공 학 박 사 학 위 논 문

셋업을 고려한 이종병렬기계 일정계획

2019년 8월

부 경 대 학 교 대 학 원

기술경영협동과정

손 준 석

공 학 박 사 학 위 논 문

셋업을 고려한 이종병렬기계 일정계획

지도교수 고 시 근

이 논문을 공학박사 학위논문으로 제출함.

2019년 8월

부 경 대 학 교 대 학 원

기술경영협동과정

손 준 석

손준석의 공학박사 학위논문을 인준함.

2019년 8월 23일

위원장 공학박사 권 혁 무 (인)

위 원 공학박사 구 평 회 (인)

위 원 공학박사 고 창 성 (인)

위 원 공학박사 김 정 배 (인)

위 원 공학박사 고 시 근 (인)

목 차

<표 차 례>

[그 림 차 례]

Scheduling of non-identical parallel machines with setup times

Jun Seok Sohn

Department of Management of Technology, The Graduate School,
Pukyong National University

Abstract

This thesis studies three scheduling problems for non-identical parallel machine systems that can be easily found in small and medium sized manufacturing companies. Common features of the three problems are as follows; 1) minimizing the makespan, 2) machine dependent setup times, and 3) machine dependent processing times. For all the problems, this thesis proposes mixed integer linear programming models, and using the models, the optimal solutions for relatively small example problems can be found by a commercial optimization software. However, since all of the three problems are NP-hard and the size of a real problem is large, some heuristic algorithms including genetic algorithm to solve the practical big-size problems in a reasonable computational time are proposed for each problem. And then, to assess the performances of the algorithms, a computational experiment is conducted in each chapter.

After an introduction to the non-identical parallel machine scheduling problem and literature review for the problems in chapter 1 and a brief explanation for the setup operations in chapter 2, the first problem out of the three problems is studied in chapter 3. A special feature of this problem is that the setup times are sequence dependent. For the problem, a mathematical model and four heuristic algorithms are proposed. Through a computational experiment, it is found that the heuristic algorithms show different performances as the problem

characteristics are changed and some simple heuristics show better performances than genetic algorithm based heuristics for the case when the numbers of jobs and/or machines are large.

A special feature of the second problem in chapter 4 is that the setup operations for all machines are performed by a single setup operator. For the problem, a mathematical model and three genetic algorithm based heuristics are proposed. From a computational experiment, it is found that some heuristic algorithms show very good performances.

Lastly, the third problem, in which the machine dependent setup time of each job can be divided into internal and external setup times, is studied in chaper 5. The internal setup refers to those setup actions that inevitably require that the machine be stopped, and the external setup refers to actions that can be taken while the machine is operating. To solve the problem, another mathematical model is developed and a genetic algorithm based heuristic is proposed. Through a computational experiment, the heuristic algorithm shows very good performances.

Keywords : non-identical parallel machine, internal and external setup, scheduling, single setup-operator, genetic algorithm

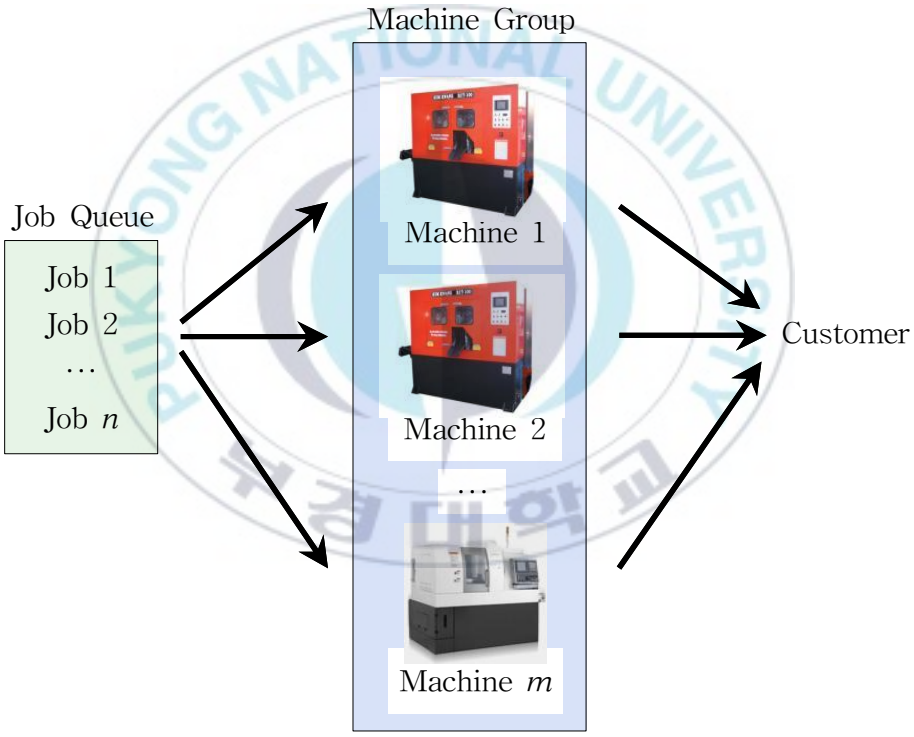
Ⅰ. 서 론

1. 연구의 배경

일반적으로 대규모 제조기업 에서는 비교적 복잡한 구조의 제품을 생산하기 위해 여러 단계의 공정으로 이루어진 생산시스템을 운영하는데, 이 경우 제품들은 각 단계의 공정을 거치며 점진적으로 완제품의 형태를 갖추어간다. 반면, 중소 제조기업의 경우 다른 제조기업(원청회사)으로부터 주문을 받아 비교적 간단한 구조의 전문화된 제품을 생산하는 경우가 많은데, 본 연구에서는 이러한 형태의 중소규모 제조기업 에서 발생하는 생산 일정계획 문제를 해결하고자 한다. 이 기업의 핵심 제조시스템은 [그림 1-1]과 같이 유사한 기계들의 그룹으로 이루어진 한 단계의 공정으로 이루어져 있는데, 여러 고객사들로부터 유사한 종류의 제품에 대한 주문을 받아 이 시스템에서 처리하고 있다.

각 주문은 그룹에 속한 기계들 중 하나를 통해 처리되는데, 이 기계들은 일반적으로 기업이 한꺼번에 일괄 구입한 것이 아니고 생산시스템을 운영하면서 필요에 따라 한 대씩 구매하였기 때문에 용도는 유사하지만 그 성능은 조금씩 차이가 나는 것이 보통이다. 이렇게 용도는 같지만 성능 및 사양이 서로 다른 기계그룹을 '이종병렬기계(Non-identical Parallel Machine)' 그룹이라고 부른다. 이러한 이종병렬기계 생산시스템은 플라스틱 사출이나 기계 가공 등 다양한 분야의 기업에서 쉽게 찾아볼 수 있다.

본 연구의 동기가 된 기업은 플라스틱 사출을 전문으로 하는 소규모 기업으로서 3~5대의 기계로 이루어진 기계그룹을 여러 개 운영하고 있다. 처리하는 주문의 종류는 매우 다양한데 유사성에 따라 주문들을 분류하여 각 기계그룹에 할당하고, 할당된 기계그룹 내에서는 기계의 셋업시간과 처리시간 등 여러 가지 상황을 고려해 각 주문을 적당한 기계에 배정한 다음 처리하게 된다.



[그림 1-1] 이종병렬기계 생산시스템의 예

이러한 상황은 비단 소규모 기업 뿐 아니라 다품종 소량생산 방식으로 항공부품 등을 제작하는 대규모 공장의 각 제조 파트에도 나타

나는 문제이기도 하다. 대규모 공장의 생산기계나 설비 배치 시 용도가 비슷한 기계들을 그룹으로 묶어 배치하게 되며, 다품종으로 여러 주문이 동시에 발생하는 경우가 많으므로 이러한 이종병렬기계의 생산 공정 문제는 다양한 장소에서 발생할 가능성이 높다고 할 수 있다. 이러한 이종병렬기계 시스템에서 처리되는 주문들은 지속형과 일회성의 두 가지 형태로 분류할 수 있다. 일회성 주문은 글자 그대로 반복성이 없는 주문인데, 지속형 주문이란 장기간에 걸쳐 지속적으로 들어오는 주문으로서 하루에 일정량 씩 매일 (혹은 자주) 납품해야하는 형태를 갖는다. 이 경우 하루 생산량이 기계의 하루 생산능력을 초과하면 한 대 이상의 기계가 그 제품의 생산을 전담해야겠지만 (전담기계의 경우는 본 연구의 고려대상이 아님) 일반적으로는 하루 생산량이 기계 생산능력보다 작아서 다른 일회성 혹은 지속형 주문들과 기계를 공유하는 것이 보통이다. 이러한 지속형 주문은 일자별 생산량을 하나의 주문으로 취급하면 납기가 서로 다른 여러 개의 주문으로 분리해서 생각할 수 있다. 그런데 이 분리된 주문들은 일회성 주문과는 다른 특징이 있는데, 그것은 이 주문들이 원래 한 종류의 제품에 대한 주문이었으므로 분리된 주문들 사이에는 기계 준비시간(셋업시간)이 없다는 것이다.

이렇게 여러 개로 나누어진 지속형 주문들과 일회성 주문들이 특정 기계군(이종병렬기계 시스템) 앞에서 처리를 기다리고 있을 때, 각 주문들을 어느 기계에 할당할 것인지 그리고 할당된 기계 안에서 처리되는 순서는 어떻게 되는지를 결정하는 것이 본 연구의 해결 과제이다. 이 과제의 해결에 있어 중요한 변수는 기계의 성능차이와 주문에 대한 셋업시간이다. 즉, 기계의 성능차이로 인해 각 주문을 어느 기계에 할당하느냐에 따라 그 주문의 처리시간이 다르게 되어 주문의

기계할당 문제가 중요해지고, 또한 주문의 처리순서에 따라 셋업시간이 달라지기 때문에 주문의 처리순서 결정문제도 중요해지는 것이다.

일정계획 문제의 목적함수로는 크게 작업 전체완료시간(makespan)의 최소화와 납기지연의 최소화가 있으나 본 연구에서는 대부분의 생산시스템에서 목표로 하는 생산성 제고를 위해 전체 작업완료시간의 최소화를 해결 과제의 목적함수로 둔다. 작업완료시간을 최소화하면 기계의 가동률이 높아지고 따라서 시스템의 생산성이 제고되는 효과를 기대할 수 있다.

여기까지는 몇 가지 특이한 조건들을 제외하면 기존의 일반적인 이종병렬기계 일정계획문제와 상당히 유사한 상황이라고 할 수 있을 것이다. 하지만, 위에서 소개한 사례기업에는 일반적인 이종병렬기계 문제와 다른 매우 독특한 상황이 존재하는데, 그것은 각 기계그룹별로 셋업 작업을 수행하는 작업자가 1명 혹은 2~3명으로 제한되어 있다는 것이다.

만약 작업자가 1명이라면 그 기계그룹에서는 한 번에 오직 한 대의 기계에서만 셋업이 진행될 수 있고, 작업자가 두 명이라면 그 기계그룹에서는 셋업이 기계 두 대까지 동시에 진행될 수 있다는 것을 의미한다. 이와 같은 셋업 작업자 수 조건은 대상시스템의 일정계획에 매우 큰 제약으로 작용한다. 본 연구에서는 작업자 조건을 고려하지 않은 일정계획 문제에서 출발하여 셋업작업자의 제약이 있는 경우(이 때 작업자 수가 기계 대수보다 작다고 가정함)로 확장한 문제들을 해결하고자 한다.

2. 국내외 연구동향

국내외적으로 생산설비의 일정계획을 다룬 연구는 1950년대 중반부터 상당히 많이 수행되어왔다. Allahverdi et al.(2008)에 의하면 수천 건의 관련 문헌을 찾을 수 있으며, 이 연구들의 대부분은 셋업시간이 무시할 수 있을 정도로 작거나 작업시간에 포함될 수 있는 형태라고 가정하였다. 셋업시간을 작업시간과 별도로 고려한 연구는 1960년대 중반부터 수행되기 시작하였으며, 관련 연구결과들은 Allahverdi et al.(1999, 2008)과 Potts & Kovalyov(2000)와 같은 서베이 논문에 잘 정리되어 있다. 이 서베이 논문들에는 셋업시간이 작업순서에 무관 및 유관한 상황, 주문들이 묶음(batch)으로 처리되거나 혹은 그렇지 않은 상황, 다양한 기계 상황(기계 한 대, 병렬 기계, 흐름 라인 등)에 대한 기존 연구문헌들을 체계적으로 정리하였다. 특히 Cheng & Sin(1990)은 병렬기계를 다룬 기존 연구문헌들을 조사하기도 하였다.

일반적인 생산일정계획 문제는 이와 같이 많이 연구되었지만 본 연구의 상황인 이종병렬기계 시스템에서 작업순서에 의존하는 셋업시간이 존재하는 경우를 다룬 연구는 그리 많지 않다. 이러한 상황을 다룬 연구는 Marsh & Montgomery(1973)가 최초로서, 그들은 동일기계 혹은 이종기계 병렬시스템에서 셋업시간을 최소화하는 휴리스틱 방법을 제안하였다. 그 후 Guinet(1990)은 순서의존적인 셋업시간을 갖는 이종병렬기계 시스템에서 평균흐름시간을 최소화하기 위한 혼합정수계획모형을 제시하고 모형을 풀기 위한 휴리스틱 알고리즘을 개발하였다. Zhu & Heady(2000)는 비슷한 상황에서 조기완료 및 지연완료를 최소화하는 혼합정수계획 모형을 개발하였으며 그 모형을 사

용해 기계가 3대이고 주문이 9개인 문제의 최적해를 적정시간 내에 도출하였다. Weng et al.(2001)은 비슷한 상황에서 각 작업들의 완료시간 가중합을 최소화하는 문제를 해결하기 위해 7개의 간단한 휴리스틱을 제안하였으며 수치실험을 통해 그 휴리스틱들을 평가하였다.

국내에서는 전태웅 & 강맹규(1995)가 셋업시간에 대한 고려 없이 이종병렬기계 시스템의 납기지연 작업수를 최소화하기 위한 Tabu Search 방법론을 제안하였다. 강용혁 등(1998)도 셋업시간을 고려하지 않은 이종병렬기계 시스템에서 각 주문들이 서로 다른 납기를 가질 때 납기지연주문의 수를 최소화하는 알고리즘을 개발하였다.

비교적 최근 들어 관련연구의 빈도가 높아지고 있는데, Hop & Nagaur(2004)는 PCB 생산라인의 이종병렬기계 상황에서 순서에 의존하는 준비시간을 포함한 총 완료시간을 최소화하기 위하여 수리적 모형을 제시한 다음 유전알고리즘 기반의 해법을 제안하였다. Agarwal et al.(2006)은 이종병렬기계시스템의 총완료시간(makespan)을 최소화하기 위해 12개의 휴리스틱과 augmented neural network 방법론을 제안하고 이 해법들을 비교·평가하였다. Li & Yang(2009)은 이종병렬기계 상황을 다룬 연구들 중에서 총 가중완료시간의 최소화를 목적으로 하는 (즉, 납기를 고려하지 않은) 연구들을 수집하고 해법에 따라 분류한 결과를 소개하였다. Tavakkoli-Moghaddam et al.(2009)은 순서 및 기계에 의존하는 셋업시간을 고려한 이종병렬기계시스템에서 지연작업의 수와 총 완료시간을 최소화하기 위한 유전알고리즘 기반의 해법을 제안하였다. Gharegozli et al.(2009)은 순서에 의존하나 기계와는 무관한 셋업시간을 가정하고 총 가중흐름시간과 총 가중지연시간을 동시에 최소화하기 위한 혼합정수 목표계획모형을 개발하였다. 이 두 연구에서는 공통적으로 주문들은 각각의 도착시간을 갖고 있어

서 이 시간 이전에는 작업이 진행될 수 없음을 가정하였다. Vallada & Ruiz(2011)는 순서 및 기계 의존적인 셋업시간을 고려한 문제를 해결하기 위해 지역탐색 알고리즘과 유전알고리즘을 결합한 알고리즘을 개발하였다. 가장 최근에는 Joo & Kim(2012)이 순서 및 기계에 의존하는 기계준비시간을 고려한 이종병렬기계 일정계획문제를 다루었다. 총 완료시간을 최소화하는 수리모형을 제시하고 유전알고리즘 및 자기진화(self-evolution) 알고리즘 기반의 두 해법을 개발하고 그 성능을 비교·평가하였다.

조금 다른 형태의 관련연구로, 강용하 등(2007)은 가공시간은 동일하지만 기계에 따라서 재작업 확률이 달라지는 상황에 대한 일정계획문제를 다루었다. 기계에 따라 재작업 확률이 달라지므로 이종병렬기계 문제로 분류될 수 있으나 각 작업에 대한 총 가공시간은 재작업의 불확실성으로 인해 확률적인 변동성을 가지게 된다. Ko et al.(2010)도 유사한 문제를 다루었다. 병렬기계에서 가공시간은 동일하지만 품질이 정규분포를 따르는 확률변수이고 그 품질에 따라 재작업이 발생하는 상황에서 납기지연을 최소화하는 주문-기계 할당 알고리즘을 개발하였다. 서정하 등(2011)은 기계에 따라 서로 다른 작업준비시간, 가공시간, 재작업확률을 가정하여 좀 더 일반화된 상황을 가정한 모형을 제시하고 평균 완료시간을 최소화하는 알고리즘을 제안하였다.

3. 기존연구의 문제점 및 본 연구의 방향

본 연구의 대상문제가 갖는 가장 중요한 특성은 1)이종병렬기계,

2)순서에 따라 변하는 셋업시간, 3)기계에 따라 변하는 셋업시간, 4)기계 가용시간, 5)주문 도착시간, 6)셋업작업자 제약 등 6개로 요약할 수 있다. 이 6개 항목을 기준으로 기존의 유사한 연구들과 본 연구과제의 차이점을 살펴본다. 우선 Zhu & Heady(2000)는 이 6개 항목 중 1), 2), 5) 등 3개 항목이 본 연구와 동일하다. 이종기계이므로 각 주문의 처리시간은 기계에 따라 다르다고 가정하였으나 셋업시간은 순서에만 의존하고 기계에는 독립적이라고 가정하였다.

또한 Tavakkoli-Moghaddam et al.(2009)은 1), 2), 3) 등 3개 항목이 본 연구와 동일하며, Gharegozli et al.(2009)은 1), 2), 5) 등 3개 항목이 동일하다. 마지막으로 Joo & Kim(2012)은 1), 2), 3), 5) 등 4개 항목이 동일하다.

이와 같이 기존의 연구들 중 본 연구과제와 동일한 상황을 다룬 연구는 아직 없었다. 4번째 항목은 계획시점 현재 그룹 내 일부 기계는 처리중인 작업이 있어서 다른 주문을 바로 처리할 수 없다는 것을 의미한다. 실제로 본 연구에서 다루는 일정계획 문제가 발생하는 시점은 하나의 기계에서 생산이 종료되었을 때이며, 계획대상 주문들은 그 시점 이전에 접수되어 있거나 미래의 어떤 시점에 접수되기로 예약되어 있는 주문들이다. 따라서 4번째 항목과 같이 계획시점에 각 기계들은 처리중인 주문을 갖고 있으며 그 완료시간(즉, 각 기계의 가용시간)은 모두 다른 값을 가지게 되는 것이다.

이와 같이 본 연구과제에서는 기존의 연구들이 간과한 현실적인 상황들을 추가적으로 고려한 문제를 1차적으로 해결하고자 한다. 그 내용은 앞서 설명한 6개 중 1) ~ 5) 항목으로 요약할 수 있다. 이 문제를 해결한 다음에는 셋업담당 작업자의 수를 제약조건으로 갖는 문제를 해결하고자 한다. 기존의 연구 중에는 셋업담당자 제약을 고려

한 연구를 찾을 수 없었으며, 따라서 본 연구에서는 앞서 언급한 것처럼 셋업담당자 수의 제약이 있는 경우와 나아가 셋업작업의 성격에 따라 내부와 외부로 나뉘는 상황까지 가정하여 연구하고자 한다.

장의 구성 순서대로 정리하면, 우선 제 2장에서는 셋업 작업에 대한 기본적인 개념에 대한 이해를 위해 실제 작업현장에서 나타날 수 있는 셋업 작업의 세부적인 내용들과 셋업 작업을 줄일 수 있는 방법들에 대하여 살펴볼 것이다.

제 3장에서는 대상문제가 갖는 가장 중요한 특성으로 제시한 1)이종병렬기계, 2)순서에 따라 변하는 셋업시간, 3)기계에 따라 변하는 셋업시간, 4)기계 가용시간, 5)주문 도착시간 등 5가지 특성을 고려한 일반적인 휴리스틱 일정계획에 대해 우선 살펴볼 예정이다.

제 4장에서는 이종병렬기계 상황에 셋업담당자 제약을 추가한 문제를 다루고자 한다. 이 문제를 해결하기 위해 우선 수리적 모형을 개발하여 규모가 작은 문제에 대한 해결방안으로 활용한다. 그러나 문제의 규모가 커지면 수리적 모형에 의한 해는 기대할 수 없으므로, 효과적인 메타휴리스틱으로 잘 알려진 유전알고리즘을 활용한 휴리스틱 해법들을 개발하고 이 해법들에 대한 비교를 통해 문제에 적합한 휴리스틱을 규명할 예정이다.

제 5장에서는 각 주문의 셋업 작업시간이 내부활동과 외부활동으로 분리되어 있는 경우의 이종병렬기계 시스템에서 한 사람(혹은 한 팀)의 셋업 작업자가 셋업을 전담할 때 이 셋업 담당자가 각 작업들의 내부 및 외부 셋업활동을 가장 효율적으로 수행하는 일정계획을 제시 하고자 한다.

마지막 6장에서는 본 연구를 간략히 정리하고 그 의의와 한계점 및 향후 연구과제에 대하여 논할 예정이다.

Ⅱ. 셋업 작업에 관한 고찰

1. 개요

개념적으로 셋업(set-up)이란, 기계작업이 이루어지기 전에 선행하여 준비해야 하는 일련의 행동이나 조치 등을 의미한다. 협의의 개념으로 기계를 작동하기 위해 행해지는 공구, 치공구류의 부착이나 교체, 좌표계의 설정, 시운전 등의 활동에서부터 공장 안에서 이루어지는 원자재의 이동이나 운반 또는 원자재를 작업에 적절한 크기로 미리 자르는 준비 작업, CNC 공작기계에 작업을 지시하는 프로그램을 작성하는 CAM (Computer Aided Manufacturing) 작업 등을 의미한다. 광의의 개념으로는 기계 가공이 가능하게 설계나 모델링을 한다거나, 필요한 자재의 구매, 치공구류의 제작, 원자재나 시제품의 품질검사 등의 공정도 셋업의 개념에 포함될 수 있을 것이다. 이러한 셋업 작업은 또한, 해당 기계를 정지해야만 수행 되는 내부 셋업 작업과 가동 기계의 정지 없이 행할 수 있는 외부 셋업 작업으로 구분되기도 한다. 셋업 작업은 기계 작업을 위해 불가피하게 수행해야 할 작업 이지만, 그 자체로 부가가치를 높이는 작업은 아니므로 가능한 줄이기 위한 노력이 필요하다. 셋업 작업을 줄이기 위해서는 우선 셋업 작업을 좀 더 깊이 있게 이해하기 위한 이론적인 고찰이 있어야 한다.

이하 제 2절 에서는 우선 셋업의 구체적인 개념과 셋업 비용에 대하여, 마지막 제 3절 에서는 셋업 작업 절약의 중요성과 셋업 작업을 줄일 수 있는 실제 개선활동에 대하여 살펴 볼 예정이다.

2. 셋업 작업의 개념과 셋업 비용

시장 경쟁이 심화되고, 제품의 수명 주기가 짧아질수록 전용 생산 라인을 구축하는 것은 기업에게 위험 부담이 큰 결정이 된다. 이러한 리스크를 줄이고 투자비용을 아끼기 위해 기업은 도입 기계를 될 수 있으면 여러 용도로 사용할 수 있기를 희망한다. 그러기 위해서는 필연적으로 작업이 변경될 때 마다 기계에 부착되어 있는 공구나 치공구류 등의 세팅을 교체하고 새로운 CAM 작업을 실시하며 시제품 등을 새로 만들게 된다. 이렇게 다른 작업을 위해 기계의 세팅을 바꾸는 작업을 셋업(set-up) 작업이라 하는데 이 절에서는 셋업 작업의 개념, 셋업 공정들의 다양한 내용과 셋업에 소요되는 비용에 대해 살펴볼 예정이다.

우선 공장 내에서 이루어지는 원자재를 이동, 정리, 운반하거나, 기계의 시운전, 절삭유 등의 주입, 치공구류 설치, 좌표 맞추기 등의 행위가 대표적인 셋업 작업에 속한다고 할 수 있다. 특히, 이런 일반적인 셋업 공정들은 작업자의 숙련도에 따라 크게 차이가 나는데 이러한 숙련도의 차이는 전체 기계 가동률과 생산성의 차이로 결과가 나타난다. 최근에는 컴퓨터의 발달로 NC공작기계에 computer를 내장한 CNC 공작기계가 널리 이용되고 있으며 통상 NC라고 하면 CNC(computer numerical control)를 지칭한다. CNC(computer numerical control)는 이미 여러 분야의 기계에 적용되어 활용되고 있으며, 여러 대의 CNC공작기계에 공작물이나 공구 등을 운반하는 자동 반송 장치와 자동화 창고, 로봇 등과 연결해서 이들을 computer로 관리하는 공장 자동화도 급속도로 보급되고 있다. 이러한 CNC 공작기계의 특징은 공작기계가 공작물을 가공하는 중에도 part program 수정이 가능하며, 단위를 쉽게 자동 변

환할 수 있고, part program을 macro 형태로 저장시켜 필요시 다시 불러 사용할 수 있으며, 비교적 품질이 균일한 생산품을 얻을 수 있어 제조원가 및 인건비를 절감할 수 있다는 것이다. 종래의 셋업 작업이 단순한 공구나 치공구류를 부착하거나 원자재의 운반 및 쉽게 가공하기 위해 예비 가공을 실시하는 등 주로 특별한 기술 보다는 단순한 반복 작업의 숙달로 실행이 가능했었던 것에 비해 CNC 공작기계와 자동화 기술의 발달로 인해 점점 더 셋업 작업이 단순 반복 작업이 아닌 기술력을 필요로 하는 과정으로 바뀌고 있다. 대표적인 예로 모델링한 3차원 설계 자료를 공작기계에 적용하기 위한 CAM(Computer Aided Manufacturing)작업의 경우 작업자의 숙련도에 따라 같은 제품을 생산하는데도 작업자의 기술 수준에 따라 다르게 프로그래밍을 한다. 그렇게 되는 이유는 설계 프로그램에 대한 이해수준과 숙련도 외에도 공작기계나 치공구류 등의 특성을 잘 아는 숙련기술자의 경우 훨씬 더 쉽고 능률적인 프로그래밍 작업이 가능하기 때문이다. 이러한 숙련 기술자의 경우 당연히 그 노임단가가 높으며, 이러한 고급인력과 이종병렬기계 간의 적절한 Machine Grouping이 그 공장의 생산성을 결정적으로 좌우할 수 있는 중요한 요인이 된다.

일반적으로 기계부품을 생산하는 개략적인 과정은 설계 모델링 작업을 시작으로 기계 가공, 시제품의 검사, 표면처리 가공, 도장, 출고검사의 순으로 이루어진다. 이 중 주요 작업 이라 할 수 있는 기계 가공을 더 자세히 살펴보면, 우선 모델링 자료를 CNC 공작기계 생산에 적용하기 위한 CAM 작업을 시작으로 자재를 적당한 크기로 절단이나 황삭 작업이 이루어지고, 메인 공작기계에 공구와 치공구류 등을 물려서 가공을 하고, 품질 검사를 실시한 후 마무리 다듬질, 최종 검사의 순서로 이루어진다. 이들 공정 중 CAM 작업과 치공구류의 설치 작업을 비교해 보면, 두 공정 모두 기계 작

업을 위한 셋업 작업이지만 이 2가지 셋업 작업은 그 성격에 있어서 차이가 있다. CAM 작업의 경우 기계가 작동하는 중에도 수행할 수 있는 외부 셋업 작업인 반면, 치공구류 설치 작업은 기계의 정지를 전제로 하는 내부 셋업 작업이다. 다른 측면에서 살펴보면, 외부 셋업 작업 중 CAM 작업의 경우 프로그래밍에 대한 전문적인 지식이 있는 엔지니어가 다루어야 할 셋업 작업인데 비하여 원자재 절단, 황삭의 경우 기능공이 다루어야 할 셋업 작업으로 작업 성격상 차이가 날 수 있다. 이러한 성격상 차이에 따라 다른 셋업 작업자를 계획하는 것도 공정계획의 중요한 고려 사항이 될 수 있다.

셋업 작업과 관련된 일련의 비용들은 제품에 부가가치가 생기지 않는 순수한 비용으로 이 셋업 비용의 절감 및 셋업 작업 시간의 감소는 생산성을 높이는 중요한 수단이다. 이하 셋업 작업과 관련된 비용에 대하여 살펴보고자 한다. 셋업 작업의 비용에는 여러 종류로 나뉠 수 있는데, 우선 가장 큰 요소는 직접 노무비 이다. 대체로 셋업 작업은 기계에 대한 기본적인 지식과 경험을 필요로 하므로 아무나 할 수 있는 작업은 아니다. 하지만, 그 종류에도 단순한 절단이나 황삭 등 비교적 기술적 난이도가 낮아 인건비가 높지 않은 셋업 작업에서부터 CAM 작업이나 정밀 품질 검사 등 기술적 난이도가 높은 작업들도 나뉜다. 이렇게 기술적 난이도가 높을수록 직접 노무비는 높아지며, 대형의 기계나 복잡한 첨단 기계일수록 현저히 높아지는 특성이 있다. 또한 새로운 셋업 작업의 경우 몇 개의 시제품을 생산하여 품질 검사를 거쳐 양산에 착수하게 된다. 이러한 시제품의 제작에 소요되는 재료비, 검사 및 인증 비용도 셋업 작업 비용에 포함 된다. 또한, 대개의 새로운 셋업 작업은 하나의 기계에만 해당하는 것이 아니라 일련의 생산 공정에 위치하는 기계군 들에 셋업 작업을 모두 실행해야 하므로 그 비용은 더욱 커지게 된다. 이렇게 부가가치를 발생하지 않는 셋업

작업의 비용이 커지는 것은 생산 현장의 생산성을 떨어뜨리는 것으로 생산 현장의 목표는 이러한 셋업 작업 시간과 비용의 감소로 집중하게 된다. 셋업 작업 비용을 줄일 수 있는 가장 좋은 방법은 가능한 셋업 작업의 횟수를 줄이는 것으로 셋업 작업 후 단일 제품을 최대한 많이 생산하는 것이 좋으나 이렇게 되면, 셋업 비용은 줄어들지만 한꺼번에 많은 양의 재고를 보유하는 문제가 생기게 된다. 생산 현장에 적정량의 재고는 필요하지만, 과다 재고의 보유는 재고 보유비용이라는 또 다른 생산성 저해 요인을 갖게 되는 것으로 바람직하지 않다. 이하 다음 마지막 절에서는 재고를 늘리지 않고 셋업 작업의 비용을 최대한 줄일 수 있게 셋업 작업을 효율적으로 줄일 수 있는 실제 개선 활동의 내용에 대하여 살펴볼 예정이다.

3. 셋업 작업의 개선

셋업 작업은 크게 4가지 범주로 분류할 수 있다. 첫 번째는 셋업의 사전 활동이다. 셋업에 필요한 공구나 기계 장비, 도구 등을 미리 준비하고 확인하는 것이다. 이러한 도구들이 제자리에 없는 것은 대단히 비생산적인 낭비요소이며 기본적인 작업장의 정리 정돈과도 연관이 되는 사항이다. 두 번째는 현재 기계에 부착된 공구, 치공구류 등을 분리하고 새로운 공구, 치공구류 등을 부착하는 일이다. 세 번째는 새로 장착된 공구, 치공구류 등을 시방서나 규격에 맞게 정확하게 조정하는 일이다. 마지막으로 시제품을 만들어 품질검사와 확인을 하고 다시 치수를 재조정하는 과정이다. 이 과정은 공구나 치공구류 등을 교체하지 않더라도 공구의 마모 등의 사유로 품질이 일정하지 않은 경우는 부정기적으로 행해질 수 있는 셋업 작업이다.

셋업 작업 개선의 중요성을 알기 쉽게 이해하기 위해, 손쉬운 예를 들

어본다면, 가령 셋업 시간이 60분이고, 단위 가공시간이 1분인 경우, 생산 로트가 600개라면 총 가공시간은 11시간이 된다. 이 경우 셋업 시간이 1/10로 줄어들면, 60개씩 10번 생산하여 같은 생산 결과를 얻을 수 있다. 이 상황을 다르게 표현하자면 셋업 시간이 줄어들수록 생산 로트가 작아질 수 있고, 다품종 소량 생산이 가능하다는 것이다. 즉, 다품종 소량생산을 하기 위해서는 필연적으로 셋업 시간을 줄일 수밖에 없고, 셋업 시간을 줄이는 것이 생산성을 향상시키는 중요한 요소가 된다는 것이다.

셋업 작업의 개선을 위해서는 우선 셋업 작업의 상세한 분석이 선행되어져야 한다. 일견 간단해 보이는 셋업 작업도 작업공정도 등의 작업분석 도구로 분석해 보면 상당히 복잡한 과정이며 그 과정을 단계마다 분리하여 낭비 제거 요소를 파악해야 한다. 전자기기의 발달로 요즘은 스마트폰을 이용한 간단한 녹화로도 셋업 작업을 손쉽게 분석할 수 있게 되었다. 일단 모든 셋업 작업의 활동이 파악되면 각 항목을 내부 셋업 작업과 외부 셋업 작업으로 분리하고 가능한 모든 외부 셋업 작업은 기계의 가동 중에 끝내야 한다. 미국의 보통 공정들의 경험에 의하면, 이러한 셋업 작업의 철저한 사전 준비만으로 50%의 셋업 시간을 줄일 수 있다고 발표된 자료도 있다.

셋업 작업의 개선을 위한 또 다른 방법으로 많은 종류의 내부 셋업 작업을 외부 셋업 작업으로 바꾸어 나가는 것이 중요하다. 이러한 경우 기계의 모듈화에 노력하여야 한다. 기계를 정지시키고 난 후 여러 장치나 공구를 해체하고 다시 조립하기보다 미리 새로운 장치나 공구 등을 몇 개의 모듈로 준비하여 대기 하고 있다가 기계가 정지되면 즉시 몇 개의 모듈만 갈아 끼우는 것이다. 이러한 모듈이 개발되어 있지 않다면, 기계의 정지 후 설치된 치공구류나 공구를 수공구 등을 사용하여 인력으로 해체하고, 다음 작업에 해당되는 공구, 치공구류 등을 다시 수공구를 활용하여 부착하는 수작업을 한다면, 기계의 내부 셋업 작업 시간이 엄청나게 길어질 것이다.

공구나 치공구류 등이 모듈화 되어 간단한 장탈착 기구로 바로 교체된다면 기계의 내부 셋업 시간이 획기적으로 줄어들 것이다. 요즘 나오는 공작기계의 경우 이미 이런 식으로 제작되고 있으며 공작기계의 구매 시 다양한 악세서리 장치 등을 함께 제공하고 있다.

셋업 작업을 줄이는 방법으로 중요한 요소 중 셋업 작업자의 반복에 의한 숙련도의 향상이다. 정확한 작업분석을 통한 개선방안의 연구와 그 방법을 실행할 셋업 작업자의 숙련도 향상에 따라 셋업 작업 시간은 비약적으로 발전할 수 있다.

제품이 다양화 될수록 그 제품을 생산하기 위한 셋업 시간은 필연적으로 생겨나지만, 이러한 셋업 시간은 대개 부가가치의 창출과 무관하고 고급인력을 장시간 활용하는 결과를 초래하여 결과적으로 생산성을 떨어뜨리게 하는 요인으로 이에 대한 단축 노력은 꾸준히 연구되고 시도 되어져 왔다. 대표적인 사례로 도요타의 SMST(single minute setup time)운동으로 이 개념은 모든 셋업 시간을 10분 이내로 낮추는 것으로 실제로 이 운동의 결과는 재고량과 비용의 7배 감소라는 엄청난 결과로 나타났다. 셋업 시간을 줄이기 위해서는 우선 셋업 작업을 내부 셋업과 외부 셋업으로 구분해야 하며, 가능한 한 셋업 작업을 외부화 하는 것이 바람직하다. 아울러 작업대나 스위치 등의 표준화를 통해서도 내부 셋업시간을 상당량 줄일 수 있다. 본 논문의 주제가 이종병렬기계를 운영하는 소규모 공장에서 셋업 시간을 고려한 일정계획을 단시간 내에 세우는 휴리스틱 방법에 관한 것인데 이러한 방법과 더불어 근본적으로 부품설계나 생산설비 배치의 단계에서부터 셋업 시간을 최소화 할 수 있는 고려를 하는 것이 중요할 것이다.

Ⅲ. 작업순서 의존형 셋업시간을 갖는 이종병렬기계의 휴리스틱 일정계획

1. 개요

본 장에서는 전 장에서 살펴본 셋업의 이론적인 개념과 작업공정 분석도, 다중활동분석표 등에 대한 이론적인 고찰을 바탕으로 작업순서 의존형 셋업시간을 가지는 이종병렬기계의 휴리스틱 일정계획에 관하여 살펴볼 예정이다. 본 연구가 대상문제가 갖는 가장 중요한 특성으로 제시한 1)이종병렬기계, 2)순서에 따라 변하는 셋업시간, 3)기계에 따라 변하는 셋업시간, 4)기계 가용시간, 5)주문 도착시간, 6)셋업작업자 제약 등 6가지 중 6)셋업작업자 제약을 제외한 5가지 특성을 고려한 일반적인 휴리스틱 일정계획에 대해 우선 알아보고자 한다. 실제로 본 장에서 다루는 일정계획 문제가 발생하는 시점은 하나의 기계에서 생산이 종료되었을 때이며, 계획대상 주문들은 그 시점 이전에 접수되어 있거나 미래의 어떤 시점에 접수되기로 예약되어 있는 주문들이다. 따라서 4번째 항목과 같이 계획시점에 각 기계들은 처리중인 주문을 가질 수 있으며 그 완료시간(즉, 각 기계의 가용시간)은 모두 다른 값을 가지게 되는 것이다. 본 장에서는 이러한 상황에서 총 완료시간의 최소화를 목적함수로 하는 최적화 문제를 해결하고자 하며, 작업자 인원에 대한 고려는 별도로 하지 않는다. 이러한 문제를 해결하기 위해서 우선 다음 절에서 최적화 모형을 세울 것이다.

2. 최적화 모형

본 절에서는 앞 절에서 소개한 문제의 수리적 모형을 개발한다. 본 장에서 다루는 문제는 Zhu & Heady('2000) 및 Joo & Kim('2012)의 문제와 매우 유사하므로 수리적 모형에 있어 큰 차이는 없다. 우선 모형에서 사용할 기호들을 먼저 설명한 다음 최적화 모형을 제시하고자 한다.

<파라메터>

i, j : 주문을 표시하는 인덱스 (i, j = 1, 2, ..., n)

k : 기계를 표시하는 인덱스 (k = 1, 2, ..., m)

a_i : 계획시점 이전에 주문 i가 도착하였으면 0, 그렇지 않으면 도착예정시간

r_k : 계획시점에 기계 k가 작업중이면 그 작업의 완료시간, 그렇지 않으면 0

s_{ik} : 기계 k에서 가장 먼저 처리되는 주문이 i일 경우의 셋업시간

s_{ijk} : 기계 k에서 주문 i가 완료되고 주문 j를 준비할 경우의 셋업시간

p_{ik} : 주문 i가 기계 k에 할당되었을 경우의 처리시간 (셋업시간 제외)

<결정변수>

x_i = 주문 i의 착수시간 (셋업시간이 0보다 큰 경우에는 셋업

착수시간)

f_i = 주문 i의 완료시간

$f_{\max}$ = 모든 주문의 완료시간 (makespan)

y_{ik} = 주문 i가 기계 k에 할당되면 1, 그렇지 않으면 0

z_{ik} = 기계 k에서 가장 먼저 처리되는 주문이 i이면 1,
그렇지 않으면 0

z_{ijk} = 기계 k에서 주문 i에 이어 주문 j가 처리되면 1,
그렇지 않으면 0

<모형>

Minimize $f_{\max}$ (3-1)

subject to

$$x_i \geq a_i \text{ for all } i \tag{3-2}$$

$$x_i + M(1 - z_{ik}) \geq r_k \text{ for all } i,\ k \tag{3-3}$$

$$x_i + \sum_{k=1}^{m}(s_{ik}z_{ik} + \sum_{\substack{j=1 \\ j \neq i}}^{n} s_{jik}z_{jik} + p_{ik}y_{ik}) = f_i \text{ for all } i \tag{3-4}$$

$$x_j + M(1 - \sum_{k=1}^{m} z_{ijk}) \geq f_i \text{ for all } i,\ j \tag{3-5}$$

$$f_i \leq f_{\max} \text{ for all } i \tag{3-6}$$

$$\sum_{k=1}^{m} y_{ik} = 1 \text{ for all } i \tag{3-7}$$

$$\sum_{i=1}^{n} z_{ik} = 1 \text{ for all } k \tag{3-8}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} z_{ijk} \leq y_{ik} \text{ for all } i,\ k \tag{3-9}$$

$$z_{ik} + \sum_{\substack{j=1 \\ j \neq i}}^{n} z_{jik} = y_{ik} \text{ for all } i,\ k \tag{3-10}$$

y_{ik}, z_{ik} : 0/1 integer for all $i,\ k$

z_{ijk} : 0/1 integer for all $i,\ j,\ k$

식 (3-1)은 본 문제가 총 완료시간을 최소화하는 문제라는 것을 보여준다. 식 (3-2)는 각 주문이 도착한(확정된) 이후에 작업이 시작될 수 있음을 나타낸다. 식 (3-3)은 각 기계가 현재 작업 중인 작업을 종료한 이후에 새로운 작업을 시작할 수 있다는 조건을 표현하고 있다. 식 (3-4)는 각 주문의 착수시간과 완료시간 사이의 관계를 나타낸다. 식 (3-5)에서는 각 기계에서 선행 주문의 작업이 완료되어야만 후속 주문의 작업이 시작될 수 있음을 표현하고 있다. 식 (3-6)은 총 완료시간이 모든 주문들의 완료시간 중 가장 큰 값이라는 것을 보여준다. 식 (3-7)은 각 주문이 한 대의 기계에만 할당되도록 해준다. 식 (3-8)은 각 기계에 최초로 배정되는 작업은 한 개라는 것을 보여준다. 식 (3-9)에 의하면 어떤 주문이 어떤 기계에 배정되지 않았다면($y_{ik} = 0$) 그 주문은 그 기계에서 다른 어떤 주문의 선행주문도 될 수 없다 ($z_{ijk} = 0$)는 것을 보여준다. 반대로 해석하면 어떤 주문이 하나의 기계에 배정되었다면($y_{ik} = 1$) 그 주문은 그 기계에서 다른 어떤 주문의

선행작업이라는 것($z_{ijk}=1$)을 보여준다. 그런데 이 식이 부등식인 것은 그 주문이 그 기계에서 최종으로 처리되는 주문일 경우 때문이다. 비슷하게 식 (3-10)에서는 어떤 주문이 하나의 기계에 배정되었다면 ($y_{ik}=1$) 그 주문은 그 기계에서 처리되는 최초의 주문($z_{ik}=1$)이거나 다른 어떤 주문의 후속주문이라는 것($z_{jik}=1$)을 보여준다.

이 모형은 문제의 크기(주문수 및 기계수)가 작은 경우 CPLEX나 LINGO 같은 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 이를 확인하기 위해 주문의 수(n)가 8이고 기계수(m)가 3인 상황의 문제를 LINGO를 사용해 해결해 보았다. 데이터는 아래와 같다.

$$a_i = (0,\ 0,\ 0,\ 2,\ 0,\ 3,\ 0,\ 4),\ r_k = (0,\ 3,\ 5),\ p_{ik} = \begin{bmatrix} 5 & 3 & 4 \\ 3 & 6 & 2 \\ 2 & 3 & 7 \\ 4 & 3 & 5 \\ 3 & 5 & 4 \\ 3 & 6 & 2 \\ 2 & 3 & 7 \\ 3 & 5 & 4 \end{bmatrix},\ s_{ik} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix},$$

$$s_{1jk} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \end{bmatrix},\ s_{2jk} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix},\ s_{3jk} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix},\ s_{4jk} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix},$$

$$s_{5jk} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix},\ s_{6jk} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix},\ s_{7jk} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \end{bmatrix},\ s_{8jk} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

속도 2.60GHz의 CPU가 장착된 PC에서 [그림 3-1]과 같은 LINGO 프로그램을 이용해 해를 구한 결과 1분 15초의 계산시간에 이 예제의 최적해를 구할 수 있었고 그 결과는 다음과 같다. 괄호 안의 숫자는 (주문번호, 착수시간, 종료시간)이다.

기계 1에 할당된 주문 및 그 순서 : (5, 1, 5) - (7, 5, 8) - (8, 8, 12)
기계 2에 할당된 주문 및 그 순서 : (3, 3, 6) - (1, 6, 9) - (4, 9, 12)
기계 3에 할당된 주문 및 그 순서 : (6, 6, 9) - (2, 9, 12)

아래 <표 3-1>은 기계 별 최적 할당주문을 나타낸 것이다.

<표 3-1> 기계 별 최적 할당주문

	1	2	3	4	5	6	7	8	9	10	11	12
기계 1		주문 5				주문 7			주문 8			
기계 2				주문 3			주문 1			주문 4		
기계 3							주문 6			주문 2		

아래 [그림 3-1]은 예제의 LINGO 프로그램의 실제 화면을 캡쳐한 그림이다.

Lingo Model - makespan1

```
MODEL:
 SETS:
      Job: a, x, f;
      Mch: r;
      JM(Job, Mch): p, s0, z0, y;
      JJ(Job, Job);
      JJM(Job, Job, Mch): s, z;
 ENDSETS

 DATA:
      n = 8;
      m = 3;

      BM = 9999;
      Job = 1..n;
      Mch = 1..m;

      a = 0 0 0 2 0 3 0 4;
      r = 0 3 5;

      p =   5 3 4  3 6 2  2 3 7  4 3 5  3 5 4  3 6 2  2 3 7  3 5 4;
      s0 =  1 1 0  1 2 1  1 0 1  2 1 0  1 0 2  1 2 1  2 1 0  1 0 2;

      s =   0 0 0  1 0 1  1 2 1  1 0 1  0 1 0  1 0 1  1 2 1  1 0 1
            0 1 1  0 0 0  1 0 2  1 1 0  1 0 2  1 1 0  1 1 0  1 0 2
            0 0 1  1 0 2  0 0 0  1 1 0  1 0 2  1 1 0  1 0 2  1 0 1
            1 1 0  1 0 2  1 0 0  0 0 0  0 1 1  1 1 0  1 1 0  1 0 2
            0 2 1  1 0 1  1 0 0  0 2 1  0 0 0  1 2 0  1 1 0  1 0 2
            0 2 1  1 0 1  1 0 0  0 2 1  1 0 1  0 0 0  1 1 0  1 0 2
            0 2 1  1 0 1  1 0 0  0 2 1  1 0 1  1 1 0  0 0 0  1 0 2
            0 2 1  1 0 1  1 0 0  0 2 1  1 0 1  1 1 0  1 0 2  0 0 0;
 ENDDATA

 MIN = fmax;

 @FOR(Job(i): x(i) > a(i));
 @FOR(JM(j, k): x(j) + BM * (1 - z0(j, k)) > r(k));

 @FOR(Job(i): x(i) + @SUM(Mch(k): s0(i, k) * z0(i, k)
      + @SUM(Job(j) | j #ne# i: s(j, i, k) * z(j, i, k)) + p(i, k) * y(i, k)) = f(i));
 @FOR(JJ(i, j) | i #ne#j: x(j) + BM * (1 - @SUM(Mch(k): z(i, j, k))) > f(i));

 @FOR(Job(i): f(i) < fmax);

 @FOR(Job(i): @SUM(Mch(k): y(i, k)) = 1);
 @FOR(Mch(k): @SUM(Job(j): z0(j, k)) = 1);
 @FOR(JM(i, k): @SUM(Job(j) | j #ne# i: z(i, j, k)) < y(i, k));
 @FOR(JM(i, k): z0(i, k) + @SUM(Job(j) | j #ne# i: z(j, i, k)) = y(i, k));

 @FOR(JM(i, k): @BIN(y(i, k)));
 @FOR(JM(i, k): @BIN(z0(i, k)));
 @FOR(JJM(i, j, k): @BIN(z(i, j, k)));

END
```

[그림 3-1] 예제의 LINGO 프로그램

그러나 잘 알려진 바처럼 이 문제는 NP-hard 에 속하므로 문제의 크기가 커질 경우 정상적인 시간 안에 최적해를 구하는 것은 불가능

한 일이다. 실제로 위 예제보다 큰 문제(주문수가 9 이상 또는 기계 수가 4 이상)의 경우에는 같은 사양의 컴퓨터에서 1시간 안에 해를 구할 수 없었다. 따라서 본 연구에서는 최적은 아니지만 정상적인 시간 안에 최적에 가까운 좋은 해를 찾을 수 있는 휴리스틱 알고리즘을 개발하는데 초점을 맞추도록 한다.

3. 단순 휴리스틱 방법론

앞에서 제시한 모형을 해결하기 위해 본 장에서는 두 개의 단순 휴리스틱과 두 개의 유전 알고리즘 기반 알고리즘을 개발하였다. 첫 번째 휴리스틱은 BH (Base Heuristic) 라고 명명하였으며 그 수행할 절차는 다음과 같다.

BH (Base Heuristic) Algorithm

Step 0. N = 계획대상 주문의 집합, M = 전체 기계 집합

Step 1. 다음과 같은 방법으로 배치대상 기계를 선택한다.

1.1) M에 포함된 모든 기계에 대해 F_k (= 기계 k 에 배치된 작업들이 모두 완료되는 시점)를 계산한다. 배치된 작업이 없다면 $F_k = r_k$ 로 놓는다.

1.2) F_k가 최소인 기계 번호를 k^* 로 놓는다.

Step 2. 다음과 같은 방법으로 선택된 기계에 배정할 주문을 선택한다.

2.1) 기계 k^*에 배정된 주문이 없다면 N에 속한 모든 주문 j 에 대해

$s_{jk^*} + p_{jk^*}$ 를 계산하고 그 값을 최소화하는 주문 j^* 를 찾는다.

2.2) 그렇지 않으면, 기계 k^*에 배정된 마지막 주문을 i 로 놓고, N에 속한 모든 주문 j 에 대해 $s_{ijk^*} + p_{jk^*}$ 를 계산한 다음, 그 값을 최소화하는 주문 j^* 를 찾는다.

2.3) 주문 j^*를 기계 k^*에 배정하고 그 주문을 집합 N에서 제거한다.

2.4) M = ∅ 이면 종료하고 그렇지 않으면 Step 1 로 간다.

이 알고리즘은 두 부분으로 구성되어 있다. 앞 부분은 기계그룹으로부터 하나의 기계를 선택하는 과정이다. 이 과정에서는 '가장 먼저 가용해지는 기계를 먼저' 선택하는 방식을 따른다. 알고리즘의 후반부에서는 선정된 기계에서 처리할 주문을 선택한다. 여기서는 '준비시간과 처리시간의 합이 최소인 주문을 먼저' 배정하는 방식을 적용하였다. 이 알고리즘은 매우 간단하여 쉽게 해를 구할 수 있다. 또한 본 장에서는 이 알고리즘의 결과를 뒤에서 제시하는 다른 알고리즘과 비교하는 기준으로 사용할 것이다.

그런데 BH 알고리즘은 이해하기 쉽고 해를 구하기도 쉽지만 한번 해가 구해지면 개선이 없는 일방향 알고리즘이므로 그로부터 구해지는 해의 품질을 보증하는 것은 쉽지 않을 것이다. 이러한 인식으로부터 다음에 제시되는 새로운 휴리스틱이 제안되었다. 이 알고리즘의 이름은 IBH (Imoroved BH) 로 명명하였으며 그 이름에서 알 수 있듯이 이 알고리즘은 BH를 개선한 것이다. 개선방법은 BH에 의해 각 기계에 배정된 주문들을 작업순서가 인접한 주문들끼리 맞바꾸는 (pairwise interchange) 것이다. 자세한 절차는 다음과 같다.

IBH (Improved Base Heuristic) Algorithm

Step 0. 초기해는 BH 알고리즘의 결과이다. M = 전체 기계 집합

Step 1. 집합 M에서 기계 하나를 임의로 선택하여 k 라고 부른다.

Step 2. 기계 k 에 연속으로 배정된 주문 i와 j에 대해 δ_{ij} (makespan 감소량)를 다음과 같이 계산한다.

Case 1. 주문 i 가 기계 k의 첫 주문이라면,

$$\delta_{ij} = s_{ik} + s_{ijk} - s_{jk} - s_{jik}$$

Case 2. 주문 j 가 기계 k의 마지막 주문이라면,

$$\delta_{ij} = s_{ijk} - s_{jik}$$

Case 3. 둘 다 아니면 주문 u를 i의 선행, v를 j의 후속주문이라 할 때

$$\delta_{ij} = s_{uik} + s_{ijk} + s_{jvk} - s_{ujk} - s_{jik} - s_{ivk}$$

Step 3. 한 개 이상의 δ_{ij}가 양수라면,

3.1) 최대의 δ_{ij}를 갖는 두 주문의 처리순서를 교환한다.

3.2) Step 2로 간다.

Step 4. 모든 δ_{ij} 값들이 0 이하라면,

4.1) 집합 M에서 기계 k를 제거한다.

4.2) M = ∅ 이면 종료하고 그렇지 않으면 Step 1 로 간다.

4. 유전 알고리즘 방법론

이제 유전 알고리즘 기반의 메타휴리스틱 알고리즘을 개발한다. 전체적인 절차는 일반적인 유전 알고리즘과 대동소이한데 염색체 표현 및 염색체-해 전환 (Decoding) 절차에 있어서 앞 절에서 제안한 휴리스틱의 개념을 도입하였다. 염색체 표현은 주문 수만큼의 (0, 1) 구간 실수를 사용하고, 이 실수 값을 사용해 각각의 주문을 기계에 할당한다. 예를 들어 계획대상 주문이 5개라면 5개의 (0, 1) 구간 실수로 염색체가 구성되고, 첫 번째 실수를 이용해 첫 번째 주문의 기계 할당이, 두 번째 실수를 이용해 두 번째 주문의 기계할당이, 순차적으로 이루어진다. 할당 방법은 각 주문이 모든 기계에 할당되는 비율을 동일하게 유지하기 위하여 구간 $(\frac{k-1}{m}, \frac{k}{m})$에 유전자 값이 포함되면 기계 k 에 배정하도록 한다. 예를 들어 기계가 3대인 경우 유전자 값이 (0, 1/3)에 속하면 해당 주문을 기계 1에 배정하고, (1/3, 2/3)에 속하면 기계 2에, (2/3, 1)에 속하면 기계 3에 배정하는 것이다. 배정 이후의 일정 생성은 앞선 BH 알고리즘과 같다. n 개의 (0, 1) 구간 실수로부터 하나의 일정이 생성되는 전체 과정을 요약하면 아래와 같다.

DECODE_1 Algorithm

Step 1. 염색체 내부의 유전자 값이 구간 $(\frac{k-1}{m}, \frac{k}{m})$에 속하면 관련되는 주문은 기계 k 에 배정한다.

Step 2. M = 모든 기계들의 집합

Step 3. M으로부터 기계 하나를 임의로 선택하여 k 라고 부른다.

Step 4. N = 기계 k 에 배정된 주문들의 집합

Step 5. N에 속한 모든 주문 i 에 대해 $s_{ik}+p_{ik}$ 를 계산하고 그 값을 최소화하는 주문을 기계 k 의 첫 주문으로 배정한다. 그 주문을 N에서 제거한다.

Step 6. 기계 k 에 배정된 마지막 주문을 i 로 놓고, N에 속한 모든 주문 j 에 대해 $s_{ijk}+p_{jk}$ 를 계산한 다음 그 값을 최소화하는 주문을 찾아 기계 k 의 다음 주문으로 배정하고 그 주문을 N에서 제거한다.

Step 7. N = ∅ 이면 Step 8 로 가고 그렇지 않으면 Step 6 으로 간다.

Step 8. 기계 k 를 M에서 제거한다.

Step 9. M=∅ 이면 종료하고 그렇지 않으면 Step 3 으로 간다.

이 알고리즘은 앞서 제안한 BH 알고리즘을 염색체-해 전환 과정에 적용한 것이다. 유전자 값에 의해 주문-기계 할당이 이루어진 다음 BH 알고리즘과 같이 '준비시간과 처리시간의 합이 최소인 주문을 먼저' 처리하는 방식을 적용하였다. 이 방식은 염색체에 의해서는 주문-기계 할당만 이루어지고 한 기계 안에서의 주문처리 순서를 결정하는 것은 휴리스틱에 의해 이루어지므로 일종의 Hybrid 형 유전 알고리즘이라고 할 수 있다.

앞에서 BH를 개선한 IBH 알고리즘을 제안하였듯이 유전 알고리즘 기반의 해법에서도 위의 DECODE_1 알고리즘을 개선할 수 있다. 개선방법은 IBH 알고리즘과 마찬가지로 각 기계 내에서 주문들의 처리순서를 교환하는 과정을 거치게 된다. 자세한 절차는 아래와 같다.

DECODE_2 Algorithm

Step 0. DECODE_1을 적용한 해를 갖고 있다. M = 모든 기계들의 집합

Step 1. M으로부터 기계 하나를 임의로 선택하여 k 라고 부른다.

Step 2. 기계 k 에 연속으로 배정된 주문 i 와 j 에 대해 δ_{ij} (makespan 감소량)를 다음과 같이 계산한다.

Case 1. 주문 i 가 기계 k 의 첫 주문이라면, $\delta_{ij} = s_{ik} + s_{ijk} - s_{jk} - s_{jik}$

Case 2. 주문 j 가 기계 k 의 마지막 주문이라면, $\delta_{ij} = s_{ijk} - s_{jik}$

Case 3. 둘 다 아니면 주문 u 를 i 의 선행, v 를 j 의 후속주문이라 할 때

$$\delta_{ij} = s_{uik} + s_{ijk} + s_{jvk} - s_{ujk} - s_{jik} - s_{ivk}$$

Step 3. 한 개 이상의 δ_{ij}가 양수라면,

3.1) 최대의 δ_{ij}를 갖는 두 주문의 처리순서를 교환한다.

3.2) Step 2로 간다.

Step 4. 모든 δ_{ij} 값들이 0 이하라면,

4.1) 집합 M에서 기계 k 를 제거한다.

4.2) M = ∅ 이면 종료하고 그렇지 않으면 Step 1 로 간다.

위에서 제시한 염색체-해 전환 절차 이외의 과정은 일반적인 유전 알고리즘의 전개와 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.

$$F_i = Z_{\max} - Z_i \tag{3-11}$$

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 연산(Genetic Operation)을 적용하여 다음 세대의 염색체를 만들어 낸다. 이 유전 연산에는 매우 다양한 방법들이 있으나 본 연구에서는 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 사용하였다. 또한 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(roulette wheel) 방법으로 두 개의 염색체를 선택한 다음 (0, 1) 구간의 난수 하나를 생성한다. 이 난수가 미리 정한 교배확률 P_c 보다 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 2에 의해 생성된 염색체에 속하는 모든 유전자에 대해 다음 과정을 수행한다. (0, 1) 구간의 난수를 하나 생성한다. 이 난수가 미리 정한 돌연변이 확률 P_m 보다 작으면 (0, 1) 구간에서 새로운 난수를 하나 생성하여 기존의 유전자를 대치한다. 여기서도 P_m의 값은 실험을 통해 선정한다.

5. 수치 실험

앞서 제시한 휴리스틱 알고리즘들의 성능을 평가하기 위해 본 절에서는 임의로 생성한 많은 예제들을 사용해 알고리즘의 결과를 비교한다. 문제의 복잡도가 주문의 수(n)와 기계 수(m)에 가장 크게 영향을 받으므로 이 두 모수들을 몇 개의 수준으로 나누어서 실험을 실시한다. 주문의 수는 20, 30, 40, 50 등의 4개 값 중에서 하나로 하고 기계의 수는 2, 3, 4, 5 등 4개 값 중에서 하나로 하여 총 16개 (n, m) 조합의 문제를 생성하여 해법을 적용하였다.

이러한 16개 각각의 (n, m) 조합에 대해 10개의 문제를 임의로 생

성한다. 문제를 생성하기 위해서는 a_i, r_k, p_{ik}, s_{ik}, s_{ijk} (i, j = 1, 2, ..., n, k = 1, 2, ..., m) 값들을 생성하여야 한다. 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. r_k 값도 $U[0, 10]$로 가정한다. 처리시간(p_{ik})과 준비시간(s_{ik} 및 s_{ijk})은 $U[5, 15]$로 가정한다.

생성된 문제들은 앞서 제시한 4개의 알고리즘(BH, IBH 알고리즘 및 DECODE_1과 DECODE_2를 적용한 유전 알고리즘)을 사용해 해결한다. 또한 본 연구에서 제시한 알고리즘의 성능을 객관적으로 평가하기 위해 Joo & Kim(2012b)이 제안하였던 알고리즘을 적용하기로 한다. 이 알고리즘도 유전 알고리즘인데 염색체로는 주문들의 처리순서를 표현하고 그 순서에 따라 각 주문을 가장 빠른 시간에 완료될 수 있는 기계를 선택하여 배정하게 된다. Joo & Kim(2012b)은 이 알고리즘을 GA_DR(Genetic Algorithm with Dispaching Rule)이라고 명명하였으며 매우 뛰어난 성능을 발휘하는 것을 보여주었다.

<표 3-2> Results of Numerical Experiments

		n=20			n=30			n=40			n=50		
		mean	min	max	mean	min	max	mean	min	max	mean	min	max
m=2	IBH	0.98	0.92	1.00	0.99	0.97	1.00	0.99	0.98	1.01	1.00	0.97	1.01
	GA1	0.93	0.87	0.98	0.95	0.93	1.00	0.95	0.93	0.97	0.98	0.96	1.00
	GA2	0.93	0.87	0.98	0.95	0.92	1.00	0.95	0.92	0.97	0.97	0.95	1.00
	J&K	0.94	0.85	1.01	1.01	0.96	1.05	1.04	1.00	1.08	1.09	1.06	1.12
m=3	IBH	0.99	0.95	1.03	0.99	0.94	1.01	1.00	0.98	0.01	1.00	0.97	1.01
	GA1	0.94	0.88	0.99	0.95	0.89	1.01	0.99	0.96	1.03	1.01	0.98	1.04
	GA2	0.92	0.87	0.96	0.96	0.89	0.99	0.98	0.95	1.02	1.00	0.97	1.02
	J&K	0.94	0.88	1.01	0.98	0.93	1.01	1.05	1.02	1.08	1.08	1.05	1.13
m=4	IBH	1.00	0.95	1.07	1.00	0.95	1.05	0.99	0.97	1.01	1.00	0.96	1.04
	GA1	0.94	0.88	0.99	0.99	0.94	1.05	1.03	0.96	1.08	1.05	1.02	1.11
	GA2	0.92	0.86	0.99	0.97	0.93	1.03	1.02	0.94	1.06	1.03	1.00	1.11
	J&K	0.90	0.86	0.97	0.98	0.93	1.05	1.03	0.95	1.09	1.08	1.05	1.14
m=5	IBH	0.98	0.93	1.00	0.99	0.92	1.03	0.98	0.94	1.00	1.00	0.97	1.02
	GA1	0.91	0.87	0.97	1.01	0.89	1.09	1.04	0.98	1.14	1.08	1.01	1.14
	GA2	0.90	0.86	0.96	0.99	0.92	1.07	1.03	0.97	1.09	1.06	1.01	1.12
	J&K	0.85	0.82	0.90	0.97	0.89	1.04	1.01	0.96	1.07	1.05	1.01	1.10

수치실험의 결과는 <표 3-2>에 정리되어 있다. 이 표의 각 값들은 분자를 해당 알고리즘으로 구한 총 완료시간(makespan)으로 하고 분모를 BH 알고리즘으로 구한 총 완료시간으로 하여 계산한 비율값이다. 따라서 값이 작을수록 BH 알고리즘에 비해 작업을 일찍 마쳤다는 의미이므로 알고리즘의 성능이 우수하다고 할 수 있다. 표에서 IBH로 표시된 행은 IBH의 결과와 BH를 비교한 것이고, GA1과 GA2는 각각 DECODE_1과 DECODE_2를 적용한 유전 알고리즘 결과를 BH와 비교한 것, 그리고 J&K는 Joo & Kim(2012b)의 GA_DR과 BH를 비교한 결과이다.

이 <표 3-2>는 16개 (n, m) 조합에 따라 16개의 영역으로 이루어져 있다. 각 영역에서는 해당 (n, m) 조합에 대해 무작위로 생성된 10개의 문제를 푼 결과를 4개 알고리즘에 대해 정리하였으며 각 알고리즘에 대해서는 3개 종류의 값을 제시하였다. 이 3개 값들은 각 알고리즘으로 10개 문제를 풀고 각 문제에 대해서 위의 비율값을 계산하여 그 평균값, 최소값, 최대값을 찾은 것이다. 예를 들어 표의 좌상귀 셀은 (n, m) = (20, 2)인 경우의 IBH 알고리즘에 대한 결과로서 그 값 0.98, 0.92, 1.00은 다음과 같은 절차에 의해 구해졌다.

1) 계획대상 작업수(n)는 20이고 기계의 수(m)가 2인 상황에서 생성된 10개의 문제에 대해 BH 알고리즘을 적용한 결과 그 최종완료시간은 158, 151, 167, 164, 156, 157, 157, 154, 159, 160 등이다.
2) 같은 문제들에 대해 IBH 알고리즘을 적용한 최종 완료시간은 146, 151, 160, 164, 153, 157, 157, 151, 159, 156 등이다.
3) IBH 결과를 BH 결과로 나누면 146/158(=0.92), 151/151(=1.00),

…, 156/160(=0.98) 등이 된다.

4) 이 10개의 값들에 대한 평균은 0.98, 최소는 0.92, 최대는 1.00이 된다.

알고리즘들의 성능을 평가하기 위해 표의 값들을 관찰해보면 다음과 같은 내용들을 발견할 수 있다.

1) 기계의 수(m)가 적은 경우 본 연구에서 제안한 유전 알고리즘이 Joo & Kim(2012b)의 알고리즘보다 우수하다. 또한 계획대상 작업수(n)가 많아질수록 본 연구에서 제안한 유전 알고리즘이 Joo & Kim(2012b)의 알고리즘보다 우수해진다. 따라서 기계당 작업수 비율(n/m)이 커질수록 본 연구의 유전 알고리즘이 상대적으로 좋은 성능을 발휘한다고 볼 수 있다.
2) 기계의 수(m) 혹은 계획대상 작업수(n)가 많아질수록 단순 휴리스틱인 IBH의 결과가 유전 알고리즘에 비해 우수해진다. 즉, 문제의 복잡도가 증가하면 유전 알고리즘의 성능이 떨어지게 된다. 물론 모든 경우에 해당되지는 않지만, 단순 휴리스틱이 유전 알고리즘보다 더 우수한 결과를 보여주는 것은 매우 이례적이라 할 수 있다.
3) 결론적으로, 기계수(m) 혹은 작업수(n)가 많은 경우(본 실험에서는 $m \times n$ 값이 대략 150 초과)에는 IBH 알고리즘을 사용하고, 그렇지 않은 경우에는 기계당 작업수 비율이 클 때(본 실험에서는 n/m 값이 대략 5 초과) DECODE_2를 적용한 유전 알고리즘을, 작을 때는 Joo & Kim(2012b)의 알고리즘을 사용하는 것이 좋다고 할 수 있다.

다음으로는 수행시간의 관점에서 알고리즘의 성능을 살펴본다. 결과는 <표 3-3>에 정리되어 있으며 4GB RAM과 2.60GHz G620 CPU가 장착된 PC에서 C++언어로 구현된 알고리즘을 가동하였을 때의 소요시간 평균값을 초단위로 표시한 것이다. 이 표도 데이터가 (n, m) 조합에 따라 정리되어 있는데, 주의한 점은 단순 휴리스틱(BH 및 IBH)의 수행시간은 표에서 찾아볼 수 없다는 것이다. 단순 휴리스틱의 경우 소요시간이 모두 0.01초 미만이었고 이 값들을 비교하는 것은 현실적으로 큰 의미를 갖기 어렵다고 판단하였기 때문이다.

아래 표를 대략적으로 훑어보면 본 장에서 제시한 유전 알고리즘(GA1 및 GA2)의 소요시간이 대체로 Joo & Kim(2012b)의 GA_DR보다 길다는 것을 발견할 수 있다. 계획대상 작업의 수(n)가 커질수록 그 차이도 커진다는 것을 알 수 있다. 또한 알고리즘의 종류에 관계없이 계획대상 작업의 수(n)가 커지면 소요시간도 길어지지만 기계의 수(m)는 소요시간에 큰 영향을 미치지 않는다는 것도 이 표를 통해 알 수 있다. 그러나 이러한 소요시간의 변화에도 불구하고 현실적인 관점에서 본다면 단순 휴리스틱은 물론이고 유전 알고리즘 기반의 휴리스틱도 실제상황에서 적용하기에 소요시간이 그리 길지 않다고 할 수 있다.

<표 3-3> Computation Time for GA-based Heuristics (seconds)

		n=20	*n*=30	*n*=40	*n*=50
m=2	GA1	0.02	0.04	0.10	0.14
	GA2	0.02	0.06	0.11	0.15
	J&K	0.02	0.04	0.06	0.08
m=3	GA1	0.02	0.06	0.09	0.15
	GA2	0.03	0.07	0.11	0.18
	J&K	0.01	0.01	0.05	0.05
m=4	GA1	0.04	0.04	0.09	0.10
	GA2	0.03	0.06	0.10	0.14
	J&K	0.01	0.04	0.03	0.09
m=5	GA1	0.03	0.04	0.09	0.11
	GA2	0.01	0.06	0.08	0.10
	J&K	0.02	0.02	0.04	0.09

6. 요약

본 장은 플라스틱 사출성형을 전문적으로 처리하는 소규모 제조기업의 사례 문제를 해결하기 위해 처리순서에 따라 작업준비시간이 달라지는 이종병렬기계 시스템의 일정계획 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 기존의 연구와 약간 다른 형태로 제시하였다. 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 두 개의 단순 휴리스틱 알고리즘과 두 개의 유전 알고리즘을 개발하였다. 다

양한 수치실험을 통해 이 알고리즘들이 비교적 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다. 특히 문제에 따라서는 단순 휴리스틱 알고리즘이 유전 알고리즘보다 오히려 나은 해를 제공하기도 하였다.

다음 장에서는 기계의 운전자에 대한 제약이 있는 상황에 대해 고려할 것이다. 즉, 한 명의 작업자가 주어진 이종병렬기계 시스템의 작업준비를 모두 담당하는 상황이다. 실제로 자동화가 많이 이루어진 현대의 생산시스템에서 이러한 상황은 쉽게 찾아볼 수 있는데, 특히 주문에 따른 다품종 소량생산을 하는 공장에서는 CNC밀링, 선반 등 기계의 종류별로 구획되어 있으며, 대개 이들 이종병렬기계를 운전하는 인력은 소수인 경우가 대부분이다.

Ⅳ. 셋업 작업자의 제약이 있는 이종병렬기계의 휴리스틱 일정계획

1. 개요

이종병렬기계의 일정계획을 소개한 기존 연구들은 각 기계에서의 기존 작업이 종료되면 그 즉시 다음 작업에 대한 셋업작업이 독립적으로 수행될 수 있다고 가정하였다. 즉, 셋업을 처리할 수 있는 작업자가 충분히 확보되어 있어서 필요할 경우 언제든지 그리고 어느 기계에서든지 셋업작업이 가능하다고 보는 것이다. 그러나 현실적으로 셋업 담당 작업자는 고도의 숙련도가 요구되는 고급 인력에 속하므로 충분한 수의 셋업 담당자를 확보하는 것은 매우 비용이 많이 들어가는 일이다. 따라서 실제 산업현장에서는 한 명의 작업자가 여러 기계를 담당하는 경우가 대부분이며, 특히 기계의 자동화가 진행될수록 이러한 현상은 더욱 심해진다.

셋업담당자 제약을 고려한 병렬기계 문제를 일반적인 일정계획문제 표현방식으로 나타내면 P,S|s_j|$C_{\max}$와 같이 쓸 수 있다. 여기서 P는 동종병렬기계를 의미하며 뒤에 숫자를 써서 기계대수를 표시할 수 있다. 또한 S는 셋업작업자(server) 제약을 의미하며 여기에도 숫자를 써서 작업자 수를 표시할 수 있다. s_j는 각 주문별로 셋업시간이 주어진다는 것을, 그리고 $C_{\max}$는 문제의 목적이 makespan 최소화라는 것을 의미한다. Kravchenko & Werner(1997)는 P,S1|s_j=1|

$C_{\max}$ 문제가 단항 NP-hard임을 보이고 P2,S1|s_j=1|$C_{\max}$ 문제에 대한 의사(pseudo) 다항 알고리즘을 개발하였다. 셋업담당자 제약을 고려한 문제의 난이도에 대한 연구는 Glass et al.(2000), Bruker et al.(2002), Haque & Armstrong(2007) 등에 의해서도 수행되었다. Abdekhodaee & Wirth(2002)와 Abdekhodaee et al.(2004, 2006)은 P2,S1|s_j=1|$C_{\max}$ 문제에서 각 주문의 셋업시간과 처리시간이 특이한 상황에 대해 최적해를 구하는 알고리즘과 여러 가지 휴리스틱 기법을 제안하였다. Glass et al.(2000)은 각 주문이 특정기계에 미리 할당되는 PD,S1|s_j|$C_{\max}$ 문제에 대한 근사해법을 제안하였다. Huang et al.(2010)은 셋업시간이 순서의존형인 PD,S1|s_{sd}|$C_{\max}$ 문제에 대한 수리모형과 유전알고리즘 기반의 해법을 개발하였다. 기계수나 사전 할당 등의 제한이 없는 일반적인 병렬기계 상황에서 셋업시간이나 처리시간에도 별다른 제한이 없는 가장 일반화된 문제는 Kim & Lee(2012)가 다루었다. 그들은 일반적인 P,S1|s_j|$C_{\max}$ 문제를 두 가지 형태의 혼합정수계획법 모형으로 표현하고 문제의 특성을 고찰하여 해를 구하는 휴리스틱 알고리즘을 개발하였다.

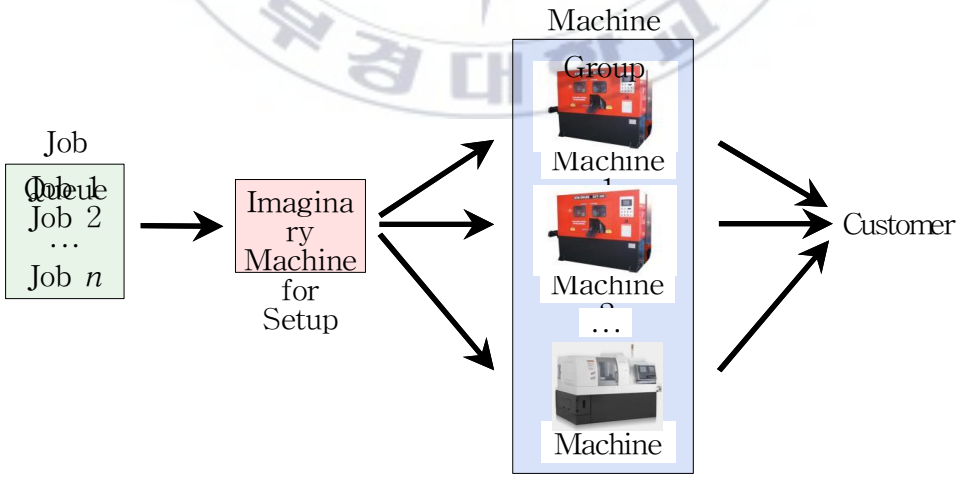
본 장에서는 앞서 소개한 이종병렬기계 상황에 셋업담당자 제약을 추가한 문제를 다루고자 하며, 이 문제는 R,S1|s_j|$C_{\max}$ 로 표현할 수 있다. 여기서 R은 동종병렬기계가 아닌 이종병렬기계를 의미한다. 이 문제를 해결하기 위해 본 장에서는 우선 수리적 모형을 개발하여 규모가 작은 문제에 대한 해결방안으로 활용한다. 그러나 문제의 규모가 커지면 수리적 모형에 의한 해는 기대할 수 없으므로, 효과적인 메타휴리스틱으로 잘 알려진 유전알고리즘을 활용한 휴리스틱 해법들을 개발하고 이 해법들에 대한 비교를 통해 문제에 적합한 휴리스틱

을 규명하도록 한다.

본 장의 나머지 부분은 다음과 같이 구성된다. 다음 절에서는 대상문제를 정수계획법의 형태로 모형화하고, 제3절에서는 대상문제를 해결하기 위한 유전 알고리즘 기반의 휴리스틱 해법을 제시하며, 제4절에서는 제안된 해법들의 성능을 평가하기 위한 수치실험을 소개한다. 마지막으로 제5절에서는 장을 요약한다.

2. 최적화 모형

본 장의 대상문제는 한 단계의 공정으로 처리가 완료되는 상황을 다루고 있지만, 모형화를 위해 각 주문의 처리시간을 셋업작업과 순수 작업시간으로 나누어 생각하기로 한다. 그러면 본 장에서 다루는 상황은 [그림 4-1]과 같은 2단계 흐름생산시스템으로 표현될 수 있다.



[그림 4-1] 셋업작업자 1명인 경우 m생산 공정 개념도

[그림 4-1]을 보면 일단 여러 개의 주문들이 생산시스템 앞에 대기하고 있다. 이 때 기계 한 대와 셋업 담당자가 가용해지면 대기 중이던 주문들 중 하나가 선택되어 셋업 작업자에 의해 해당 기계에 맞추어 셋업 작업이 착수된다. 셋업작업이 완료된 순간 그 주문의 나머지 작업(셋업을 제외한 순수한 작업)이 착수되고 작업자는 다른 기계가 가용할 경우 다른 주문에 대한 셋업작업을 시작할 수 있다. 따라서 그림과 같이 한 명의 셋업 담당자를 가상의 기계 한 대로 이루어진 1단계 공정으로 보고, 실제 기계군을 이종병렬기계로 이루어진 2단계 공정으로 보는 모형이 만들어지게 되는 것이다. 다만 여기서 셋업시간은 1단계 공정의 처리시간으로 취급되므로 이 그림의 각 공정에는 셋업시간이 별도로 고려될 필요가 없다.

한 개 이상의 단계가 병렬기계로 이루어진 다단계 흐름생산라인의 일정계획문제는 Hybrid flow shop scheduling 이라는 분야로 분류되며 상당히 많은 연구가 이루어져 왔다(Ruiz & Vazquez-Rodriguez, 2010). Lee & Park(1999)은 첫 번째 단계가 두 대의 이종병렬처리기계로 이루어져 있고 두 번째 단계는 한 대의 기계로 이루어진 혼합흐름 생산시스템에서 전체작업완료시간을 최소화하기 위한 알고리즘을 개발하였다. 또한 Jeong & Jeong(2000)은 각 단계가 이종병렬기계들로 구성된 다단계 생산시스템의 생산일정계획 지원시스템을 개발하기 위해 작업완료시간, 납기, 가공시간, 긴급률 등을 고려한 비용최소화 방법론을 연구하였다.

병렬기계를 포함하는 흐름라인 일정계획문제도 이와 같이 상당히 많이 연구되었지만 본 연구와 같은 형식의 제약조건을 다룬 연구는 아직 없었다. 그림이 표현하고 있는 본 연구의 대상시스템에서 단계

1의 가상기계는 원래 단계 2에 소속된 기계의 셋업작업 만을 분리시킨 공정이다. 따라서 단계 1의 작업시간은 그 주문이 후속단계에서 어느 기계에 투입될 것인지에 따라 달라지게 된다. 또한 1단계 가상기계의 가용 여부도 2단계에 속한 기계들의 가용 여부에 종속되어 있다. 즉, 단계 1의 가상기계에서 현재 처리중인 작업이 없다 하더라도 단계 2의 병렬기계가 모두 작업 중이라면 단계 1에서 작업이 시작될 수는 없는 것이다.

모형의 개발을 위해서는 문제를 명확하게 정의해야 하며 이를 위해 다음과 같이 가정하도록 한다. 우선 문제를 해결하는 시점은 작업자가 하나의 셋업작업을 마치는 순간이다. 즉, 셋업작업을 마치는 시점에 그 때까지 알려진 주문 정보와 기계 정보를 사용해 새로운 작업일정을 작성하는 것이다. 그리고 작업자의 셋업작업이 끝나는 순간 기계의 나머지 처리가 바로 시작된다고 가정한다. 마지막으로 각 주문의 셋업시간은 그 주문이 어느 기계에서 처리되느냐에 따라 변하지만 처리순서와는 무관하다고 가정한다. 이러한 가정 하에서 모형을 개발하였으며, 모형에서 사용되는 기호에 대해 먼저 아래와 같이 설명한 다음 개발된 모형을 소개한다.

<파라메터>

i, j : 주문을 표시하는 인덱스 (i, j = 1, 2, ..., n)

k : 기계를 표시하는 인덱스 (k = 1, 2, ..., m)

a_i : 계획시점 이전에 주문 i가 도착하였으면 0, 그렇지 않으면 도착예정시간

r_k : 계획시점에 기계 k가 작업 중이면 그 작업의 완료시간,

그렇지 않으면 0

s_{ik} : 주문 i가 기계 k에 할당되었을 경우의 셋업시간

p_{ik} : 주문 i가 기계 k에 할당되었을 경우의 처리시간 (셋업시간 제외)

M : 충분히 큰 수

<결정변수>

x_{1i} = 주문 i의 1단계 공정 착수시간 (즉, 셋업 착수시간)

x_{2i} = 주문 i의 2단계 공정 착수시간 (즉, 기계 작업 착수시간)

f_i = 주문 i의 2단계 공정 완료시간 (1단계 완료시간은 2단계 착수시간과 동일하므로 별도의 변수 없음)

$f_{\max}$ = 모든 주문의 완료시간 (makespan)

u_i = 1단계 공정에서 가장 먼저 처리되는 주문이 i이면 1,
그렇지 않으면 0

u_{ij} = 1단계 공정에서 주문 i에 이어 주문 j가 처리되면 1,
그렇지 않으면 0

y_{ik} = 주문 i가 기계 k에 할당되면 1, 그렇지 않으면 0

z_{ik} = 기계 k에서 가장 먼저 처리되는 주문이 i이면 1,
그렇지 않으면 0

z_{ijk} = 기계 k에서 주문 i에 이어 주문 j가 처리되면 1,
그렇지 않으면 0

<모형>

Minimize $f_{\max}$ (4-1)

subject to

$x_{1i} \geq a_i$ for all i (4-2)

$x_{1i} + M(1 - z_{ik}) \geq r_k$ for all $i,\ k$ (4-3)

$$x_{1i} + \sum_{k=1}^{m} s_{ik} y_{ik} = x_{2i} \text{ for all } i \quad (4\text{-}4)$$

$$x_{2i} + \sum_{k=1}^{m} p_{ik} y_{ik} = f_i \text{ for all } i \quad (4\text{-}5)$$

$$f_i \leq x_{1j} + M(1 - \sum_{k=1}^{m} z_{ijk}) \text{ for all } i \neq j \quad (4\text{-}6)$$

$x_{2i} \leq x_{1j} + M(1 - u_{ij})$ for all $i \neq j$ (4-7)

$f_i \leq f_{\max}$ for all i (4-8)

$$\sum_{k=1}^{m} y_{ik} = 1 \text{ for all } i \quad (4\text{-}9)$$

$$\sum_{i=1}^{n} z_{ik} = 1 \text{ for all } k \quad (4\text{-}10)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} z_{ijk} \leq y_{ik} \text{ for all } i,\ k \quad (4\text{-}11)$$

$$z_{ik} + \sum_{\substack{j=1 \\ j \neq i}}^{n} z_{jik} = y_{ik} \text{ for all } i,\ k \quad (4\text{-}12)$$

$$\sum_{i=1}^{n} u_i = 1 \tag{4-13}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} u_{ij} \leq 1 \text{ for all } i \tag{4-14}$$

$$u_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} u_{ji} = 1 \text{ for all } i \tag{4-15}$$

u_i : 0/1 integer for all i

u_{ij} : 0/1 integer for all i, j

y_{ik}, z_{ik} : 0/1 integer for all i, k

z_{ijk} : 0/1 integer for all i, j, k

식 (4-1)은 본 문제가 총 완료시간을 최소화하는 문제라는 것을 보여준다. 제약조건 (4-2)는 각 주문의 셋업작업이 그 주문의 도착(확정) 시점 이후에 시작될 수 있음을 나타낸다. 식 (4-3)은 각 주문의 셋업작업이 그 주문을 처리하는 기계가 가용해지는 시점(r_k) 이후에 시작될 수 있다는 조건을 표현하고 있다. 식 (4-4)는 1단계 (셋업) 작업이 끝나자마자 2단계 작업이 시작되어야 함을 표현하며, 식 (4-5)는 각 주문이 2단계 공정에서 완료되는 시점이 언제인지를 나타낸다. 식 (4-6)은 셋업을 위한 기계 가용성을 표현하는데, 한 기계에서 선후관계를 갖는 주문들은 선행 주문이 2단계 공정에서 완료(f_i)되어야 후속 주문의 셋업이 착수(x_{1j})될 수 있다는 것을 보여준다. 식 (4-7)은 셋업을 위한 작업자 가용성을 표현하는데, 1단계 공정(즉, 셋업 작업

자)에서 선후관계를 갖는 주문들은 선행 주문의 셋업이 완료(x_{2i})되어야 후속 주문이 셋업이 착수(x_{1j})될 수 있음을 보여준다. 식 (4-8)은 총 완료시간이 모든 주문들의 완료시간 중 가장 큰 값이라는 것을 보여준다. 식 (4-9)는 각 주문이 한 대의 기계에만 할당될 수 있도록 해준다. 식 (4-10)은 각 기계에 최초로 배정되는 주문은 한 개라는 것을 의미한다. 식 (4-11)에 의하면 어떤 주문이 어떤 기계에 배정되지 않았다면($y_{ik} = 0$) 그 주문은 그 기계에서 다른 어떤 주문의 선행주문도 될 수 없다($z_{ijk} = 0$)는 것을 보여준다. 반대로 해석하면 어떤 주문이 하나의 기계에 배정되었다면($y_{ik} = 1$) 그 주문은 그 기계에서 다른 어떤 주문의 선행작업이라는 것($z_{ijk} = 1$)을 보여준다. 그런데 이 식이 부등식인 것은 그 주문이 그 기계에서 최종으로 처리되는 주문일 경우 때문이다. 비슷하게 식 (4-12)에서는 어떤 주문이 하나의 기계에 배정되었다면($y_{ik} = 1$) 그 주문은 그 기계에서 처리되는 최초의 주문($z_{ik} = 1$)이거나 다른 어떤 주문의 후속주문이라는 것($z_{jik} = 1$)을 보여준다. 식 (4-13)은 최초로 셋업을 수행하는 주문은 하나일 수밖에 없음을 보여준다. 식 (4-14)에 의하면 셋업담당자가 주문 i 다음에 처리하는 주문은 한 개이거나 없다. 없는 경우는 그 주문이 마지막 주문인 경우이다. 반대로 식 (4-15)에 의하면 어떤 주문의 셋업순서는 처음이거나 혹은 다른 어떤 주문의 뒤 순서이다.

이 모형도 문제의 크기(주문수 및 기계수)가 작은 경우 CPLEX나 LINGO 같은 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 예를 들기 위해 다음과 같이 주문의 수가 7이고 기계수가 3인 문제를 생각한다.n = 7, m = 3, $\boldsymbol{a}$ = (0, 0, 0, 2, 6, 0, 8), $\boldsymbol{r}$ = (0, 3, 5),

$$\boldsymbol{s} = \begin{bmatrix} 2\,3\,1 \\ 2\,1\,3 \\ 1\,2\,2 \\ 2\,3\,2 \\ 1\,3\,4 \\ 2\,1\,2 \\ 3\,2\,3 \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} 9\,6\,8 \\ 6\,9\,5 \\ 5\,7\,9 \\ 8\,6\,9 \\ 5\,9\,7 \\ 6\,9\,5 \\ 5\,7\,9 \end{bmatrix}.$$

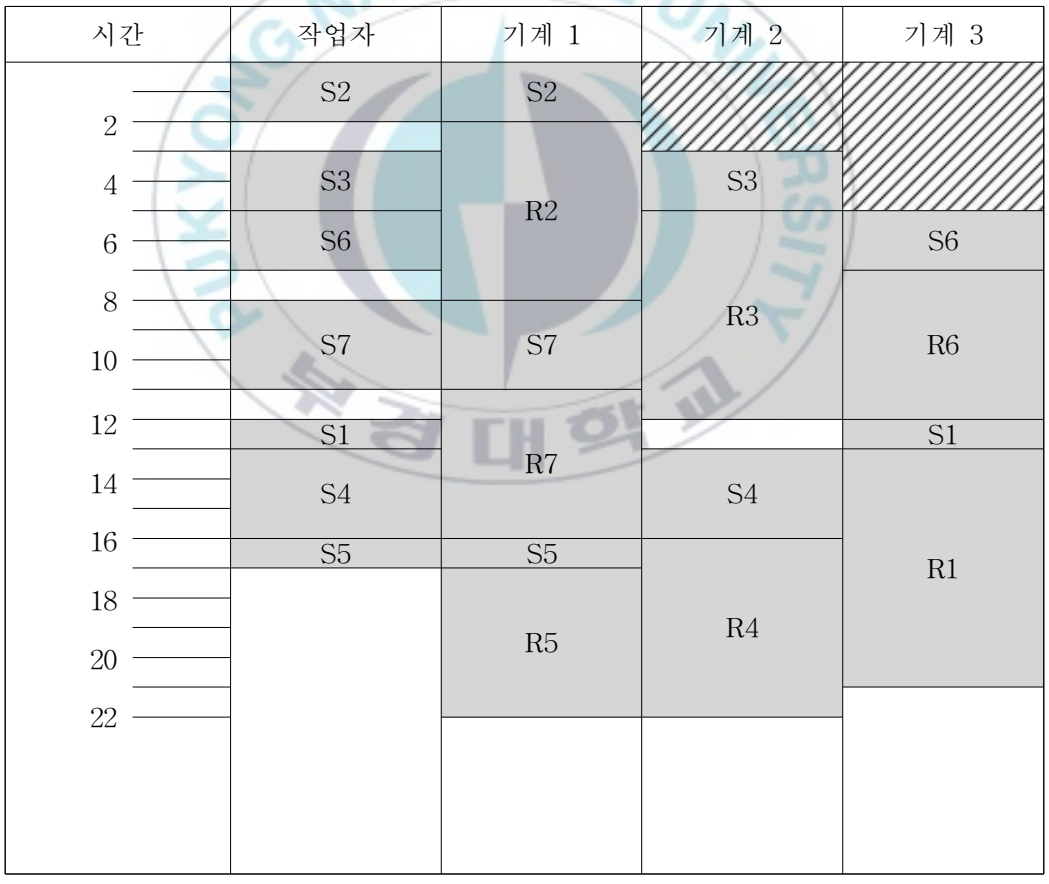
이 데이터를 사용해 [그림 4-2]와 같은 LINGO 프로그램으로부터 최적해를 구할 수 있다.

Lingo Model - op1_makespan

```
MODEL:
 SETS:
       Job: a, x1, x2, u0, f;
       Mch: r;
       JM(Job, Mch): p, s, z0, y;
       JJ(Job, Job): u;
       JJM(Job, Job, Mch): z;
 ENDSETS

 DATA:
       n = 7;
       m = 3;

       BM = 9999;
       Job = 1..n;
       Mch = 1..m;

a = 0 0 0 2 6 0 8 ;
r = 0 3 5 ;
p = 9 6 8   6 9 5   5 7 9   8 6 9   5 9 7   6 9 5   5 7 9 ;
s = 2 3 1   2 1 3   1 2 2   2 3 2   1 3 4   2 1 2   3 2 3 ;

 ENDDATA

 MIN = fmax;

 @FOR(Job(i): x1(i) > a(i));
 @FOR(JM(i, k): x1(i) + BM * (1 - z0(i, k)) > r(k));

 @FOR(Job(i): x1(i) + @SUM(Mch(k): s(i, k) * y(i, k)) = x2(i));
 @FOR(Job(i): x2(i) + @SUM(Mch(k): p(i, k) * y(i, k)) = f(i));
 @FOR(JJ(i, j) | i #ne#j: x1(j) + BM * (1 - @SUM(Mch(k): z(i, j, k))) > f(i));
 @FOR(JJ(i, j) | i #ne#j: x1(j) + BM * (1 - u(i, j)) > x2(i));

 @FOR(Job(i): f(i) < fmax);

 @FOR(Job(i): @SUM(Mch(k): y(i, k)) = 1);
 @FOR(Mch(k): @SUM(Job(j): z0(j, k)) = 1);

 @FOR(JM(i, k): @SUM(Job(j) | j #ne# i: z(i, j, k)) < y(i, k));
 @FOR(JM(i, k): z0(i, k) + @SUM(Job(j) | j #ne# i: z(j, i, k)) = y(i, k));

 @SUM(Job(i): u0(i)) = 1;
 @FOR(Job(i): @SUM(Job(j) | j #ne# i: u(i, j)) < 1);
 @FOR(Job(i): u0(i) + @SUM(Job(j) | j #ne# i: u(j, i)) = 1);

 @FOR(Job(i): @BIN(u0(i)));
 @FOR(JM(i, k): @BIN(y(i, k)));
 @FOR(JJ(i, j): @BIN(u(i, j)));
 @FOR(JM(i, k): @BIN(z0(i, k)));
 @FOR(JJM(i, j, k): @BIN(z(i, j, k)));

END
```

[그림 4-2] 예제의 LINGO 프로그램(작업자 1인)

이 최적해는 [그림 4-3]과 같이 작업관리 분야에서 사용되는 Man-Machine Chart 형태로 표현할 수 있다. 이 그림에서 작업자와 기계1에 공통적으로 표시된 S2는 2번 주문에 대한 셋업작업을 의미하며 기계1에만 표시된 R2는 주문2의 기계 처리시간을 의미한다. 그림을 살펴보면 주문들의 셋업처리 순서는 2-3-6-7-1-4-5 이고 기계1의 주문 처리순서는 2-7-5, 기계2는 3-4, 기계3은 6-1임을 알 수 있다. 또한 최종 완료시간은 22가 된다.



[그림 4-3] Man-Machine Chart 형태로 표현한 최적 스케줄

3. 혼합 유전알고리즘 방법론

본 장의 대상 문제는 크기(주문수 및 기계수)가 작은 경우 CPLEX나 LINGO 같은 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 그러나 잘 알려진 바처럼 이 문제는 NP-hard 에 속하므로 문제의 크기가 커질 경우 현실적인 시간 안에 최적해를 구하는 것은 불가능한 일이다. 실제로 앞 절에서 제시한 예제(주문수가 7이고 기계수가 3)의 최적해는 LINGO를 사용해 4GB RAM과 2.60GHz G620 CPU가 장착된 PC에서 12분 25초만에 구할 수 있었으나 주문의 수가 8이고 기계수가 3인 문제는 1시간 안에 해가 구해지지 않아 프로그램을 중단할 수밖에 없었다.

본 장에서는 현실적인 크기의 문제를 해결하기 위해 혼합 유전알고리즘(Hybrid Genetic Algorithm)을 제안한다. 대상문제의 해결과정이 1)주문을 기계에 할당하고 2)할당된 주문들의 순서를 결정하는 두 단계로 이루어져 있다고 보고, 첫 단계인 할당 문제는 유전 알고리즘의 염색체를 통해 해결하고, 두 번째 단계인 순서결정 문제는 각 할당(즉, 염색체)에 대해 간단한 휴리스틱 알고리즘을 적용하여 해결하도록 한다. 다시 말해 전체적인 절차는 일반적인 유전 알고리즘과 대동소이한데 염색체-해 전환 (Decoding) 절차에 간단한 휴리스틱의 개념을 도입한 것이다. 그 절차를 설명하면 다음과 같다.

우선 염색체 표현은 주문 수(n) 만큼의 (0, 1) 구간 실수를 사용한다. 이 실수 값을 사용해 각각의 주문을 기계에 할당하는데, 그 방법은 각 주문이 각 기계에 할당되는 기대비율을 동일하게 유지하기 위하여 구간 $(\frac{k-1}{m}, \frac{k}{m})$에 유전자 값이 포함되면 그 주문을 기계 k 에

배정하도록 하는 것이다. 예를 들어 주문이 5개이고 기계가 3대인 경우 염색체는 (0.567, 0.132, 0.982, 0.248, 0.724)와 같은 5개의 실수로 구성될 수 있고, 첫 번째 유전자 값인 0.567에 의해 첫 번째 주문의 기계할당이 이루어지는데, 그 값이 (1/3, 2/3)에 속하므로 첫 번째 주문은 기계 2에 배정된다. 마찬가지 방식으로 두 번째 주문은 0.132 $\in$ (0/3, 1/3)이므로 기계 1에, 세 번째 주문은 0.982 $\in$ (2/3, 3/3)이므로 기계 3에 할당된다. 같은 방법으로 주문 4와 5는 기계 1과 3에 각각 할당된다. 이렇게 하나의 염색체에 대해 주문-기계 할당이 이루어지면 그 다음 절차로는 간단한 휴리스틱에 의해 각 기계에 대한 주문들의 처리순서가 결정되는데 본 연구에서는 세 가지 방식을 제안한다.

첫 번째 방식은 작업시간이 긴 주문을 우선적으로 처리하는 LPT(Longest Processing Time) 규칙을 사용한다. Pinedo(2002)에 의하면 기본적인 병렬기계문제에서 LPT 규칙은 상당히 훌륭한 Worst Case Bound를 가지는 좋은 휴리스틱이다. 따라서 본 연구에서는 주문들이 기계에 할당되어 있는 경우 처리순서를 결정하는 첫 번째 방법으로 LPT 규칙을 사용하였다. 염색체 하나가 주어졌을 때 이 염색체를 해로 전환(Decoding)하는 전체과정은 다음과 같다.

***DECODE_LPT* Algorithm**

Step 1. 염색체 내부의 i 번째 유전자 값이 구간 $(\frac{k-1}{m}, \frac{k}{m})$에 속하면 주문 i 를 기계 k 에 배정하고, 기계 k 에 배정된 주문들의 집합을 J_k 로, 한 개 이상의 주문이 할당된 기계들의 집합을 M 으로 놓는다.

Step 2. 작업자가 가용해지는 시점을 $avail$ 로, 기계 k 가 가용해지는 시점을 ms_k 로 놓는다. 그러면 $avail = 0$, $ms_k = r_k$ for $k \in M$ 이다.

Step 3. $M = \varnothing$ 이면 ms_k (k = 1, 2, ..., m) 중 최대값을 $f_{\max}$ 로 놓고 알고리즘을 종료한다.

Step 4. M 에 속한 k 중에서 ms_k가 최소가 되는 k 를 k^* 로 놓는다. 만약 $J_{k^*} = \varnothing$ 이면 M 에서 k^*를 제거하고 <Step 3>으로 간다.

Step 5. J_{k^*} 에 속한 모든 주문 i 에 대해 max{$avail$, a_i, ms_{k^*}} + s_{ik^*} + p_{ik^*} 를 계산하고 그 값이 최대가 되는 주문 i^* 를 기계 k^* 의 다음 처리 주문으로 한다. 주문 i^* 를 집합 J_{k^*} 에서 제거한다.

Step 6. $avail$ = max{$avail$, a_{i^*}, ms_{k^*}} + s_{ik^*} 와 같이 $avail$ 을 재계산한다.

Step 7. ms_{k^*} = $avail$ + p_{ik^*} 와 같이 ms_{k^*} 를 재계산하고 <Step 4>로 간다.

앞서 2절에서 문제를 해결하는 시점을 작업자가 하나의 셋업작업을 마치는 순간이라고 가정하였다. 위 알고리즘에서 사용된 변수 $avail$ 이 바로 이 시점을 의미한다. 재계산된 각 $avail$ 시점에서 향후 가장 빨리 가용해지는 기계를 찾고 (Step 4), 그 기계에 배정되었지만 아직 착수되지 않은 주문들 중에서 예상종료시간이 가장 큰 주문을 찾아 다음 착수주문으로 하는 (Step 5) 것이 이 알고리즘의 핵심내용이다. 여기서 각 주문의 예상종료시점은 현 시점($avail$)과 그 주

문의 가용시점(a_i), 그리고 해당기계의 가용시점(ms_{k^*}) 중 최대값과 셋업(s_{ik^*}) 및 처리시간(p_{ik^*})의 합으로 계산할 수 있다.

이 알고리즘의 [Step 5]에서는 예상종료시간이 가장 큰 주문을 선택함으로써 LPT 규칙을 적용하였다. 이 방식과의 비교를 위해 본 연구에서는 일정계획 분야에서 많이 사용되는 또 하나의 규칙인 SPT(Shortest Processing Time) 규칙도 적용하기로 한다. 이 방식은 위의 ***DECODE_LPT*** 알고리즘에서 [Step 5]의 '최대'를 '최소'로 바꿔주기만 하면 된다. 이 방식의 이름은 ***DECODE_SPT*** 알고리즘으로 하며 다른 모든 내용은 ***DECODE_LPT*** 알고리즘과 동일하므로 따로 설명하지는 않기로 한다.

본 연구에서 제안하는 세 번째 Decoding 방식은 두 단계 흐름생산시스템의 총 완료시간을 최소화하는 문제에서 최적해를 찾아주는 Johnson 규칙을 활용하는 것이다. 앞서 설명한 것처럼 본 연구에서 다루는 문제는 개념적으로 두 단계 문제와 유사한 부분이 있다. 물론 Johnson 규칙이 적용되는 단순한 두 단계 문제와는 다르지만, 주문들이 기계에 할당되어 있는 경우 그 규칙을 적용하여 주문들의 처리순서를 결정하는 것이 가능하며 그 절차는 다음과 같다.

***DECODE_Johnson* Algorithm**

Step 1. 염색체 내부의 i 번째 유전자 값이 구간 $(\frac{k-1}{m}, \frac{k}{m})$에 속하면 주문 i 를 기계 k 에 배정하고, 기계 k 에 배정된 주문들의 집합을 J_k 로, 한 개 이상의 주문이 할당된 기계들의 집합을 M 으로 놓는다.

Step 2. 작업자가 가용해지는 시점을 $avail$ 로, 기계 k 가 가용해지는 시점을 ms_k 로 놓는다. 그러면 $avail = 0$, $ms_k = r_k$ for $k \in M$ 이다.

Step 3. $M = \varnothing$ 이면 ms_k (k = 1, 2, ..., m) 중 최대값을 $f_{\max}$ 로 놓고 알고리즘을 종료한다.

Step 4. M 에 속한 k 중에서 ms_k가 최소가 되는 k 를 k^* 로 놓는다. 만약 $J_{k^*} = \varnothing$ 이면 M 에서 k^*를 제거하고 <Step 3>으로 간다.

Step 5. J_{k^*} 에 속한 주문 i 중에서 max{$avail$, a_i, ms_{k^*}} - $avail$ + $s_{ik^*} < p_{ik^*}$ 를 만족하는 주문들의 집합을 U 로, 그렇지 않은 주문들의 집합을 V 로 놓는다.

Step 6. $U \neq \varnothing$ 이면 U 에 속한 i 중에서 max{$avail$, a_i, ms_{k^*}} - $avail$ + s_{ik^*} 가 최소가 되는 i 를 i^* 로 놓고, $U = \varnothing$ 이면 V 에 속한 i 중에서 p_{ik^*} 가 최대가 되는 i 를 i^* 로 놓는다. 주문 i^* 를 기계 k^* 의 다음 처리 주문으로 한다. 주문 i^* 를 집합 J_{k^*} 에서 제거한다.

Step 7. $avail$ = max{$avail$, a_{i^*}, ms_{k^*}} + s_{ik^*} 와 같이 $avail$ 을 재계산한다.

Step 8. ms_{k^*} = $avail$ + p_{ik^*} 와 같이 ms_{k^*} 를 재계산하고 <Step 4>로 간다.

이 알고리즘에서도 재계산된 각 $avail$ 시점에서 향후 가장 빨리 가용해지는 기계를 찾는 과정(Step 4)은 앞서 보았던 ***DECODE_LPT***

알고리즘과 동일하다. 그 기계에서 다음에 착수할 주문을 선정할 때 Johnson 규칙을 사용한다. 일반적으로 Johnson 규칙을 적용하는 방법은 1) 선행기계의 처리시간이 후속기계 처리시간보다 짧은 주문이 있다면 그 주문들 중에서 선행기계 처리시간이 최소인 주문을 먼저 처리하고, 2) 그런 주문이 없다면 후속기계 처리시간이 최대인 주문을 먼저 처리하는 것이다. 본 연구에서는 이 규칙을 적용하기 위해 주문 i 의 선행기계 처리시간으로 max{$avail$, a_i, ms_{k^*}} - $avail$ + s_{ik^*} 을 사용하였는데, 이 값은 현 시점을 기준으로 그 주문의 셋업을 종료할 때까지의 예상소요시간을 의미한다. 물론 후속기계의 처리시간으로는 실제 처리시간인 p_{ik^*} 를 사용하면 된다.

위에서 제시한 염색체-해 전환 (Decoding) 절차 이외의 과정은 일반적인 유전 알고리즘의 전개와 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.

$$F_i = Z_{\max} - Z_i \tag{4-16}$$

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 연산(Genetic Operation)을 적용하여 다음 세대의 염색체를 만들어 낸다. 이 유전

연산에는 매우 다양한 방법들이 있으나 본 연구에서는 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 사용하였다. 또한 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_c 보다 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 다음 과정을 수행한다. (0, 1) 구간의 난수를 하나 생성한다. 이 난수가 미리 정한 돌연변이 확률 P_m 보다 작으면 (0, 1) 구간에서 새로운 난수를 하나 생성하여 기존의 유전자를 대치한다. 여기서도 P_m의 값은 실험을 통해 선정한다.

4. 수치 실험

앞서 제시한 혼합 유전알고리즘들의 성능을 평가하기 위해 본 절에서는 임의로 생성한 예제들을 사용해 알고리즘의 결과를 비교한다. 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 알고리즘들의 성능을 서로 비교하도록 한다. 문제의 규모는 주문수(n)와 기계수(m)가 결정하므로 이 두 모수들의 수준을 각 단계별로 다르게 통제하도록 한다. 우선 첫 번째 단계에서는 주문수를 5와 6 중에서 하나로 하고 기계수를 2와 3 중 하나로 하여 총 4개 (n, m) 조합에 대해 실험을 수행한다. 두 번째 단계에서는 주문의 수를 20, 30, 40, 50 등 4개 값 중에서 하나로 하고 기계의 수는 2, 3, 4, 5 등 4개 중에서 하나로 하여 총 16개 (n, m) 조합의 문제를 생성하여 실험을 진행한다.

각각의 (n, m) 조합에 대해서는 5개의 문제를 임의로 생성한다. 문제를 생성하기 위해서는 a_i, r_k, p_{ik}, s_{ik}, s_{ijk} (i, j = 1, 2, ..., n, k = 1, 2, ..., m) 값들을 생성하여야 하는데 그 방법은 다음과 같다. 우선 a_i 값들을 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. r_k 값도 $U[0, 10]$로 가정한다. 마지막으로 처리시간(p_{ik})과 준비시간(s_{ik} 및 s_{ijk})은 모두 $U[5, 15]$로

가정하였다.

<표 4-1> Results of small problems(작업자 1인)

		n=5				n=6			
		makespan			LINGO comp. time	makespan			LINGO comp. time
		John	LPT	SPT		John	LPT	SPT	
m=2	mean	1.02	1.03	1.05	8.2	1.05	1.08	1.05	305.6
	min	1.00	1.02	1.02	3	1.00	1.02	1.02	180
	max	1.05	1.05	1.08	15	1.09	1.18	1.09	389
m=3	mean	1.01	1.02	1.04	15.0	1.01	1.03	1.05	933.8
	min	1.00	1.00	1.03	9	1.00	1.00	1.00	355
	max	1.03	1.09	1.09	21	1.03	1.08	1.10	1336

첫 번째 단계의 실험결과는 <표 4-1>에 정리되어 있다. 앞서 설명한 바와 같이 이 단계에서는 4개의 (n, m) 조합에 대해 실험을 수행하였다. 우선 각 조합에 대해 5개의 문제를 임의로 생성하고, 각 문제에 대한 최적해를 LINGO 시스템으로 구한 다음, 본 연구에서 제안한 3개의 유전알고리즘으로 각각 해를 구한다. 예를 들어 표의 좌상부 셀은 (n, m) = (5, 2)인 경우의 ***DECODE_Johnson*** 알고리즘에 대한 결과로서 그 값 1.02, 1.00, 1.05는 다음과 같은 절차에 의해 구해졌다.

1) 계획대상 작업수(n)가 5이고 기계의 수(m)가 2인 상황에서 생성된 5개의 서로 다른 문제에 대해 LINGO 시스템으로 최적해를

구한 결과 그 최종완료시간(makespan)은 41, 40, 40, 38, 42 등이었다.

2) ***DECODE_Johnson*** 알고리즘을 적용한 유전알고리즘으로 같은 문제를 푼 결과는 42, 42, 40, 39, 43 등이었다.

3) ***DECODE_Johnson*** 알고리즘의 결과를 최적 결과로 나누면 42/41=1.02, 42/40=1.05, 40/40=1.00, 39/38=1.03, 43/42=1.02 등이 된다.

4) 이 5개의 값들에 대한 평균은 1.02, 최소는 1.00, 최대는 1.05가 되고 이 세 값들을 표에 수록하였다.

같은 방법으로 ***DECODE_LPT*** 와 ***DECODE_SPT*** 에 대해 계산한 결과가 그 오른쪽에 표시되어 있다. 그 다음 오른쪽에 표시된 8.2, 3, 15 등 3개 값들은 LINGO 시스템의 계산시간을 표시한 것이다. 실제 5개 문제에 대한 계산시간은 3초, 8초, 15초, 5초, 10초였으며 이 값들에 대한 평균값, 최솟값, 최댓값 등을 표시하였다. 유전알고리즘에 대한 계산시간은 표시하지 않았는데 이것은 모든 문제에 대해 유전알고리즘 계산시간이 0.1초 미만이었기 때문이다.

<표 4-1>의 값들을 관찰해보면 본 연구에서 제안한 알고리즘들의 성능에 대한 개략적인 평가를 할 수 있으며 그 결과는 다음과 같이 정리할 수 있다.

1) 세 개의 휴리스틱을 비교해보면 ***DECODE_Johnson***이 다른 두 휴리스틱에 비해 거의 모든 경우에서 우수한 해를 제공해주었다. 전체 생성된 20개 문제 중 한 개 문제에서 ***DECODE_LPT***가 ***DECODE_Johnson***보다 나은 결과를 보여주기도 하였으나

DECODE_SPT가 ***DECODE_Johnson***보다 나은 경우는 단 한 번도 없었다.

2) 최적해와 휴리스틱을 비교해보면 ***DECODE_Johnson***의 경우 평균값을 기준으로 최적해에 비해 1~5% 정도 전체 완료시간(makespan)이 지연되었다. 또한 최악의 경우(최대값)에도 그 오차는 10% 미만이었다. 따라서 ***DECODE_Johnson*** 알고리즘을 사용할 경우 운이 좋으면 최적해를, 운이 나빠도 최적해보다 10% 미만의 나쁜 해를 구할 수 있으므로 현실적인 측면에서 우수한 휴리스틱이라고 할 수 있다.

3) LINGO 시스템을 통한 최적해 도출시간을 비교해보면 기계수(m)보다는 작업수(n)가 그 시간에 많은 영향을 주는 것을 발견할 수 있다. 이것은 앞서 2절의 수리모형에서 14개의 식으로 주어진 제약조건들 중 기계수(m)에만 영향을 받는 식은 (10) 한 개이고, 기계수(m)와 작업수(n) 모두에 영향을 받는 식은 (3), (11), (12) 등 3개, 그리고 작업수(n)에만 영향을 받는 식은 9개라는 사실을 통해 이해할 수 있다.

<표 4-2> Results of large problems(작업자 1인)

		n=20		n=30		n=40		n=50	
		LPT	SPT	LPT	SPT	LPT	SPT	LPT	SPT
m=2	mean	0.96	0.97	0.95	0.96	0.96	0.96	0.96	0.96
	min	0.93	0.95	0.94	0.95	0.94	0.95	0.94	0.94
	max	0.99	0.98	0.96	0.96	0.97	0.98	0.97	0.97
m=3	mean	1.01	1.02	1.01	1.02	1.01	1.02	1.00	1.01
	min	1.00	1.02	0.99	1.01	1.00	1.01	0.99	1.00
	max	1.02	1.03	1.02	1.02	1.01	1.03	1.01	1.02
m=4	mean	1.01	1.02	1.00	1.02	1.01	1.02	1.00	1.01
	min	0.99	1.00	0.99	1.01	1.00	1.01	0.99	1.00
	max	1.03	1.03	1.02	1.02	1.02	1.03	1.01	1.02
m=5	mean	1.00	1.01	1.00	1.01	1.00	1.02	1.01	1.02
	min	0.98	0.99	1.00	1.01	0.99	1.01	1.00	1.01
	max	1.03	1.02	1.01	1.02	1.01	1.03	1.02	1.03

<표 4-2>는 두 번째 단계의 실험결과를 보여주고 있다. 이 실험은 주문의 수를 20, 30, 40, 50 중에서 하나로 하고 기계의 수는 2, 3, 4, 5 중 하나로 하는 총 16개 (n, m) 조합의 문제에 대한 실험이다. 이런 규모의 문제에 대해서는 LINGO 시스템을 통한 최적해를 현실적인 시간 내에 구할 수가 없으므로, 본 연구에서 제시한 3가지 형태의 유전알고리즘으로 해를 구한 다음 ***DECODE_Johnson*** 알고리즘의 결과를 기준으로 나머지 두 알고리즘의 결과들을 비교하였다. 예를 들어 <표 4>의 좌상부 셀은 (n, m) = (20, 2)인 경우의 ***DECODE_Johnson*** 알고리즘과 ***DECODE_LPT*** 알고리즘을 비교한 결과로서 그 값 0.96, 0.93, 0.99는 다음과 같은 절차에 의해 구해졌다.

1) 계획대상 작업수(n)가 20이고 기계의 수(m)가 2인 상황에서 생성된 5개의 서로 다른 문제에 대해 ***DECODE_Johnson*** 알고리즘을

적용한 유전알고리즘으로 해를 구한 결과 그 최종완료시간(makespan)은 151, 144, 139, 141, 142 등이었다.

2) ***DECODE_LPT*** 알고리즘을 적용한 유전알고리즘으로 같은 문제를 푼 결과는 141, 136, 137, 134, 138 등이었다.
3) ***DECODE_LPT*** 알고리즘의 결과를 ***DECODE_Johnson*** 알고리즘의 결과로 나누면 0.93, 0.94, 0.99, 0.95, 0.97 등이 된다.
4) 이 5개의 값들에 대한 평균은 0.96, 최소는 0.93, 최대는 0.99이다.

그 오른쪽에 표시된 0.97, 0.95, 0.98은 같은 방법으로 ***DECODE_Johnson*** 알고리즘과 ***DECODE_SPT*** 알고리즘에 대해 계산한 결과이고, 표에 제시되 다른 값들은 다른 (n, m) 조합에 대해 실시한 같은 종류의 비교결과를 정리한 것이다. 이 실험에서는 LINGO 최적해를 구하지 않았으므로 <표 4-1>과 달리 <표 4-2>에는 계산시간에 대한 결과가 없다. 앞서 첫 단계의 실험에서도 유전 알고리즘의 수행시간이 0.1초 미만으로 현실적인 의미가 적어 수록하지 않았지만, 이번 실험에서도 모든 유전 알고리즘의 수행시간이 가장 긴 경우에도 0.2초 정도로 짧아 현실적인 의미가 없다고 보았기 때문에 수록하지 않았다.

알고리즘의 성능을 평가하기 위해 <표 4-2>의 값들을 관찰해보면 다음과 같은 현상들을 발견할 수 있다.

1) 기계수(m)가 2인 경우에는 작업수(n)에 관계없이 ***DECODE_LPT*** 와 ***DECODE_SPT***가 ***DECODE_Johnson***보다 항상 우수하였다. 평균값으로 비교하면 ***DECODE_LPT***는 4~5%, ***DECODE_SPT***는 3~4% 정도 ***DECODE_Johnson***보다 짧은 완료시간(makespan)을

보여주었다. 한편 기계수(m)가 3 이상이 되면 ***DECODE_Johnson***이 다른 두 알고리즘에 비해 우수하였다. 평균 완료시간(makespan)을 비교하였을 때 ***DECODE_LPT***보다 0~1% 작았고 ***DECODE_SPT***보다는 1~2% 작았다.

2) ***DECODE_LPT***와 ***DECODE_SPT***를 비교하면 기계수(m)와 작업수(n)에 관계없이 거의 모든 경우에서 ***DECODE_LPT***가 더 우수하였다. 비록 두 알고리즘을 직접 비교하지는 않았지만 간접 비교를 통해 평균 완료시간(makespan)을 비교하면 ***DECODE_LPT***가 0~2% 작은 값을 보여주었다. 예를 들어 (n, m)=(50, 2)일 때는 ***DECODE_LPT***와 ***DECODE_SPT***를 ***DECODE_Johnson***과 비교한 평균값이 모두 0.96이므로 ***DECODE_LPT***와 ***DECODE_SPT***를 비교한 값을 0으로 보았고, (n, m)=(40, 5)일 때는 그 값들이 1.00과 1.02이므로 두 알고리즘의 차이를 2%라고 본 것이다.

위 결과는 대체로 예상에 부합되는 것으로 볼 수 있으나 기계수(m)가 2대인 경우에는 상당히 특이한 결과를 보여주고 있다. 즉, 대부분의 경우 ***DECODE_Johnson***이 세 개의 유전알고리즘 중에서 가장 좋은 결과를 만들어 주었는데 기계수(m)가 2대인 경우에서만 ***DECODE_LPT***가 가장 우수한 결과를 생성한 것이다. 더구나 첫 번째 단계의 실험에서는 기계수(m)가 2대인 경우에도 ***DECODE_Johnson***이 우수하였다는 점을 고려한다면 확실히 <표 4-2>에서 m=2 인 경우의 결과는 예상하지 못한 결과였다.

실험결과를 종합하여 정리하면, 우선 계산시간 측면에서, 세 가지 유전알고리즘 모두 매우 짧은 시간 안에 해를 구하였으므로 현장에서의 적용에 아무런 문제가 없다고 판단된다. 해의 품질 측면에서는,

DECODE_Johnson을 적용한 유전알고리즘이 전체적으로 가장 골고루 우수한 해를 생산해주었다. 단, 작업수(n)가 많은 경우에는 기계수(m)가 2대일 때 ***DECODE_LPT***가 가장 우수한 해를 발견하였으나 그 차이가 그렇게 크지 않으므로 ***DECODE_Johnson***을 사용해도 큰 문제는 없을 것으로 생각된다. 그러나 가능하다면 두 알고리즘을 모두 적용하여 두 개의 해를 구한다음 둘 중 우수한 해를 적용한다면 좀 더 나은 결과를 기대할 수 있을 것이다.

5. 요약

본 장은 플라스틱 사출성형을 전문적으로 처리하는 소규모 제조기업의 사례 문제를 해결하기 위해 작업자 한 명이 기계 여러 대의 셋업을 전담하는 상황에 대한 이종병렬기계 시스템의 일정계획 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 제시하고, 이 모형의 유효성을 평가하기 위하여 최적화 소프트웨어를 사용해 작은 크기의 예제를 해결하였다. 실험결과 작은 크기의 문제는 최적화 모형을 사용해 최적해를 구할 수 있었으나, 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 유전알고리즘 기반의 휴리스틱 알고리즘을 개발하였다. 다양한 수치실험을 통해 본 연구에서 제시된 알고리즘들이 매우 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다.

다음 장에서는 셋업시간이 내부작업과 외부작업으로 나누어질 수 있다는 현실적인 가정을 추가하여 진행할 것이다.

V. 내부 및 외부 셋업을 수행하는 이종병렬기계의 일정계획

1. 개요

앞 장에서 셋업 작업자 제약이 있는 일정계획에 대하여 개략적으로 살펴보았다. 이 장에서는 셋업작업을 좀 더 자세히 구분하여, 살펴볼 예정이다. 도요타의 JIT 생산시스템의 관점에서 본다면 셋업 활동은 생산되는 제품에 부가가치를 전혀 발생시키지 않으면서 단지 비용만 발생시키는 비부가가치 활동이다. 이 비용에는 여러 가지 종류의 비용이 포함되지만 가장 우선적으로 직접 인건비를 들 수 있다. 셋업 활동은 주로 해당 기계에 대한 전문적인 지식이나 경험을 소지한 고도의 숙련공에 의해 이루어지므로 비싼 인건비가 투입되는 것이 보통이다. 그 외에도 고가의 설비를 셋업시간 동안 사용하지 못함으로 인한 기회비용, 셋업 후 일반적으로 생산해야 하는 테스트 제품에 대한 비용 등 많은 비용이 셋업으로 인해 발생한다. 일찍이 도요타 생산시스템에서는 이러한 셋업활동의 낭비적 특성에 주목하여 셋업활동에 소요되는 시간을 최소화하기 위해 많은 노력을 기울여왔다(Moden, 1983).

도요타 생산시스템에서 셋업시간의 단축을 위해 사용한 기법들이 여러 가지 있지만 그 중 가장 중요한 원칙은 셋업에 필요한 여러 활동들을 내부(internal)활동과 외부(external)활동으로 구분하는 것이다.

내부활동은 해당 기계가 반드시 물리적 혹은 기계적으로 정지하고 있는 상태에서만 수행할 수 있는 활동으로 그 대표적인 예로는 기계 작동에 필요한 여러 공구나 지그(jig) 장치들을 떼어내거나 부착하는 작업이 있다. 반면 외부활동은 그 기계가 정지하지 않고 현재 작동하고 있는 도중에 할 수 있는 일을 의미한다. 그 예로는 기계가공을 위해 사전에 실시하는 CAM(Computer Aided Manufacturing) 프로그램을 작성하거나, 셋업에 필요한 공구나 기계장치, 원자재 등을 미리 확인하고 준비하는 활동이 있다.

셋업시간 단축을 위해서는 우선 셋업에 필요한 모든 활동들을 파악하여 그 활동이 내부활동인지 외부활동인지를 분류하여야 한다. 그 다음에는 모든 외부활동들을 기계가 가동하고 있는 동안에 미리 종료되도록 조치한다. 그리고 한걸음 더 나아가 공구의 모듈화와 같은 기법을 통해 내부 셋업활동을 최대한 외부 셋업활동으로 전환시키는 것도 셋업시간 단축을 위한 중요한 원칙으로 알려져 있다.

이러한 노력을 통해 각 주문의 셋업시간을 내부활동과 외부활동으로 분리하였다면 일정계획 문제에서도 그 결과를 반영하여야 할 것이다. 그러나 일정계획을 다루는 연구 결과들 중에서 셋업시간이 내부 및 외부활동 시간으로 나누어진 상황을 다루는 연구결과는 찾아보기가 어렵다. Sabouni & Logendran(2013)이 이런 형태의 셋업시간을 고려한 연구를 수행하였으나 두 기계 흐름생산라인에서의 묶음 일정계획(group scheduling) 문제를 다루고 있어 본 연구의 대상인 이종병렬기계 시스템에 적용하기는 어려울 것으로 보여진다.

따라서 본 장에서는 각 주문의 셋업시간이 내부활동과 외부활동으로 분리되어 있는 경우의 이종병렬기계 시스템에서 한 사람(혹은 한 팀)의 셋업 작업자가 셋업을 전담할 때 이 셋업 담당자가 각 작업들

의 내부 및 외부 셋업활동을 가장 효율적으로 수행하는 일정을 구하고자 한다. 즉, 3장에서 수행하였던 R,S1|s_j|$C_{\max}$ 문제를 셋업시간이 내부활동시간과 외부활동시간으로 나누어져 있는 상황으로 확장하고자 하는 것이다.

본 장의 나머지 부분은 다음과 같이 구성된다. 다음 절에서는 대상문제를 정수계획법의 형태로 모형화하고 소규모 예제를 통해 이 모형의 유효성을 검정한다. 제3절에서는 대상문제를 해결하기 위한 유전 알고리즘 기반의 휴리스틱 해법을 제시하며, 제4절에서는 제안된 해법들의 성능을 평가하기 위한 수치실험을 소개한다. 마지막으로 제5절에서는 본장을 개략적으로 요약한다.

2. 최적화 모형

모형의 개발을 위해서는 우선 문제의 명확한 정의가 필요하다. 다음과 같은 가정들을 통해 문제를 보다 명확하게 정의하도록 한다. 첫째, 하나의 문제가 형성되고 해결되어져야 하는 시점(즉, 계획시점)은 셋업 작업자가 내부이든 외부이든 하나의 셋업작업을 마치는 순간이다. 만약 셋업 작업자가 유휴인 상태에서 새로운 주문이 도착하였다면 그 시점에서도 새로운 문제가 만들어지고 해결되어야 한다. 즉, 하나의 셋업작업을 마치는 시점 또는 작업자 유휴 상태에서 새로운 주문이 도착한 시점에 그 때까지 알려진 주문 정보와 기계 정보를 사용해 셋업 작업자가 다음으로 처리해야할 셋업 일정을 작성하는 것이 본 문제의 목표이다. 둘째, 모든 주문들은 외부셋업, 내부셋업, 기계작업이 순차적으로 수행되어야 한다. 이미 외부셋업 작업이 완료된 주

문이 존재하더라도 모형에서는 그 주문의 외부셋업 작업시간을 0으로 놓고 내부셋업 작업 이전에 수행하는 것으로 본다. 셋째, 각 작업의 내부셋업이 종료된 순간에는 기계 작업이 바로 시작되나 외부셋업과 내부셋업 사이에는 시간 간격이 발생할 수 있다. 마지막으로, 각 주문의 셋업시간은 그 주문이 어느 기계에서 처리되느냐에 따라 변하지만 처리순서와는 무관하다고 가정한다. 이러한 가정 하에서 모형을 개발하였으며, 모형에서 사용되는 기호에 대해 먼저 설명한 다음 개발된 모형을 소개한다.

<파라메터>

i, j : 주문을 표시하는 인덱스 (i, j = 1, 2, ..., n)

k : 기계를 표시하는 인덱스 (k = 1, 2, ..., m)

a_i : 계획시점 이전에 주문 i가 도착하였으면 0, 그렇지 않으면 도착예정시간

r_k : 계획시점에 기계 k가 작업 중이면 그 작업의 완료시간, 그렇지 않으면 0

se_{ik} : 주문 i가 기계 k에 할당되었을 경우의 외부셋업시간, 계획시점에 이미 완료되었다면 0

si_{ik} : 주문 i가 기계 k에 할당되었을 경우의 내부셋업시간

p_{ik} : 주문 i가 기계 k에 할당되었을 경우의 기계 처리시간 (셋업시간 제외)

M : 충분히 큰 수

<결정변수>

x_{1i} = 주문 i의 외부셋업 착수시간

x_{2i} = 주문 i의 외부셋업 종료시간

x_{3i} = 주문 i의 내부셋업 착수시간

x_{4i} = 주문 i의 내부셋업 종료시간으로 기계작업 착수시간과 동일

f_i = 주문 i의 작업 완료시간

$f_{\max}$ = 모든 주문의 완료시간 (makespan) = max{f_i}

u_i = 주문 i의 외부셋업 작업이 모든 작업 중 가장 빨리 수행되면 1, 그렇지 않으면 0

$u_{2i-1,2j-1}$ = 주문 i의 외부셋업 작업에 이어 주문 j의 외부셋업 작업이 처리되면 1, 그렇지 않으면 0

$u_{2i-1,2j}$ = 주문 i의 외부셋업 작업에 이어 주문 j의 내부셋업 작업이 처리되면 1, 그렇지 않으면 0

$u_{2i,2j-1}$ = 주문 i의 내부셋업 작업에 이어 주문 j의 외부셋업 작업이 처리되면 1, 그렇지 않으면 0

$u_{2i,2j}$ = 주문 i의 내부셋업 작업에 이어 주문 j의 내부셋업 작업이 처리되면 1, 그렇지 않으면 0

y_{ik} = 주문 i가 기계 k에 할당되면 1, 그렇지 않으면 0

z_{ik} = 기계 k에서 가장 먼저 처리되는 주문이 i이면 1, 그렇지 않으면 0

z_{ijk} = 기계 k에서 주문 i에 이어 주문 j가 처리되면 1, 그렇지 않으면 0

<모형>

Minimize $f_{\max}$ (5-1)

subject to

$x_{1i} \geq a_i$ for all i (5-2)

$x_{3i} + M(1 - z_{ik}) \geq r_k$ for all $i,\ k$ (5-3)

$$x_{1i} + \sum_{k=1}^{m} se_{ik} y_{ik} = x_{2i} \text{ for all } i \quad (5\text{-}4)$$

$x_{2i} \leq x_{3i}$ for all i (5-5)

$$x_{3i} + \sum_{k=1}^{m} si_{ik} y_{ik} = x_{4i} \text{ for all } i \quad (5\text{-}6)$$

$$x_{4i} + \sum_{k=1}^{m} p_{ik} y_{ik} = f_i \text{ for all } i \quad (5\text{-}7)$$

$$f_i \leq x_{3j} + M(1 - \sum_{k=1}^{m} z_{ijk}) \text{ for all } i \neq j \quad (5\text{-}8)$$

$f_i \leq f_{\max}$ for all i (5-9)

$x_{4i} \leq x_{3j} + M(1 - u_{2i,2j})$ for all $i \neq j$ (5-10)

$x_{2i} \leq x_{3j} + M(1 - u_{2i-1,2j})$ for all i, j (5-11)

$x_{4i} \leq x_{1j} + M(1 - u_{2i,2j-1})$ for all $i \neq j$ (5-12)

$x_{2i} \leq x_{1j} + M(1 - u_{2i-1,2j-1})$ for all $i \neq j$ (5-13)

$$\sum_{k=1}^{m} y_{ik} = 1 \text{ for all } i \quad (5\text{-}14)$$

$$\sum_{i=1}^{n} z_{ik} = 1 \text{ for all } k \quad (5\text{-}15)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} z_{ijk} \leq y_{ik} \text{ for all } i,\ k \tag{5-16}$$

$$z_{ik} + \sum_{\substack{j=1 \\ j \neq i}}^{n} z_{jik} = y_{ik} \text{ for all } i,\ k \tag{5-17}$$

$$\sum_{i=1}^{n} u_i = 1 \tag{5-18}$$

$$u_{2i,2i-1} = 0 \text{ for all } i \tag{5-19}$$

$$\sum_{\substack{j=1 \\ j \neq 2i-1}}^{2n} u_{2i-1,j} = 1 \text{ for all } i \tag{5-20}$$

$$\sum_{\substack{j=1 \\ j \neq 2i}}^{2n} u_{2i,j} \leq 1 \text{ for all } i \tag{5-21}$$

$$u_i + \sum_{\substack{j=1 \\ j \neq 2i-1}}^{2n} u_{j,2i-1} = 1 \text{ for all } i \tag{5-22}$$

$$\sum_{\substack{j=1 \\ j \neq 2i}}^{2n} u_{j,2i} = 1 \text{ for all } i \tag{5-23}$$

u_i : 0/1 integer for all i

u_{ij} : 0/1 integer for all $i,\ j$

$y_{ik},\ z_{ik}$: 0/1 integer for all $i,\ k$

z_{ijk} : 0/1 integer for all $i,\ j,\ k$

식 (5-1)은 본 문제가 총 완료시간을 최소화하는 문제라는 것을 보여준다. 제약조건 (5-2)는 각 주문의 외부셋업 작업이 그 주문의 도착(확정) 시점(a_i) 이후에 시작될 수 있음을 나타낸다. 식 (5-3)은 각 주문의 내부셋업 작업이 그 주문을 처리하는 기계가 가용해지는 시점(r_k) 이후에 시작될 수 있다는 조건을 표현하고 있다. 식 (5-4)는 외부셋업 작업의 종료시간을 계산해준다. 식 (5-5)에 의해 내부셋업은 외부셋업 종료 이후에 시작할 수 있다. 식 (5-6)은 내부셋업 작업의 종료시간을 계산해주며, 식 (5-7)은 각 작업의 완료시간을 계산해준다. 식 (5-8)에 의하면 한 기계에서 선후관계를 갖는 주문들은 선행주문이 완료(f_i)된 이후 후속주문의 내부셋업이 착수(x_{3j})될 수 있다. 식 (5-9)는 각 주문들의 완료시간과 전체 완료시간의 관계를 표현한다. 식 (5-10)에 의하면 주문 i의 내부셋업 작업에 이어 주문 j의 내부셋업 작업이 처리될 경우 주문 j의 내부셋업 착수시간은 주문 i의 내부셋업 종료시간 이후이다. 식 (5-11)~(5-13)도 셋업의 종류는 다르지만 식 (5-10)과 같은 구조이다. 식 (5-14)는 각 주문이 한 대의 기계에만 할당될 수 있도록 해준다. 식 (5-15)는 각 기계에 최초로 배정되는 주문은 한 개라는 것을 의미한다. 식 (5-16)에 의하면 주문 i가 기계 k에 배정되지 않았다면($y_{ik}=0$) 그 주문은 그 기계에서 다른 어떤 주문의 선행주문도 될 수 없다($z_{ijk}=0$)는 것을 보여준다. 반대로 해석하면 주문 i가 기계 k에 배정되었을 경우($y_{ik}=1$) 그 주문은 그 기계에서 다른 어떤 주문 앞에 처리되는 주문이라는 것($z_{ijk}=1$)을 보여준다. 그런데 이 식이 부등식인 것은 그 주문이 그 기계에서 최종으로 처리되는 경우 때문이다. 비슷하게 식 (5-17)에서는 주문 i가 기

계 k에 배정되었을 경우($y_{ik}=1$) 그 주문은 그 기계에서 처리되는 최초의 주문($z_{ik}=1$)이거나 다른 어떤 주문의 후속주문이라는 것($z_{jik}=1$)을 보여준다. 식 (5-18)은 최초로 외부셋업을 수행하는 주문은 하나일 수밖에 없음을 보여준다. 식 (5-19)는 내부셋업이 외부셋업보다 선행할 수 없음을 의미한다. 식 (5-20)에 의하면 한 주문의 외부셋업이 종료된 이후에는 그 주문의 내부셋업이나 다른 주문의 외부 혹은 내부셋업 중 하나가 반드시 이루어져야 한다. 반면 식 (5-21)에 의하면 한 주문의 내부셋업이 종료된 이후에는 다른 주문의 외부 혹은 내부셋업 중 하나가 수행될 수 있다. 이 식이 부등식인 것은 그 주문이 마지막 주문인 경우에는 후속 작업이 없을 수도 있기 때문이다. 식 (5-22)는 주문 i의 외부셋업이 가장 먼저 이루어지거나 혹은 다른 주문의 내부 혹은 외부셋업 이후에 이루어진다는 것을 보여준다. 마지막으로 식 (5-23)은 어떤 주문의 내부셋업 이전에는 반드시 하나의 다른 셋업작업이 있어야 함을 나타낸다.

이 모형은 문제의 크기(주문 수 및 기계 수)가 작은 경우 CPLEX나 LINGO 같은 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 예를 들기 위해 다음과 같이 주문 수(n)가 5이고 기계 수(m)가 3인 문제를 생각한다.

$\boldsymbol{a}$ = (0, 0, 0, 5, 10), $\boldsymbol{r}$ = (0, 3, 10),

$$\boldsymbol{se} = \begin{bmatrix} 2 & 4 & 5 \\ 3 & 5 & 2 \\ 3 & 3 & 6 \\ 7 & 6 & 4 \\ 2 & 5 & 6 \end{bmatrix}, \quad \boldsymbol{si} = \begin{bmatrix} 3 & 2 & 4 \\ 6 & 5 & 4 \\ 6 & 7 & 4 \\ 3 & 3 & 4 \\ 5 & 3 & 4 \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} 35 & 35 & 40 \\ 30 & 26 & 33 \\ 37 & 32 & 30 \\ 32 & 38 & 35 \\ 35 & 30 & 28 \end{bmatrix}.$$

속도 2.60GHz의 CPU가 장착된 PC에서 [그림 5-1]과 같은 LINGO

프로그램을 이용해 해를 구한 결과 29초의 계산시간에 최적해를 구할 수 있었고 그 결과는 <표 5-1>과 같이 정리할 수 있었다.

Lingo Model - op1_makespan_separate_5_2

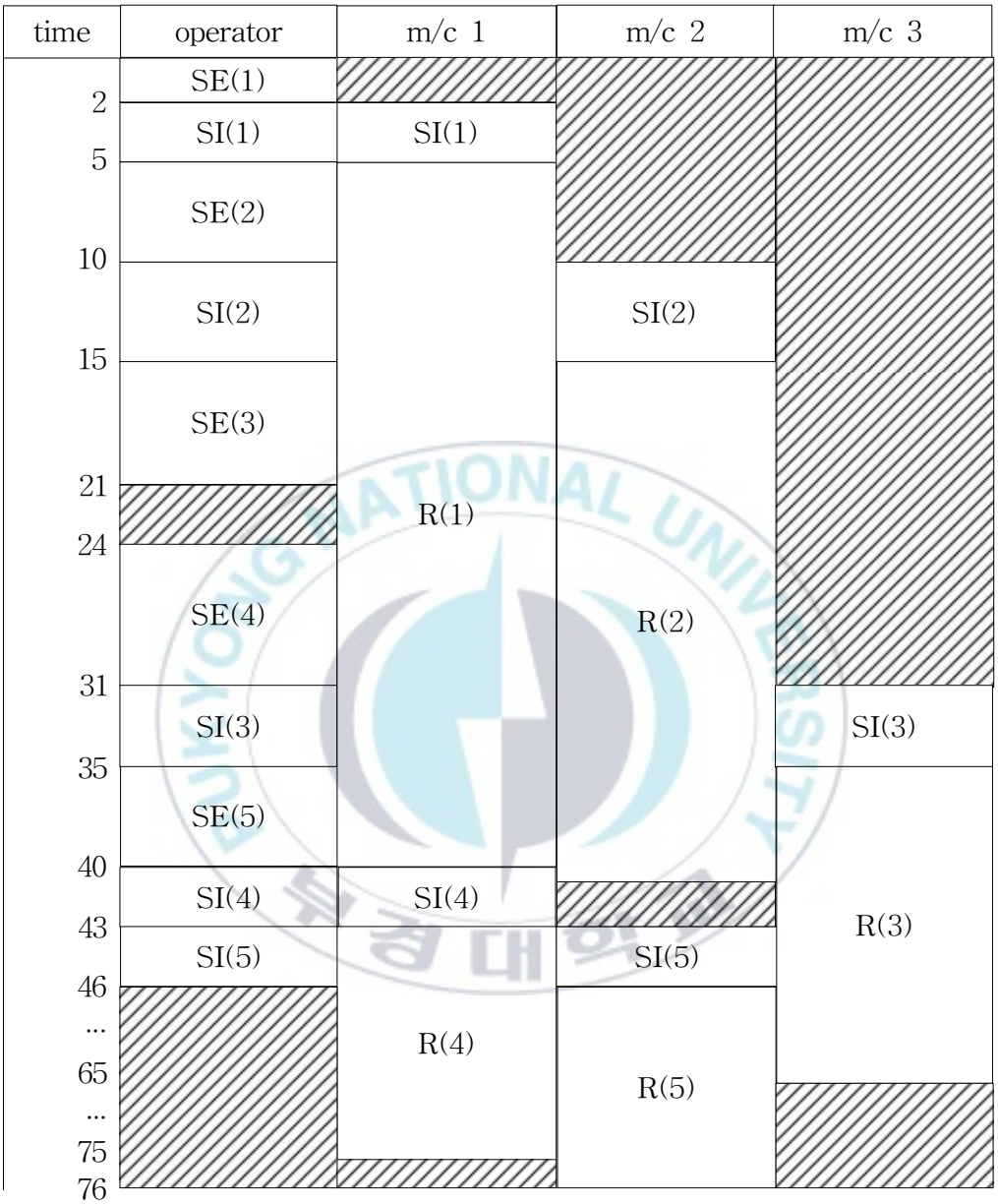
```
MODEL:
 SETS:
      Job: a, x1, x2, x3, x4, f, u0;
      Job2;
      Mch: r;
      JM(Job, Mch): p, se, si, z0, y;
      JJ(Job, Job);
      J2J2(Job2, Job2): u;
      JJM(Job, Job, Mch): z;
 ENDSETS

 DATA:
   n = 5; n2 = 10; m = 3; BM = 9999;
   Job = 1..n; Job2 = 1..n2;  Mch = 1..m;
   a = 0 0 0 5 10 ;     r = 0 3 10 ;
   p = 35 35 40   30 26 33   37 32 30   32 38 35   35 30 28  ;
   se = 2 4 5   3 5 2   3 3 6   7 6 4   2 5 6 ;
   si = 3 2 4   6 5 4   6 7 4   3 3 4   5 3 4 ;
 ENDDATA

 MIN = fmax;

 @FOR(Job(i): x1(i) > a(i));
 @FOR(JM(i, k): x3(i) + BM * (1 - z0(i, k)) > r(k));
 @FOR(Job(i): x1(i) + @SUM(Mch(k): se(i, k) * y(i, k)) = x2(i));
 @FOR(Job(i): x2(i) < x3(i));
 @FOR(Job(i): x3(i) + @SUM(Mch(k): si(i, k) * y(i, k)) = x4(i));
 @FOR(Job(i): x4(i) + @SUM(Mch(k): p(i, k) * y(i, k)) = f(i));

 @FOR(JJ(i, j) | i #ne# j: f(i) < x3(j) + BM * (1 - @SUM(Mch(k): z(i, j, k))));
 @FOR(JJ(i, j) | i #ne# j: x3(j) + BM * (1 - u(2*i, 2*j)) > x4(i));
 @FOR(JJ(i, j): x3(j) + BM * (1 - u(2*i-1, 2*j)) > x2(i));
 @FOR(JJ(i, j) | i #ne# j: x1(j) + BM * (1 - u(2*i, 2*j-1)) > x4(i));
 @FOR(JJ(i, j) | i #ne# j: x1(j) + BM * (1 - u(2*i-1, 2*j-1)) > x2(i));
 @FOR(Job(i): f(i) < fmax);

 @FOR(Job(i): @SUM(Mch(k): y(i, k)) = 1);
 @FOR(Mch(k): @SUM(Job(j): z0(j, k)) = 1);
 @FOR(JM(i, k): @SUM(Job(j) | j #ne# i: z(i, j, k)) < y(i, k));
 @FOR(JM(i, k): z0(i, k) + @SUM(Job(j) | j #ne# i: z(j, i, k)) = y(i, k));

 @SUM(Job(i): u0(i)) = 1;
 @FOR(Job(i): u(2*i, 2*i-1) = 0);
 @FOR(Job(i): @SUM(Job2(j) | j #ne# 2*i-1: u(2*i-1, j)) = 1);
 @FOR(Job(i): @SUM(Job2(j) | j #ne# 2*i: u(2*i, j)) < 1);
 @FOR(Job(i): u0(i) + @SUM(Job2(j) | j #ne# 2*i-1: u(j, 2*i-1)) = 1);
 @FOR(Job(i): @SUM(Job2(j) | j #ne# 2*i: u(j, 2*i)) = 1);

 @FOR(Job(i): @BIN(u0(i)));
 @FOR(JM(i, k): @BIN(y(i, k)));
 @FOR(J2J2(i, j): @BIN(u(i, j)));
 @FOR(JM(i, k): @BIN(z0(i, k)));
 @FOR(JJM(i, j, k): @BIN(z(i, j, k)));

END
```

[그림 5-1] 예제의 LINGO 프로그램(내외셋업분리)

<표 5-1> Result of The Example Problem(내외셋업분리)

Job (i)	machine assigned	x_{1i}	x_{2i}	x_{3i}	x_{4i}	f_i
1	1	0	2	2	5	40
2	2	5	10	10	15	41
3	3	15	21	31	35	65
4	1	24	31	40	43	75
5	2	35	40	43	46	76

이 <표 5-1>은 각 주문별로 그 주문이 할당된 기계 번호와 외부셋업 착수 및 종료, 내부셋업 착수 및 종료, 그리고 기계작업 종료시간을 자세히 보여준다. 만들어진 최적 일정계획에 의하면 기계 1에는 주문 1과 4가, 기계 2에는 주문 2와 5가, 그리고 기계 3에는 주문 3이 할당되었고, 5개 주문의 예정 완료시간은 각각 40, 41, 65, 75, 76 이며, 따라서 전체 완료시간은 이 값들 중 최대인 76이다. 숫자로 채워진 이 표를 보다 쉽게 이해하려면 [그림 5-2]와 같은 Man-Machine Chart 형태로 결과를 표현하는 것이 유용할 수도 있다. 그림에서 SE(i)는 주문 i에 대한 외부 셋업작업을, SI(i)는 주문 i에 대한 내부 셋업작업을, 그리고 R(i)는 주문 i에 대한 기계 작업시간을 의미한다. 이 그림을 보면 작업자와 각 기계의 작업일정을 시간대 별로 쉽게 파악할 수 있다.



[그림 5-2] Result of the example problem(내외셋업분리)

3. 혼합 유전알고리즘 방법론

본 연구의 대상 문제는 크기(주문수 및 기계수)가 작은 경우 최적화 소프트웨어를 사용하여 쉽게 풀 수 있다. 그러나 잘 알려진 바처럼 이 문제는 NP-hard 에 속하므로 문제의 크기가 커질 경우 현실적인 시간 (하나의 셋업이 끝나거나 새로운 주문이 도착했을 때 작업자의 다음 작업을 결정해야하는 시간) 안에 최적해를 구하는 것은 불가능한 일이다. 실제로 앞 절에서 제시한 예제(주문수가 5이고 기계수가 3)의 최적해는 LINGO를 사용해 4GB RAM과 2.60GHz G620 CPU가 장착된 PC에서 29초만에 구할 수 있었으나 주문의 수가 6인 문제의 해결에는 17분이 소요되었고 주문의 수가 7이 되면 1시간 안에 해가 구해지지 않아 프로그램을 중단할 수밖에 없었다. 또한 주문과 기계의 수가 같은 경우에도 데이터에 따라 계산시간은 10배 이상 차이가 나는 경우가 많았다. 따라서 문제의 크기와 데이터의 변화에도 현실적인 시간 안에 안정적으로 해를 찾을 수 있는 방법을 개발하는 것이 필요하다.

본 연구에서는 현실적인 크기의 문제를 해결하기 위해 혼합 유전알고리즘(Hybrid Genetic Algorithm)을 제안한다. 대상문제의 해결과정이 1)작업자가 각 주문별 외부 및 내부 셋업작업들을 처리하는 순서를 결정하고 2)각 작업을 기계에 할당하는 두 단계로 이루어져 있다고 보고, 첫 단계인 순서결정 문제는 유전 알고리즘의 염색체를 통해 해결하고, 두 번째 단계인 할당 문제는 각 처리순서(즉, 염색체)에 대해 간단한 휴리스틱 알고리즘을 적용하여 해결하도록 한다. 다시 말해 전체적인 절차는 일반적인 유전 알고리즘과 대동소이한데 염색체-해 전환 (decoding) 절차에 간단한 휴리스틱의 개념을 도입한 것

이다. 그 절차를 설명하면 다음과 같다.

우선 염색체 표현은 각 주문번호를 두 번씩 포함하는 정수열을 사용한다. 예를 들어 4개의 주문을 처리해야 한다면 (2, 1, 1, 3, 2, 4, 3, 4)와 같은 길이 8의 정수열이 하나의 염색체가 될 수 있다. 내부셋업이 외부셋업보다 빨리 수행될 수는 없으므로 이 염색체에 의한 작업자의 작업은 주문 2의 외부셋업, 주문 1의 외부 및 내부셋업, 주문 3의 외부셋업, 주문2의 내부셋업, ..., 그리고 마지막으로 주문 4의 내부셋업과 같은 순서로 이루어지게 된다.

이와 같이 하나의 염색체에 의해 셋업 작업자의 작업순서가 정해지면 다음으로는 각 주문을 기계에 할당하여 완성된 하나의 일정을 생성하여야 한다. 또한 일정을 생성하면서 그 일정에 의한 주문처리 총 완료시간($f_{\max}$)도 같이 계산하여야 하는데 이 과정을 위해 다음과 같은 간단한 휴리스틱 알고리즘을 제안한다.

***DECODE* Algorithm**

Step 0. 셋업 작업자가 가용해지는 시점을 $avail$ 로, 기계 k 가 가용해지는 시점을 ms_k 로 놓는다. 그러면 $avail = 0$, $ms_k = r_k$ 이다. 또한 $i = 0$, $f_{\max} = 0$ 으로 놓는다.

Step 1. $i = i + 1$ 로 놓는다. $i > 2n$ 이면 알고리즘을 종료한다.

Step 2. j = 염색체 내부의 i 번째 유전자 값으로 놓는다. 주문 j 의 외부 및 내부셋업이 모두 처리되었으면 <Step 1>로, 두 셋업 모두 처리되지 않았다면 <Step 3>으로, 외부셋업은 처리되고 내부셋업이 처리되지 않았다면 <Step 5>로 간다.

Step 3. $avail$ 시점 현재 모든 기계에 주문들이 할당되어 있고 각 기

계에 마지막으로 할당된 주문들이 모두 외부셋업 작업만 수행된 상태라면, 각 기계 k 에 대해 max{$avail$, ms_k}+ si_{lk} 를 계산하여 최소가 되는 기계를 찾아 그 기계에 이미 할당되어 외부셋업이 수행된 주문 l 의 내부셋업을 먼저 처리한다. x_{3l} = max{$avail$, ms_k}, $avail$ = x_{4l} = x_{3l}+ si_{lk}, ms_k = f_l = x_{4l}+ p_{lk}, $f_{\max}$ = max{$f_{\max}$, ms_k} 로 놓는다.

Step 4. 주문이 할당되어 외부셋업만 수행된 상태가 아닌 (즉, 주문이 하나도 할당되지 않았거나, 할당되었다면 할당된 주문의 외부 및 내부셋업이 모두 수행된 상태의) 기계 k 에 대해 max {$avail$, ms_k}+ se_{jk}+si_{jk}를 계산하여 최대가 되는 기계를 찾아 주문 j 를 그 기계에 할당하고 외부셋업을 처리한다. x_{1j} = max{$avail$, a_j}, $avail$ = x_{2j} = x_{1j}+ se_{jk} 로 놓고 <Step 1>로 간다.

Step 5. 주문 j 가 할당된 기계를 k 라고 하고 내부셋업을 처리한다. x_{3j} = max{$avail$, ms_k}, $avail$ = x_{4j} = x_{3j}+ si_{jk}, ms_k = f_j = x_{4j}+ p_{jk}, $f_{\max}$ = max{$f_{\max}$, ms_k} 로 놓고 <Step 1>로 간다.

앞서 2절에서 문제를 해결하는 시점을 하나의 셋업작업을 마치는 시점 또는 작업자 유휴 상태에서 새로운 주문이 도착한 순간이라고 가정하였다. 위 알고리즘에서 사용된 변수 $avail$ 이 바로 이 시점을 의미한다. 알고리즘은 재계산된 각 $avail$ 시점에 다음 처리대상 셋업작업을 각 기계에 할당해나가는 과정을 설계한 것이다. <Step 3>에서

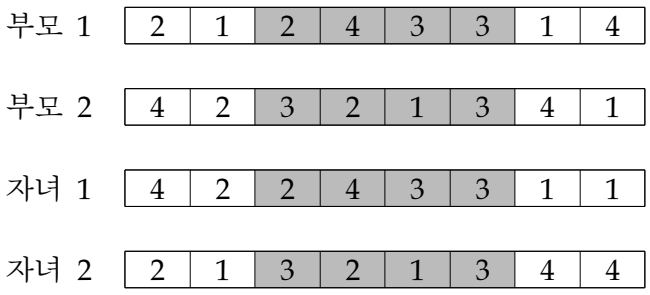
는 선택된 주문 j 의 외부셋업을 가능하면 빨리 수행하기 위해 max {$avail$, ms_k}+ si_{lk} 값이 최소가 되는 기계를 찾아 그 기계에 이미 할당되어 외부셋업이 수행된 주문 l 의 내부셋업을 먼저 처리하였고, <Step 4>에서는 기본적인 병렬기계문제에서 훌륭한 결과를 제공해주는 LPT(Longest Processing Time) 규칙(Pinedo, 2002)을 응용하기 위해 max{$avail$, ms_k}+ se_{jk}+si_{jk} 값이 최대가 되는 기계를 찾아 선택된 주문을 할당하였다.

위에서 제시한 염색체-해 전환 절차 이외의 과정은 일반적인 유전알고리즘의 전개와 유사하다. 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.

$$F_i = Z_{\max} - Z_i \tag{5-24}$$

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 대상으로는 교배(crossover) 및 돌연변이(mutation)와 같은 유전 연산(genetic operation)을 적용하여 다음 세대의 염색체를 만들어 낸다. 우선 돌연변이 연산은 임의로 선택된 하나의 염색체에서 임의로 선택된 두 유전자를 서로 맞교환하는 방식을 적용하였다.

한편 교배 연산은 [그림 5-3]과 같이 순서결정 문제에서 많이 사용되는 방법(Gen & Cheng, 1997)인 순서교배(order crossover)와 유사한 방식을 사용하였다. 이 연산을 위해서는 두 개의 부모 염색체를 임의로 선택한 다음 염색체 중간에 두 개의 절점을 임의로 선정하여야 한다. 아래 그림은 주문의 수가 4개(즉, 염색체의 길이는 8)일 때, 두 개의 부모 염색체가 선택되어 있고 두 절점은 유전자 (2, 3) 사이와 유전자 (6, 7) 사이인 상황을 보여주고 있다. 먼저 자녀 1 염색체를 생성하기 위해 부모 1 염색체의 두 절점 사이 유전자인 (2, 4, 3, 3)을 복사한다. 그리고 부모 2 염색체의 나머지 부분을 각 주문번호가 2개 이하가 되는 조건 하에서 복사한다. 부모 2의 첫 번째와 두 번째 유전자인 4와 2는 조건을 만족하므로 그대로 복사한다. 하지만 부모 2의 7번째 유전자인 4는 이미 두 번 등장하였기 때문에 뛰어 넘고 부모 2의 8번째 유전자인 1을 자녀 1의 7번째 자리에 복사한다. 부모 유전자 번호가 염색체 길이를 초과하게 되면 주문번호 순으로 두 번 미만 등장한 주문번호를 차례로 기입하는데 이 예제에서는 주문번호 1이 한번만 나타났으므로 기입하면 자녀 1 염색체가 완성된다. 자녀 2 염색체도 비슷한 방법으로 생성할 수 있다.



[그림 5-3] An Example Crossover

위와 같은 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_c 보다 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 대해 다음 과정을 수행한다. (0, 1) 구간의 난수를 하나 생성한다. 이 난수가 미리 정한 돌연변이 확률 P_m 보다 작으면 그 염색체에서 임의의 두 유전자를 선택하여 서로 맞교환한다. 여기서도 P_m의 값은 실험을 통해 선정한다.

알고리즘의 유효성을 검증하기 위해 2절에서 사용하였던 예제를 본 알고리즘으로 해결하였다. 앞서 최적해를 구하였던 컴퓨터에서 테스트한 결과 0.1초 미만의 시간에 해를 구할 수 있었다.

그 결과를 <표 5-1>과 같은 형태로 정리하면 <표 5-2>와 같다. 앞서 최적 완료시간은 76 이었지만 이 표를 통해 유전 알고리즘의 결과는 82 라는 사실을 알 수 있다. 총 완료시간이 최적해에 비해 약 7.9% 길어진 다소 미흡한 해가 구해졌지만 본 연구에서 제시한 알고리즘으로 해가 구해진다는 것은 확인할 수 있었다.

<표 5-2> Result of The Example Problem(내외셋업분리)

Job (i)	machine assigned	x_{1i}	x_{2i}	x_{3i}	x_{4i}	f_i
1	1	0	2	5	8	43
2	1	19	22	43	49	79
3	2	2	5	8	15	47
4	3	15	19	22	26	61
5	2	26	31	49	52	82

4. 수치 실험

앞서 제안한 혼합 유전알고리즘의 성능을 평가하기 위해 본 절에서는 임의로 생성한 예제들을 사용해 알고리즘의 해를 최적해와 비교한다. 문제의 규모가 커지면 LINGO 시스템을 사용해 최적해를 구할 수가 없으므로 문제의 규모를 결정하는 주문수(n)와 기계수(m)를 적절한 수준으로 통제하도록 한다. 주문수가 6 이상이 되면 최적해를 구하는 시간이 급격하게 증가하여 실험이 어려워지므로, 주문수는 3,

4, 5 중에서 하나로 하고 기계수는 2와 3 중 하나로 하여 총 6개 (n, m) 조합에 대해 실험을 수행한다.

주문수와 기계수 이외의 모수들은 a_i, r_k, p_{ik}, s_{ik}, s_{ijk} (i, j = 1, 2, ..., n, k = 1, 2, ..., m) 등이 있으며 이들은 다음과 같은 절차를 통해 생성하고자 한다. 우선 a_i 값들을 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 b 사이에서 임의로 생성된 정수라는 의미이다. r_k 값은 기계 중 하나를 임의로 골라 0 으로 놓고 나머지는 $U[0, 20]$로 가정하였다. 기계 처리시간(p_{ik})은 $U[25, 40]$으로, 준비시간(se_{ik} 및 si_{ik})은 모두 $U[3, 8]$로 가정하였다.

<표 5-3> Results of Small Problems(내외셋업분리)

		m=2		m=3	
		makespan ratio	LINGO comp time	makespan ratio	LINGO comp time
n=3	mean	1.01	< 1	1.03	< 1
	min	1.00	< 1	1.00	< 1
	max	1.06	< 1	1.15	< 1
n=4	mean	1.01	3.8	1.03	1.9
	min	1.00	1	1.00	1
	max	1.08	7	1.15	8
n=5	mean	1.01	15.9	1.03	60.8
	min	1.00	8	1.00	6
	max	1.05	31	1.13	381

실험결과는 <표 5-3>에 정리되어 있다. 앞서 설명한 바와 같이 이 단계에서는 6개의 (n, m) 조합 대해 실험을 수행하였다. 우선 각 조합에 대해 20개의 문제를 임의로 생성한 다음, 각 문제에 대한 해를 LINGO 시스템과 본 연구에서 제안한 유전알고리즘을 통해 각각 구하고, 두 해의 비율을 계산하여 평균, 최소, 최대를 표시하였다. 또한 각 문제에 대한 LINGO 시스템의 계산시간을 측정하여 그 평균, 최소, 최대를 표시하였다. 유전알고리즘에 대한 계산시간은 별도로 표시하지 않았는데 그 이유는 모든 문제에 대해 유전알고리즘 계산시간이 0.1초 미만이었기 때문이다. 표를 이해하기 위해 표의 좌하부 셀에 기입된 값 1.01, 1.00, 1.05, 15.9, 8, 31 등을 구하는 절차를 설명하면 아래와 같다.

1) 계획대상 주문수(n)가 5이고 기계수(m)가 2인 상황에서 생성된 20개의 서로 다른 문제에 대해 LINGO 시스템으로 최적해를 구한 결과 그 최종완료시간은 103, 99, 104, ..., 108 등이었다.
2) 본 연구의 유전알고리즘으로 같은 문제를 푼 결과는 103, 99, 108, ..., 114 등이었다.
3) 유전 알고리즘의 결과를 최적해로 나누면 1.00, 1.00, 1.04, ..., 1.06 등이 된다.
4) 이 20개 값들의 평균 1.01, 최소 1.00, 최대 1.05 등을 표에 수록하였다.
5) 또한 이 20개 문제에 대한 LINGO 계산시간은 17초, 14초, 13초, ..., 13초 등이었고 이 값들의 평균은 15.9초, 최소는 8초, 최대는 31초였다.

<표 5-3>의 값들을 관찰해보면 본 연구에서 제안한 알고리즘들의 성능에 대한 개략적인 평가를 할 수 있으며 그 결과는 다음과 같이 정리할 수 있다.

1) 유전 알고리즘은 6개의 모든 (n, m) 조합에서 최적해를 찾았으며, 최악의 경우 최적해에 비해 15% 떨어지는 해를 제공하였고, 평균적으로는 최적해에 비해 1%~3% 정도 떨어지는 해를 찾아주었다. 문제 크기에 따라 이 성능에는 차이가 있겠지만 이 규모만 놓고 볼 때는 매우 우수한 결과라고 판단된다.
2) 앞서 2절의 최적화 모형에 기계보다는 주문에 관련된 변수와 제약조건이 많다는 점에서 예측할 수 있듯이 최적해 도출시간은 기계수보다는 주문수에 더 민감하게 반응하였다.
3) 주문수가 같다면 기계수가 많을 때보다는 적을 경우에 유전 알고리즘의 성과가 우수하였다. 기계수가 적으면 경우의 수가 줄어들기 때문에 우연탐색(random search)에 기반한 유전알고리즘의 특성에 부합되는 결과라고 판단된다.

앞선 실험결과는 본 연구에서 제안한 알고리즘의 우수성을 보여주고 있으나 실험이 매우 작은 주문수 및 기계수를 대상으로 이루어져 일반화하는데 문제가 있다. 그러나 주문수 및 기계수가 많아지면 최적해를 구할 수가 없으므로 알고리즘의 결과를 비교할 대상이 없으므로 실험이 불가능한 것이 문제이다. 본 연구에서는 <표 5-3>과 같은 절대평가는 불가능하지만 규모가 큰 문제에 대한 실험도 수행하였다. 이 실험은 주문의 수를 10, 20, 30, 40 중에서 하나로 하고 기계의 수는 2, 3, 4 중 하나로

하는 총 12개 (n, m) 조합의 문제에 대한 실험이다. 별도로 실험결과 데이터를 제시하지는 않았다.

실험결과는 알고리즘의 계산시간 및 일관성 측면에서 정리할 수 있다. 우선 계산시간을 보면 기계수는 별 영향이 없었고 주문수에 따라 변하였는데, 주문수가 20 이하일 때는 모든 문제에서 1초 미만에 해를 구할 수 있었고 주문수가 30인 경우에는 평균 2.0초(최대 3초), 40인 경우에는 평균 3.5초(최대 4초)에 해를 구할 수 있었다. 따라서 문제의 규모가 현실적인 범위 안에서 매우 커지더라도 현실적인 시간에 해를 구하는데 아무런 문제가 없었다.

유전알고리즘은 우연탐색의 특성을 가지므로 알고리즘을 수행할 때마다 해가 달라질 수 있다. 따라서 알고리즘의 일관성 문제가 대두되는데 이 해의 변동성은 문제의 규모가 커지면 같이 커지게 된다. 실험에서는 같은 문제에 대해 알고리즘을 10회 반복해서 수행하였는데 그 결과 주문수가 커지면 해의 변동성도 커지는 현상을 발견하였다. 주문수가 10일 때 같은 문제를 반복해서 풀면 문제에 따라 4%에서 15% 사이에서 해가 변동하였고, 주문수가 40이 되면 그 폭은 8%에서 31% 수준까지 확대되었다. 따라서 알고리즘을 한번 수행하는 것 보다는 충분한 횟수만큼 반복수행하여 우수한 해를 고르는 것이 유리하다. 물론 여기에는 수행시간에 대한 제약을 고려되어야 하는데 위에서 보았듯이 주문수가 40인 경우에도 평균 1회 수행시간이 3.5초이므로 10회 정도 수행하여도 큰 문제가 없으리라 생각된다.

5. 요약

본 장에서는 각 주문의 셋업시간이 내부활동과 외부활동으로 분리되어 있는 경우의 이종병렬기계 시스템에서 한 사람(혹은 한 팀)의 셋업 작업자가 셋업을 전담할 때 이 셋업 담당자가 각 주문의 내부 및 외부 셋업활동을 가장 효율적으로 수행하는 일정을 구하기 위한 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 제시하고, 이 모형의 유효성을 평가하기 위하여 최적화 소프트웨어를 사용해 작은 크기의 예제를 해결하였다. 실험결과 작은 크기의 문제는 최적화 모형을 사용해 최적해를 구할 수 있었으나, 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 유전알고리즘 기반의 휴리스틱 알고리즘을 개발하였다. 수치실험을 통해 본 연구에서 제시된 알고리즘이 매우 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다.

Ⅵ. 결 론

1. 연구의의 및 요약

본 연구는 중소규모 제조 기업에서 주로 발견되는 용도는 같지만 성능 및 사양이 서로 다른 기계그룹 즉 '이종병렬기계(Non-identical Parallel Machine)' 그룹에서 발생되는 일정계획 문제를 주로 다루고 있다. 특히 기존의 연구들과 차이가 나는 부분은 기존의 연구에서 비교적 간과되었던 기계를 가동시키기 위해 먼저 수행되는 작업인 셋업 시간과 셋업 작업자 인원의 제약 및 셋업작업의 형태를 내부 셋업 작업과 외부셋업 작업으로 구분 적용하여 다루었다는 점이다.

제 2장 에서는 셋업 작업을 잘 이해하기 위해 셋업 작업의 개념과 셋업 비용, 실제 현장에서 일어나는 셋업의 종류들과 이러한 셋업 작업제 3장 에서는 처리순서에 따라 작업준비시간이 달라지는 일반적인 이종병렬기계 시스템에서 작업자 제약에 대한 고려가 없는 일정계획 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 기존의 연구와 약간 다른 형태로 제시하였다. 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 두 개의 단순 휴리스틱 알고리즘과 두 개의 유전 알고리즘을 개발하였다. 다양한 수치실험을 통해 이 알고리즘들이 비교적 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다. 특히 문제에 따라서는 단순 휴리스틱 알고리즘이 유전 알고리즘보다 오히려 나은 해를 제공하기도 하였다.

제 4장에서는 셋업 작업자 한 명이 기계 여러 대의 셋업을 전담하는 제약을 추가한 상황에 대한 이종병렬기계 시스템의 일정계획 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 제시하고, 이 모형의 유효성을 평가하기 위하여 최적화 소프트웨어를 사용해 작은 크기의 예제를 해결하였다. 실험결과 작은 크기의 문제는 최적화 모형을 사용해 최적해를 구할 수 있었으나, 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 유전알고리즘 기반의 휴리스틱 알고리즘을 개발하였다. 다양한 수치실험을 통해 제시된 알고리즘들이 매우 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다.

제 5장에서는 각 주문의 셋업시간이 내부활동과 외부활동으로 분리되어 있는 경우의 이종병렬기계 시스템에서 한 사람(혹은 한 팀)의 셋업 작업자가 셋업을 전담할 때 이 셋업 담당자가 각 주문의 내부 및 외부 셋업활동을 가장 효율적으로 수행하는 일정을 구하기 위한 알고리즘을 개발하였다. 우선 문제의 상황을 잘 표현해주는 수리계획 모형을 제시하고, 이 모형의 유효성을 평가하기 위하여 최적화 소프트웨어를 사용해 작은 크기의 예제를 해결하였다. 실험결과 작은 크기의 문제는 최적화 모형을 사용해 최적해를 구할 수 있었으나, 이 최적화 모형은 널리 알려진 것처럼 NP-hard 문제이므로 현실적인 크기의 문제를 해결하기 위해 유전알고리즘 기반의 휴리스틱 알고리즘을 개발하였다. 수치실험을 통해 제시된 알고리즘이 매우 짧은 시간 안에 좋은 품질의 해를 제공해준다는 사실을 확인하였다.

이 연구의 가장 큰 의의는 종래의 이종병렬기계 일정계획 연구에서 자주 다루어지지 않았던 셋업 시간 및 셋업 작업자의 수에 대한 현실적인 제약을 고려한 일정계획 알고리즘을 개발하여 이종병렬기계

그룹을 운영하는 중소규모 제조 기업들에게 신속하게 보다 나은 알고리즘을 제공함으로써 생산성 향상과 재고자원에 대한 낭비요소를 사전에 방지하는 효과를 주는데 있다. 아울러 이러한 알고리즘은 중소규모 제조업체 뿐만 아니라, 다품종 소량 생산을 주로 하는 항공정비창의 제작공장 기계작업장 등에서도 우선순위와 효율적인 생산을 가능하게 만드는 도구로서 유용할 것으로 생각된다.

2. 연구의 한계점과 향후 연구과제

본 연구의 한계점으로 첫 번째 셋업 작업자 수의 제약이 1인(1팀)인 경우만 다루었는데 현실적으로 셋업 작업자의 수가 2인 이상인 경우에 대한 향후 추가적인 연구가 필요할 것으로 생각된다.

두 번째로는 주문의 납기를 고려한 문제의 해결이 필요할 것으로 생각된다. 즉, 납기지연의 최소화 또는 납기에 대한 선행 완료 및 지연을 모두 최소화하는 문제 등을 생각할 수 있다.

마지막으로 셋업시간이 처리 순서에 종속적인 상황에 대해서도 추가적인 연구가 필요할 것으로 생각된다.

참고문헌

Abdekhodaee, A.H. and Wirth, A. (2002), Scheduling parallel machines with a single server: some solvable cases and heuristics, Computers & Operations Research, 29(3), 295-315.

Abdekhodaee, A.H., Wirth, A. and Gan, H.S. (2004), Equal processing and equal setup cases of scheduling parallel machines with a single server, Computers & Operations Research, 31(11), 1867-1889.

Abdekhodaee, A.H., Wirth, A. and Gan, H.S. (2006), Scheduling two parallel machines with a single server: the general case, Computers & Operations Research, 33(4), 994-1009.

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.

Brucker, P., Dhaenens-Flipo, C., Knust, S., Kravchenko, S.A. and Warner, F. (2002), Complexity results for parallel machine problems with a single server, Journal of Scheduling, 5(6), 429-457.

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.

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.

Gen, M. and Cheng, R. (1997), Genetic Algorithms and Engineering Design, John Wiley & Sons, New York, U.S.A..

Gharehgozli, A.H., Tavakkoli-Moghaddam, R. and Zaerpour, N. (2009), A fuzzy-mixed integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates, Robotics and Computer-Integrated Manufacturing, 25(4-5), 853-859.

Glass, C.A., Shafransky, Y.M. and Strusevich, V.A. (2000), Scheduling for parallel dedicated machines with a single server, Naval Research Logistics, 47(4), 304-328.

Guinet, A. (1990), Textile production systems: a succession of

non-identical parallel processor shops, Journal of the Operational Research Society, 42(8), 655-671.

Haque, L. and Armstrong, M.J. (2007), A survey of the machine interference problem, European Journal of Operational Research, 179(2), 469-482.

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.

Huang, S.H., Cai, L. and Zhang, X. (2010), Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server, Computers & Industrial Engineering, 58(1), 165-174.

Jeong, N.K. and Jeong, M.Y. (2000), Scheduling system for non-identical parallel production system, Korean Management Science Review, 17(2), 67-73.

Johnson, S.M. (1954), Optimal two and three stage production schedules with setup times included, Naval Research Logistics Quarterly, 1(1), 61-68.

Joo, C.M. (2012), Scheduling of restricted non-identical parallel machine with sequence and machine dependent setup times, Journal of the Korea Management Engineers Society, 17(3), 31-45.

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.

Kim, M.Y. and Lee, Y.H. (2012), MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server, Computers & Operations Research, 39(11), 2457-2468.

Koh, S.G. and Lim, D.J. (2014), Single machine scheduling for minimizing earliness/tardiness penalties, Journal of the Korea Management Engineers Society, 19(1), 109-119.

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.

Koh, S.G., Seo, W.C. and Lim, D.J. (2015), Single machine scheduling for minimizing earliness/tardiness penalties with sequence-dependent setup times, Journal of the Korea Management Engineers Society, 20(2), 133-149.

Kravchenko, S.A. and Warner, F. (1997), Parallel machine scheduling problems with a single server, Mathematical and Computer Modelling, 26(12), 1-11.

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.

Li, K. and Yang, S.L. (2009), Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms, Applied Mathematical Modelling, 33(4), 2145-2158.

Marsh, J.D. and Montgomery, D.C. (1973), Optimal procedure for scheduling jobs with sequence-dependent changeover times on parallel processors, AIIE Technical Papers, 279-286.

Monden, Y. (1983), Toyota production system: Practical Approach to Production Management, Industrial Engineering and Management Press, Atlanta, U.S.A..

Pinedo, M. (2002), Scheduling: Theory, Algorithms, and Systems, Prentice Hall, New Jersey, U.S.A..

Ruiz, R. and Vazquez-Rodriguez, J.A. (2010), The hybrid flow shop scheduling problem, European Journal of Operational Research, 205(1), 1-18.

Sabouni, M.T.Y. and Logendran, R. (2013), Carryover sequence-dependent group scheduling with the integration of internal and external setup times, European Journal of Operational Research, 224(1), 8-22.

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.

Weng, M.X., Lu, J. and Ren, H. (2001), Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective, International Journal of Production Economics, 70(3), 215-226.

Zhu, Z. and Heady, R.B. (2000), Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach, Computers & Industrial Engineering, 38(2), 297-305.