Journal of the Korean Institute of Industrial Engineers
Vol. 47, No. 1, pp. 001-010, February 2021.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2021.47.1.001

<Original Research>

유전 알고리즘을 이용한 두 단계 이종병렬기계 시스템의 일정계획

고시근[1†] · 서원철[1] · 김상운[2]

[1]부경대학교 시스템경영공학부 / [2]㈜엠임팩트씨엔씨

Two-Stage Non-Identical Parallel Machine Scheduling with Genetic Algorithm

Shiegheun Koh[1] · Wonchul Seo[1] · Sangwoun Kim[2]

[1]Dept of Systems Management & Engineering, Pukyong National University
[2]Management Impact Consulting & Coaching Co., Ltd.

This research deals with a scheduling problem for two-stage flow line, where each stage of the line consists of non-identical parallel machines. We first present a mixed integer linear programming formulation for the problem, and using this formulation, we can easily find the optimal solutions for small-sized problems with any commercial optimization software. However, since the problem is NP-hard and the size of a real problem is large in general, two genetic algorithm based heuristics are proposed to solve the practical big-size problems in a reasonable computational time. And then, to assess the performances of the algorithms, we conduct some computational experiment, from which we found both the heuristic algorithms show very good performances.

Keywords: Scheduling, Non-Identical Parallel Machines, Two-Stage Flow Line

1. 서 론

하나의 작업이 여러 대의 기계들 중 한 기계에서 처리되는 상황을 다루는 병렬기계 일정계획(parallel machine scheduling) 문제들은 일반적으로 시스템 내의 모든 기계들이 완전히 동일하다고 가정하고 있다(Cheng and Sin, 1990; Dawande *et al*., 2005). 하지만 현실적으로 병렬기계시스템은 생산시스템을 운영하면서 필요에 따라 기계를 추가 구매하거나 다른 용도로 사용되던 기계를 수정함으로써 이루어지는 경우가 많기 때문에 기계들의 성능이나 사양이 서로 다른 경우가 대부분이다. 이렇게 서로 다른 사양의 기계들이 같은 작업에 투입되는 상황을 기존의 병렬기계 시스템과 구분하기 위해 '이종병렬기계(non-identical parallel machines)' 시스템이라고 부르며 제조 및 서비스를 포함한 다양한 분야의 기업에서 많이 찾아볼 수 있다(Randhawa and Kuo, 1997; Kim *et al*., 2002).

일정계획을 다룬 많은 연구들이 '모든 주문의 처리가 최종적으로 완료되는 총 완료시간(makespan)을 최소화하는 것'을 목적함수로 하고 있는데, 이종병렬기계 환경에서 총 완료시간을 최소화하는 문제들은 1970년대부터 본격적으로 연구되었다. 우선 Horowits and Sahni(1976), Ibarra and Kim(1977), De and Morton(1980), Davis and Jaffe(1981), Potts(1985), Lenstra *et al*. (1990), Hariri and Potts(1991) 등이 이종병렬기계 시스템의 총 완료시간을 최소화하기 위한 근사 알고리즘을 개발하였다. 그 후 Van de Velde(1993)는 분지-한계를 통한 최적 알고리즘과 휴리스틱 알고리즘을 제안하였고, Piersma and Van Dijk(1996)은 기존의 지역탐색 알고리즘을 개선하여 개선된 해를 제공하는

이 논문은 부경대학교 자율창의학술연구비(2019년)에 의하여 연구되었음.
† 연락저자 : 고시근 교수, 48513 부산광역시 남구 용소로 45, 부경대학교 시스템경영공학부, Tel : 051-629-6484, Fax : 051-629-6478, E-mail : sgkoh@pknu.ac.kr
2020년 6월 5일 접수; 2020년 9월 22일 수정본 접수; 2020년 9월 29일 게재 확정.

새로운 알고리즘을 개발하였다. 또한 Srivastava(1998)는 타부 서치 기법을 활용한 알고리즘을 개발하였고, Hop and Nagaur (2004)는 PCB 생산라인의 이종병렬기계 시스템에서 총 완료시간을 최소화하기 위한 수리적 모형을 제시한 다음 유전 알고리즘 기반의 해법을 제안하였다. Ghirardi and Potts(2005)는 RBS (recovering beam search)를 활용한 휴리스틱 알고리즘을 제안하였고, Agarwal *et al*.(2006)은 12개의 휴리스틱과 augmented neural network 방법론을 제안하고 이 해법들을 비교 · 평가하였다. Tavakkoli-Moghaddam *et al*.(2009)은 순서 및 기계에 의존하는 셋업시간을 고려한 이종병렬기계시스템에서 지연작업의 수와 총 완료시간을 최소화하기 위한 유전 알고리즘 기반의 해법을 제안하였다. 이 연구에서 주문들은 각각의 도착시간을 갖고 있어서 도착 이전에는 작업이 진행될 수 없다고 가정하였다. Vallada and Ruiz(2011)는 순서 및 기계 의존적인 셋업시간을 고려한 문제를 해결하기 위해 지역탐색 알고리즘과 유전 알고리즘을 결합한 알고리즘을 개발하였다. Joo and Kim(2012)도 이들과 비슷한 문제를 다루었는데, 총 완료시간을 최소화하는 수리모형을 제시하고 유전 알고리즘 및 자기진화(self-evolution) 알고리즘 기반의 두 해법을 개발한 다음 그 성능을 비교 · 평가하였다. 그 후 Koh and Mahardini(2014)는 실시간 계획에서 필요한 기계가용시간 조건을 포함시켜 모형을 수정하고 그 모형을 해결하기 위한 다양한 휴리스틱을 제안하였다.

한 개 이상의 단계가 병렬기계로 이루어진 다단계 흐름생산라인은 혼합흐름생산시스템(hybrid flow shop)이라고 불리며 생산현장에서 종종 발견된다. 실제로 저자들은 스프링 제조기업에서 스프링 형상 제조공정과 열처리 공정이 이종병렬기계의 흐름으로 이루어진 시스템을 목격한 바 있다. 이러한 시스템에 대한 일정계획문제도 최근 상당히 많은 관심을 받고 있는데(Ruiz and Vazquez-Rodriguez, 2010), 우선 Narashimhan and Panwalker(1984)가 두 번째 작업장에 이종병렬기계가 배치된 2단계 혼합흐름생산시스템에서 총 기계유휴시간 및 총 작업대기시간을 최소화하는 CMD(cumulative minimum deviation) 규칙을 제안하였다. 다음으로 Gupta and Tunc(1991)는 두 번째 작업장에 동종병렬기계가 설치된 상황에서 총 완료시간을 최소화하기 위한 근사해 발견 알고리즘을 제안하였고, Gupta *et al*.(1997)은 첫 번째 작업장에 동종병렬기계가 설치된 상황에서 총 완료시간을 최소화하기 위한 분지-한계 알고리즘과 몇 가지 휴리스틱을 제안하였으며, Lee and Park(1999)은 첫 번째 단계가 두 대의 이종병렬기계로 이루어져 있고 두 번째 단계는 한 대의 기계로 이루어진 혼합흐름생산시스템에서 총 완료시간을 최소화하기 위한 알고리즘을 개발하였다. 또한 Jeong and Jeong(2000)은 각 단계가 이종병렬기계들로 구성된 다단계 생산시스템의 생산일정계획지원시스템 개발사례를 소개하였다. 그러나 이 논문에서는 최적화 모형이나 최적화 알고리즘을 제시하지는 않았다. 그 후 Chen and Chen(2009)은 병목기반 휴리스틱을 활용하여 이종병렬설비로 이루어진 유연흐름라인의 작업완료시간을 최소화하는 기법을 제안하였다.

본 연구에서는 두 단계가 모두 이종병렬기계로 이루어진 2단계 혼합흐름생산시스템의 일정계획 문제를 다루고자 한다. 저자들이 조사한 바에 의하면 두 단계가 모두 기계대수 제한이 없는 이종병렬기계 시스템의 총 완료시간 최소화 문제에 대한 최적화 모형구축 및 해법을 다룬 연구는 아직 없었다. 따라서 본 연구에서는 일단 연구대상 문제를 혼합정수계획법(MILP; Mixed Integer Linear Programming) 모형으로 표현하고 규모가 작은 문제들을 이용해 모형의 유효성을 검증한다. 또한 규모가 큰 문제를 해결하기 위한 방법론으로 유전알고리즘을 적용한 휴리스틱 해법을 제안하고 그 효과에 대해서도 검증하도록 한다.

본 연구의 나머지 부분은 다음과 같이 구성된다. 다음 장에서는 대상문제를 정수계획법의 형태로 모형화하고, 제 3장에서는 대상문제를 해결하기 위한 유전 알고리즘 기반의 휴리스틱 해법을 제시하며, 제 4장에서는 제안된 해법들의 성능을 평가하기 위한 수치실험을 소개한다. 마지막으로 제 5장에서는 논문을 요약하고 추후에 수행되어야할 연구과제를 제시한다.

2. 수리 모형

본 절에서는 본 연구의 대상문제를 정수계획법 형태의 수리모형으로 표현하고자 한다. 모형 개발을 위해서는 문제를 명확하게 정의해야 하며 이를 위해 다음과 같이 대상 시스템에 대해 가정하도록 한다. 1) 본 연구의 대상 시스템은 두 개의 기계군으로 이루어져 있으며 처리대상 주문은 이 두 기계군을 반드시 순서대로 거쳐야 한다. 2) 주문이 각 기계군을 거칠 때는 그 기계군 내 하나의 기계에서 작업이 이루어진다. 3) 셋업시간은 순서와 무관하여 각 작업시간에 포함될 수 있으므로 별도로 고려하지 않는다. 4) 주문들의 이동시간은 고려하지 않는다. 5) 하나의 기계에서 일단 작업이 시작되면 그 작업이 끝날 때까지 멈출 수 없다. 6) 같은 단계라도 어느 기계에서 작업이 수행되느냐에 따라 작업수행시간이 달라진다. 7) 문제를 해결하는 (즉, 일정계획이 필요한) 시점은 선행기계군의 기계 중 적어도 하나에서 작업이 끝나 그 기계가 유휴상태로 변하는 순간이다. 만약 도착해 있는 주문이 하나도 없는 상황에서 선행기계가 유휴상태로 되었다면 그 이후 첫 주문이 도착하는 시점이 문제해결 시점이다. 즉, 선행기계군에서 하나 이상의 기계가 유휴상태이고 대기 중인 주문이 하나 이상 있는 상황에서 새로운 작업일정을 작성하는 것이 본 연구의 목적이다.

위와 같은 가정 하에서 모형을 개발하였으며, 모형에서 사용되는 기호에 대해 먼저 아래와 같이 설명한 다음 개발된 모형을 소개한다.

<모형 파라미터>

n : 처리 대상 주문 수

m_1 : 선행 기계군에 속한 기계 수

m_2 : 후행 기계군에 속한 기계 수
i, j : 주문을 표시하는 인덱스 ($i, j = 1, 2, \cdots, n$)
k : 선행기계를 표시하는 인덱스($k = 1, 2, \cdots, m_1$)
l : 후행기계를 표시하는 인덱스($l = 1, 2, \cdots, m_2$)
a_i : 계획시점 이전에 주문 i가 도착하였으면 0, 그렇지 않으면 도착 예정시간
r_{1k} : 선행기계 k의 가용시점(즉, 계획시간 이전에 할당된 작업의 완료 시간)
r_{2l} : 후행기계 l의 가용시점(즉, 계획시간 이전에 할당된 작업의 완료 시간)
p_{ik} : 주문 i가 선행기계 k에 할당되었을 경우의 처리시간
q_{il} : 주문 i가 후행기계 l에 할당되었을 경우의 처리시간
M : 충분히 큰 수

<결정변수>
s_{1i}, f_{1i} : 주문 i의 선행 공정 착수시간 및 완료시간
s_{2i}, f_{2i} : 주문 i의 후행 공정 착수시간 및 완료시간
$f_{\max}$: 모든 주문의 완료시간 (makespan)
x_{ik} : 주문 i가 선행기계 k에 할당되면 1, 그렇지 않으면 0
y_{il} : 주문 i가 후행기계 l에 할당되면 1, 그렇지 않으면 0
u_{ik} : 선행기계 k가 가장 먼저 처리하는 주문이 i이면 1, 그렇지 않으면 0
u_{ijk} : 선행기계 k에서 주문 j가 주문 i에 이어 처리되면 1, 그렇지 않으면 0
w_{il} : 후행기계 l이 가장 먼저 처리하는 주문이 i이면 1, 그렇지 않으면 0
w_{ijl} : 후행기계 l에서 주문 j가 주문 i에 이어 처리되면 1, 그렇지 않으면 0

<모형>

Minimize $f_{\max}$ (1)

subject to

$$s_{1i} \geq a_i \quad \text{for all } i \quad (2)$$

$$s_{1i} + M(1 - x_{ik}) \geq r_{1k} \quad \text{for all } i, k \quad (3)$$

$$s_{1i} + \sum_{k=1}^{m_1} p_{ik} x_{ik} = f_{1i} \quad \text{for all } i \quad (4)$$

$$f_{1i} \leq s_{2i} \quad \text{for all } i \quad (5)$$

$$s_{2i} + M(1 - y_{il}) \geq r_{2l} \quad \text{for all } i, l \quad (6)$$

$$s_{2i} + \sum_{l=1}^{m_2} q_{il} y_{il} = f_{2i} \quad \text{for all } i \quad (7)$$

$$f_{2i} \leq f_{\max} \quad \text{for all } i \quad (8)$$

$$f_{1i} \leq s_{1j} + M(1 - \sum_{k=1}^{m_1} u_{ijk}) \quad \text{for all } i \neq j \quad (9)$$

$$f_{2i} \leq s_{2j} + M(1 - \sum_{l=1}^{m_2} w_{ijk}) \quad \text{for all } i \neq j \quad (10)$$

$$\sum_{k=1}^{m_1} x_{ik} = 1 \quad \text{for all } i \quad (11)$$

$$\sum_{l=1}^{m_2} y_{il} = 1 \quad \text{for all } i \quad (12)$$

$$\sum_{i=1}^{n} u_{ik} = 1 \quad \text{for all } k \quad (13)$$

$$\sum_{i=1}^{n} w_{il} = 1 \quad \text{for all } l \quad (14)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} u_{ijk} \leq x_{ik} \quad \text{for all } i, k \quad (15)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ijl} \leq y_{il} \quad \text{for all } i, l \quad (16)$$

$$u_{ik} + \sum_{\substack{j=1 \\ j \neq i}}^{n} u_{jik} = x_{ik} \quad \text{for all } i, k \quad (17)$$

$$w_{il} + \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{jil} = y_{il} \quad \text{for all } i, l \quad (18)$$

$$x_{ik}, u_{ik} \in \{0, 1\} \quad \text{for all } i, k$$

$$y_{il}, w_{il} \in \{0, 1\} \quad \text{for all } i, l$$

$$u_{ijk} \in \{0, 1\} \quad \text{for all } i, j, k$$

$$w_{ijl} \in \{0, 1\} \quad \text{for all } i, j, l$$

식 (1)은 본 문제가 총 완료시간을 최소화하는 문제라는 것을 보여준다. 제약조건 (2)는 각 주문의 선행기계 착수가 그 주문의 도착 시점 이후에 가능함을, 식 (3)은 각 주문의 선행기계 착수가 그 주문을 처리하는 기계의 가용시점(r_{1k}) 이후에 가능함을 표현하고 있다. 식 (4)는 각 주문의 선행기계 착수시간과 완료시간 사이의 관계를 설명하고 있다. 식 (5)는 각 주문의 후행기계 작업이 선행기계 작업 종료 이후에 착수될 수 있음을 의미한다. 식 (6)과 식 (7)은 식 (3)과 식 (4)의 조건들이 후행기계 작업에 적용된 것이다. 식 (8)은 총 완료시간이 모든 주문들의 후행기계 작업 완료시간 중 가장 큰 값이라는 것을 보여준다. 식 (9)는 선행기계군에서, 식 (10)은 후행기계군에서 연속처리되는 두 주문에 대해 먼저 처리되는 주문의 작업이 완료된 후에 이어지는 주문의 작업이 시작될 수 있음을 의미한다. 식 (11)에 의하면 하나의 주문은 선행기계군의 기계 중 단 하나에서 처리된다. 식 (13)은 선행기계군의 각 기계에 최초로 배정되는 주문은 한 개라는 것을 의미한다. 식 (15)에 의하면 어떤 주문이 선행기계군의 어떤 기계에 배정되지 않았다면 ($x_{ik} = 0$) 그 주문은 그 기계에서 다른 어떤 주문의 선행주문도 될 수 없다($u_{ijk} = 0$)는 것을 보여준다. 반대로 해석하면 어떤 주문이 선행기계군 내 하나의 기계에 배정된 경우($x_{ik} = 1$) 그 주문은 그 기계에서 다른 어떤 주문의 선행작업이라는 것 ($u_{ijk} = 1$)을 보여준다. 그런데 이 식이 부등식인 것은 그 주문이 그 기계에서 최종으로 처리되는 주문일 경우 때문이다. 비슷하게 식 (17)에서는 어떤 주문이 선행기계군 내 한 기계에

배정되었다면($x_{ik}=1$) 그 주문은 그 기계에서 처리되는 최초의 주문($u_{ik}=1$)이거나 다른 어떤 주문의 후속주문이라는 것($u_{jik}=1$)을 보여준다. 식 (12), 식 (14), 식 (16), 식 (18)은 각각 식 (11), 식 (13), 식 (15), 식 (17)의 조건들이 후행기계군에 적용된 식이다.

이 모형의 유효성을 검증하기 위해 규모가 작은 예제를 만들어 최적화 소프트웨어를 사용해 해결해보도록 한다. 다음과 같이 주문의 수가 6이고 선, 후행 기계수가 각각 3, 2인 문제($n=6$, $m_1=3$, $m_2=2$)를 생각한다;

$$a=(0,0,0,0,4,12),\ r_1=(5,3,0),\ r_2=(7,10),\ p=\begin{bmatrix}8&6&5\\9&8&6\\5&7&8\\6&7&5\\4&6&7\\5&7&9\end{bmatrix},\ q=\begin{bmatrix}3&5\\7&6\\5&7\\8&5\\7&5\\6&4\end{bmatrix}.$$

이 예제의 데이터를 살펴보면 계획대상 주문 6개 중 4개는 이미 도착해 있고($a_i=0$, i=1, 2, 3, 4) 2개의 주문은 시점 4 및 12에 도착할 예정($a_5=4$, $a_6=12$)임을 알 수 있다. 또한 선행기계군의 기계 3대 중 2대는 시점 5 및 시점 3에 현재 진행중인 작업이 완료될 예정이고 나머지 한 대는 작업이 끝나 현재 유휴상태임을 보여준다. 즉, 앞서 설명한 가정 (7)에 의해 새로운 일정계획이 필요한 시점임을 알 수 있다.

위 예제에 대한 최적해를 LINGO를 사용해 구하였다. 2.60 GHz G620 CPU와 4GB RAM이 장착된 PC에서 1분 23초만에 총 완료시간이 24인 최적해를 구할 수 있었고, 그 결과로 얻어진 일정계획을 표로 정리하면 <Table 1>과 같이, 간트차트로 표현하면 <Figure 1>과 같이 나타낼 수 있다. 단, 간트차트에서 빗금으로 표시된 부분은 해당 기계가 현재 가동중으로 새로운 작업의 시작이 불가능함을 표시하며 J와 붙어있는 숫자는 주문번호를 의미한다.

3. 유전알고리즘 방법론

앞서 언급하였듯이 본 연구의 대상 문제는 크기(주문수 및 기계수)가 작은 경우 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 그러나 잘 알려진 바처럼 (Pinedo, 2002) 이 문제보다 간단한 일반적인 병렬기계 일정계획 문제도 NP-hard에 속하므로 문제의 크기가 커질 경우 현실적인 시간 안에 최적해를 구하는 것은 불가능한 일이다. 실제로 앞 절에서 제시한 예제(주문수가 6이고 기계수는 선행 3대 및 후행 2대)의 최적해는 LINGO를 사용해 1분 23초라는 길지 않은 시간에 구할 수 있었으나 주문의 수가 7개 이상이 되면 30분 안에 해가 구해지지 않아 현장에서 사용하기는 어려울 것으로 생각된다. 현실적으로 선행기계군의 기계 중 하나가 작업을 마치는 순간 대기중인 주문들 중 어느 주문이 가장 먼저 그 기계에 투입되어야 하는지를 즉시 결정할 수 있어야 하므로, 이에 부응하기 위해서는 10여 개 이상의 주문이 있을 경우에도 최소한 수초 이내에 답을 찾을 수 있는 해법이 필요할 것이다. 따라서 본 연구에서는 주문의 수가 수십개일 경우에도 안정적으로 빠른 시간 안에 해를 구하기 위해 유전알고리즘 기반의 휴리스틱 방법론을 개발하도록 한다.

Table 1. Optimal Schedule Table

Job	a_i	machine assigned (stage 1)	start time (stage 1)	finish time (stage 1)	machine assigned (stage 2)	start time (stage 2)	finish time (stage 2)
1	0	2	3	9	1	9	12
2	0	3	5	11	1	12	19
3	0	3	11	19	1	19	24
4	0	3	0	5	2	10	15
5	4	2	9	15	2	15	20
6	12	1	15	20	2	20	24

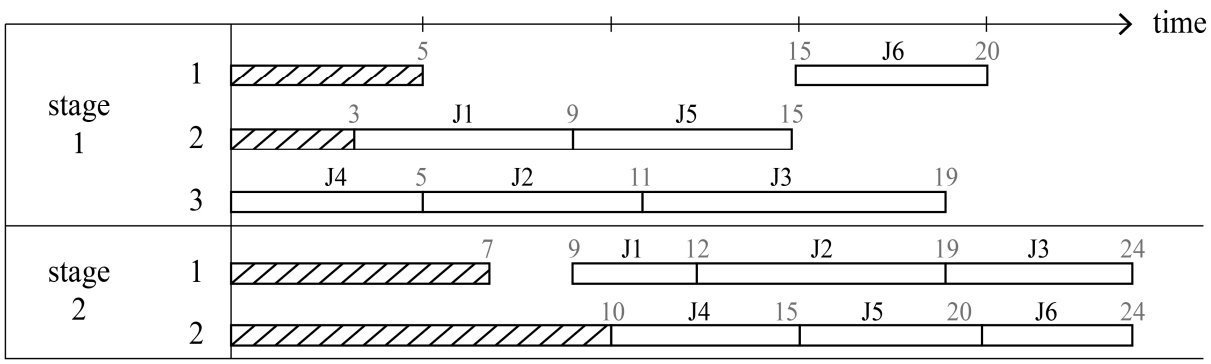

Figure 1. Optimal Schedule in Gantt Chart

본 연구의 대상문제에 대한 답을 구하기 위해서는 선, 후행 각 단계에서 1) 각 주문을 어느 기계에 할당할 것인지와 2) 각 기계에 할당된 주문들의 처리 순서는 어떻게 할 것인지의 두 종류 문제를 해결하여야 한다. 유전알고리즘을 사용해 어떤 문제를 해결하려면 그 문제의 답을 염색체(chromosome)로 표현(encoding)하는 것이 필요한데, 위와 같이 성격이 상이한 두 문제의 답을 하나의 염색체로 표현하는 것은 간단치 않으므로, 본 연구에서 제안하는 유전알고리즘에서는 두 문제 중 한 문제의 답은 염색체로 표현하고 나머지 한 문제의 답은 간단한 휴리스틱을 이용해 구하는 방법을 사용하였다. 즉, 주문을 기계에 할당하는 문제를 염색체로 표현하면 그 할당 안에서 주문들의 처리순서는 간단한 휴리스틱을 사용해서 해결하고, 반대로 주문들의 처리순서를 염색체로 표현하면 기계할당을 간단한 휴리스틱으로 해결하는 것이다. 따라서 본 연구에서 제안하는 알고리즘은 순수한 유전알고리즘에 간단한 휴리스틱이 가미된 혼합유전알고리즘(hybrid genetic algorithm)이라고 할 수 있다.

우선 주문을 기계에 할당하는 과정에서 염색체가 사용되고 각 기계에 할당된 주문들의 순서결정에 휴리스틱을 사용하는 절차에 대해 먼저 설명한다. 일단 염색체 표현(encoding) 방법은 염색체의 한 원소인 유전자(gene)를 (0, 1) 구간 실수로 표현하고, 하나의 염색체를 구성하는 유전자의 수는 주문 수의 두 배(2*n*)로 하는 것이다. 각 염색체의 앞쪽 *n*개 유전자는 각 주문이 선행기계군의 기계 중 어느 기계에 할당되는지를 표시한다. 그 방법은 *i*번째 유전자 값이 구간($\frac{k-1}{m_1}$, $\frac{k}{m_1}$)에 포함되면 주문 *i*를 기계 k(k = 1, 2, ⋯, m_1) 에 배정하도록 하는 것이다. 염색체의 뒤쪽 *n*개의 유전자도 비슷한 방법으로 각 주문을 후행기계군의 기계에 할당하는데 사용한다. 예를 들어 주문이 4개이고 선행기계군은 3대, 후행기계군은 2대로 이루어진 문제라면 염색체는 (0.56, 0.13, 0.98, 0.24, 0.72, 0.03, 0.74, 0.33) 과 같은 8개의 실수로 구성될 수 있다. 첫 번째 유전자 값인 0.56에 의해 첫 번째 주문의 선행기계 할당이 이루어지는데, 그 값이 (1/3, 2/3)에 속하므로 첫 번째 주문은 선행기계 2에 배정된다. 또한 두 번째 주문은 0.13 ∈ (0/3, 1/3)이므로 선행기계 1에, 세 번째 주문은 0.98 ∈ (2/3, 3/3)이므로 선행기계 3에, 네 번째 주문은 0.24 ∈ (0/3, 1/3)이므로 선행기계 1에 각각 할당된다. 비슷한 방법으로 다섯 번째 유전자 0.72는 주문 1의 후행기계 할당에 사용되는데 그 값이 (1/2, 2/2)에 속하므로 첫 번째 주문은 후행기계 2에 배정된다. 같은 방법으로 주문 2는 0.03 ∈ (0/2, 1/2)이므로 후행기계 1에, 주문 3은 0.74 ∈ (1/2, 2/2)이므로 후행기계 2에, 주문 4는 0.33 ∈ (0/2, 1/2)이므로 후행기계 1에 각각 할당된다.

이와 같은 방법으로 염색체로부터 각 주문들의 선행 및 후행기계 할당이 이루어지고 나면 각 기계에 할당된 주문들의 처리순서를 결정해야 하고 나아가 그 염색체에 대한 일정계획이 확정되어야 한다. 이 과정에서는 본 연구의 목적함수인 총 완료시간을 최소화하는 방향으로 간단한 휴리스틱을 만들어 사용하는데, 선행기계군의 경우 착수가능한 주문들 중에서 처리시간이 가장 작은 주문을 먼저 처리하도록 하였다. 이것은 두 단계 일정계획문제의 고전적 해법인 Johnson 규칙(Johnson, 1954)을 활용한 것이다. 반면, 후행기계군의 경우에는 처리시간과 무관하게 먼저 도착하는 (즉, 선행기계의 처리가 먼저 끝나는) 주문을 먼저 처리하도록 한다. 이상과 같이 하나의 염색체로부터 그에 대응하는 하나의 일정계획을 생성하는 절차(decoding)를 구조적으로 표현하면 아래와 같다.

***ASSIGN_FIRST* Algorithm**

Step 1. 염색체 내부의 i번째 유전자 값이 구간($\frac{k-1}{m_1}$, $\frac{k}{m_1}$)에 속하면 주문 i를 선행기계 k (k = 1, 2, ..., m_1) 에 배정하고 기계 k에 배정된 주문들의 집합을 J_k로 놓는다.

Step 2. 선행기계 k가 가용해지는 시점을 ms_k로 놓는다. 그러면 $ms_k = r_{1k}$이다.

Step 3. $k = 1$로 놓는다.

Step 4. $k > m_1$이면 <Step 8>로 간다.

Step 5. 만약 $J_k = \varnothing$ 이면 $k = k+1$로 놓고 <Step 4>로 간다.

Step 6. J_k에 속한 주문 i 중 선행기계 k에서 가장 빨리 작업을 마칠 수 있는 (즉, $\max\{a_i, ms_k\}+p_{ik}$ 값이 가장 작은) 주문을 i^* 로 놓는다.

Step 7. J_k에서 주문 i^*를 제거하고 ms_k=$\max\{a_{i^*}, ms_k\}+p_{i^*k}$ 와 같이 ms_k를 재계산한다. 주문 i^*의 선행기계작업 완료시간을 $f_{1i^*} = ms_k$로 놓고 <Step 5>로 간다.

Step 8. 염색체 내부의 $n+i$번째 유전자 값이 구간($\frac{l-1}{m_2}$, $\frac{l}{m_2}$)에 속하면 주문 i를 후행기계 l(l = 1, 2, ⋯, m_2) 에 배정하고 기계 l에 배정된 주문들의 집합을 J_l로 놓는다.

Step 9. 후행기계 l이 가용해지는 시점을 ms_l로 놓는다. 그러면 $ms_l = r_{2l}$이다.

Step 10. $l = 1$로 놓는다.

Step 11. $l > m_2$이면 ms_l($l = 1, 2, \cdots, m_2$) 중 최대값을 $f_{\max}$로 놓고 알고리즘을 종료한다.

Step 12. 만약 $J_l = \varnothing$ 이면 $l = l+1$로 놓고 <Step 11>로 간다.

Step 13. J_l에 속한 주문 i 중 선행기계작업 완료시간(f_{1i})이 가장 작은 주문을 i^*로 놓는다. J_l에서 주문 i^*를 제거하고 ms_l=$\max\{f_{1i^*}, ms_l\}+q_{i^*l}$과 같이 ms_l을 재계산한 다음 <Step 12>로 간다.

이번에는 순서를 바꾸어 염색체로 주문처리순서를 표현하고 각 주문의 기계할당에 휴리스틱을 사용하는 절차에 대해 설명한다. 사용하는 염색체의 형태는 이전과 동일하게 (0, 1) 구간의 실수를 주문 수의 두 배 만큼 나열하도록 하였다. 각 염색체의 앞쪽 *n*개 유전자는 각 주문이 선행기계에 투입되는 순서를 표현한다. 이를 위해서는 이 *n*개의 유전자 값을 정렬(sorting)하여야 하는데, 정렬한 결과 *i*번째 유전자의 크기가 *j*번째라면 주문 *i*를 선행기계에 *j*번째로 투입하도록 한다. 비슷한 방법으로 뒤쪽 *n*개의 유전자는 각 주문의 후행기계 투입 순서를 표현할 수 있다. 앞서 사용한 염색체(0.56, 0.13, 0.98, 0.24, 0.72, 0.03, 0.74, 0.33)의 경우 앞쪽 4개 유전자를 정렬해보면 가장 큰 값(0.98)을 갖는 세 번째 주문을 선행기계에서 가장 먼저 처리하고 가장 작은 값(0.13)을 갖는 주문 2를 선행기계에서 가장 늦게 처리하게 된다. 같은 방법으로 뒤쪽 유전자 4개를 정렬해보면 후행기계에서의 처리순서는 주문 3, 주문 1, 주문 4, 주문 2와 같은 순서가 됨을 알 수 있다.

이와 같이 각 염색체를 사용해 각 주문들의 선행 및 후행기계 처리순서를 결정하고 나면, 각 주문들을 실제로 선행 및 후행 기계에 할당하고 나아가 실제 처리시간이 결정해야 한다. 이 과정에서는 가장 고전적인 한 단계 동종병렬기계 일정계획 문제의 최적해를 구할 때처럼(Pinedo, 2002) 먼저 작업이 완료되는 기계에 순서가 결정된 주문들을 차례로 할당하는 단순 휴리스틱을 사용하였다. 이상과 같은 방법으로 하나의 염색체로부터 해를 구하는 과정(decoding)을 체계적으로 정리하면 아래 알고리즘과 같다.

***SEQUENCE_FIRST* Algorithm**

Step 1. 선행기계 k가 가용해지는 시점을 ms_k로 놓는다. 그러면 $ms_k = r_{1k}$이다.

Step 2. 염색체의 앞쪽 n개 유전자가 모두 0보다 작으면 <Step 6>으로 간다.

Step 3. 염색체의 앞쪽 n개 유전자 중 가장 큰 값에 해당하는 주문을 i^*로 놓는다.

Step 4. ms_k 값이 가장 작은 선행기계를 k^*로 놓고 주문 i^*를 기계 k^*에 배정한 다음 염색체의 i^*번째 유전자 값을 -1로 놓는다.

Step 5. $f_{1i^*} = ms_{k^*} = \max\{a_{i^*}, ms_{k^*}\} + p_{i^*k^*}$ 로 놓고 <Step 2>로 간다.

Step 6. 후행기계 l이 가용해지는 시점을 ms_l로 놓는다. 그러면 $ms_l = r_{2l}$이다.

Step 7. 염색체의 뒤쪽 n개 유전자가 모두 0보다 작으면 ms_l 중 최대값을 $f_{\max}$로 놓고 알고리즘을 종료한다.

Step 8. 염색체의 뒤쪽 n개 유전자 중 가장 큰 값에 해당하는 주문을 i^*로 놓는다.

Step 9. ms_l 값이 가장 작은 선행기계를 l^*로 놓고 주문 i^*를 기계 l^*에 배정한 다음 염색체의 $(n+i^*)$번째 유전자 값을 -1로 놓는다. $ms_{l^*} = \max\{f_{1i^*}, ms_{l^*}\} + q_{i^*l^*}$ 로 재계산하고 <Step 7>로 간다.

위에서 제시한 염색체 표현방법 (representation) 이외의 유전알고리즘 전개는 다음 내용과 같다. 대상문제가 최소화 문제이므로 적합도 값(F_i : fitness value)은 아래와 같이 모집단 내에서 목적함수 값(즉, 총 완료시간) 중 최대값($Z_{\max}$)과 해당 염색체의 목적함수 값(Z_i)의 차이를 사용한다.

$$F_i = Z_{\max} - Z_i \tag{19}$$

다음은 재생(reproduction) 과정으로 이전 세대의 각 염색체를 그 적합도 값에 따라 다음 세대로 복제하는 절차이다. 본 연구에서는 가장 쉽고도 보편적 방법인 룰렛휠 방법을 사용하였다. 즉, 적합도 값에 비례한 크기의 확률값으로 이전 세대의 염색체를 선택하여 다음 세대의 염색체를 생성하는 것이다. 선택된 염색체를 대상으로 교배(crossover) 및 돌연변이(mutation)와 같은 유전 연산을 적용하여 다음 세대의 염색체를 만들어 낸다. 이 유전 연산에는 매우 다양한 방법들이 있으나 본 연구에서는 역시 가장 쉽고 널리 알려진 방법을 적용한다. 교배는 선택된 두 염색체를 대상으로 한 개의 지점을 선택하여 각각의 유전자를 서로 맞교환하는 방식(one-cut exchange method)을 사용하였고, 돌연변이에서는 임의로 선택된 유전자 값을 (0, 1) 구간의 새로운 난수로 교체하는 방식을 적용하였다. 이러한 재생과정에서는 각 세대의 최우수 염색체가 후속 세대로 넘어가지 않는 경우가 발생할 수 있으므로 본 연구에서는 각 세대의 최우수 염색체 두 개를 선정하여 유전연산을 적용하지 않고 다음 세대로 바로 복사하는 정책(elitist policy)을 사용하였다. 재생과정을 정리하면 아래의 EVOLVE 알고리즘과 같다.

***EVOLVE* Algorithm**

Step 1. 최우수 염색체 두 개는 다음 세대에서 그대로 사용한다.

Step 2. 룰렛휠 방법으로 두 개의 염색체를 선택한 다음 (0, 1) 구간의 난수 하나를 생성한다. 이 난수가 미리 정한 교배확률 P_c보다 작으면 위에서 설명한 교배절차에 따라 새로운 염색체 두 개를 생성하고, 그렇지 않으면 두 염색체를 그대로 사용한다. 염색체의 수가 모집단 수에 이를 때까지 이 과정을 반복한다. P_c의 값은 실험을 통해 적당한 값을 찾아 사용한다.

Step 3. 위의 <Step 2>에 의해 생성된 염색체에 속하는 모든 유전자에 대해 (0, 1) 구간의 난수를 하나 생성한 다음, 이 난수가 미리 정한 돌연변이 확률 P_m보다 작으면 (0, 1) 구간에서 새로운 난수를 하나 생성하여 기존의 유전자를 대치한다. 여기서도 P_m의 값은 실험을 통해 선정한다.

앞서 2장에서 LINGO를 사용해 최적해를 구했던 예제를 이 알고리즘으로 풀어보았다. 두 가지 염색체 표현방법 중 어떤 방법을 사용하든 프로그램 수행시간은 0.1초 미만이었고, 최종 목적함수값(즉, 총 완료시간)은 *ASSIGN_FIRST* 알고리즘의 경우 24로 최적해와 일치하였으나 *SEQUENCE_FIRST* 알고리즘의 경우 26으로 최적해에 미치지 못하는 결과를 보여주었다. 이어지는 4장에서 보다 광범위한 시험을 수행하겠지만, 이 사례만 보아도 본 연구에서 제안한 알고리즘이 충분히 짧은 계산시간과 나쁘지 않은 해의 품질을 제공한다는 것을 알 수 있다.

4. 수치 실험

앞서 제시한 유전알고리즘들의 성능을 평가하기 위해 본 절에서는 임의로 생성한 예제들을 사용해 수치실험을 시행한다. 이 과정은 다음과 같은 두 단계로 나누어 진행하기로 한다. 첫 번째 단계에서는 LINGO 시스템을 사용해 최적해를 구할 수 있을 정도로 규모가 작은 문제들을 생성하고 그에 대한 해를 본 연구에서 제안한 알고리즘들과 LINGO 시스템을 사용해 각각 구하여 비교하는 작업을 수행한다. 그 다음으로 두 번째 단계에서는 최적해를 구하기 어려운 비교적 큰 규모의 문제를 생성하여 제안한 두 알고리즘들의 성능을 서로 비교하도록 한다. 또한 이 단계에서는 LINGO 시스템을 30분 동안 수행하고 그 때까지의 최고해를 구하여 두 유전알고리즘의 결과와 비교하기도 한다.

문제의 규모는 주문수(n)와 기계수(m_1 및 m_2)가 결정하므로 이 세 값들의 수준을 각 단계별로 다르게 통제하도록 한다. 우선 첫 번째 단계에서는 주문수를 5와 6 중에서 하나로 하고 기계수를 2와 3 중 하나로 하여 총 8개(n, m_1, m_2) 조합에 대해 실험을 수행한다. 두 번째 단계에서는 주문의 수를 10, 20, 30 중에서 하나로 하고 기계의 수는 2, 3, 4 중에서 하나로 하여 총 27개 (n, m_1, m_2) 조합의 문제를 생성해 실험을 진행한다.

각각의 (n, m_1, m_2) 조합에 대해서는 5개의 문제를 임의로 생성하여 반복실험을 수행한다. 문제를 생성하기 위해서는 a_i, r_{1k}, r_{2l}, p_{ik}, q_{il}($i = 1, 2, \cdots, n$, $k = 1, 2, \cdots, m_1$, $l = 1, 2, \cdots, m_2$) 값들을 생성하여야 하는데 그 방법은 다음과 같다. 우선 a_i값들을 생성하기 위해 계획대상 주문의 반은 계획시점에 이미 접수되어 있다고 가정하였고(즉, $a_i = 0$), 나머지 반은 $a_i = U[0, 10]$으로 가정하였다. 여기서 $U[a, b]$는 a와 b사이에서 임의로 생성된 정수라는 의미이다. 선행기계 중 하나가 방금 작업을 마쳤음을 표현하기 위해 $r_{11} = 0$으로 놓고 나머지 r_{1k}값은 $U[0, 10]$로 가정한다. r_{2l}값은 $l \le m_1$인 경우에는 선행기계 작업을 마친 다음에 후행기계 작업까지 마쳐야 하므로 기 생성된 r_{1l}값에 $U[0, 10]$를 더한 값으로 놓았고 $l > m_1$인 경우라면 $U[0, 10]$이라고 가정하였다. 마지막으로 처리시간(p_{ik} 및 q_{il})은 모두 $U[5, 10]$으로 가정하였다.

Table 2. Results of Small Problems

n	m_1		m_2 = 2			m_2 = 3		
			ASS	*SEQ*	time	*ASS*	*SEQ*	time
5	2	mean	1.02	1.02	9.4	1.01	1.00	6.0
		min	1.00	1.00	2	1.00	1.00	3
		max	1.04	1.04	14	1.04	1.00	11
	3	mean	1.00	1.00	11.8	1.06	1.01	1.2
		min	1.00	1.00	2	1.00	1.00	1
		max	1.00	1.00	24	1.16	1.05	2
6	2	mean	1.05	1.05	158.8	1.05	1.04	113.4
		min	1.00	1.03	16	1.00	1.00	36
		max	1.07	1.07	345	1.11	1.07	388
	3	mean	1.00	1.02	1286.8	1.06	1.03	51.6
		min	1.00	1.00	282	1.00	1.00	2
		max	1.00	1.04	-	1.18	1.09	172

ASS : result of *ASSIGN_FIRST* algorithm
SEQ : result of *SEQUENCE_FIRST* algorithm

첫 번째 단계의 실험결과는 <Table 2>에 정리되어 있다. 앞서 설명한 바와 같이 이 단계에서는 8개의 (n, m_1, m_2) 조합에 대해 실험을 수행하였다. 우선 각 조합에 대해 5개의 문제를 임의로 생성하고, 각 문제에 대한 최적해를 LINGO 시스템으로 구한 다음, 본 연구에서 제안한 2개의 유전알고리즘으로 각각 해를 구한다. 예를 들어 표의 좌상부 셀은 (n, m_1, m_2) = (5, 2, 2)일 때 *ASSIGN_FIRST* 알고리즘의 결과와 최적해를 비교한 결과로서 그 값 1.02, 1.00, 1.04는 다음과 같은 절차에 의해 구해졌다.

1) (n, m_1, m_2) = (5, 2, 2)인 상황에서 임의로 생성된 5개의 서로 다른 문제에 대해 LINGO 시스템으로 최적해를 구한 결과 그 최종완료시간(makespan)은 25, 25, 27, 24, 26 등이었다.
2) *ASSIGN_FIRST* 알고리즘을 적용한 유전알고리즘으로 같은 문제를 푼 결과는 26, 26, 27, 24, 26 등이었다.
3) *ASSIGN_FIRST* 알고리즘의 결과를 최적 결과로 나누면 26/25 = 1.04, 26/25 = 1.04, 27/27 = 1.00, 24/24 = 1.00, 26/26 = 1.00 등이 된다.
4) 이 5개의 값들에 대한 평균 1.02, 최소 1.00, 최대 1.04 등을 표에 수록하였다.

같은 예제들에 대해 *SEQUENCE_FIRST*와 최적해를 비교한 결과가 위 값들의 우측 셀에 1.02, 1.00, 1.04로 표시되어 있으며, 그 다음 오른쪽에는 최적해를 구하기 위한 LINGO 시스템의 컴퓨팅 시간을 표시하였다. 앞서 생성한 5개 문제에 대한 계산시간은 10초, 2초, 14초, 14초, 7초였으므로 이 값들에 대한 평균 9.4, 최소 2, 최대 14 등이 표시된 것이다. 유전알고리즘에 대한 계산시간은 별도로 표시하지 않았는데 이것은 모든 문제에 대해 유전알고리즘 계산시간이 0.1초 미만이었기 때문이다. 한편 표의 아래쪽을 보면 (n, m_1, m_2) = (6, 3, 2)인 경우에는 계산시간의 최댓값이 기록되지 않았는데, 이것은 생성된

5개 문제 중 한 문제에 대한 최적해를 LINGO 시스템이 1시간 이 경과하도록 찾지 못하여 시스템을 강제로 멈추었기 때문이다. 따라서 이 경우의 평균과 최솟값은 나머지 4개 문제의 계산시간을 정리한 것이다.

<Table 2>의 값들을 관찰해보면 본 연구에서 제안한 알고리즘들의 성능에 대한 개략적인 평가를 할 수 있으며 그 결과는 다음과 같이 정리할 수 있다.

1) *ASSIGN_FIRST* 알고리즘의 경우 모든 (n, m_1, m_2) 조합에서 min 값은 1.00이고 max 값은 1.18 이하이다. 이것은 모든 조합에서 생성된 5개 문제 중 적어도 한 개 이상에서 최적해를 찾았음을 의미하며 또한 최악의 경우에도 최적해에 비해 18% 정도 나쁜 해를 찾아 주었음을 의미한다.
2) *SEQUENCE_FIRST* 알고리즘의 경우에도 (6, 2, 2)를 제외한 모든 (n, m_1, m_2) 조합에서 min 값은 1.00이고, max 값은 모든 경우에서 1.09 이하이다. 최적해를 찾지 못한 경우가 한 번 있었다는 것은 *ASSIGN_FIRST* 알고리즘에 비해 부족한 점이지만 최악에 대한 결과는 *SEQUENCE_FIRST* 알고리즘이 우수하였다.
3) (n, m_1, m_2) 조합에 따른 두 알고리즘의 우열에 일관성을 발견하기는 어려웠다. 따라서 우수한 해를 원한다면 알고리즘 수행시간이 매우 짧으므로 두 알고리즘의 해를 모두 구하여 둘 중 우수한 결과를 사용할 수도 있다. 또한 우연변동이 존재하는 유전알고리즘의 특성을 고려할 때 알고리즘을 두 번 이상 실행하고 그 중 최상의 결과를 사용하는 것도 알고리즘 수행시간이 충분히 짧으므로 가능할 것이다.

다음으로 두 번째 단계의 실험결과는 <Table 3>에 정리하였다. 이 실험에서는 주문의 수를 10, 20, 30 중에서 하나로 하고 선행 및 후행 기계의 수는 2, 3, 4 중 하나로 하는 총 27개(n, m_1, m_2) 조합의 문제를 관찰하였다. 이런 규모의 문제에 대해서는 LINGO 시스템을 통한 최적해를 현실적인 시간 내에 구할 수가 없으므로, 본 연구에서 제시한 2가지 형태의 유전알고리즘으로 해를 구한 다음 *SEQUENCE_FIRST* 기반 알고리즘의 결과(총 완료시간)를 *ASSIGN_FIRST* 기반 알고리즘의 결과로 나눈 값을 표에 제시하였다. 따라서 표에 나타난 값이 1보다 크면 *SEQUENCE_FIRST* 알고리즘의 결과가 *ASSIGN_FIRST* 알고리즘보다 크므로 *ASSIGN_FIRST* 알고리즘이 우수하다는 것을 의미하고, 반대로 결과값이 1보다 작으면 *SEQUENCE_FIRST* 알고리즘이 우수하다는 것을 의미한다.

또한 이 표에는 앞서 언급한 것처럼 최적해 대신 계산시간을 30분으로 제한한 LINGO 시스템의 해를 유전알고리즘의 해와 비교한 결과를 제시하였다. 27개의 각 조합에서 생성된 5개의 문제 중 첫 번째 문제에 대해 LINGO 시스템의 결과와 두 유전알고리즘의 결과 중 더 나은 해를 비교한 것이다. 예를 들어, 이 표의 좌상부 셀은 (n, m_1, m_2)=(10, 2, 2)인 경우에 대한 결과로서 표시된 값 1.01/0.94/1.08과 0.98은 다음과 같은 절차에 의해 구해졌다.

Table 3. Results of Large Problems

n	m_1		m_2		
			2	3	4
10	2	mean/ min/ max opt in 30min	1.01/0.94/1.08 0.98	1.00/0.98/1.02 0.95	0.99/0.97/1.02 0.96
	3		1.02/1.00/1.06 1.00	0.97/0.95/1.00 0.97	0.96/0.89/1.00 0.97
	4		1.02/0.98/1.08 1.00	0.97/0.91/1.03 1.00	0.99/0.93/1.04 1.03
20	2	mean/ min/ max opt in 30min	1.06/1.03/1.10 0.75	1.02/1.00/1.06 0.80	1.01/0.99/1.03 0.85
	3		1.03/1.01/1.07 0.84	1.05/0.98/1.10 0.76	1.00/0.98/1.02 0.77
	4		1.01/0.99/1.04 0.79	1.02/0.98/1.08 0.72	1.01/0.96/1.07 0.63
30	2	mean/ min/ max opt in 30min	1.09/1.06/1.14 0.43	1.05/1.04/1.07 0.70	1.01/0.99/1.04 0.84
	3		1.04/1.01/1.07 0.63	1.07/1.01/1.11 0.68	1.06/1.01/1.10 0.68
	4		1.04/1.02/1.05 0.74	1.04/1.01/1.07 0.74	1.03/0.98/1.07 0.51

1) 계획대상 작업수가 10개이고 선행 및 후행기계의 수가 모두 2대인 상황에서 생성된 5개의 서로 다른 문제에 대해 *ASSIGN_FIRST* 기반의 유전알고리즘으로 해를 구한 결과 그 최종완료시간은 45, 40, 47, 37, 42 등이었다. 한편 동일한 5개 문제에 대한 *SEQUENCE_FIRST* 기반 유전알고리즘의 결과는 45, 42, 44, 40, 42 등이었다.
2) *SEQUENCE_FIRST* 기반 유전알고리즘의 결과를 *ASSIGN_FIRST* 기반 유전알고리즘의 결과로 나누면 각각 1.00, 1.05, 0.94, 1.08, 1.00 등이 된다.
3) 이 5개의 값들에 대한 평균, 최소, 최대는 각각 1.01, 0.94, 1.08이다.
4) 5개의 문제 중 첫 번째 문제에 대해 30분 동안 LINGO 시스템을 수행한 결과 최고해는 46이었다. 두 유전알고리즘의 결과 (45, 45) 중 더 나은 해(45)를 LINGO 시스템의 결과 46으로 나누면 0.98이 된다.

<Table 3>의 값들을 관찰해보면 본 연구에서 제안한 두 유전알고리즘들의 상대적인 성능을 비교해볼 수 있으며 그 결과는 다음과 같이 정리할 수 있다.

1) 계획대상 주문수(n)가 커지면 대체로 표의 값들이 증가하는 경향을 보여준다. 이것은 n이 커질 때 *ASSIGN_FIRST* 알고리즘이 좀 더 우수한 해를 제공한다는 것을 의미한다. 그러나 경우에 따라 반대의 경우도 나타나므로 정확히 n이 얼마일 때 *ASSIGN_FIRST* 알고리즘이 우수하다고 단정하기는 매우 어렵다.
2) 두 알고리즘의 우열에 (m_1, m_2) 조합이 미치는 영향을 발견하기는 어렵다. 위에서 n의 영향에 대해 언급하였지만, 전체적으로 (n, m_1, m_2) 조합에 따라 하나의 알고리즘을 선택하는

것은 위험하다고 생각된다.

3) 최저의 min 값은 0.89이고 최대의 max 값은 1.14이다. 즉, 두 알고리즘의 차이는 가장 클 때 11~14% 정도이므로 대부분의 경우 큰 차이가 없다고 볼 수 있다.
4) 계획대상 주문수(n)가 10개일 때는 LINGO 시스템의 결과와 유전알고리즘의 결과에 큰 차이가 없으나 주문수가 커질수록 유전알고리즘의 결과가 LINGO 시스템의 결과보다 더 우수해진다. 즉, 주문수가 커질수록 본 연구에서 제안한 유전알고리즘의 효용성이 커진다고 할 수 있다.
5) 관찰 내용을 종합하면, 앞서 <Table 2>와 비슷하게, 두 알고리즘의 우열에 일관성을 발견하기 어려우므로 우수한 해를 원한다면 알고리즘 수행시간이 매우 짧은 성질을 이용하여 두 개의 알고리즘을 모두 두 번 이상 실행하여 총완료시간이 최소가 되는 결과를 적용하는 것이 가능할 것이다.

5. 결 론

본 연구에서는 두 개의 단계가 모두 이종병렬기계로 이루어진 두단계 흐름생산라인의 일정계획문제를 다루었다. 우선 문제의 상황을 잘 표현해주는 혼합정수계획(Mixed Integer Linear Program) 모형을 최초로 제시하였고, 이 모형의 유효성을 평가하기 위하여 최적화 소프트웨어를 사용해 작은 크기의 예제들을 해결하였다. 실험 결과 작은 크기의 문제는 최적화 모형을 사용해 최적해를 쉽게 구할 수 있었으나, 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 두 종류의 유전알고리즘 기반 휴리스틱 알고리즘을 개발하였다. 다양한 수치실험을 통해 본 연구에서 제시된 알고리즘들이 매우 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다.

저자들은 본 연구의 후속으로 다음과 같은 내용을 고려하고 있다. 우선 셋업시간을 고려한 연구이다. 본 연구에서는 셋업시간이 처리시간에 포함될 수 있는 상황을 가정하였으나 좀 더 현실적인 모형이 되려면 셋업시간을 처리시간에서 독립시켜야 하고, 나아가 셋업시간이 처리순서에 종속적인 상황에 대해서도 고려해야 할 것이다. 두 번째로는 목적함수의 변화이다. 본 연구에서는 총 완료시간을 목적함수로 사용하였으나 납기를 고려한 목적함수가 더 중요한 경우도 있을 것이다. 또 다른 문제로는 셋업 작업자 제약을 고려한 연구도 가능할 것이다. 일반적으로 한 명(또는 팀)의 작업자가 여러 대의 기계에 대한 셋업을 처리하는 경우가 많기 때문에 이러한 제약을 추가하는 것도 현실적인 의미가 있으리라 생각한다.

참고문헌

Agarwal, A., Colak, S., Jacob, V. S., and Pirkul, H. (2006), Heuristics and Augmented Neural Networks for Task Scheduling with Non-Identical Machines, *European Journal of Operational Research*, **175**(1), 296-317.

Chen, C. L. and Chen, C. L. (2009), A Bottleneck-based Heuristicfor Minimizing Makespan in a Flexible Flow Line with Unrelated Parallel Machines, *Computers and Operations Research*, **36**(11), 3073-3081.

Cheng, T. C. E. and Sin, C. C. S. (1990), A State-of-the-Art Review of Parallel-Machine Scheduling Research, *European Journal of Operational Research*, **47**(3), 271-292.

Davis, E. and Jaffe, J. M. (1981), Algorithms for Scheduling Tasks on Unrelated Processors, *Journal of the ACM*, **28**(4), 721-736.

Dawande, M., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C. (2005), Sequencing and Scheduling in Robotic Cells : Recent Developments, *Journal of Scheduling*, **8**(5), 387-426.

De, P. and Morton, T. E. (1980), Sequencing to Minimize Makespan on Unrelated Parallel Processors, *Decision Sciences*, **11**(4), 586-602.

Ghirardi, M. and Potts, C. N. (2005), Makespan Minimization for Scheduling Unrelated Parallel Machines : A Recovering Beam Search Approach, *European Journal of Operational Research*, **165**(2), 457-467.

Gupta, J. N. D., Hariri, A. M. D., and Potts, C. N. (1997), Scheduling a Two-Stage Hybrid Flow Shop with Parallel Machines at the First Stage, *Annals of Operations Research*, **69**, 171-191.

Gupta, J. N. D. and Tunc, E. A. (1991), Schedules for a Two-Stage Hybrid Flowshop with Parallel Machines at the Second Stage, *International Journal of Production Research*, **29**(7), 1489-1502.

Hariri, A. M. A. and Potts, C. N. (1991), Heuristics for Scheduling Unrelated Parallel Machines, *Computers and Operations Research*, **18**(3), 323-331.

Hop, N. V. and Nagaur, N. N. (2004), The Scheduling Problem of PCBs for Multiple Non-Identical Parallel Machines, *European Journal of Operational Research*, **158**(3), 577-594.

Horowitz, E. and Sahni, S. (1976), Exact and Approximate Algorithms for Scheduling Non-Identical Processors, *Journal of the ACM*, **23**(2), 317-327.

Ibarra, O. H. and Kim, C. E. (1977), Heuristic Algorithm for Scheduling Independent Tasks on Nonidentical Processors, *Journal of the ACM*, **24**(2), 280-289.

Jeong, N. K. and Jeong, M. Y. (2000), A Scheduling Support System for Non-Identical Parallel Machine Lines, *Korean Management Science Review*, **17**(2), 67-73.

Johnson, S. (1954), Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included, *Naval Research Logistic Quarterly*, **1**, 61-68.

Joo, C. M. and Kim, B. S. (2012), Non-Identical Parallel Machine Scheduling with Sequence and Machine Dependent Setup Times Using Meta-Heuristic Algorithms, *Industrial Engineering & Management Systems*, **11**(1), 114-122.

Kim, D. W., Kim, K. H., Jang, W., and Chen, F. F. (2002), Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing, *Robotics and Computer Integrated Manufacturing*, **18**(3-4), 223-231.

Koh, S. G. and Mahardini, K. A. (2014), Heuristics for Non-Identical Parallel Machine Scheduling with Sequence Dependent Setup Times, *Journal of the Korean Institute of Industrial Engineers*, **40**(3), 305-312.

Lee, J. S. and Park, S. H. (1999), Scheduling for Two Stage Mixed Flow Production System with Non-Identical Parallel Machines, *Journal of the Korean Institute of Industrial Engineers*, **25**(2), 254-265.

Lenstra, J. K., Shmoys, D. B., and Tardos, E. (1990), Approximation

Algorithm for Scheduling Unrelated Parallel Machines, *Mathematical Programming*, **46**(1), 259-271.
Narashimhan, S. L. and Panwalker, S. S. (1984), Scheduling in a Two-Stage Manufacturing Process, *International Journal of Production Research*, **22**(4), 555-564.
Piersma, N. and Van Dijk, W. (1996), A Local Search Heuristic for Unrelated Parallel Machine Scheduling with Efficient Neighborhood Search, *Mathematical and Computer Modeling*, **24**(9), 11-19.
Pinedo, M. (2002), *Scheduling : Theory, Algorithms, and Systems*, Prentice Hall, New Jersey, U.S.A..
Potts, C. N. (1985), Analysis of a Linear Programming Heuristic for Scheduling Unrelated Parallel Machines, *Discrete Applied Mathematics*, **10**(2), 155-164.
Randhawa, S.U. and Kuo, C.H. (1997), Evaluating Scheduling Heuristics for Non-Identical Parallel Processors, *International Journal of Production Research*, **35**, 969-981.
Ruiz, R. and Vazquez-Rodriguez, J. A. (2010), The Hybrid Flow Shop Scheduling Problem, *European Journal of Operational Research*, **205**(1), 1-18.
Srivastava, B. (1998), An Effective Heuristic for Minimizing Makespan on Unrelated Parallel Machines, *Journal of the Operational Research Society*, **49**(8), 886-894.
Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, F., Izadi, M., and Sassani, F. (2009), Design of a Genetic Algorithm for Bi-Objective Unrelated Parallel Machines Scheduling with Sequence-Dependent Setup Times and Precedence Constraints, *Computers & Operations Research*, **36**(12), 3224-3230.
Vallada, E. and Ruiz, R. (2011), A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times, *European Journal of Operational Research*, **211**(3), 612-622.
Van de Velde, S. L. (1993), Duality-based Algorithms for Scheduling Unrelated Parallel Machines, *ORSA Journal on Computing*, **5**(2), 192-205.

저자소개

고시근 : 고려대학교 산업공학과에서 1986년 학사학위를 취득하고 KAIST 산업공학과에서 1988년 석사, 1993년 박사학위를 취득하였다. 1991년부터 현재까지 부경대학교 시스템경영공학부에 재직 중이다. 관심분야는 생산일정계획 및 공급사슬 최적화에 대한 이론적 접근과 현장 적용을 위한 컴퓨터시스템 개발이다.

서원철 : POSTECH 산업경영공학과에서 학사 및 박사학위를 취득하고, 삼성디스플레이, 한국지식재산연구원을 거쳐 현재는 부경대학교 시스템경영공학부 부교수로 재직 중이다. 주요 연구관심분야는 Patent Mining, Technology Intelligence, Technology Opportunity Identification 이다.

김상운 : 경성대학교 산업공학과에서 1990년 학사학위를 취득하고 부경대학교 시스템경영공학과에서 2005년 석사 후 2013년에는 부경대학교 MOT 대학원 박사과정을 수료하였다. 현재 'DQS Korea LLC'의 경영시스템 선임 심사원 및 '㈜엠임팩트씨엔씨'의 수석컨설턴트로 활동하고 있으며 주요 관심분야는 중소기업 경영혁신 활동, 글로벌 기업 경영시스템 사례분석, 경영개선 기법 등이다.