job scheduling, and local search heuristic based on 2-opt algorithm is suggested for reducing computational time. In order to verify the effectiveness of our method, we compared the algorithms' performance. The results show that the suggested decomposition structure can find acceptable solutions within a reasonable time. Also, we present schematized results for field application.

Keywords : Printing Process, Scheduling, Mixed Integer Programming, 2-OPT Algorithm

1. 서 론

제조업 생산 현장에서 적은 시간과 비용으로 더 많은 제품을 생산하려면 효율적인 생산 계획은 필수적이다. 좋은 생산 계획을 위해서는 납기, 전체 생산 시간 등의 고려해야 할 요소가 많은데, 산업군에 따른 생산 방식과 공정의 특성이 다르기 때문에 어느 요소에 가중치를 두는지에 따라 문제의 형태가 달라진다. 본 연구에서는 다양한 형태의 스케줄링 문제 중 실제 인쇄 산업 현장에서의 스케줄링 최적화 문제를 다루었다.

인쇄 기업은 고객의 주문에 따라 필요한 잉크를 설비에 배치하고 제품을 생산한다. 주문 사항은 각기 다르고 주문 이전에 미리 생산할 수도 없기 때문에, 다품종 소량 생산 방식의 운영은 불가피하다. 이에 맞춰 설비의 잉크통에 담긴 잉크 색상만 교체하면 작업이 가능한 설비를 사용하는 것이 일반적이다. 이러한 설비에서 작업자는 생산 스케줄에 따라 해당 순서에서 인쇄되는 주문이 필요로 하는 색상의 잉크를 잉크통에 담는다. 이 때, 빈 잉크통이 없으면 기존에 다른 색상 잉크가 담긴 잉크통을 세척해야 하는데, 세척 시간은 기존에 담긴 색상과 새로 담을 색상의 조합에 의해 결정된다. 따라서 주문이 필요로 하는 색상 구성에 따라 세척할 잉크통을 결정하는 문제가 스케줄링의 중요한 요인으로 작용한다.

문제점은 주문 사항이 다양해짐과 함께 발생한다. 인쇄 공정에서 주문 사항의 다양화는 업체가 생산해야 할 품종

Received 18 October 2021; Finally Revised 23 December 2021; Accepted 24 December 2021
† Corresponding Author : sage@kitech.re.kr

수가 늘어나는 것과 같다. 다양해진 만큼 여러 대의 설비를 도입하면 쉽게 대응할 수 있지만, 이런 방식에는 많은 자금이 필요하고 추후에 수요가 줄어들게 되면 더 큰 문제를 야기할 수도 있다. 따라서 효율적인 생산스케줄을 수립하는 방식의 대응이 일반적이다. 하지만 설비 용량과 원재료 수, 주문 사항 등의 생산 과정 관리에 필요한 정보들의 전산화 수준이 낮은 중소기업에서는 스케줄링이 작업자의 경험과 직감에 의존하여 수작업으로 이루어지고 있다. 이러한 방식은 시간이 많이 소요될 뿐만 아니라 효율적인 스케줄을 만들기도 어렵다. 비효율적인 스케줄은 준비 작업의 빈도를 늘리게 되고, 이는 곧 가동률과 작업 능률 저하 및 불량률의 상승으로 귀결된다. 특히 인쇄 공정의 경우 잦은 준비 작업으로 인해 잉크통 세척 빈도가 증가하면 그 과정에서 환경적 이슈도 발생하게 된다.

이에 본 연구에서는 새로운 유형의 스케줄링 문제와 혼합정수계획법을 이용한 수리모형을 제시하고, 그에 대한 해법으로 2 - OPT 알고리즘을 응용한 발견적 기법을 제시하고자 한다.

2. 관련 연구

본 연구에서 다루는 스케줄링 문제는 작업 순서에 따라 비용이 가변적으로 발생하는 점에서 외판원 문제(TSP)와 유사함을 보인다. TSP 문제를 해결하기 위한 연구는 여러 알고리즘을 개발하는 방향으로 진행되어 왔다.

최적해를 보장하는 exact algorithm으로 Dantzig, Fulkerson and Johnson(1954), Miller, Tucker and Zemlin(1960)은 혼합정수계획법을 이용한 수리모형을 제시하였고[9], Bellman (1962)은 동적계획법을 제안하였다[2]. Garfinkel(1973)은 분지한계법을 적용한 알고리즘을 제시하였다[3]. 하지만 exact algorithm은 최적해 도출에 상당히 오랜 시간을 요구했기 때문에, 이후 최적해를 보장하지 않더라도 짧은 시간 내에 결과를 도출할 수 있는 휴리스틱 알고리즘 연구가 진행되었다. Rosenkrantz, et al.(1977)은 최근접 이웃해 알고리즘을 제시하였고[13], Lin and Kerninghan(1973)은 r-opt 알고리즘을 제시하였다[11]. Kirkpatrick, et al.(1983)은 담금질 기법을 제시하였으며[8], Glover(1989, 1990)은 타부 탐색 기법을 제시하였다[4, 5].

TSP 문제 해결을 위한 연구를 기반으로, 순서 의존 준비 시간을 갖는 공정에 대한 스케줄링 최적화 문제도 많은 연구자들에 의해 다뤄져 왔다. Gupta(1986)는 순서 의존 준비 시간을 갖는 흐름 공정의 일정 계획을 위해 TSP에 기반한 근사 알고리즘을 제시하였다[6]. 작업 투입 시점과 순서 의존적인 준비 시간이 있는 공정에서 Shin, et al.(2001)은 타부 탐색 기법으로 단일 기계에 대한 일정 계획을 제시하였고[14], Joo(2009)는 개미군집 시스템으로 병렬 기계에 대한 일정 계획을 제시하였다[7]. Lee and Jeong(2020)은 휴리스틱 알고리즘을 통해 긴급 주문에 대처할 수 있도록 하였다[10]. 흐름 공정에서 확장되어 잡샵 스케줄링 문제에 대해서는 Artigues and Feillet(2008)은 분지한계법을 제시하였다[1]. Nadesri, et al.(2010)은 담금질 기법을 제시하고 다른 휴리스틱 알고리즘과 성능을 비교하였으며[12], Vela, et al.(2010)은 유전 알고리즘과 이웃해 탐색 알고리즘을 혼합한 기법을 개발하였다[15]. 최근 연구로는 Yoo, et al.(2019, 2020)이 딥러닝과 강화학습 기반의 실시간 일정 계획 모델을 개발하여 납기 위반과 작업 준비 비용을 최소화하였다[17, 18].

그러나 본 연구에서 다루는 문제는 일반적인 TSP와 다르게 준비 시간이 순서에 의존적일 뿐 아니라, 같은 순서로 작업하더라도 설비 배치에 따라 다르게 발생할 수 있다. 유사한 경우에 대한 연구로 Yu(2015)는 전체 순서에서 작업의 위치에 따라 다른 준비 비용을 갖는 PCB 생산 공정의 일정 계획법을 제시하였다[19]. 하지만 인쇄 공정과는 생산 환경이 다른 점에서 동일하게 적용하기 어렵다. 이에 본 연구를 통해 인쇄 공정의 특성을 반영하여 준비 시간을 단축시킬 수 있는 스케줄링 방법을 제시한다.

3. 문제 소개 및 수리모형

3.1 문제 소개

본 연구에서 다루는 생산 환경은 한 라인에서 여러 주문을 인쇄하는 공정이다. 각 주문은 필요한 색상 구성이 있고, 이를 위해 해당 색상의 잉크를 설비 라인에 있는 잉크통에 담는다. 이 때 설비 라인 용량에 제약이 있어 잉크통의 수는 제한적이다. 여러 설비 라인에서 각각 한 종류의 주문만 인쇄하면 필요한 잉크를 모두 담을 수 있지만, 본 연구에서는 여러 종류의 주문을 한 설비 라인이 인쇄하는 경우를 다루려 한다. 주문이 필요로 하는 색상 구성이 바뀌었을 때, 연속된 두 주문에 필요한 색상이 잉크통보다 많은 경우가 발생할 수 있다. 이 경우, 앞선 주문의 인쇄가 완료된 후에 다음 주문의 인쇄를 위해 잉크통의 세척이 불가피하다.

이러한 조건에서 각 주문의 인쇄 시간이 작업 순서에 상관없이 일정하면, 전체 공정의 작업 시간을 최소화하는 문제는 잉크통의 세척 시간을 최소화하는 문제와 동일하다. 세척 시간의 총합은 주문을 인쇄하는 순서에 의해 결정되는데, 여기서 주문을 노드, 세척 시간을 노드 간 거리로 보면 일종의 외판원 문제(TSP)로 취급할 수 있다.

TSP 모형의 비용 c_{ij}는 주문 i를 인쇄한 직후에 주문 j를 인쇄할 때 발생하는 세척 시간을 의미하고, 이는 전체 작업 순서 중 어느 위치에 오더라도 일정하게 적용된다. 그러나 실제 인쇄 공정에서는 전체 작업 순서와 그에 따라 어느잉크통을 세척하고 어떤 색을 담는지가 비용 계산에 영향을 줄 수 있다. 이러한 영향을 3.1.1에서 예시를 통해 설명한다.

3.1.1 인쇄 공정의 특성 예시

예를 들어, 3개의 주문에서 사용 가능한 색상은 6종류가 있다고 가정하였을 때, 각각의 주문 1, 2, 3이 필요로 하는 색상과 색상 간 교체 시간은 각각 <Table 1>, <Table 2>와 같다.

<Table 1> Illustration of Color Requirements for Jobs

Job	Color					
	C	M	Y	K	O	G
1	1	1	1	1	0	0
2	0	1	1	0	1	0
3	1	0	1	0	0	1

<Table 2> Illustration of color-to-color Cleaning Time

Color	C	M	Y	K	O	G
C	0	1	2	3	4	5
M	3	0	2	5	1	4
Y	4	2	0	1	3	5
K	2	4	5	0	4	1
O	1	3	4	5	0	2
G	5	4	3	2	1	0

잉크통을 4개로 가정하면, 가능한 모든 경우에서 한 주문의 인쇄가 끝나면 적어도 1개의 잉크통을 세척해야 한다. 먼저 인쇄 순서를 '1 - 2 - 3'으로 가정하고, 인쇄 순서에 따라 잉크통의 세척 결정 및 잉크 할당이 전체 비용에 어떻게 영향을 주는지 살펴보자.

가장 먼저 주문 1을 인쇄하기 위해 필요한 색상 수와 잉크통 수가 동일하므로 4개의 잉크통에 차례로 C, M, Y, K를 담는다(1번: C, 2번: M, 3번: Y, 4번: K). 다음으로 주문 2를 인쇄하기 위해 필요한 색상은 M, Y, O다. 현재 모든 잉크통에 잉크가 담겨 있기 때문에 주문 2에서 사용되지 않는 잉크를 담은 1번(C)과 4번(K) 잉크통 중 한 개는 세척해야 한다. 여기서 1번을 세척하거나 4번을 세척하는 두 가지로 경우가 나뉜다.

1번을 세척하는 경우 O, M, Y, K가, 4번을 세척하는 경우 C, M, Y, O가 잉크통에 담긴다. 세척 시간은 두 경우 모두 4로 같다. 따라서 주문 2의 인쇄가 완료되는 시점까지 발생하는 세척 시간은 같지만, 잉크통들에 담긴 색상이 다르므로 이후 작업에 영향을 줄 수 있다.

주문 3은 색상 C, Y, G를 필요로 한다. 이를 위해 앞서 1번 잉크통을 세척했다면 현재 O, M, K가 담긴 잉크통 중 어디에 C, G를 담을지 고려하여 2개의 잉크통을 세척해야 하는 반면, 앞서 4번 잉크통을 세척했다면 현재 C와 Y가 담겨 있으므로 M, O가 담긴 잉크통 중 1개만 세척하여 G를 담으면 된다. 여기서 교체되는 색과 교체하는 색의 조합에 따라서도 세척 시간은 달라질 수 있다. 각각의 경우에서 <Table 3>, <Table 4>와 같다.

<Table 3> Variation of Total Cleaning Time by Color Change when ink Container 1 was Cleaned

Color change	Cleaning time	Total
O → C, M → G	1 + 4 = 5	4 + 5 = 9
O → C, K → G	1 + 1 = 2	4 + 2 = 6
O → G, M → C	2 + 3 = 5	4 + 5 = 9
O → G, K → C	2 + 2 = 4	4 + 4 = 8
M → C, K → G	3 + 1 = 4	4 + 4 = 8
M → G, K → C	4 + 2 = 6	4 + 6 = 10

<Table 4> Variation of Total Cleaning Time by Color Change when Ink Container 4 was Cleaned

Color change	Cleaning Time	Total
M → G	4	4 + 4 = 8
O → G	2	4 + 2 = 6

<Table 3, 4>의 인쇄 순서는 같지만, 작업 준비 방식에 따라 전체 세척 시간이 달라진다.

c_{ij}는 전체 작업 순서에서 주문 i, j의 위치에도 영향을 받는다. <Table 1>과 <Table 2>의 예로 다시 돌아가서, 이번에는 작업 순서를 '1 - 3 - 2'와 '3 - 2 - 1'로 가정하자. 두 경우 모두 주문 3 직후에 주문 2를 인쇄하는 경우가 존재하고, 이 때의 세척 시간 c_{32}의 변화는 다음과 같다.

<Table 5> Variation of Cleaning Time by order which is 1 - 3 - 2

Order	Number of the ink container				Cleaning time
	1	2	3	4	
1	C	M	Y	K	-
1 → 3	C	M	Y	K→ G	1
3 → 2	C→ O	M	Y	G	5

'1 - 3 - 2'의 순서로 인쇄하는 경우, 먼저 주문 1을 인쇄하기 위해 C, M, Y, K를 잉크통에 담는다. 주문 1의 인쇄 후, 주문 3의 인쇄를 위해 1개의 잉크통은 세척해야 한다.

여기서 K가 담긴 4번 잉크통을 세척하고 G를 담았다고 가정한다. 그 다음, 주문 2에 필요한 O를 담기 위해 이번에는 C가 담긴 1번 잉크통을 세척한다. 따라서 c_{32}는 'C→O'에 필요한 세척 시간으로 5가 된다.

<Table 6> Variation of Cleaning Time by the order which is 3 - 2 - 1

Order	Number of the ink container				Cleaning time
	1	2	3	4	
3	C	Y	G	-	-
3 → 2	C→ M	Y	G → O	-	2
2 → 1	M	Y	O → C	K	1

'3 - 2 - 1'의 순서로 인쇄하는 경우, 먼저 주문 3이 필요로 하는 색상 C, Y, G를 잉크통에 담고, 1개의 잉크통은 비워 둔다. 주문 2를 인쇄하기 위해 C가 담긴 1번 잉크통과 G가 담긴 3번 잉크통을 세척 후, 각각 M과 O로 교체한다. 이 때 c_{32}는 'C→M'과 'G→O'에 필요한 세척 시간을 더한 2가 된다. 이는 앞선 경우의 c_{32}와 다름을 알 수 있다.

이처럼 인쇄 공정의 스케줄링에 있어 작업 순서와 세척하는 잉크통 결정은 중요한 요인으로 작용하고, 이는 스케줄링 전반에 영향을 미친다. 따라서 전체적인 관점에서의 일정 계획을 필요로 하는 측면에서 외판원 문제와 비슷하면서도 동일한 방식으로는 최적해를 도출하기 어려운 구조로 이루어져 있다고 볼 수 있다.

3.2 수리모형

위 같은 특성을 혼합정수계획법으로 구현하였다.

본 연구에서는 k개의 잉크통이 있는 단일 설비 라인이 i개의 주문을 인쇄한다고 가정한다. 설비 용량을 고려하여 주문은 m개의 색상 중 k개 이하로 사용할 수 있고, 이를 이진행렬 R_{ic}로 정의하였다. 라인에서 효율적 설비 배치표를 정의하기 위하여 작업 순서 Slot을 정의하였고, 순서 Slot에서의 잉크통별 사용 색상을 계산할 수 있도록 하였다. 이 때 단일 라인에서 모든 주문을 인쇄해야 하기 때문에 순서 Slot의 수는 주문 수 와 같다($i=l$). 색상 조합에 따라 교체에 필요한 세척 시간은 $S_{c_1c_2}$로 정의하였는데, c_1과 c_2가 같을 때의 $S_{c_1c_2}$는 0으로 세척이 발생하지 않는다.

이를 바탕으로 인쇄 공정의 스케줄링 모형을 구성하기 위해 필요한 변수들과 제약식을 다음과 같이 정의한다.

<Set>

I: 주문 집합, $I=\{1, 2, \cdots, i\}$

O: 순서 집합, $O=\{1, 2, \cdots, l\}$

K: 잉크통 집합, $K=\{1, 2, 3, \cdots, k\}$

C: 색상 집합, $C=\{c_1, c_2, \cdots, c_m\}$

<Parameter>

R_{ic}: 주문 i를 수행하는데 색상 c가 필요하면 1, 그렇지 않으면 0

$S_{c_1c_2}$: c_1을 c_2로 교체하는데 필요한 세척 시간

<Decision Variable>

x_{il}: 주문 i가 l번째로 수행되면 1, 그렇지 않으면 0, $[x_{il}\ \forall i,l]=x\in \boldsymbol{X}$

y_{ckl}: l번째 작업이 색상 c를 잉크통 k에 사용하면 1, 그렇지 않으면 0

$z_{c_1c_2l}$: l번째 작업 직전에 색상을 c_1에서 c_2로 변경하는 잉크통이 있으면 1, 그렇지 않으면 0

$$\text{minimize} \sum_{c_1\in C}\sum_{c_2\in C}\sum_{l\in O} S_{c_1c_2}z_{c_1c_2l} \tag{1}$$

subject to:

$$\sum_{l\in O} x_{il}=1\ \ \forall i\in I \tag{2}$$

$$\sum_{i\in I} x_{il}=1\ \ \forall l\in O \tag{3}$$

$$\sum_{k\in K} y_{ckl} \geq \sum_{i\in I} R_{ic}x_{il}\ \ \forall c\in C,\ l\in O \tag{4}$$

$$\sum_{k\in K} y_{ckl} \leq 1\ \ \forall c\in C,\ l\in O \tag{5}$$

$$\sum_{c\in C} y_{ckl} \leq 1\ \ \forall k\in K,\ l\in O \tag{6}$$

$$\sum_{c\in C} y_{ck(l-1)} \leq \sum_{c\in C} y_{ckl} \forall k\in K,\ l\in O|l>1 \tag{7}$$

$$z_{c_1c_2l} \geq y_{c_1k(l-1)}+y_{c_2kl}-1 \quad \forall c_1,\ c_2\in C,\ k\in K,\ l\in O|l>1 \tag{8}$$

$$x_{il}\in\{0,1\}\ \ \forall i\in I,\ l\in O \tag{9}$$

$$y_{ckl}\in\{0,1\}\ \ \forall c\in C,\ k\in K,\ l\in O \tag{10}$$

$$z_{c_1c_2l}\in\{0,1\}\ \ \forall c_1,\ c_2\in C,\ l\in O \tag{11}$$

목적함수 (1)은 세척 시간을 의미하고, 전체 공정의 세척 시간 총합을 최소화한다. 제약식 (2)~(3)은 모든 주문이 각 순서 별로 한 번씩만 인쇄되어야 함을 의미한다. 제약식 (4)는 주문이 인쇄되는 순서의 잉크통에는 해당 주문이 필요로 하는 색상이 있게 하는 식이다. 제약식 (5)는 색상이 여러 잉크통에 중복하여 담기는 것을 제한하고, 제약식 (6)은 잉크통이 여러 색상을 중복하여 담는 것을 제한한다. 제약식 (7)은 한 번 채워진 잉크통이 다시 비워질 수 없도록 한다. 제약식 (8)은 세척이 발생하는 조건에 대한 식이다. 제약식 (9)~(11)은 결정변수의 조건에 관한 식이다.

위 수리모형은 혼합정수계획법으로 구현했기 때문에, 상용 최적화 프로그램 Gurobi(Ver. 9.1.2)를 이용하여 해를 탐색하였다. 상세한 결과는 5장에 있다.

4. 해법

상기 수리모형은 주문 수가 늘어남에 따라 계산 시간이 증가하여 최적해 도출에 어려움이 있다(5장 참고). 이에 본 연구에서는 문제를 2단계로 분할하는 방식을 도입하였다. 먼저, 상위 부분 문제에서는 전체 작업 순서를 결정한다. 하위 부분 문제에서는 작업 순서가 결정된 상태에서, 세척 시간 최소화를 위한 잉크통 별 잉크 할당을 결정하게 된다.

4.1 상위 부분 문제 해법

상위 부분 문제에서는 전체 작업의 순서를 결정한다. 이때, 원 문제의 목적 함수인 세척 시간은 잉크 할당이 완료되어야 알 수 있다. 이는 하위 부분 문제에서 결정하기 때문에, 상위 부분 문제는 잉크 할당이 완료되지 않은 상태에서 작업순서를 결정하게 된다. 따라서 잉크 할당에 따른 세척 시간을 알 수 없는 상위 부분 문제는 기존의 목적함수를 적용하기 어렵다. 이에 본 연구에서는 상위 부분 문제를 위한 새로운 목적 함수를 개발하였다. 새로운 목적함수는 잉크 할당 없이 작업 순서만으로 세척 시간을 예측할 수 있도록 근사 함수의 형태로 디자인하였다. 세척 시간의 예측을 위해서는 일반적으로 비례 관계에 있는 세척 횟수를 사용하였다. 세척 횟수는 연속된 작업들에서 동일한 색상을 많이 사용할수록 줄어드는 경향이 있고, 본 연구에서는 이 점을 활용하여 아래와 같이 근사 함수를 설계했다.

아래 <Figure 1>은 연속적으로 진행되는 작업에 대하여 특정 파라미터($step\ size = 4$) 상황에서 세척 횟수를 예측하는 방식이다. 예를 들어, 4번의 연속된 작업에서 색상별로 사용횟수를 측정하였을 때, 색상 A가 4회, 색상 B가 3회, 색상 C가 2회, 색상 D가 2회, 색상 E가 1회 사용되었다고 가정하자. 4번의 작업 동안 계속해서 A를 사용해야 하므로 A가 담긴 잉크통을 세척하지 않는 것이 적절하다. E는 같은 기간 동안 1회 사용되므로 세척 횟수를 증가시키는 요인으로 볼 수 있다.

상위 부분 문제에서는 <Figure 1>에서 색상 A와 같이 연속된 작업들에서 계속해서 사용되는 색상의 수를 최대화한다. 입력 파라미터로 $step\ size = p$가 있다면 <Figure 1B>와 같이 각 스텝에서 각 컬러의 사용 횟수를 측정 후 정렬한다. 목적 함수는 사용 횟수가 많은 색상의 수를 단계적으로 최대화(다목적 최적화)한다.

이를 위하여 추가된 변수 및 상위 부분 문제는 아래와 같다.

<Notation>

p: 목적 함수 계산을 위한 $step\ size$

t_{lc}^{q}: 주문 순서 l부터 그 이후의 p 개의 작업 순서 $(l+p-1)$ 안에서 색상 c가 정확히 q번 사용되었으면 1, 아니면 0

$$\text{maximize} \sum_{\substack{l \in L \\ l \le |L|-(p-1)}} \sum_{c \in C} t_{lc}^{p} \tag{12}$$

$$\text{maximize} \sum_{\substack{l \in L \\ l \le |L|-(p-2)}} \sum_{c \in C} t_{lc}^{(p-1)}$$

$$\cdots$$

$$\text{maximize} \sum_{\substack{l \in L \\ l \le |L|-1}} \sum_{c \in C} t_{lc}^{2}$$

subject to: (2), (3)

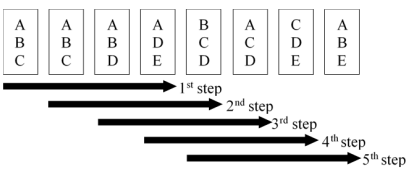
$$t_{lc}^{q} \le \frac{1}{q} \sum_{i \in I} \sum_{l \le l' < l+p} R_{ic} x_{il} \quad \forall 2 \le q \le p,\ l \in O,\ c \in C \tag{13}$$

$$t_{lc}^{q} \le 1 - \frac{1}{bigM} \left(\sum_{i \in I} \sum_{l \le l' < l+p} R_{ic} x_{il} - q \right) \quad \forall 2 \le q \le p,\ l \in O,\ c \in C \tag{14}$$

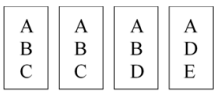
$$t_{lc}^{q} \in \{0,1\} \quad \forall 2 \le q \le p,\ l \in O,\ c \in C \tag{15}$$

목적함수 (12)는 사용 횟수가 많은 색상의 수를 순차적으로 최대화하는 목적함수이다. 제약식 (13)~(14)는 binary 변수의 특성을 이용하여 각 step마다 색상 별로 사용된 횟

(A) Method of generating partial order when $step\_size$ is 4



(B) Method of encoding objective value in 1st step



The number of colors used 4 times: 1; A
The number of colors used 3 times: 1; B
The number of colors used 2 times: 2; C, D

<Figure 1> Illustration of upper-level algorithm

수를 계산한다. 즉, 주문 순서 l을 포함한 p개의 작업 동안 색상 c가 q보다 적게 사용되면 제약식 (13)에 의해 t_{lc}^{q}가 0이 되고, 많이 사용되면 제약식 (14)에 의해 t_{lc}^{q}가 0이 되게 하여 정확히 q번 사용돼야만 1이 되도록 한다. 마지막으로 제약식 (15)은 변수 제약식이다.

4.1.1 발견적 기법

위 상위 부분 문제는 다목적 최적화로 exact solution을 찾기가 어려워 문제를 푸는데 많은 계산 시간이 소요된다. 이에 따라 현장 적용성을 고려하여 실용적인 시간 내에 좋은 해를 탐색하기 위한 발견적 기법으로 2 - OPT 기반의 이웃해 탐색 알고리즘(Local Search Heuristic)을 구현하였다.

알고리즘은 임의의 초기해로부터 매 시행마다 2 - OPT 기반의 이웃해를 생성하고, <Figure 1>에서 설명한 목적함수를 통해 가장 개선된 해로 이동한다. 알고리즘의 구조는 Wikipedia[16]을 참고하였고, 구체적인 알고리즘의 내용은 다음과 같다.

<Algorithm 1> Pseudocode of upper-level problem

```
Input: jobs, color requirements, step_size
N←|jobs| // number of jobs
Generate initial solution S
best_value←f(S) // objectivefunction value of S
repeat
    change = 0
    for s=1 to N−1 do
        for e=s+1 to N do
            add S[1] to S[s−1] in order to S'
            add S[s]to S[e] in reverse order to S'
            add S[e+1]to end in order to S'
            if best_value < f(S') then
                clear solution_list
                add S' to solution_list
                best_value←f(S')
                change = 1
            else if best_value = f(S') then
                add S' to solution_list
            endif
        Endfor
    Endfor
    S←S*∈solution_list
// choose a solution randomly
until change = 0
if size(solution_list)=1 then
    return S*∈solution_list
best_value←0
q←1
while (step_size −q>1)
    for x∈solution_list do
        if best_value < f_q(x) then
            clear optimal_list
            add x to optimal_list
            best_value←f_q(x)
        else if best_value = f_q(x) then
            add x to optimal_list
        endif
    Endfor
    if size(optimal_list)=1 then
        break
    Endif
    solution_list←optimal_list
    q+=1
Endwhile
return S*∈optimal_list
// choose a solution randomly
```

4.2 하위 부분 문제 해법

하위 부분 문제는 상위 부분 문제의 결과인 작업 순서를 기반으로 세척시간을 최소화하는 잉크 할당 방식을 결정한다. 상위 부분 문제에서 결정된 작업 순서를 $\hat{x}_{il}$(주문 i가 l번째로 수행되면1, 그렇지 않으면 0)라고 하였을 때, 잉크 할당 및 세척과 그에 따른 세척 시간을 계산하는 하위 부분 문제는 다음과 같은 수리모형으로 나타낼 수 있다.

$$minimize \sum_{c_1 \in C} \sum_{c_2 \in C} \sum_{l \in O} S_{c_1 c_2} z_{c_1 c_2 l} \tag{16}$$

subject to: (5), (6), (7), (8), (10), (11)

$$\sum_{k \in K} y_{ckl} \geq \sum_{i \in I} R_{ic} \hat{x}_{il} \quad \forall c \in C,\ l \in O \tag{17}$$

목적 함수인 (16)은 기존 수리 모형의 목적 함수인 (1)을 적용하였다. 제약식 (17)은 기존 수리모형의 제약식 (4)에서 x_{il}에 $\hat{x}_{il}$을 적용한 형태이다.

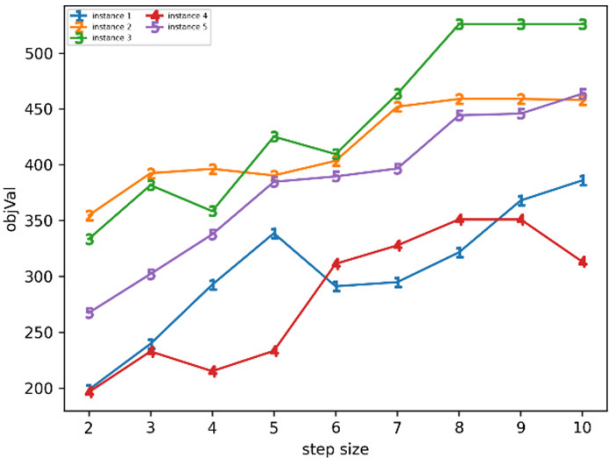
계산에 많은 시간이 소요되어 Gurobi로 결과를 얻기 어려운 상위 부분 문제와 달리, 하위 부분 문제는 비교적 큰 규모의 문제에서까지 최적해를 산출할 수 있었다. 따라서 추가적인 계산 시간 단축보다 개선된 해를 얻기 위해 Gurobi로 해를 탐색하였다.

5. 실험 및 결과 분석

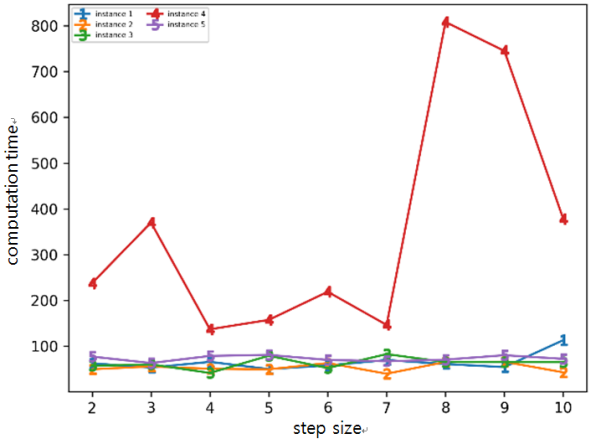
본 연구에서 제시하는 알고리즘은 Python 언어로 구현하였다. 상위 부분 문제의 목적 함수에서 사용하는 파라미터는 파라미터를 변경하는 여러 번의 실험을 통해 적정한 값을 도출한다. 해법의 성능을 검증하기 위해, 데이터를 임의로 생성하여 3장에서 제시한 수리모형을 Gurobi(Ver. 9.1.2)로 계산한 최적해 값과 결과를 비교한다. 실험은 Python 3.8을 이용하여 2.9GHz hexa core CPU, 16GB RAM의 환경에서 수행하였다.

5.1 상위 부분 문제 파라미터 결정 실험

상위 부분 문제 목적 함수의 파라미터 $step_size$를 결정한다. 2~10의 범위에서 분석을 진행한다. 기본 가정 사항은 동일하고, 단일 설비 라인에 6개의 잉크통이 있다고 가정한다. 주문 수는 20, 색상 수는 10으로 하는 임의의 데이터를 무작위로 생성하고, 색상 간 교체에 필요한 세척 시간도 15~30에서 무작위로 생성한다. 이러한 예제를 각각 5개씩 만들어 파라미터를 가변적으로 적용하며 실험한다. 실험은 동일한 예제에 대해 10회 반복하고 목적 값의 평균값으로 경향성을 분석한다.$step_size$를 증가시키며 적용한 실험에서 목적 값은 <Figure 2>와 같이 $step_size$와 함께 증가하는 경향을 보인다. 그러나 계산 시간은 <Figure 3>와 같이 $step_size$에 따른 경향성이 크게 나타나지 않는다. 따라서 이후 실험부터는 $step_size$를 2로 적용하여 실험한다.



<Figure 2> Sensitivity Analysis on Objective Function by the Step Size



<Figure 3> Sensitivity Analysis on Computation Time by the Step Size

5.2 해법 검증 실험

본 연구에서 제시한 해법을 검증하기 위해 3장에서 제시한 MIP 수리모형의 Gurobi로 산출한 최적해와 비교한다.

6개의 잉크통을 보유한 단일 설비 라인으로 가정하고, 주문은 5~150개, 색상은 10~30개인 공정모형을 대상으로 실험하였다. 이러한 공정모형을 48개로 구분하였고, 각 모형마다 임의로 생성한 예제를 대상으로 실험하였다. 문제의 규모가 증가함에 따라 계산 시간 문제로 최적해를 구할 수 없는 경우가 있기 때문에, 계산 시간이 2시간 경과 시에 실험을 중단하고 그 때의 값을 최적해로 간주하였다.

<Table 7>은 두 방법으로 구한 최적해와 계산 시간을 비교하여 성능을 평가하였다(Gap은 MIP 수리모형의 Gurobi 최적해 대비 발견적 기법의 최적해의 비율로 변화율에 해당). 색상이 10개이고 주문이 5~7개로 작은 규모의 문제에서는 MIP 수리모형도 우수한 최적해 값을 도출하였다. 하지만 주문이 7개인 경우에는 계산 시간이 급격하게 증가하였고, 더 큰 규모의 문제에서는 2시간 내에 값을 도출하지 못했다. 색상이 20개, 30개인 예제에서도 각각 주문이 6개, 5개 이상부터 2시간 내에 값을 도출하지 못했다.

본 연구에서 제시한 해법은 계산 시간에 있어 우수한 값을 나타내었다. 주문이 30개인 예제까지 대부분 10분 내에 최적해를 도출하였다. 또한, 같은 구간에서 최적해 값도 색상이 20개, 30개인 예제에서 전반적으로 MIP 수리모형보다 평균 16.23% 감소되어 개선된 값을 나타내었다.

이처럼 본 연구에서 제시하는 해법은 더 큰 규모의 문제까지 평균적으로 실용적인 시간 내에 개선된 최적해를 도출하였다. 하지만 문제의 규모가 더 커지면 기존처럼 계산 시간이 증가하였다. 또한, 알고리즘 단계에서 해를 무작위로 선택하는 방식 때문에 결과의 변동성을 충분히 통제하지 못하였다.

<Table 7> Result Comparison between MIP and Our Solution

$\|I\|$	$\|C\|$	MIP			Our solution				
		Objective value	CPU time (sec)	MIP Gap (%)	Objective value	CPU time (sec)[1]		MIP Gap (%)	Gap[2] (%)
						Upper level	Lower level		
5	10	53*	27.92	0	54	0	0.22	0	1.89
	20	126*	545.11	0	143	0	2.83	0	13.49
	30	157	7200	67.52	163	0	12.84	0	3.82
6	10	101*	1730.01	0	125	0.02	2.2	0	23.76
	20	148	7200	65.54	153	0.02	13.2	0	3.38
	30	196	7200	88.27	199	0.02	29.01	0	1.53
7	10	104*	6312.88	0	148	0.03	2.08	0	42.31
	20	163	7200	77.91	150	0.02	14.14	0	-7.98
	30	218	7200	99.08	242	0.04	56.12	0	11.01
8	10	85	7200	44.70	90	0.05	2.84	0	5.88
	20	261	7200	89.27	245	0.07	14.52	0	-6.13
	30	207	7200	100	204	0.05	81.08	0	-1.45
9	10	91	7200	72.53	98	0.08	0.98	0	7.69
	20	221	7200	96.83	260	0.1	44.34	0	17.65
	30	369	7200	100	337	0.07	102.62	0	-8.67
10	10	121	7200	71.07	201	0.08	11.67	0	66.12
	20	355	7200	99.71	266	0.13	27.65	0	-25.07
	30	478	7200	100	440	0.13	105.28	0	-7.95
12	10	117	7200	94.02	182	0.25	16.05	0	55.56
	20	505	7200	100	465	0.23	33.27	0	-7.92
	30	620	7200	100	521	0.15	104.34	0	-15.97
14	10	134	7200	98.51	161	0.47	19.28	0	20.15
	20	425	7200	100	371	0.46	1097.701	0	-12.71
	30	593	7200	100	536	0.42	241.14	0	-9.61
16	10	270	7200	99.63	338	0.74	24.43	0	25.19
	20	555	7200	100	419	0.68	124.08	0	-24.50
	30	903	7200	100	649	0.69	227.16	0	-28.13
18	10	197	7200	100	249	0.97	33.02	0	26.40
	20	655	7200	100	489	1	303.26	0	-25.34
	30	769	7200	100	513	1.09	7198.91	5.65	-33.29
20	10	280	7200	100	265	1.42	14.43	0	-5.36
	20	852	7200	100	587	1.67	165.97	0	-31.10
	30	1299	7200	100	924	1.53	491.27	0	-28.87
30	10	394	7200	100	545	8.19	225.53	0	38.32
	20	1022	7200	100	704	8.23	7191.77	0.99	-31.12
	30	2474	7200	100	1096	7.67	1669.33	0	-55.70
40	10	472	7200	100	429	27.8	7172.2	5.83	-9.11
	20	2179	7200	100	1074	27.04	7172.96	1.4	-50.71
	30	4566	7200	100	1262	33.03	7166.97	1.66	-72.36
50	10	776	7200	100	667	69.63	7130.37	1.65	-14.05
	20	2910	7200	100	1347	53.34	7146.661	5.42	-53.71
	30	5589	7200	100	1701	53.39	7146.61	3.12	-69.57
100	10	3436	7200	100	1237	1039.7	6160.3	13.74	-64.00
	20	7191	7200	100	2749	949.71	6250.29	67.55	-61.77
	30	12870	7200	100	3278	963.67	6236.33	79.80	-74.53
150	10	4577	7200	100	5443	5574	1626	90.83	18.92
	20	14504	7200	100	6037	5713	1487	87.54	-58.38
	30	TIME**	-	-	13430	5781	1419	95.26	-

Note: 1) (CPU time of upper-level) + (CPU time of lower-level) $\leq$ 7200

2) {(Objective value of MIP) - (Objective value of our solution)} / (Objective value of MIP)

* Optimal solution of a certain case.

** Any feasible solutions cannot be found during 2 hours.

5.3 실험 결과 적용

5.2의 실험을 통해 본 연구에서 제시하는 해법의 성능을 검증하였다. 성능의 우수성도 중요하지만, 실제 현장에서 적용하려면 최적화된 작업 순서와 설비 배치 방식이 중요하다. 따라서 본 연구에서 제시하는 해법으로 얻은 최적화된 작업 순서와 설비 배치 방식을 도식화해야 할 필요가 있다.

7개의 잉크통이 있는 단일 설비 라인에서 30개의 주문을 인쇄하는 공정모형을 가정한다. 사용 가능한 색상은 10가지가 있고 색상 간 교체에 필요한 세척 시간은 15~30의 값을 갖는다.

주문은 사용 가능한 10개의 색상 중 1~7개를 사용할 수 있고, 색상 간 교체에 필요한 세척 시간은 15~30의 값을 갖는다. 주문 정보와 색상별 세척시간은 범위 내에서 무작위로 생성하였고 각각 <Table 8>, <Table 9>과 같다.

<Table 8> Illustration of Color Requirements for Jobs

Job #	Using color
Job 1	Y, K
Job 2	M, Y, K, G, V
Job 3	VM, M, Y, LK, O, G, V
Job 4	C, LK, K
Job 5	C, VM, M, Y, LK, K
Job 6	LC, M, LK, K, O, G, V
Job 7	C, Y, LK, O, V
Job 8	M, Y, K, V
Job 9	C, K, O
Job 10	VM, G, V
Job 11	LC, M, LK, K, O, G
Job 12	LC, VM, Y, LK, K, O, V
Job 13	LC, C, VM, M, Y, LK, K
Job 14	LC, C, M, LK, G, V
Job 15	LC, M, Y, LK, O, G, V
Job 16	C, Y, K, O, G
Job 17	LC, VM, M, LK, K, G
Job 18	LC, VM, M, Y, G
Job 19	LC, C, M, Y, LK, O
Job 20	LC, VM, LK, O, G, V
Job 21	VM, G
Job 22	LC, C, M, Y, LK, G
Job 23	LC, C, Y, K, O, G, V
Job 24	C, VM, M, LK, G
Job 25	LC, C, Y, LK, K
Job 26	M, LK, K, G, V
Job 27	LK, O, V
Job 28	VM, M
Job 29	VM, LK, K
Job 30	Y, LK

<Table 9> Illustration of color-to-color Cleaning Time

Color	LC	C	VM	M	Y	LK	K	O	G	V
LC	0	28	22	24	24	22	19	25	16	17
C	26	0	26	15	18	21	28	21	26	18
VM	26	16	0	17	29	21	22	27	27	18
M	30	21	20	0	18	20	27	26	26	29
Y	16	20	25	19	0	15	20	20	16	30
LK	28	28	17	16	27	0	21	25	22	18
K	21	15	16	27	15	27	0	28	20	26
O	26	18	15	29	17	25	16	0	20	21
G	27	17	25	18	24	15	28	18	0	23
V	15	23	24	29	16	25	28	29	16	0

<Table 10> Illustration of Color Assignment Plan Obtained through Our Solution

Order slot		Number of the ink container						
		1	2	3	4	5	6	7
1	Job 1			Yellow	blacK			
2	Job 8							
3	Job 2							
4	Job 26		Magenta				Green	
5	Job 6	Light Cyan						
6	Job 11			Light blacK				
7	Job 12						Vivid Magenta	Violet
8	Job 3				Magenta	Orange		
9	Job 15							
10	Job 23	Green			Cyan			
11	Job 16							
12	Job 9							
13	Job 4							
14	Job 25	Magenta	Yellow	blacK			Light Cyan	
15	Job 5					Vivid Magenta		
16	Job 13							
17	Job 18				Green			
18	Job 17							
19	Job 24			Cyan				
20	Job 14							
21	Job 22				Orange	Violet		Light blacK
22	Job 19							
23	Job 7							
24	Job 27							
25	Job 20		Green	Vivid Magenta				
26	Job 10							
27	Job 21							
28	Job 28				blacK			
29	Job 29							
30	Job 30				Yellow			
Objective value		377						

<Table 10>은 <Table 8>과 <Table 9>을 입력으로 하여 얻은 결과를 도식화하여 실제 현장에서 적용할 수 있도록 하였다. 7개의 잉크통을 가진 설비 라인은 30개의 주문을 인쇄하기 위해 30개의 작업 순서 Slot을 갖는다. <Table 10>의 'Order slot'이 작업 순서 Slot에 해당하고, 주문을 최적화된 순서대로 배치하였다. 'Number of the ink container'는 설비 라인의 잉크통에 해당하고, 간트 차트와 유사한 형태로 구성하였다. <Table 10>의 경우, 'Job 1 - 8 - 2 - 26 - 6 -11 - 12 - 3 - 15 - 26 - 16 - 9 - 4 - 25 - 5 - 13 - 18 - 17 - 24 - 14 - 22 - 19 - 7 - 27 - 20 - 10 - 21 - 28 - 29 - 30'의 순서로 작업하였다. 1번 잉크통은 4번째 작업까지 사용하지 않고, 5~7번째에 LC를, 8~13번째에 G를, 14~30번째에 M을 담는다. 2번 잉크통은 1번째 작업에서 사용하지 않고, 2~6번째에 M을, 7~24번째에 Y를, 25~30번째에 G를 담는다. 나머지 잉크통도 동일한 방식으로 해석하여 잉크를 배치한다. 이와 같이 설비를 배치하였을 때 발생하는 세척 시간의 총합은 377이다.

6. 결론

본 연구에서는 작업 순서 및 잉크 할당에 따라 준비 시간이 변화하는 오프셋 인쇄 공정의 스케줄링 최적화 문제를 다루었다. 기존 연구에서 다뤄지지 않았기 때문에 새로운 유형의 스케줄링 문제를 정의하였고, 혼합정수계획법을 이용한 수리모형을 제시하였다. 제시한 수리모형은 문제의 규모가 증가함에 따라 계산 시간이 증가하여 최적해 도출이 어려웠고, 이에 본 연구에서는 문제를 분할하고 2 - OPT 알고리즘을 적용한 발견적 기법을 해법으로 제시하였다. 제시한 해법의 성능은 원래의 수리모형과 비교하여 검증하였다. 실험 결과, 제시한 해법을 적용하여 기존 수리모형보다 전체 세척 시간이 평균 16.23% 가량 감소하였고, 더 큰 규모의 문제까지 실용적인 시간 내에 값을 도출하였다. 또한, 현장에서의 적용을 고려하여 최적화된 스케줄을 도식화하여 나타내었다.

본 연구에서는 빠르게 의사결정이 이뤄져야 하는 현장에서의 일정 계획 수립 문제를 자동화하고자 하였다. 적용 가능성을 고려하여, 최적해를 보장하지는 않지만 짧은 시간 내에 개선된 결과를 얻을 수 있는 발견적 기법을 제시하였다. 그러나 본 연구에서 개발한 알고리즘은 무작위로 선택하는 방식으로 인해 결과가 변동적인 한계가 있다. 또한, 다중 설비 라인과 함께 100여 개 이상의 주문이 있고, 무수히 많은 색상을 사용 가능하여 자주 사용되는 색상과 그렇지 않은 색상으로 나뉘며, 납기일이 있어 주/월 단위의 일정 계획이 필요한 실제 데이터와 달리, 이를 기반으로 일부 제약을 생략하여 생성한 모형 데이터는 그 규모가 다르다. 따라서 이러한 본 연구의 한계를 바탕으로 향후에는 다중 설비 도입 여부 및 납기일 등의 제약을 고려하는 연구로의 확장과 알고리즘의 개선이 필요하다.

Acknowledgment

This research was a part of the project titled 'forest science-technology R&D program (2021383A00-2123-0101)', funded by the Korea Forestry Promotion Institute (Korea National Arboretum), Korea.

This research was a part of the R&D reserve project, funded by the Korea Institute of Industrial Technology, Korea.

References

[1] Artigues, C., Feillet, D., A branch and bound method for the job-shop problem with sequence-dependent setup times, *Annals of Operations Research*, 2008, Vol. 159, No. 1, pp. 135-159.

[2] Bellman, R., Dynamic Programming Treatment of the Travelling Salesman Problem, *Journal of the ACM*, 1962, Vol. 9, No. 1, pp. 61-63.

[3] Garfinkel, R.S., Technical Note-On Partitioning the Feasible Set in a Branch-and-Bound Algorithm for the Asymmetric Traveling-Salesman Problem, *Operations Research*, 1973, Vol. 21, No. 1, pp. 340-343.

[4] Glover, F., Tabu Search - Part Ⅰ ORSA, *Journal on Computing*, 1989, Vol. 1, No. 3, pp. 190-206.

[5] Glover, F., Tabu Search - Part Ⅱ ORSA, *Journal on Computing*, 1990, Vol. 2, No. 1, pp. 4-32.

[6] Gupta, J.N.D., Flowshop Schedules With Sequence Dependent Setup TIMES, *Journal of the Operations Research Society of Japan*, 1986, Vol. 29, No. 3, pp. 206-219.

[7] Joo, C.M., An Improved Ant Colony System for Parallel-Machine Scheduling Problem with Job Release Times and Sequence-Dependent Setup Times, *Journal of the Korean Institute of Industrial Engineers*, 2009, Vol. 35, No. 4, pp. 218-225.

[8] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., Optimization by Simulated Annealing, *Science*, 1983, Vol. 220, No.4598, pp. 671-680.

[9] Laporte, G., The traveling salesman problem: An overview of exact and approximate algorithms,

European Journal of Operational Research, 1992, Vol. 59, No. 2, pp. 231-247.

[10] Lee, J.Y., Jeong, B.J., Heuristic Algorithms for a Two-Machine Flowshop Scheduling Problem with Urgent Jobs and Sequence-Dependent Setup Times, *Korean Management Science Review*, 2020, Vol. 37, No. 1, pp. 47-60.

[11] Lin, S., Kernighan, B.W., An effective heuristic algorithm for the traveling-salesman problem, *Operations Research*, 1973, Vol. 21, No. 2, pp. 498-516.

[12] Nadesri, B., Ghomi, S.M.T.F., Aminnayeri, M., A high performing metaheuristic for job shop scheduling with sequence-dependent setup times, *Applied Soft Computing*, 2010, Vol. 10, No. 3, pp. 703-710.

[13] Rosenkrantz, D.J., Stearns, R.E., Lewis II, P.M., An analysis of several heuristics for the traveling salesman problem, *SIAM Journal on Computing*, 1977, Vol. 6, No. 3, pp. 563-581.

[14] Shin, H.J., Kim, S.S., Ko, K.S., A Tabu Search Algorithm for Single Machine Scheduling Problem with Job Release Times and Sequence-dependent Setup Times, *Journal of the Korean Institute of Industrial Engineers*, 2001, Vol. 27, No. 2, pp. 158-168.

[15] Vela, C.R., Varela, R., González, M.A., Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times, *Journal of Heuristics*, 2010, Vol. 16, No. 1, pp. 139-165.

[16] WIKIPEDIA, https://en.wikipedia.org/wiki/2-opt.

[17] Yoo, W.S., Kim, S.J., Kim, K.H., Real-Time Scheduling Scheme based on Reinforcement Learning Considering Minimizing Setup Cost, *The Journal of Society for e-Business Studies*, Vol. 25, No. 2, pp. 15-27.

[18] Yoo, W.S., Seo, J.H., Lee, D.H., Kim, D.H., Kim, K.H., Scheduling Generation Model on Parallel Machines with Due Date and Setup Cost Based on Deep Learning, *The Journal of Society for e-Business Studies*, Vol. 24, No. 3, pp. 99-110.

[19] Yu, S.Y., Scheduling of Production Process with Setup Cost depending Job Sequence, *Management & Information Systems Review*, 2015, Vol. 34, No.2, pp. 67-78.

ORCID

Jae Kyeong Moon | https://orcid.org/0000-0002-4889-9620
Hyun Seop Uhm | https://orcid.org/0000-0003-1185-4151
Hyun Chul Tae | https://orcid.org/0000-0002-2277-0722