Journal of the Korean Institute of Industrial Engineers
Vol. 42, No. 3, pp. 173-181, June 2016.
ISSN 1225-0988 | EISSN 2234-6457
http://dx.doi.org/10.7232/JKIIE.2016.42.3.173

<Application Research>

단일 호이스트 생산시스템에서 다양한 주문을 처리하기 위한 분지한계 기반의 휴리스틱 일정계획

이정구[1] · 김정배[2] · 고시근[3†]

[1]대한항공(주) 항공우주사업본부 품질경영부/ [2]부경대학교 기술경영전문대학원/
[3]부경대학교 시스템경영공학부

Branch-and-Bound Based Heuristic Scheduling for the Single-Hoist and Multiple-Products Production System

Jungkoo Lee[1] · Jeongbae Kim[2] · Shiegheun Koh[3]

[1]Quality Management Department, Aerospace Business Division, Korean Air Co.
[2]Graduate School of MOT, Pukyong National University
[3]Department of Systems Management and Engineering, Pukyong National University

This paper deals with the single-hoist and multiple-products scheduling problem. Although a mixed integer linear programming model for the problem was developed earlier, a branch-and-bound based heuristic algorithm is proposed in this paper to solve the big-size problems in real situation. The algorithm is capable of handling problems incorporating different product types, jobs in the process, and tank capacities. Using a small example problem the procedure of the heuristic algorithm is explained. To assess the performance of the heuristic we generate a bigger example problem and compare the results of the algorithm proposed in this paper with the optimal solutions derived from the mathematical model of earlier research. The comparison shows that the heuristic has very good performance and the computation time is sufficiently short to use the algorithm in real situation.

Keywords: Hoist Scheduling, Heuristic, Branch-and-Bound

1. 서 론

호이스트(Hoist)를 사용한 다단계 생산시스템은 전기 도금이나 폴리머 코팅과 같이 표면처리가 필요한 다양한 산업분야에서 널리 찾아볼 수 있다. 이 시스템에서 각 작업단계(Workstation)는 기계적 처리를 행하는 설비가 아니라 화학 처리용 액체가 채워진 일종의 탱크이다. 각 제품의 생산과정은 그 제품에 필요한 탱크들을 순서대로 방문하여 각 탱크별로 정해진 시간동안 제품을 그 탱크의 액체에 담그는 공정으로 이루어진다.

이러한 공정을 액침(Immersion)이라고 부르며, 생산라인의 각 탱크는 그 탱크 고유의 액침 공정을 수행하게 된다. 한 제품이 방문해야 하는 탱크의 순서를 액침순서라고 하는데, 단일 종류의 품목이 생산되는 라인이라면 모든 제품들이 동일한 액침순서에 따라 생산되겠지만, 현실적으로는 하나의 생산라인에서 다양한 종류의 품목에 대한 처리가 별개의 액침순서에 따라 동시에 진행되는 것이 일반적인 상황이다.

다양한 품목의 제품이 호이스트 한 대로 처리되는 생산라인의 한 예를 <Figure 1>에서 볼 수 있다. 그림의 좌측 부분은 투

이 논문은 부경대학교 자율창의학술연구비(2016년)에 의하여 연구되었음.
† 연락저자 : 고시근 교수, 48513 부산광역시 남구 용소로45, 부경대학교 시스템경영공학부, Tel : 051-629-6484, Fax : 051-629-6478, E-mail : sgkoh@pknu.ac.kr
2015년 12월 7일 접수; 2016년 3월 23일 수정본 접수; 2016년 4월 26일 게재 확정.

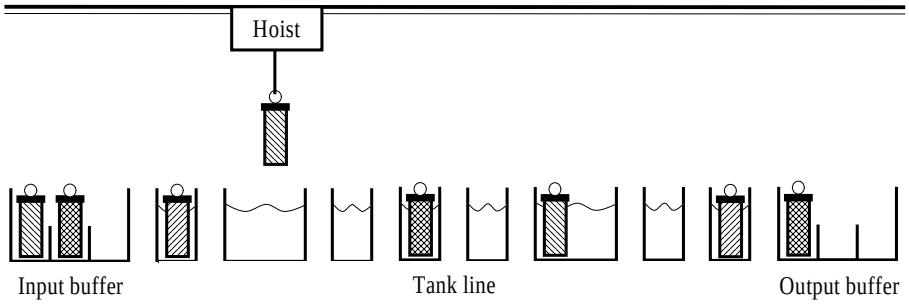

Figure 1. An Example of the Single Hoist Production Line

입버퍼(Input buffer)로서 이 생산라인에 도착한 처리대상 제품이 라인에 투입되기 전에 임시로 대기하는 공간이다. 반면 우측 부분은 완료버퍼(Output buffer)로서 처리를 마친 제품이 생산시스템을 떠나기 전에 대기할 수 있는 공간이다. 이 그림에서는 두 버퍼와 탱크 라인이 일직선으로 구성되어 있지만 경우에 따라서 일직선이 아니어도 별 문제는 없을 것이다. 실제로 본 연구의 계기가 된 기업의 생산시스템은 30개의 탱크가 U자형으로 배치되어 있다. 각 탱크에서의 처리시간은 탱크에 따라 정해지기도 하지만 탱크 뿐 아니라 처리대상 제품에 따라서도 달라질 수 있는 상황이 보다 일반적인 환경이라고 할 수 있다. 이 처리시간은 대부분 시간창(Time window)으로 표시함으로써, 각 제품이 그 탱크에 머물러야 하는 최소 시간과 머물 수 있는 최대 시간을 나타내는데, 일부 탱크의 경우에는 최소시간만 주어지고 최대 시간은 한계가 없어 매우 긴 시간을 머물 수도 있다.

이 생산시스템에서 호이스트의 역할은 물자 운반으로서, 투입버퍼에 있는 처리대상 품목을 가져와서 그 품목에 고유한 처리순서에 따라 탱크를 방문한 다음 마지막으로 완료버퍼로 운반하는 것이다. 호이스트가 어떤 제품을 필요한 탱크로 운반한 다음 그 제품의 액침공정이 수행될 때 호이스트는 그 탱크 위에서 공정이 끝날 때까지 대기할 수도 있고 다른 제품의 운반을 위해 다른 위치로 이동할 수도 있다. 또한 <Figure 1>에서 볼 수 있듯이 생산라인 내부에는 임시저장 공간이 없으므로 한 탱크에서 들어 올려진 제품은 다음 순서의 탱크로 즉시 이동하여야 한다. 따라서 후속 탱크에 다른 제품이 현재 처리중이라면 그 이동은 불가능하게 된다. 이런 절차에 따라 투입버퍼 및 각 탱크에 존재하는 제품들이 모두 완료버퍼로 이동하면 처리가 완료되었다고 본다.

호이스트에 의해 수행되는 이동작업은 네 가지 단위 작업으로 구성되는데, 첫 번째는 하나의 제품을 내려놓은 빈 호이스트가 이동이 필요한 제품을 갖고 있는 탱크로 이동하는 것이다. 그 탱크에 도착한 후에는 대상 제품을 호이스트에 장착하는 작업이 이루어지고, 제품을 장착한 상태에서 그 제품의 다음 목적지(탱크 혹은 완료버퍼)로 이동하게 된다. 목적지에 도착한 이후에는 마지막으로 제품을 호이스트와 분리하는 작업이 이루어진다. 호이스트의 이동시간은 물품이 장착된 상태인지 그렇지 않은지에 따라 달라지는 것이 보통이며, 각 경우 이동시간은 이동거리에만 관련이 있다.

이러한 생산시스템에서 처리대상 제품을 주문(Job)이라고 간주하면, 각 주문에 대해 필요한 공정은 탱크들의 부분순서(Partial sequence)로 정의된다. 예를 들어 어떤 주문의 처리를 위한 공정이 2-4-5-6과 같이 주어졌다면, 이 주문은 2번 탱크, 4번 탱크, 5번 탱크, 6번 탱크를 순서대로 거쳐야 함을 의미하는 것이다. 동일 품목에 대한 주문이 두 개 이상일 수도 있으며 이 경우 각 주문은 동일한 공정순서를 갖게 된다. 각 탱크는 동시처리가 가능한 주문의 수로 표시되는 용량(Capacity)을 갖고 있다. <Figure 1>의 경우 8개 탱크 중에서 2번째와 6번째 탱크는 처리용량이 2이고 나머지는 모두 1인 것을 발견할 수 있다. 한편 투입 및 완료버퍼의 경우 그림에서는 편의상 용량을 3으로 표시하였지만 제한이 없다고 가정한다.

시스템의 효과적인 운영을 위해서는 주문들의 처리 및 운송을 동시에 고려하는 호이스트 일정계획 문제가 필요해지는데, Kumar(1994)에 의하면 이 호이스트 일정계획은 시스템의 생산성을 결정하는 가장 중요한 요인이며, 호이스트 일정계획에 따라 평균 주문 대기시간이 20%까지 줄어들 수 있다고 한다. 따라서 본 연구에서는 <Figure 1>과 같은 형태의 생산시스템을 효과적으로 운영하기 위한 호이스트 일정계획 방안을 탐색하고자 한다.

본 연구의 나머지 부분은 다음과 같이 구성된다. 다음 절에서는 관련 선행연구에 대해 소개하고, 제 3장에서 대상문제에 대한 명확한 정의를 내린 다음, 제 4장에서는 본 연구에서 제안하고자 하는 휴리스틱 방법론에 대해 소개한다. 제 5장에서는 제안된 해법의 성능을 평가하기 위한 수치실험 결과를 소개하고, 마지막으로 제 6장에서는 논문을 요약하고 추후 연구과제를 제시한다.

2. 선행 연구 조사

관련 연구는 Phillips and Unger(1976)가 한 대의 호이스트가 설치된 단일 품목의 제품을 생산하는 전기도금 생산라인을 분석하면서 시작되었는데, 이 연구에서는 호이스트가 정해진 이동순서를 계속 반복하도록 계획되었다. 이렇게 계속 반복되는 고정된 이동순서를 사이클(cycle)이라고 하며, 한 사이클 동안

에는 한 개의 주문이 시스템에 들어오고 한 개의 주문이 시스템을 떠나게 된다. 이러한 사이클 방식으로 계획을 작성하면 문제의 난이도를 낮출 수 있으나 문제의 현실성 또한 낮아질 수밖에 없다.

사이클 방식의 호이스트 일정계획에 대해서는 많은 연구들이 수행되었다. 우선 Shapiro and Nuttle(1988)은 분지한계법을 이용한 해법을 개발하여 작은 규모의 문제에 대한 최적해를 구할 수 있도록 하였다. Lei and Wang(1989)은 호이스트 일정계획 문제가 모든 주문이 동일한 품목으로 된 가장 단순한 문제의 경우에도 NP-hard라는 것을 보였고, 그 후 그들(Lei and Wang, 1994)은 한 사이클에 두 개 이상의 주문이 시스템에 들어올 수 있는 확장된 사이클 문제에 대한 분지한계법을 개발하였다. Chen *et al*.(1998)은 주문들을 탱크에 배정하는 첫 번째 단계와 호이스트의 이동순서를 결정하는 두 번째 단계로 이루어진 2단계 분지한계법을 제안하였다. 한편 Lim(1997)은 유전알고리즘을 사용해 현실적인 규모의 문제를 해결할 수 있는 방안을 제안하였다. Xu and Huang(2004)은 생산성과 더불어 환경문제까지 고려한 연구를 수행하였으며, Kuntay *et al*.(2006)은 같은 문제를 해결하기 위해 두 단계 최적화 알고리즘을 개발하였다. 최근에는 Liu *et al*.(2012)이 생산성 최대화, 에너지 절약, 물 사용량 최소화 등의 세 가지 목적함수를 갖는 문제를 다루기도 하였다.

그런데 실제로 호이스트를 사용하는 생산라인은 단일품목에 대한 사이클 계획이 가정하는 상황에 비해서 훨씬 복잡한 것이 현실이다. 단일 호이스트가 다양한 품목을 처리하는 생산라인 문제에 대해서는 Hindi and Fleszar(2004)가 CSP(Constraint Satisfaction Problem)를 응용한 해법을 개발한 다음 그 해법이 기존의 해법에 비해 우수한 해를 제공한다고 주장하였다. 또한 Paul *et al*.(2007)도 같은 문제에 대해 ATW(Adaptive Time Window) 휴리스틱을 개발하고 그 우수성을 주장하였다. 하지만 이런 연구들은 계획 수립시점에 생산라인에서 이미 작업중인 품목들에 대한 고려가 없었다. 일반적으로 생산시스템에는 다양한 처리대상 주문들이 랜덤하게 도착하므로 호이스트 일정계획 문제에서도 이러한 동적 특성이 반영되어야 한다. Yin and Yih(1992)와 Yih(1994)는 이러한 상황에서 이미 생산라인에 투입된 주문들의 일정은 고정하고 새로이 도착하여 아직 생산라인에 투입되지 않은 주문들만 대상으로 일정계획을 작성하였다.

계획 시점에 생산라인에서 처리중인 주문들을 고려한 상황은 Zhao *et al*.(2013)의 연구에서 볼 수 있다. 그들이 RDHS(Real-time Dynamic Hoist Scheduling)라고 부른 그 문제는 (1) 공정 내용이 서로 다른 다양한 형태의 주문들이 (2) 예고없이 시스템에 도착하여 현재 생산라인 내부에서 진행중인 주문들과 함께 재계획되어야 하는 상황을 다룬다. 이 재계획에서는 호이스트의 현재 위치와 처리중인 주문들의 실시간 상황이 고려됨으로써, 계획결과가 현재의 시스템 상황과 매끄럽게 연결되어야 한다. 또한 (3) 일부 탱크의 경우 둘 이상의 주문을 동시에 처리할 수 있다는 처리능력(Tank capacity) 조건을 반영하였다. 이러한 형태의 일반적인 문제를 처음 다룬 그들은 단일 호이스트 상황의 RDHS 문제에 대해 혼합정수선형계획법(MILP : Mixed Integer Linear Programming) 모형을 개발하였고, 이 모형을 사용하면 빠른 시간 안에 확실한 해를 보장할 수 있다고 하였다. 그러나 이후 그 모형의 일부 제약조건을 Tian *et al*.(2013)이 개선하여 모형의 처리시간을 단축하였음에도 불구하고, 저자의 경험에 의하면 이 모형은 현실적인 규모(10개 이상의 탱크와 10개 이상의 주문)의 문제에 대해서는 적정시간 내에 해를 제시해주지 못하였다. 따라서 본 연구에서는 Zhao *et al*.(2013)과 유사한 문제에 대해 현실적인 규모의 경우에도 빠른 시간 안에 양호한 품질의 해를 제공해주는 휴리스틱 방법을 개발하고자 한다.

3. 문제 정의

앞에서 설명하였듯이 본 연구에서는 Zhao *et al*.(2013)과 매우 유사한 문제를 다룬다. 문제를 명확화하기 위해 필요한 핵심 가정들을 정리하면 다음과 같다.

(1) 처리대상 주문의 수는 유한하며, 이 중 일부는 이미 생산라인에 투입되어 공정이 진행 중이고 나머지는 투입버퍼에서 대기 중이다. 각 주문을 처리하기 위한 공정순서와 탱크들은 알려져 있다.
(2) 각 주문들의 공정순서와 탱크는 서로 다를 수 있다.
(3) 각 주문이 각 탱크에서 처리되는 시간은 주어진 최소치와 최대치 사이에 있어야 한다.
(4) 각 탱크(혹은 버퍼) 사이의 물자이동은 한 대의 호이스트에 의해 수행된다.
(5) 문제의 목적은 시스템에 존재하는 처리대상 주문을 모두 완료(완료버퍼로 이동)하기까지의 시간(Makespan)을 최소화하는 것이다.
(6) 호이스트는 한 번에 하나의 주문만을 처리하며 일단 하나의 목적지를 향해 움직이기 시작하면 중간에 멈추지 않는다.
(7) 하나의 탱크에서 일단 하나의 액침 작업이 시작되면 그 작업이 종료되기 전에는 중단될 수 없다.
(8) 각 탱크는 고장이 없고 액침공정의 셋업시간은 따로 고려하지 않는다.
(9) 각 탱크(혹은 버퍼) 사이의 호이스트 이동시간은 빈 상태와 장착상태의 두 가지 형태로 나누어지며 그 값은 알려져 있다.
(10) 각 탱크는 동시에 처리가능한 주문수로 표시되는 용량을 가지며, 두 버퍼는 그 값이 무한대이다.
(11) 일정계획이 수행되는 시점은 각 주문이 도착한 직후라고 가정한다.

위의 가정들을 바탕으로 문제의 구조를 설명하기 위해서는 문제 해결에 필요한 데이터를 정의하는 것이 좋을 것으로 생

각된다. 아래에서 그 데이터 및 관련 기호를 설명한다. 기호들은 Zhao *et al.*(2013)의 기호를 참조하였지만 대부분은 수정하여 사용하였다.

(1) 탱크의 수 : 투입버퍼와 완료버퍼를 포함한 탱크의 수를 NT로 표시한다. 실제 화학처리용 탱크가 10개라면 NT는 12가 된다. 앞으로는 탱크라고 하면 두 버퍼를 포함한 개념으로 이해하기로 한다.
(2) 주문의 수 : 계획 시점에 투입버퍼에 대기 중이거나 탱크에서 처리중인 주문의 수는 NJ로 표시한다. 계획시점 이전에 이미 완료버퍼에 도착한 주문은 제외한다.
(3) 호이스트 현 위치 : 계획 시점의 호이스트 위치는 HP로 표시하며 그 값은 탱크 번호(1, 2, ⋯, NT) 중 하나이다.
(4) 탱크의 능력(capacity) : 각 탱크의 동시 처리능력은 C_i (i = 1, 2, ⋯, NT)로 표시하며, 앞선 가정의 의해 $C_1 = C_{NT} = \infty$ 이다.
(5) 각 주문의 현 위치 : 각 주문의 현 위치는 CP_j (j = 1, 2, ⋯, NJ)로 표시하며 그 값은 탱크 번호(1, 2, ⋯, NT) 중 하나이다.
(6) 주문별 경과시간 : 계획 시점 현재 각 주문이 현 위치에 얼마나 머물렀는지를 $ELAP_j$(j = 1, 2, ⋯, NJ) 로 표시한다.
(7) 주문별 공정순서 : 각 주문별 공정흐름은 NXT_{ij} (i = 1, 2, ⋯, NT, j = 1, 2, ⋯, NJ)로 표시하며 그 값은 다음에 방문해야 하는 탱크번호이다. 만약 그 값이 0이라면 그 주문의 공정순서에는 해당 탱크가 들어있지 않음을 뜻한다. 예를 들어 NXT_{23}의 값이 0이라면 주문 3은 2번 탱크를 방문하지 않는다는 의미이고, 그 값이 4라면 주문 3이 2번 탱크를 방문한 다음에 4번 탱크를 방문해야 한다는 것을 의미한다. 한편 $NXT_{NT,j}$ 값은 모든 주문이 완료버퍼를 방문하지만 다음 탱크가 없으므로 0 값을 가진다.
(8) 처리시간 : 버퍼를 제외하고 NXT_{ij} 값이 0이 아닌 (i, j) 쌍에 대해서는 처리시간이 정의되어야 한다. 처리시간은 하한 TL_{ij}와 상한 TU_{ij}로 정의된다. Zhao *et al.*(2013)은 처리시간이 주문과는 독립적으로 각 탱크에 따라서 정해진다고 가정하였으나 본 연구에서는 좀 더 일반적으로 주문에 따라서도 변할 수 있음을 가정하였다. 실제로 본 연구가 대상으로 하는 현장의 경우 동일한 탱크임에도 주문에 따라 처리시간이 다른 현상을 볼 수 있었다.
(9) 호이스트 이동시간 : 각 탱크 사이의 호이스트 이동시간은 호이스트에 물품이 장착된 경우에는 ML_{ik} (i = 1, 2, ⋯, NT, k = 1, 2, ⋯, NT)이고 빈 상태에서는 ME_{ik} (i = 1, 2, ⋯, NT, k = 1, 2, ⋯, NT)이다.

다음에는 문제의 이해를 돕고 다음 절의 해법을 설명하기 위한 간단한 예제를 소개한다. 이 예제는 Zhao *et al.*(2013)의 예제를 기초로 하고 처리시간 및 호이스트 이동시간 등의 수치들은 본 연구의 사례기업 상황을 반영하여 현실적으로 수정한 것이다.

NT = 8, NJ = 5, HP = 1, C_i = (∞, 1, 1, 1, 2, 1, 1, ∞),
CP_j = (7, 6, 4, 3, 1), $ELAP_j$ = (33, 3, 1, 4, 12),

$$NXT_{ij} = \begin{bmatrix} 0&0&0&0&2 \\ 0&0&0&0&3 \\ 0&0&0&4&4 \\ 0&0&5&5&5 \\ 0&0&6&6&8 \\ 0&7&7&7&0 \\ 8&8&8&8&0 \\ 0&0&0&0&0 \end{bmatrix},\ TL_{ij} = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&3 \\ 0&0&0&15&15 \\ 0&0&1&1&1 \\ 0&0&40&40&40 \\ 0&5&5&5&0 \\ 30&30&30&30&0 \\ 0&0&0&0&0 \end{bmatrix},$$

$$TU_{ij} = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&10 \\ 0&0&0&25&25 \\ 0&0&5&5&5 \\ 0&0&60&60&60 \\ 0&15&15&15&0 \\ 50&50&50&50&0 \\ 0&0&0&0&0 \end{bmatrix},$$

$$ME_{ik} = \begin{bmatrix} 0&0.3&0.6&0.9&1.2&1.5&1.8&2.1 \\ 0.3&0&0.3&0.6&0.9&1.2&1.5&1.8 \\ 0.6&0.3&0&0.3&0.6&0.9&1.2&1.5 \\ 0.9&0.6&0.3&0&0.3&0.6&0.9&1.2 \\ 1.2&0.9&0.6&0.3&0&0.3&0.6&0.9 \\ 1.5&1.2&0.9&0.6&0.3&0&0.3&0.6 \\ 1.8&1.5&1.2&0.9&0.6&0.3&0&0.3 \\ 2.1&1.8&1.5&1.2&0.9&0.6&0.3&0 \end{bmatrix},$$

$$ML_{ik} = \begin{bmatrix} 0&2.3&2.6&2.9&3.2&3.5&3.8&4.1 \\ 2.3&0&2.3&2.6&2.9&3.2&3.5&3.8 \\ 2.6&2.3&0&2.3&2.6&2.9&3.2&3.5 \\ 2.9&2.6&2.3&0&2.3&2.6&2.9&3.2 \\ 3.2&2.9&2.6&2.3&0&2.3&2.6&2.9 \\ 3.5&3.2&2.9&2.6&2.3&0&2.3&2.6 \\ 3.8&3.5&3.2&2.9&2.6&2.3&0&2.3 \\ 4.1&3.8&3.5&3.2&2.9&2.6&2.3&0 \end{bmatrix}.$$

이 예제의 데이터를 설명하면 다음과 같다. 대상 시스템은 두 개의 버퍼와 6개의 탱크로 이루어져 있고, 계획시점 현재 5개의 주문이 시스템에 존재한다. 호이스트는 현재 1번 탱크(즉, 투입버퍼) 위에 위치하고 있다. 탱크의 용량은 두 버퍼의 경우 무한하고, 5번 탱크(투입버퍼를 제외하면 실제로는 4번 탱크)는 2개, 나머지는 모두 1개이다. 각 주문의 현 위치를 보면 1, 2, 3, 4번 주문은 공정에 투입되어 현재 7, 6, 4, 3번 탱크에서 각각 처리 중이고, 5번 주문은 아직 공정에 투입되지 않고 투입버퍼에 대기 중이다. 각 주문이 현재의 탱크에 도착한지는 각각 33분, 3분, 1분 4분, 12분이 경과하였다. NXT 행렬을 보면 각 주문의 남은 공정들을 알 수 있다. 예를 들어 3번 주문은 현재 위치한 4번 탱크의 작업이 끝나면 5, 6, 7번 탱크를 거쳐 마지막으로 완료버퍼인 8번 탱크로 이동하게 된다. TL 및 TU 행렬을 보면 3번 주문의 5번 탱크 작업은 40분에서 60분 사이에 이루어져야 함을 알 수 있다. 각 탱크에서의 처리시간은 주문에 따라 달라질 수 있지만 이 예제에서는 Zhao *et al.*(2013)의 모형에 적용하기 위해 한 탱크에서의 처리시간을 모든 주문에 대해 동일하게 정하였다. 마지막으로 호이스트 이동시간은 빈 이동일 경우 바로 옆 탱크로 이동하는데 18초(0.3분)가 걸리고, 물품이 장착된 이동의 경우에는 장착 및 분리 작

업을 위해 2분이 추가되어 나타난다.

이 문제의 해는 NXT 행렬에서 0이 아닌 값을 갖는 17개의 요소들에 대한 처리순서로 표시할 수 있다. 요소들을 살펴보면 1번 주문이 1개, 2번 주문은 2개, …, 마지막으로 5번 주문은 5개가 포함되어 있다. 따라서 가능(feasible) 및 불가능(infeasible) 해를 포함하여 해의 총 수는 이 17개의 주문번호 (1, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5)를 일렬로 세우는 방법의 수인 $\frac{17!}{1!2!4!5!5!}$ 개가 된다. 즉, 문제의 난이도는 NXT 행렬에 0이 아닌 요소가 몇 개 있느냐에 따라 결정되는 것이다.

최적화 소프트웨어 LINGO를 사용해 Zhao *et al*.(2013)의 모형을 이 예제에 적용하여 최적해를 구해 보았다. 4GB RAM과 2.60GHz G620 CPU가 장착된 PC에서 최적해를 구하는데 소요된 시간은 0.01초 미만으로 이 예제의 경우 매우 빠른 시간에 해를 구할 수 있었으며, 그 결과 최적의 주문 처리순서는 (3, 1, 2, 4, 4, 5, 2, 3, 5, 3, 5, 5, 4, 3, 4, 5, 4)이고 그 때의 완료시간(Makespan)은 120.3분이었다.

저자들은 Zhao *et al*.(2013)의 모형이 현실 문제에 어느 정도 적용이 가능한지를 조사하기 위해 NT가 12인 경우(두 개의 버퍼와 10개의 화학처리용 탱크)에 대한 예제를 만들어 LINGO를 사용해 풀어 보았다. 그 결과 주문의 수가 7개인 경우는 8초만에 최적해를 구할 수 있었고, 주문의 수가 8개인 경우에는 3분 9초, 9개인 경우는 1시간 55분 17초가 소요되었다. 주문의 수가 10개인 경우에는 시스템을 강제로 종료하기까지 5시간 동안 해를 구하지 못하였다. 따라서 앞서 언급하였던 본 연구의 사례기업(탱크의 수가 30개)에는 적용할 수가 없었고, 해의 품질이 어느 정도 저하되더라도 현실적인 시간 안에 해를 구할 수 있는 휴리스틱 방법론의 개발이 필요하게 되었다.

4. 휴리스틱 방법론

본 연구에서 제안하는 휴리스틱은 분지한계법에 기초를 두고 있다. 저자들은 본 연구의 사례기업에 Zhao *et al*.(2013)의 모형이 적용될 수 없음을 확인한 다음 유전 알고리즘의 적용 가능성을 타진하였다. 앞 절에서 제시한 예제의 경우 17개의 주문번호에 대한 최적 순서를 결정하는 문제이므로 일반적인 순서결정 문제와 유사하게 유전 알고리즘을 적용해보았으나 랜덤으로 생성한 해가 실행가능해일 확률이 1% 미만으로 너무 낮아서 유전 알고리즘으로는 해를 구할 수가 없었다.

두 번째로 시도한 방법은 분지한계법을 사용한 최적해 탐색이었다. 그러나 이 방법도 문제의 크기가 커지면 Zhao *et al*.(2013)의 모형보다 오히려 더 긴 시간이 소요되었다. 따라서 이 분지한계법을 간략화하여 분지의 수에 제한을 둠으로써 소요시간을 단축하고 비교적 우수한 해를 찾으려는 시도를 하게 되었다. 아래에서 이 절차를 '분지 정책'과 '하한 계산'의 두 단계로 나누어 설명하도록 한다.

4.1 분지 정책 (Branching Policy)

본 휴리스틱의 핵심은 현재의 부분일정(Partial schedule)에 새로운 일정을 추가할 때 필요한 가지(Branch)의 수를 줄이는 것이다. 앞 절의 예를 사용해 설명하도록 한다. 현재 고려대상 부분일정이 (5, 3, 1)라고 하자. 이 부분일정의 의미는 가장 먼저 5번 주문을 1번 탱크(투입 버퍼)에서 2번 탱크로 옮기고, 다음으로 3번 주문을 4번 탱크에서 5번 탱크로 옮긴 다음, 1번 주문을 7번 탱크에서 8번 탱크로 (주문 처리 완료) 옮기는 순으로 첫 3개의 작업이 이루어진다는 것이다. 이 부분일정에 다음 작업을 추가하기 위해서는 5개 주문 전체에 대해 가능성을 타진해야 하므로 5개의 가지가 필요해진다. 그러나 본 연구에서는 문제에 대한 약간의 관찰을 통해 가능성이 적은 가지를 생략하고자 한다. 그 절차는 다음과 같다.

(1) 이미 완료한 주문은 분지 대상에서 제외한다. 예를 들어 부분일정 (5, 3, 1)의 경우 1번 주문은 이미 완료버퍼로 이동하여 처리가 종료되었으므로 분지 대상에서 제외하여야 한다는 것이다.

(2) 후속 탱크의 용량을 고려하여 이동불가 주문을 제외한다. 예를 들어 부분일정 (5, 3, 1)의 경우 5번 주문의 현 위치는 2번 탱크이고 다음 탱크는 3번이다. 그런데 3번 탱크의 용량은 1이고 현재 4번 주문이 작업 중이므로 이동이 불가능하게 된다. 또한 5번 탱크의 3번 주문도 마찬가지로 6번 탱크가 작업 중이므로 이동이 불가능하며, 따라서 5번과 3번 주문을 분지 대상에서 제외할 수 있다.

(3) 호이스트 대기시간을 줄여주는 주문을 선택한다. 부분일정 (5, 3, 1)의 경우 이미 3개의 주문을 분지 대상에서 제거하였으므로 남은 두 후보는 2번 및 4번 주문이다. 이제 호이스트가 이 두 주문을 처리할 경우 발생하는 대기시간을 계산한다. 이 계산을 위해서는 최초의 호이스트 위치로부터 현 부분일정을 수행하면서 변경되는 시간 값들을 계속 보유하고 있어야 한다. 예제의 경우 호이스트가 1번 주문을 처리한 직후의 상황을 보면, 일단 호이스트의 위치는 8번 탱크(완료버퍼) 위에 있고, 2번 주문은 6번 탱크에 있는데 그 탱크에서의 경과시간은 11.1분이며, 4번 주문은 3번 탱크에서 12.1분 동안 작업이 진행된 상황이다. 2번 주문의 6번 탱크 작업은 처리시간이 (5, 15) 구간에 있어야 하므로 호이스트가 현 위치에서 6번 탱크로 이동하는 0.6분 후 대기 없이 바로 이동작업을 시작할 수 있다. 반면 4번 주문의 3번 탱크 작업은 처리시간이 (15, 25)인데 빈 호이스트 이동시간이 1.5분이므로 호이스트의 3번 탱크 도착시점이 13.6분이 되고 따라서 (5, 3, 1) 다음에 4번 주문을 처리하려면 호이스트가 3번 탱크 위에서 1.4분(= 15-13.6) 동안 대기하여야 한다. 이렇게 각 분지 후보 주문을 현재의 부분일정 바로 다음에 처리할 경우 호이스트의 대기시간이 얼마나 발생하는지를 조사하여 대기가 없는 주문은 모두 분지 대상으로 포함시키고, 대기가 발생하는 경우에는 대기시간이 가장 짧은 주문 2개만

을 분지 대상으로 한다.

(4) 문제의 크기가 커지면 분지 수를 2개로 한정하여도 알고리즘 수행시간이 길어지게 된다. 또한 주문의 수가 많으면 대기시간이 없는 주문들의 수도 늘어나 분지의 수가 더 늘어나게 된다. 따라서 알고리즘 초반에는 다양한 가능성을 열어두기 위하여 대기시간이 없는 주문들을 모두 분지대상에 포함시키고 그 숫자가 2개 미만일 경우에는 대기시간이 짧은 주문 2개 까지 분지대상으로 한 뒤, 알고리즘 후반에 가서는 분지의 수를 무조건 1개로 줄이기로 하였다. 그 방법은 미리 정해진 비율값(*BRATIO*)을 알고리즘 수행 시 파라메터로 사용하는 것인데, 예를 들어 이 비율값이 1/3이었다면 위의 예에서 부분일정의 길이가 17/3 이하일 경우에는 분지의 수를 2개 이상으로 그 이후에는 분지의 수를 1개로 하게 된다. 이 *BRATIO* 값을 이용하면 알고리즘 수행시간과 해의 품질을 적절히 조정할 수 있다. 즉, 문제의 크기가 커서 알고리즘 수행시간이 길어지면 0에 가까운 *BRATIO* 값을 사용하여 시간을 줄이고, 문제의 크기가 작아서 알고리즘 수행시간에 대한 부담이 없는 경우에는 1에 가까운 *BRATIO* 값을 사용하여 해의 품질을 높일 수 있게 되는 것이다.

4.2 하한(Lower Bound) 계산

위의 분지 정책에 따라 분지가 이루어지면 기존의 부분일정에 하나의 주문이 추가된 부분일정들이 형성되는데 그 다음에 수행할 단계는 그 부분일정들의 가능해 여부(Feasibility)를 확인하는 것이다. 이 작업은 그 부분일정이 진행되었을 경우 각 주문이 현재 처리 중인 탱크 작업의 처리시간 상한을 초과하는지를 확인하는 것으로 이루어진다. 앞선 예에서 만약 앞쪽 두 작업에 대한 부분일정이 (5, 1)로 형성되었다면 1번 주문의 이동을 마친 시점에 4번 탱크에 있는 3번 주문의 경과시간이 7.1분이 되어 처리시간 상한인 5분을 초과하게 된다. 따라서 부분일정 (5, 1)은 불가능해가 되고 그 부분일정은 앞으로의 고려대상에서 제거할 수 있게 된다.

가능해 여부를 확인한 다음에는 그 부분일정의 최적해 하한을 계산하는 것이 필요하다. 계산하는 방법은 현재까지의 경과시간에 아래 3개 항목 중 최대값을 더하는 것이다. 앞선 예에서 부분일정 (5, 3, 1)의 경우, 마지막 작업인 1번 주문의 이동이 끝난 시점이 8.1분(= 2.3+0.6+2.3+0.6+2.3)이므로 이 8.1분에 아래 3개 중 최대값을 더한다.

(1) 완료되지 않은 각 주문에 대해, 시스템 내에 그 주문만 존재한다는 가정 아래서 완료시간을 계산하고, 가장 큰 값을 찾는다. 앞선 예에서 부분일정 (5, 3, 1)이 실행된 상황에서는 미완료 주문이 2번, 3번, 4번, 5번 등 4개이다. 이 중 5번 주문의 완료시간을 계산해보자. 5번 주문은 현재 2번 탱크에 있고 경과시간은 5.8분이다. 현재 호이스트의 위치가 8번 탱크이므로 빈 호이스트가 2번 탱크로 가면 1.8분이 경과되므로 처리 경과시간은 7.6분(= 5.8+1.8)이 된다. 5번 주문의 2번 탱크 처리시간 하한은 3분이므로 바로 3번 탱크로 이동할 수 있고 그 이동시간은 2.3분이다. 3번 탱크에서 최소 처리시간(15분) 동안 작업한 후 4번 탱크로 이동(2.3분)하고, 4번 탱크에서도 최소 처리시간(1분) 동안 작업한 후 5번 탱크로 이동(2.3분)한다. 5번 탱크에서 최소시간(40분) 작업 후 완료버퍼로 이동(2.9 분)하면 5번 주문의 처리가 완료되고, 총 소요시간은 67.6분(= 1.8+2.3+15+2.3+1+2.3+40+2.9)이 된다. 2번, 3번, 4번 주문에 대해서도 같은 방법으로 완료시간을 계산하여 가장 큰 값을 찾고 그 결과를 MX_1이라고 하자.

(2) 버퍼를 제외한 각 탱크에 대해 그 탱크를 필요로 하는 주문들의 총 처리시간을 계산한다. 앞선 예제에서는 버퍼를 제외한 6개의 탱크(2번~7번) 각각에 대해 그 탱크의 처리를 완료하기 위한 최소 시간치를 계산하면 된다. 부분일정(5, 3, 1) 상황에서 5번 탱크를 대상으로 계산해본다. 1번 및 2번 작업은 5번 탱크와 관계가 없으므로 무시한다. 3번 주문은 현재 5번 탱크에 있고 그 처리 경과시간이 2.9분(= 0.6+2.3)이므로 처리시간 하한(40분)까지 37.1분이 필요하다. 또한 처리 후 다음 탱크까지 2.3분이 필요하므로 5번 탱크와 관련하여 3번 주문을 위해 필요한 시간은 39.4분이다. 다음으로 4번 주문의 처리를 위해서는 4번 탱크에서 가져와(2.3분) 최소시간 동안의 작업(40분) 후 다음 탱크로 이동(2.3분)하기 위해 총 44.6분이 필요하다. 마지막으로 5번 주문을 가져와서(2.3분) 최소시간 작업(40분)하고 다음 순서인 8번 탱크로 이동(2.9분)하는데 45.2분이 필요하다. 따라서 부분일정(5, 3, 1) 상황에서 남아있는 모든 주문이 5번 탱크를 통과하는데 최소한 129.2분이 필요하다. 나머지 탱크들(2, 3, 4, 6, 7번)에 대해서도 같은 방법으로 최소 필요시간을 구한 다음 이 값들 중에서 최대값을 찾아 MX_2라고 하자.

(3) 남아 있는 작업들을 모두 처리하기 위한 호이스트의 최소 이동시간을 계산한다. 다시 한 번 부분일정(5, 3, 1) 상황을 보면, 이 부분일정은 NXT 행렬에서 0이 아닌 값을 갖는 17개의 작업 중에서 현재까지 3개가 처리되었다는 의미이므로 앞으로 14개의 작업이 남게 되고 이 작업들을 처리하기 위한 물품 장착 상태의 호이스트 이동시간을 모두 더하면 앞으로 호이스트가 수행해야할 최소한의 시간을 계산할 수 있다. 이 값을 MX_3라고 하자.

이제 이 3개 값을 사용해 각 부분일정이 가질 수 있는 완료시간 하한을 계산할 수 있는데, 앞의 예에서 부분일정(5, 3, 1)의 하한은 8.1+max{MX_1, MX_2, MX_3}로 계산할 수 있다.

4.3 알고리즘

본 연구에서 제안하는 단순화된 분지-한계 알고리즘은 다음과 같이 정리될 수 있다.

Algorithm **SimpBNB**

단계 0 : 분지가 이루어지지 않은 현재 상황의 최소 완료시간 CurMin 값을 ∞로 둔다.

단계 1 : 첫 번째 분지한계

  단계 1.1 : 실행가능한 모든 주문(다음 순서 탱크가 비어있는 주문)에 대해 분지한다.

  단계 1.2 : 각 분지가 불가능해이면 고려대상에서 제거한다.

  단계 1.3 : 각 분지의 부분일정에 대한 경과시간을 계산한다.

단계 2 : 고려대상 분지가 없으면 종료한다(단, CurMin = ∞이면 해를 찾지 못함).

단계 3 : 분지한계

  단계 3.1 : 현재 경과시간이 최대인 분지를 찾아 제 4.1절 정책에 따라 분지한다.

  단계 3.2 : 각 분지가 불가능해이면 고려대상에서 제거한다.

  단계 3.3 : 각 분지의 부분일정에 대한 경과시간을 계산한다.

  단계 3.4 : 각 분지의 부분일정이 완료일정인 경우

    단계 3.4.1 : 경과시간 < CurMin이면 CurMin = 경과시간

    단계 3.4.2 : 그 분지는 고려대상에서 제거한다.

  단계 3.5 : 완료 일정이 아닌 경우

    단계 3.5.1 : 그 분지에 대한 하한을 계산한다.

    단계 3.5.2 : 하한이 CurMin보다 크면 그 분지는 고려대상에서 제거한다.

단계 4 : <단계 2>로 간다.

이 알고리즘을 앞서 LINGO 시스템을 실행한 컴퓨터에서 C++ 언어로 구현하고 같은 예제를 풀어 보았다. 이 문제는 크기가 작아서 *BRATIO* 값을 1로 두었다. 즉, 길이에 관계없이 모든 부분일정에서 분지의 수를 2개 이상으로 하였다. 그 결과 알고리즘 수행시간은 앞선 LINGO 최적해와 마찬가지로 0.01초 미만이었으나 해는 약간 달랐다. 주문 처리순서는 (3, 1, 2, 4, 4, 2, 5, 3, 5, 3, 5, 5, 4, 3, 4, 5)로 나왔고, 그 때의 완료시점(Makespan)은 123.7분으로서 앞선 최적 완료시간 120.3에 비해 2.8% 정도 부족한 결과를 보여 주었다.

5. 수치 실험

본 절에서는 앞 절에서 소재한 휴리스틱 방법의 성능을 평가하기 위해 *NT*가 12인 좀 더 현실적인 크기의 문제를 사용해 최적해와 비교해보고자 한다. 앞서 제 3장에서 언급하였던 것처럼 *NJ*가 10인 경우에는 LINGO를 통해서는 최적해를 구할 수가 없었으므로 *NJ*가 10보다 작은 경우에 대해서 최적해와 비교하였다. 비교하는 방법은 *NT*를 12로 고정한 상황에서 *NJ*가 7, 8, 9 등 3개 값을 가질 때 LINGO를 통해 최적해를 구하고 그 결과와 본 연구의 휴리스틱 결과를 비교하였다. 휴리스틱은 *BRATIO* 값을 0, 0.25, 0.5, 0.75, 1.0 등 5개 값으로 변화시키면서 실행하였고, 결과로 도출된 주문 전체 처리시간(Makespan)과 결과를 도출하기 위한 계산시간을 각각 비교하였다. 그 결과는 <Table 1>에 정리되어 있다.

Table 1. Performance Comparison of the Heuristic with Optimal Solution

NJ			7	8	9
LINGO(Optimal)		Makespan	100	100	100
		Com. Time	8	189	6,917
Heuristic	BRATIO = 0.0	Makespan	100.67	-	-
		Com. Time	< 1	-	-
	BRATIO = 0.25	Makespan	100.67	102.45	105.17
		Com. Time	< 1	< 1	< 1
	BRATIO = 0.5	Makespan	100.59	100.52	100.00
		Com. Time	< 1	2	22
	BRATIO = 0.75	Makespan	100.59	100.52	100.00
		Com. Time	< 1	90	730
	BRATIO = 1.0	Makespan	100.59	100.52	100.00
		Com. Time	< 1	126	915

표에 제시된 수치를 이해하기 위해 *NJ*가 7인 경우에 대한 도출과정을 설명한다. 우선 최적 Makespan이 100이라고 표시되어 있는데 실제 계산값은 1,189로서 비교를 위해 기준값을 100으로 잡은 것이다. 휴리스틱에 의한 Makespan 값은 *BRATIO* 값에 따라 각각 1,197, 1,197, 1,196, 1,196, 1,196 등이 도출되었고 최적해 1,189를 100으로 놓았으므로 이 값들은 각각 100.67, 100.67, 100.59, 100.59, 100.59 등으로 변환되어야 한다. 즉, 이 결과는 최적해에 비해 각각 0.67%, 0.67%, 0.59%, 0.59%, 0.59% 만큼 나쁜 결과를 보여주고 있다. 계산시간 측면에서는 *NJ*가 7인 경우 최적해 도출시간이 8초였는데 휴리스틱 수행시간은 *BRATIO* 값에 관계없이 모두 1초 미만이었다.

*NJ*가 8 및 9인 경우에도 비슷한 방법으로 표의 항목들을 구할 수 있었다. 다만 *BRATIO*가 0일 때는 휴리스틱이 해를 찾지 못하여 해당란은 공란으로 비워져 있다. 또한 앞서 언급한 것처럼 *NJ*가 10인 경우에는 LINGO를 통한 최적해를 구할 수 없어서 이 표에는 *NJ* 값을 9까지만 정리하였다. 그러나 휴리스틱으로는 *NJ*가 10일 때도 *BRATIO* 값이 0이 아니면 해를 구할 수 있었으며, 소요시간은 *BRATIO* 값이 0.25, 0.5, 0.75, 1.0일 때 각각 1초 미만, 59초, 722초, 893초였다. 특이한 점은 여기서 *BRATIO* 값이 0.75 및 1.0일 때의 소요시간 722초 및 893초는 *NJ*가 9일 때보다 작다는 것인데 이것은 일반적인 현상이라기보다는 본 예제의 특수성에 기인한 것으로 보인다.

하나의 예제로 일반적인 특성을 논하기는 어렵지만 <Table 1>을 잘 관찰하면 본 연구에서 제안하는 휴리스틱의 대략적인 특성을 파악할 수 있다. 우선 해의 품질을 보면 휴리스틱의 Makespan은 최적해에 비해 최악의 경우 5.17% 나쁜 결과를 보여주었고 최선의 경우 최적해와 동일하였다. 대부분의 경우 최적해

와 1% 안쪽의 차이를 보여주었으며 이것은 휴리스틱의 성능이 매우 뛰어남을 보여준다. 계산시간 측면에서도 *BRATIO* 값만 잘 선택한다면 1분 이내에 결과를 도출할 수 있으므로 현장에서 실제로 사용하는데 아무런 문제가 없을 것으로 생각된다. 다만 *BRATIO* 값에 따라 해를 구하지 못하기도 하고 소요시간에도 차이가 있다는 것이 문제점으로 지적될 수 있다. 하지만 현장 상황에서는 탱크의 수가 이미 정해져 있고 주문의 수도 큰 변화가 없다고 본다면 대략적인 *BRATIO* 값을 사전 테스트를 통해 알 수가 있으므로 실제 사용에 큰 문제는 없을 것이라고 생각한다.

사실상 <Table 1>은 최적해와 비교할 수 있는 가장 큰 사이즈의 문제를 다루고 있다. 앞서 언급한 것처럼 본 연구의 동기가 된 기업의 경우 30개의 탱크를 사용하고 있는데, 이런 규모의 문제는 최적해를 구할 수가 없으므로 휴리스틱의 성능 검증용 문제로는 사용할 수가 없었다. 하지만 제안된 휴리스틱을 사례 기업에 적용하기 위해 실제 자료와 아주 유사한 데이터를 사용한 예제를 구성해보았다. 탱크는 30개이고 처리대상 주문의 수는 10개로 하였다. 실제로 이 기업의 경우 동시에 처리하는 주문의 수는 통상 5개 미만인데 예제에서는 이것보다 충분히 큰 수를 사용한 것이다. 또한 실제와 같이 각 주문이 거쳐야 할 탱크의 수는 8~10개 정도로 하였다. *BRATIO* 값을 0부터 0.1씩 증가시키며 알고리즘을 테스트하였는데 0.5까지는 해를 찾지 못하였고 0.6 이상에서는 해를 구할 수 있었다. 해를 구한 경우 소요시간은 모두 1초와 2초 사이였다. 또한 해의 품질을 보면, *BRATIO* 값이 0.6일 때의 Makespan 값을 100이라고 했을 때 0.7일 때는 97.1, 0.8일 때는 95.4, 그리고 0.9와 1.0일 때는 95.1이었다.

여기서 한 가지 의문은 문제의 크기가 훨씬 큰데도 불구하고 탱크 30개인 문제의 계산시간이 앞선 예제보다 훨씬 짧았다는 점이다. 이것은 알고리즘 수행과정을 보면 이해가 될 수 있는데, 뒤쪽 예제의 경우 매우 많은 분지(즉, 부분해)들이 불가능해로 결정되어 더 이상의 고려대상에서 제거되었다는 점을 관찰할 수 있었다. 불가능해로 결정되는 것은 일부 탱크의 처리시간이 상한시간을 초과하는 경우인데, 이 예제의 경우 각 탱크의 처리시간 상한과 하한의 차이가 이전 예제에 비해 작았고 이로 인해 불가능해가 많이 생성되었으며 따라서 이것이 짧은 계산시간의 원인이었다고 생각된다.

6. 결 론

본 연구에서는 금속부품의 표면처리 작업을 수행하는 다단계 화학처리 생산시스템의 사례문제를 다루고 있다. 문제의 핵심은 이 생산시스템에서 물자의 이동을 담당하는 호이스트의 효율적인 사용을 위한 일정계획을 작성하는 것이다. 연구대상 시스템은 관련 연구들 중에서 가장 일반적이고 현실적인 상황을 다루고 있는데, 그 특징으로는 1) 다양한 처리조건을 갖는 다양한 주문을 다루고 있고, 2) 이미 생산시스템에 투입되어 작업 중인 주문들도 계획대상에 포함시켰으며, 3) 단위공정들이 각각의 처리용량을 가질 수 있다는 것 등이다. 수리적 모형을 이용한 최적해는 문제의 크기가 현실적으로 커질 경우 적정 시간 안에 도출이 불가능하므로 본 연구에서는 분지한계 기반의 휴리스틱을 개발하였다. 최적해를 구할 수 있는 크기의 문제를 통해 휴리스틱의 결과와 최적해를 비교해본 결과 해의 품질에 큰 차이가 없었는데 소요시간 측면에서는 문제의 크기가 커질 경우 휴리스틱이 훨씬 빨리 해를 도출해주었다. 따라서 본 연구에서 제안하는 휴리스틱은 문제의 크기가 클 경우 매우 활용도가 높을 것으로 기대된다.

저자들은 본 연구의 후속으로 다음과 같은 내용에 대한 연구를 계획하고 있다. 우선 본 연구의 휴리스틱을 호이스트의 수가 2대 이상인 경우에 대해 확대 적용할 예정이다. 또한 공정순서가 가변적인 주문이 존재하는 상황, 생산라인 중간에 대기용 탱크가 존재하는 상황, 생산시스템 내에 머물 수 있는 주문의 수에 제약이 있는 상황 등 여러 가지 생산환경에 맞게 본 연구의 휴리스틱을 수정하여 적용하는 연구도 수행할 예정이다.

참고문헌

Chen, H., Chu, C., and Proth, J. M. (1998), Cyclic scheduling of a hoist with time window constraints, *IEEE Transactions on Robotics and Automation*, **14**(1), 144-152.

Hindi, K. S. and Fleszar, K. (2004), A constraint propagation heuristic for the single-hoist multiple-products scheduling problem, *Computers and Industrial Engineering*, **47**(1), 91-101.

Kumar, P. R. (1994), Scheduling semiconductor manufacturing plants, *IEEE Control System*, **14**(6), 33-40.

Kuntay, I., Xu, Q., Uygun, K., and Huang, Y. (2006), Environmentally conscious hoist scheduling for electroplating facilities, *Chemical Engineering Communications*, **193**(3), 273-292.

Lei, L. and Wang, T. J. (1989), A proof : The cyclic HSP is NP-complete, Technical Report 89-0016, Graduate School of Management, Rutgers University.

Lei, L. and Wang, T. J. (1994), Determining optimal cyclic hoist schedules in a single-hoist electroplating line, *IIE Transactions*, **26**(2), 25-33.

Lim, J. M. (1997), A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line, *Computers and Industrial Engineering*, **33**(3/4), 789-792.

Liu, C. W., Zhao, C. Y., and Xu, Q. (2012), Integration of electroplating process design and operation for simultaneous productivity maximization, energy saving, and fresh water minimization, *Chemical Engineering Science*, **68**(1), 202-214.

Paul, H. J., Bierwirth, C., and Kopfer, H. (2007), A heuristic scheduling procedure for multi-item hoist production lines, *International Journal of Production Economics*, **105**(1), 54-69.

Phillips, L. W. and Unger, P. S. (1976), Mathematical programming solution of a hoist scheduling programm, *AIIE Transactions*, **28**(2), 219-225.

Shapiro, G. W. and Nuttle, H. L. (1988), Hoist scheduling for a PCB electroplating, *IIE Transactions*, **20**(2), 157-167.

Tian, N., Che, A., and Feng, J. (2013), AIChE Letter: Real-time hoist scheduling for multistage material handling process under uncertainties, *AIChE Journal*, **59**(4), 1046-1048.

Xu, Q. and Huang, Y. (2004), Graph-assisted optimal cyclic hoist scheduling for environmentally benign electroplating, *Industrial and Engineering Chemistry Research*, **43**(26), 8307-8316.

Yih, Y. (1994), An algorithm for hoist scheduling problems, *International Journal of Production Research*, **32**(3), 501-516.

Yin, N. C. and Yih, Y. (1992), Crane scheduling in a flexible electroplating line : A tolerance-based approach, *Journal of Electronics Manufacturing*, **2**(4), 137-144.

Zhao, C., Fu, J., and Xu, Q. (2013), Real-time dynamic hoist scheduling for multistage material handling process under uncertainties, *AIChE Journal*, **59**(2), 465-482.