공학박사 학위논문

다품목 화공처리 라인의 실시간 단일 호이스트 일정계획

2018년 8월

부경대학교 대학원

기술경영협동과정

이 정 구

공학박사 학위논문

다품목 화공처리 라인의 실시간 단일 호이스트 일정계획

지도교수 고 시 근

이 논문을 공학박사 학위논문으로 제출함

2018년 8월

부경대학교 대학원

기술경영협동과정

이 정 구

이정구의 공학박사 학위논문을 인준함

2018년 8월 24일

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 >

Real Time Single Hoist Scheduling for Multi-Item Chemical Processing Line

Jung-Koo Lee

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

Abstract

This thesis deals with a single-hoist scheduling problems for a multistage production line in which a computer-controlled hoist system performs the transportation of parts between stages. The production line consists of a series of tanks filled with a variety of chemical solutions for surface processing of the metal parts, and the processing of parts is compiled by soaking them into tanks for a certain time and in particular order. Because the system produces various items, there are distinct soaking sequences for the diverse items produced on the line.

Although some MILP (mixed integer linear programming) models for the single-hoist and multiple-products scheduling problem were developed earlier, the production line that motivated this study has a special feature that was not considered in the earlier studies. The company produces parts for assembling an aircraft and an order from customer consists of a set of parts. In order that the set is transported by the hoist and soaked in a tank all together, a tool for installing parts, that is called a rack, is needed. In order to insert a new job into the tank line, an empty rack is needed. When the process of a job is finished, the parts

that were installed on the rack are released, and the empty rack can be used to install a new job. In this thesis a new MILP model that considers this rack condition is proposed.

Applying the model to a commercial optimization software, the optimal solutions for relatively small problems can be found in very short times. For the case of realistic industry-size problems, however, the MILP model cannot find an optimal solution in a reasonable computation time. To solve such problems this thesis developed a branch-and-bound based heuristic algorithm, in which the number of branches from a node is variable with the problem size. For a very big problem, for example, the number of branches from a node is restricted to just one, while it can be two or more for the small problems. To evaluate the performance of the heuristic algorithm, the biggest problems that can be solved by the MILP model are generated and used for comparing the results of the heuristic algorithms to optimal solutions. The comparison showed that the heuristic has very good performance and the computation time is sufficiently short to use the algorithm in real situation.

Keyword : Hoist, Scheduling, Heuristic. Branch-and-bound

I. 서론

1. 연구의 배경 및 목적

호이스트(Hoist)를 사용한 생산시스템은 도금이나 코팅과 같은 표면처리가 필요한 항공산업이나 자동차 산업 등 다양한 산업 분야에서 널리 사용되고 있다. 표면처리(Surface treatment)라 함은 부품의 금속재료 표면상에 이종 재질을 전기적, 물리적, 화학적 처리 방법 등을 통해 보호 표면을 생성시킴으로써 소지 금속의 방청, 외관 미화, 내마모성, 전기 절연, 전기 전도성 부여 등의 폭넓은 목적을 달성 시키고자 하는 일련의 조작들을 말한다. 즉, 환경의 영향으로 인한 금속의 부식 방지나 표면 특성을 개선하기 위해 피막을 적용하는 공정을 표면처리라 한다. 항공산업에서 수행하는 표면처리는 일반적으로 화공처리에 의해 이루어지므로 표면처리공정을 화공처리공정이라 한다. 호이스트를 사용한 생산시스템은 다양한 산업에서 적용되고 있지만 여기서는 항공산업의 화공처리공정에 적용되고 있는 호이스트 생산시스템에 대해서 알아보고자 한다.

항공기 부품 제작 산업의 특징으로 다품종 소량 생산과 높은 신뢰성이 요구된다. 항공기는 생산되는 대수는 적으나 소요되는 부품이 수십 만개 이상으로 부품수가 매우 많고 형상이나 재질이 다양하기 때문에 제작 공정이 복잡하며, 대량 생산에 많은 어려움이 있다. 특히 항공기는 여객 수송을 담당하는 초고속 운송 수단이면서 지상을 떠나 공간을 이동하기 때문에 고도의 안전성이 요구된다. 따라서 항공기 운항 시 발생할 수 있는 사고에 대한 사전예방 및 안전한 비행을 위해 이에 사용되는 항공기 부품은 고도의 품질과 신뢰성이 요구된다. 항공기는 제작기간이 길고 한 대 가격이 고가이다 보니 다른 운송수단보다 안전하고 오래 사용할 수 있게 부품이 제작되어야 한다. 특히 운항 중 번개나 비, 구름, 수분, 온도 등의 여러 가지 환경 조건에 의한 영향을 받을

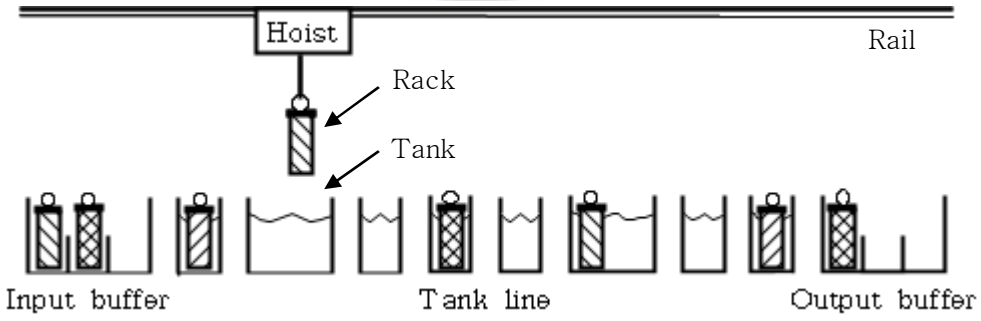
수 있으며, 대륙간 이동으로 인해 염분 등의 영향을 많이 받는다. 그러므로 항공기 부품은 부식이 발생될 수 있는 여러 가지 환경적인 요인이 존재하게 되며, 부식이 발생하게 되면 항공기의 안전성에 큰 영향을 미치게 된다. 그래서 항공기 부품은 부식이 발생되지 않도록 금속의 방청과 부품을 오래 사용할 수 있는 내마모성, 그리고 공중에서 번개 등의 영향을 최소화 하기 위한 전기 전도성 부여 등 여러 가지 목적으로 표면처리를 하게 된다. 즉, 항공기 부품 제작에서 사용되는 모든 금속 부품들은 부식 방지와 내마모성 그리고 전기 전도성의 부여 등의 목적으로 화공처리공정을 수행해야 한다. 금속 부품의 가공 공정은 일반적으로 선반, 밀링 등의 여러 가지 기계로 절삭 가공을 하는 기계 가공공정과 압축, 프레스 등의 기계와 수작업으로 금속 성형을 하는 판금가공 공정으로 나눌 수 있다. 기계가공 장비에서 가공된 부품과 판금가공 장비에서 성형된 부품들은 화공처리공정으로 이동하게 된다.

화공처리공정은 화학 처리용 액체가 채워진 탱크(Tank)에서 각 부품들이 필요로 하는 각각의 공정을 수행하게 된다. 각 부품의 화공처리공정은 그 부품이 필요로 하는 처리 작업에 맞게 각각의 탱크들을 순서대로 방문하여 탱크별로 정해진 시간 동안 부품을 탱크에 담그고 꺼내는 공정으로 이루어진다. 각각의 부품을 탱크에 담그고 꺼내고 또 다른 탱크로 이동할 때 사용되는 장비가 호이스트이다. 즉, 호이스트를 이용하여 다양한 부품들에 대해 화공처리에 필요한 탱크들을 순서대로 방문(이동)하여 탱크에 담그고, 그 탱크 공정이 완료되면 부품을 꺼내어 다음 수행할 탱크를 방문하여 탱크 공정을 진행하게 된다. 그렇게 필요로 하는 모든 탱크의 작업을 수행한 후 화공처리공정을 완료하게 된다. 이렇게 호이스트를 이용하여 부품을 생산하는 공정을 호이스트 생산시스템 또는 호이스트 생산라인이라고 한다.

[그림 1-1] 단일 호이스트 생산라인의 예를 보여주고 있다. 호이스트 생산라인은 부품을 고정시키는 랙과 랙을 이동시키는 호이스트, 그리고 화학처리를 위해 화학 액체가 담겨있는 탱크가 있으며, 부품을 랙에 고정시키고 탱크

처리 공정을 수행하려고 대기하고 있는 투입 버퍼(Input buffer)와 모든 탱크의 공정이 완료되어 랙에서 부품을 제거하고 다음 작업을 대기하고 있는 완료 버퍼(Out buffer)로 구성된다. 호이스트 생산라인에서 사용되는 호이스트가 1개일 때 단일 호이스트라고 하며, 2개 이상의 호이스트가 사용될 때 다중 호이스트라고 한다. 그러므로 작업의 양이나 처리되는 공정에 따라 1개의 라인에 2개 이상의 호이스트를 구성하여 생산의 능률을 향상시킬 수 있다. 그리고 탱크들이 모여있는 화공처리라인도 1개로 모든 작업을 처리할 수 있으며, 생산되는 부품의 특성에 따라 각각의 화공처리 공정에 맞게 2개 이상의 라인도 구성할 수 있다. 예를 들어 수행하는 화공처리 공정이 다양하고 생산하는 작업 양이 많지 않다면 1개의 라인으로 호이스트 생산시스템을 구성하는 것이 적합하며, 특정 화공처리공정에서 작업이 집중적으로 이루어 지거나 전체 작업 양이 많아서 추가적으로 라인이 더 필요하다면 공정 특성에 맞게 2~3개의 화공처리라인에서 처리할 수 있도록 구성하여 사용할 수도 있다.

본 논문의 사례 기업의 화공처리라인을 예를 들면 30개의 탱크가 배치된 구조에 1개의 호이스트가 있으며, 그리고 투입 버퍼와 완료 버퍼로 구성되어 있는 1개의 화공처리라인을 가지고 있다. 그리고 여기서 16개의 각기 다른 화공처리공정을 수행하고 있다.



[그림 1-1] 단일 호이스트 생산라인의 예

항공산업의 단일 호이스트 생산시스템에서는 항공기의 종류와 형상, 재질 및 크기 등의 차이와 많은 부품 수로 인해 다양한 작업이 존재한다. 이러한 특징으로 인해 화공처리공정에서는 다음과 같은 제약조건이 발생하게 된다.

첫 번째는 부품의 다양성으로 인해 발생하는 화공처리공정의 다양성이다. 사례 기업의 생산 라인에는 8개의 서로 다른 화공처리공정이 있으며, 동일 공정이라도 부품이나 고객 별로 탱크에 담그는 시간과 수행하는 탱크 공정의 차이가 발생하여 세부적으로는 16개의 서로 다른 화공처리공정들이 존재하고 있다. 이런 공정의 다양성으로 인해 생산 일정계획 수립에 많은 어려움이 발생하게 된다.

두 번째는 화공처리공정의 전체 완료 시간과 각 탱크 별 처리 하한 시간 및 상한 시간이다. 호이스트 이동 시간까지 포함하면 하나의 화공처리공정을 완료하는데 소요되는 시간은 1시간에서 26시간까지 다양하며 평균 2.5시간 정도 소요된다. 그리고 각 탱크 별로 탱크에 담그는 시간은 최소 시간인 하한 시간과 최대 시간인 상한 시간으로 구분된다. 그래서 부품을 탱크에 담근 후 하한 시간이 지난 다음에 꺼낼 수가 있으며 상한 시간이 경과되기 전에 탱크에서 꺼내야 하는 제한 시간이 존재하게 된다. 호이스트 생산시스템에서 생산성을 증대하기 위하여 하나의 호이스트로 여러 개의 작업을 동시에 처리해야 될 때 탱크의 처리 하한 시간과 상한 시간이 호이스트의 이동에 제약을 주는 요인으로 작용한다. 전체 소요시간이 짧은 공정은 각 탱크에서 처리되는 하한 시간과 상한 시간이 짧게 설정되어 있으며, 전체 소요시간이 긴 공정은 각 탱크에서 처리되는 하한 시간과 상한 시간이 길게 구성되어 있다. 그러므로 전체 소요시간이 짧은 시간으로만 작업을 구성하게 되면 짧은 시간으로 인해 호이스트 이동에 제약을 주게 되고, 긴 시간으로만 작업을 구성하게 되면 후속 작업의 탱크 대기 시간이 많이 증가하게 된다. 그러므로 각 작업의 전체 완료 시간과 각 탱크 별 처리되는 하한 시간 및 상한 시간에 따라 어떻게 호이스트 생산 계획을 수립하는가에 의해 화공처리공정의 생산성은 많은 차이가 발생하

게 된다.

세 번째는 한번에 이동할 수 있는 부품의 수 이다. 탱크의 크기는 고정되어 있으므로 한번에 처리할 수 있는 부품의 수는 부품의 크기와 부품을 장착할 수 있는 랙에 의해 결정된다. 부품들은 탱크로 이동하기 위해 호이스트에 장착될 수 있어야 하고, 탱크에 담그고 꺼내고 이동하는 동안 떨어지면 안되며, 또한 탱크에 담겨 화학 용액 처리를 할 때 부품 간에 접촉이 되어있지 않아야 한다. 접촉되어 있으면 접촉된 부분에 화학 용액이 잘 침투되지 않아 부품 표면에 화학 처리가 제대로 수행되지 않으므로 부품은 일정한 간격을 유지한 채 고정된 상태에서 화학 용액이 담긴 탱크에 담겨져야 한다. 이런 이유로 부품을 고정하고 호이스트로 이동할 수 있게 하는 장비가 필요하며 이 장비를 랙이라고 한다. 사례 기업에서는 대, 중, 소 크기로 부품을 장착할 수 있는 랙이 있으며, 부품의 크기에 따라 대, 중, 소 의 랙을 선택하여 큰 부품은 3개, 작은 부품은 200~300개 까지도 랙에 장착할 수 있다. 부품의 수에 따라 랙에 장착하는 시간은 많은 차이가 발생한다. 2~3개의 부품은 15~20분 내에 랙에 장착할 수 있지만, 200~300개의 부품은 1시간에서 1시간 30분 정도 장착 시간이 소요된다. 즉, 적은 수의 부품으로만 작업을 구성하게 되면 랙 장착 시간은 줄어들 수 있으나 하루에 생산되는 부품의 수는 작아지게 된다. 그리고 많은 수의 부품으로만 작업을 구성하게 되면 랙에 부품을 장착하는 시간이 많이 소요되므로 장착 작업이 완료되기 전까지 탱크라인으로 작업을 투입하지 못한다. 그러므로 부품의 수에 따라 랙에 장착하는 시간과 1일 처리할 수 있는 작업의 수는 차이가 발생하므로 각 작업에 부품의 수를 적절히 배치하여야 효율적인 생산 일정계획을 수립 할 수 있다.

네 번째는 수시로 추가적인 작업이 계속 발생할 수 있다는 것이다. 화공처리 공정은 기계가공공정과 판금가공공정에서 완료된 모든 금속 부품들이 거쳐가야 되는 필수 공정이다. 가공 작업이 방금 완료된 부품이라도 생산 일정의 긴급 여부에 따라 바로 화공처리라인에 투입되어야 할 경우가 수시로 발생된다.

그때마다 호이스트 생산 일정계획을 재 수립하여야 할 필요성이 제기되는데, 현재 진행되고 있는 작업과 대기하고 있는 작업에서 추가로 발생되는 작업의 수행을 위해 호이스트 생산일정을 재 수립한다는 것은 결코 쉬운 일이 아니다. 그렇기 때문에 추가적인 작업이 발생하게 되면 진행중인 작업을 고려한 일정 재 조정의 문제가 발생되어 일정계획 수립을 더욱 더 어렵게 한다.

위의 4가지 제약조건으로 인해 호이스트 생산 일정계획을 수립한다는 것은 많은 어려움이 따르게 된다. 사례 기업의 단일 호이스트 생산시스템에서 동시에 처리할 수 있는 작업은 2~3개이며, 여러 가지 제약조건으로 하루에 15~20개의 처리할 수 있음에도 불구하고 1일 11개의 작업으로 일정계획을 수립하고 있다.

그러므로 단일 호이스트 생산시스템에서 다양한 작업을 처리하기 위해서는 작업들의 처리 및 운송을 동시에 고려하는 호이스트 생산 일정계획이 매우 중요하며 어떻게 일정계획을 수립하는가에 따라 1일 처리할 수 있는 작업의 능률은 달라지게 된다. Kumar(1994)에 의하면 호이스트 일정계획은 시스템의 생산성을 결정하는 가장 중요한 요인이며, 호이스트 일정계획에 따라 작업 대기시간이 평균 20%까지 줄어들 수 있다고 하였다.

즉, 단일 호이스트 생산시스템에서 생산성을 제고하기 위해서는 전체 작업 처리 완료시간(Makespan ; 향후 완료 시간으로 지칭함)을 최소화하는 호이스트 일정계획의 방안 수립에 대한 연구가 필요하다. 따라서 본 연구에서는 완료 시간을 최소화 할 수 있는 효율적인 단일 호이스트 생산시스템의 일정계획 방안에 대해 탐색하고자 한다.

2. 연구의 구성

본 연구에서는 단일 호이스트 생산시스템에서 다양한 작업에 대해 완료 시간을 최소화하기 위한 효율적인 호이스트 일정계획 방안을 탐색하고자 하며, 본 연구의 나머지 부분은 다음과 같이 구성된다.

제 2장에서는 항공기 생산공정과 화공처리 공정에 대해 자세히 알아보고, 문제 해결을 위한 기존 연구 분석을 하였다. 특히 본 연구의 문제와 매우 유사한 상황의 문제를 다룬 Zhao et al.(2013)의 실시간 동적 호이스트 일정계획(RDHS, Real Time Dynamic Hoist Scheduling)에 대한 연구를 집중적으로 분석하였다. 그리고 예제를 생성하고 실험을 통해 결과를 확인해 보았다.

제 3장은 단일 호이스트 생산시스템에서 다양한 작업을 처리하기 위해 분지한계 기반의 휴리스틱 일정계획을 제안하였다. 2장과 동일한 문제로 현실적인 규모의 경우에도 빠른 시간 안에 양호한 품질의 해를 제공할 수 있는 휴리스틱 방법론을 제안하고 수치실험을 수행하였다.

제 4장에서는 사례 기업의 호이스트 생산시스템 일정계획 방법에 대해 알아보았다. 사례기업에서는 어떠한 방법과 규칙으로 일정계획을 수행하는지 그 방법과 규칙을 알아보고 그에 따라 실험된 결과를 휴리스틱 결과와 비교 분석해 보았다. 그리고 일정계획 방법을 알아보는 과정에서 제한된 치공구(랙)를 사용하는 특수한 형태의 제약조건을 추가적으로 확인할 수 있었다.

제 5장에서는 2장에서 관찰하였던 수리모형에 4장에서 확인한 랙 제약조건을 추가한 확장형 수리모형을 제안하였다. 예제를 통해 이 모형의 유효성을 살펴본 다음 3장에서 제안하였던 분지한계 기반의 휴리스틱 모형을 간략화하여 이 모형에 적용하였다. 예제를 통해 분석해 본 휴리스틱 방법론이 4장에서 보여주었던 사례기업의 일정계획 방법보다 얼마나 우수한지를 보여준다.

마지막으로 제 6장에서는 결론으로 본 연구의 결과를 요약하고, 추후 연구과제를 제시하였다.

II. 기존 연구 분석

1. 개요

항공산업의 화공처리공정은 호이스트를 사용한 생산시스템으로 다양한 유형의 작업을 처리하고 있다. 항공산업의 단일 호이스트 생산시스템에서는 항공기의 종류와 형상, 재질, 크기 등에 따라 다양한 부품이 존재하고 부품의 수가 매우 많다. 그렇기 때문에 화공처리공정을 수행할 때 다양한 화공처리공정과 작업 처리 완료 시간 그리고 한번에 이동할 수 있는 부품의 수 및 수시로 추가적인 작업이 발생하는 제약이 발생한다. 이런 여러 가지 제약조건으로 인해 호이스트 일정계획을 수립한다는 것은 많은 어려움이 따르게 된다. 그러므로 단일 호이스트 생산시스템의 전체 완료 시간을 최소화 하기 위해서는 효율적인 호이스트 일정계획 방안 수립에 대한 연구가 필요하다. 이 연구를 위해 항공산업에서 부품 제작이 어떻게 이루어지는지 그리고 호이스트 생산 일정계획에 대해 어떤 연구가 수행되었는지를 먼저 분석해봐야 한다. 이 장에서는 항공산업의 부품 제작 공정과 이전 연구에 대해 알아보도록 하겠다.

제 2절에서는 기존 연구 분석에 앞서 항공산업의 부품 제작 공정에 대해서 알아보았다. 항공기 기체 구성품은 기계가공, 판금가공, 복합제 그리고 조립과 화공처리 공정 등에 의해 부품 제작이 이루어진다. 항공산업의 화공처리공정을 설명하기 위해서는 금속 가공 작업을 수행하는 기계가공과 판금가공 공정을 알아야 하며, 복합제와 조립 공정에 대해서도 이해하여야 하기 때문에 전반적으로 항공기 기체의 부품 제작 공정에 대해 설명하였다.

제 3절에서는 호이스트 생산시스템의 일정계획에 대한 기존 연구 분석을 하였다. 제 4절에서는 기존 연구 중 실시간 동적 호이스트 일정계획 모형에 대해 설명하였으며, 사례 기업에 적용 가능한지 예제를 통해 실험을 수행하고 결과를 분석해 보았다. 마지막으로 제 5절에서는 이장을 요약하였다.

2. 항공기 생산 공정

항공기의 구성품은 동체와 날개 등의 항공기 기체와 유압 체계, 착륙장치, 엔진, 항공 전자장비 등의 항공기 작동 시스템으로 크게 나눌 수 있다. 항공기 기체를 제작하기 위해 필요한 부품을 생산하는 공정은 기계가공공정, 판금가공공정, 복합제공정과 조립공정으로 나눌 수 있다. 사용되는 재료로는 금속부품과 비금속 부품으로 나누며, 금속 부품은 알루미늄 합금과 티타늄 합금, 스테인리스 등의 재질로 구성되어 있고, 비금속 부품은 탄소섬유와 여러 가지 화학재료로 제작되는 복합제로 구성된다. 제작된 부품들은 항공기 동체와 날개 등의 기체 조립에 사용되고 있다.

항공기 기체를 조립하려면 먼저 각각의 부품을 제작하여야 한다. 부품들은 기계가공공정과 판금가공공정, 그리고 복합제공정에서 가공하게 된다. 기계가공공정과 판금가공공정에서 생산된 모든 금속 부품은 화공처리공정을 수행하게 된다. 화공처리공정을 수행하고 난 후 항공기 부품은 부식과 이종 금속간의 마찰에 의한 표면을 보호하고 외관 미화를 위해 마지막 공정으로 페인트를 도포한다. 복합제를 포함한 모든 부품들은 마지막 공정에 페인트 공정이 있으며, 페인트 공정이 완료되면 검사 후 부품 제작이 완료된다. 완료된 부품은 그 부품 그대로 납품되기도 하지만 대부분 조립공정에서 조립되어 기체 구성품이나 완제기 형태로 고객에게 납품하게 된다.

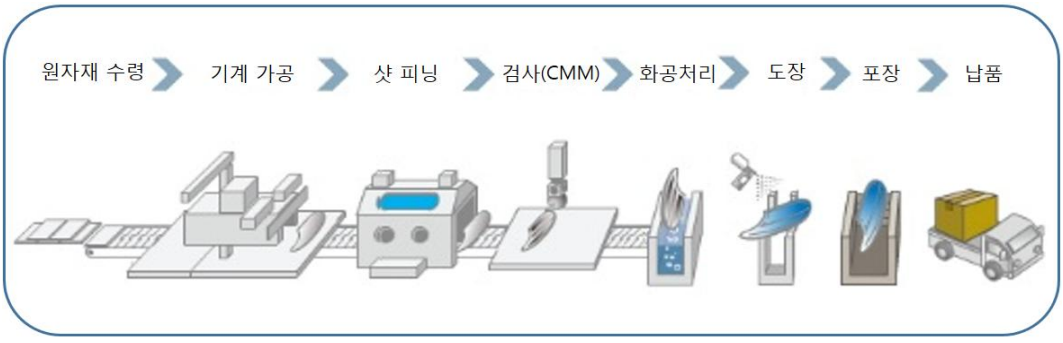
2.1 기계가공공정

기계가공공정은 [그림 2-1]과 같이 원자재 수령 – 기계가공 – 샷피닝 – 검사 – 화공처리 – 도장(Paint) – 포장의 순서로 이루어 진다. 기계가공공정은 각 부품의 크기에 맞게 절단된 원자재를 수령하는 것에서부터 시작한다. 그리고 부품의 크기나 형상에 따라 여러 가지의 가공 기계로 투입하게 된다.

강도가 요구되는 구조물(Structure)에 사용되는 부품의 경우 복잡한 형상을 가지고 있으며 그 복잡한 형상은 기계가공 장비에 의해 제작된다. 기계가공 장비는 선반과 밀링 등의 절삭기계를 사용하나, 대부분이 CNC 장비로서 복잡한 형상을 가공하기 위해 3축(Axis)과 5축 장비가 많이 사용되고 있다. 그리고 스핀들(Spindle)이 1개에서 3개까지 장착된 장비를 사용하는데, 크기가 커서 가공 시간이 많이 소요되는 부품들은 장비의 생산성을 높이기 위해 스핀들을 2~3개 사용하여 동시에 여러 개의 부품을 가공할 수가 있다. 기계가공공정을 마친 부품들은 표면경도와 피로강도를 향상시키기 위해 샷피닝(Shot Peening) 공정을 수행하게 된다. 이 공정은 샷 이라고 하는 작은 구슬들을 표면에 빠르고 강하게 때려 표면경도 및 피로강도를 생기게 하는 작업이다.

샷피닝 공정을 수행한 부품들은 화공처리공정으로 이동하게 된다. 화공처리공정에서는 부품의 재질 및 사용 용도에 적합한 탱크 공정을 처리하게 된다. 화공처리공정 완료 후 Primer 와 Top Coat 등의 도장(Paint) 공정을 수행한다. 도장 작업은 부식 방지와 표면 보호 그리고 외관상의 목적으로 수행하게 되며, 마지막 최종 검사를 하고 나면 부품의 가공 공정을 완료하게 된다.

가공이 완료된 부품은 단일 부품으로 납품되기도 하나, 대부분 조립을 위해 조립 작업장으로 이동 된다. [그림 2-1]은 항공기 부품의 기계 가공품의 생산공정을 그림으로 나타낸 것이다.



[그림 2-1] 항공기부품 기계가공품의 생산공정

2.2 판금가공공정

판금가공공정은 [그림 2-2]와 같이 원자재 수령 – 판금 가공 – 열처리 – 검사 경도/전도도 측정 – 화공 처리 – 도장 – 포장의 순서로 이루어 진다. 항공기 부품 중 Skin 이나 얇은 두께의 구조물들을 Sheet Metal 이리고 하는데 이런 얇은 두께의 금속들은 기계가공을 하지 못하기 때문에 금속성형이라는 판금가공공정을 수행하게 된다.

판금가공공정도 기계가공공정과 마찬가지로 원자재 수령 후 제품 가공이 가능한 크기로 절단하는 작업을 먼저 수행한다. 제품 가공이 가능한 크기로 절단된 자재들은 Press, 전단, 롤링, 수작업 등의 여러 가지 판금가공공정을 거쳐 부품 성형 작업을 수행하게 된다.

부품 성형 작업이 완료된 부품들은 재료의 강도를 높이기 위해 열처리 작업을 수행하게 된다. 판금가공 부품들은 두께가 얇은 판으로 되어있기 때문에 기계가공 부품처럼 샷피닝을 수행하지 못하고, 열처리로서 재료의 강도를 높이거나 잔류응력 또는 가공경화를 없애는 작업을 수행하게 된다.

열처리가 완료된 판금가공 부품들은 화공처리공정으로 이동하게 되며, 화공처리공정 이후의 공정은 기계가공 공정과 동일하게 진행하게 된다. [그림 2-2]는 항공기 부품의 판금 가공품의 생산공정을 그림으로 나타낸 것이다.

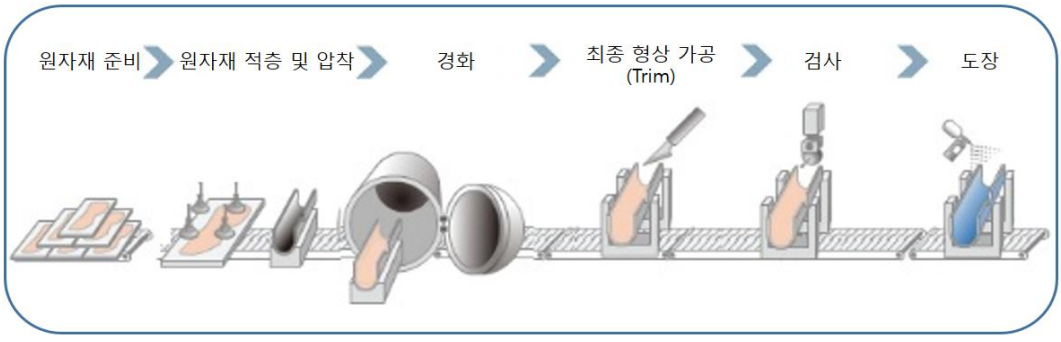
[그림 2-2] 항공기부품 판금가공품의 생산공정

2.3 복합제공정

복합제공정은 [그림 2-3]과 같이 원자재 준비 – 원자재 적층 및 압착 – 경화 – 최종 형상 가공(Trim) – 검사 – 도장의 순서로 이루어 진다. 복합제공정은 탄소섬유와 여러 가지 화학성분의 화합재료들을 사용하여 여러 겹으로 적층하고, 압착한 후 경화 작업과 최종 형상을 가공하는 공정을 수행한다.

항공기는 공중을 날아다니는 운송 수단으로써 이륙과 착륙 시에 무게에 영향을 매우 많이 받게 된다. 운항 중에도 무게가 많이 나가게 되면 연료소모가 많아져 비용이 증가하며, 연료의 연소로 인한 이산화탄소의 배출도 많아지게 된다. 비행의 안전과 비용 그리고 환경 문제로 항공기 기체 구조물들을 기존의 금속제품보다 가벼우면서 강도 면에서 우수한 복합제 제품이 많이 개발되었다. 초창기에는 항공기 외피에 많이 사용되었으나 현재는 항공기 구조물까지 복합제 제품으로 제작되고 있다. 특히 복합제 제품은 금속제품과 달리 부식이 되지 않아 항공기의 안전성 측면에서도 훌륭한 재료로 인식되고 있다.

복합제 제품은 원자재 준비 후 가공하고자 하는 틀에 원자재를 적층하고 압착 한 뒤 Oven 에서 경화하는 작업을 수행한다. 경화 후 최종 형상가공 작업을 하고 검사를 수행 한 후 도장 작업을 수행하게 된다. [그림 2-3]은 항공기 부품의 복합제 제품의 생산공정을 그림으로 나타낸 것이다.



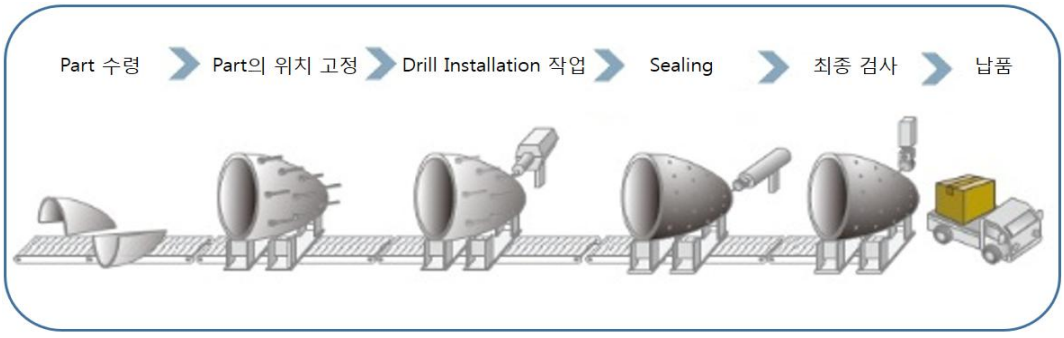
[그림 2-3] 항공기부품 복합제 제품의 생산공정

2.4 조립공정

조립공정은 [그림 2-4]와 같이 부품 수령 – 부품의 위치 고정 – Drill 및 Installation 작업 – Sealing – 최종 검사 – 납품의 순서로 이루어 지게 된다. 기계가공공정 및 판금가공공정 그리고 복합제공정에서 완료된 부품들은 여러 가지 결합체(Fastener)로 결합하여 항공기 기체 구성품으로 조립하게 된다. 사례 기업에서는 항공기 날개, 기체 구조물, Door Assy 등을 제작하여 항공기 제작사로 납품을 하고 있으며, 헬기 및 무인기 등 여러 가지 항공기 완제기 제작 사업도 진행하고 있는데, 납품되는 모든 제품들은 부품들을 가공 후 조립하여 기체 구성품 또는 완제품으로 제작하게 된다.

항공기는 공중에서 빠른 속도로 비행을 하고 기후와 바람의 영향을 많이 받아 진동이 많이 발생한다. 만약 항공기를 선박처럼 용접으로 제작한다면 항공기는 얼마 지나지 않아 진동으로 인해 용접면에 균열(Crack)이 발생되어 기체가 손상될 것이다. 그래서 항공기의 안전을 위해 항공기는 용접을 하지 않고 대신 Bolt, Rivet 이나 Pin 등 여러 가지 결합체로 모든 기체를 조립하게 된다.

조립공정에서 부품 수령 후 Jig 를 통해 부품의 위치를 고정하고 Drill 및 Install 작업과 Sealing 작업을 한 후 최종 검사를 수행하고 납품하게 된다. [그림 2-4]는 항공기 부품의 조립 제품 생산공정을 그림으로 나타낸 것이다.



[그림 2-4] 항공기부품 조립 제품의 생산공정

2.5 화공처리공정

항공산업의 화공처리공정은 부품의 금속재료 표면상에 전기적, 물리적, 화학적 처리 방법 등을 통해 금속의 방청, 내마모성, 전기 전도성 부여 등의 목적을 위해 금속에 피막을 적용하는 공정이라 할 수 있다. [그림 1-1]은 단일 호이스트 생산라인의 예로서 화공처리 생산라인은 화학용액이 채워져 있는 여러 개의 탱크와 탱크 위에서 호이스트를 이동 시킬 수 있는 레일 및 부품을 고정시켜 부품을 이동하고 탱크에 담글 수 있는 랙과 랙을 탱크에서 탱크로 운반하는 호이스트로 되어있다. 그리고 탱크 작업 전 랙에 부품을 고정시키고 화공처리 작업을 대기하고 있는 투입 버퍼와 탱크 작업 완료 후 부품을 랙에서 제거하고 새로운 작업을 위해 완료된 랙을 보관하는 완료 버퍼로 구성된다고 할 수 있다.

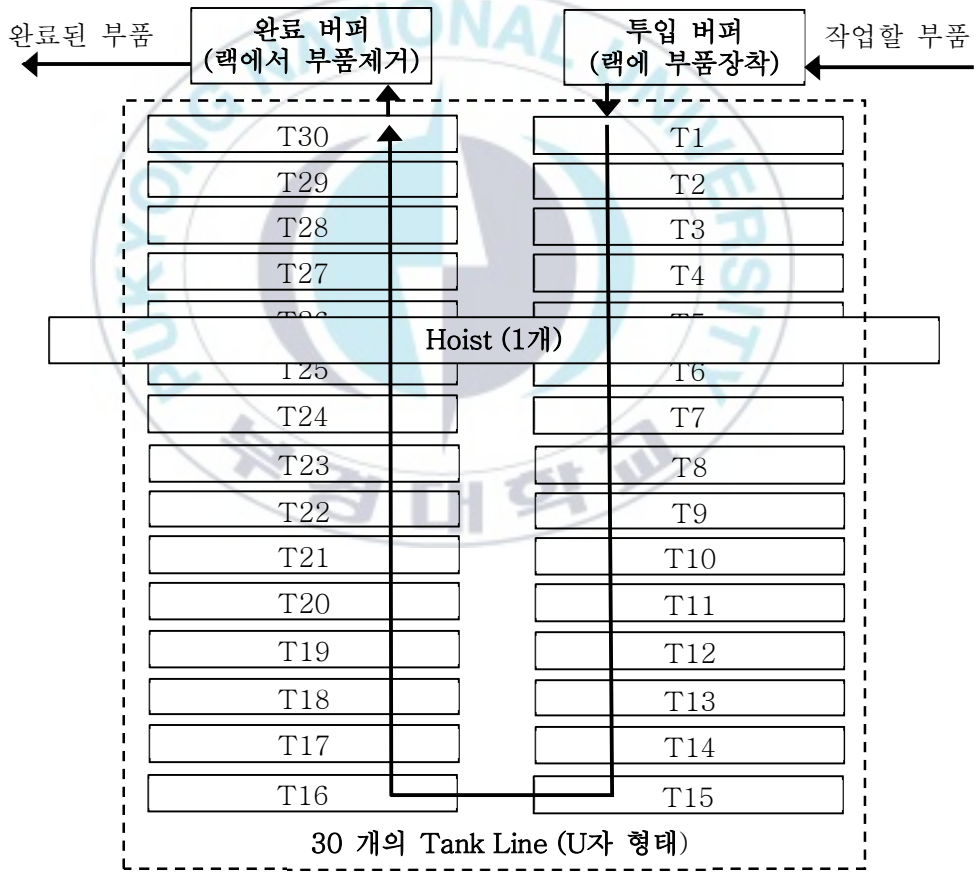
화공처리라인의 구성

- Tank : 금속표면에 피막을 적용하기 위해 화학 액체들이 들어있는 용기
- Rack : Tank로 이동하고 담그기 위해 부품들을 고정시키는 장비
- Hoist : Rack을 올리고 내리고 탱크 사이를 이동시킬 수 있는 장비
- Rail : Hoist 가 이동될 수 있게 설치된 강철제로 된 구조물
- Input Buffer : 부품을 Rack에 고정하거나, 작업을 대기하고 있는 공간
- Output Buffer : 작업 완료 후 부품을 Rack 에서 제거하기 위한 공간

즉, 호이스트 생산시스템에서 작업이란 부품들이 고정된 랙을 호이스트를 이용하여 투입 버퍼를 출발하여 정의된 화공처리공정에 따라 각각의 탱크를 순차적으로 방문하여 필요한 화공처리공정을 수행 한 후 완료 버퍼에 도착하기까지라고 할 수 있다. 그리고 투입 버퍼는 호이스트 작업을 위해 랙에 부품을 장착하는 장소로서 작업을 준비하고 탱크라인으로 투입을 위해 대기하고

있는 공간이며, 완료 버퍼는 작업 완료 후 부품을 랙에서 제거하는 공간이다. 부품의 제거가 완료되면 다음 작업을 위해 랙은 투입 버퍼로 이동하게 된다.

사례 기업의 호이스트 생산시스템은 [그림 2-5]과 같이 1개의 호이스트를 사용하여 U자 형태로 30개의 탱크가 배치되어 있으며 투입 버퍼와 완료 버퍼로 구성되어있는 화공처리라인을 가지고 있다. 30개 탱크에서 16개의 각기 다른 화공처리공정을 수행하고 있으며, 작업할 부품이 투입 버퍼로 들어오면 랙에 부품을 장착하여 탱크 라인으로 투입한다. 탱크 작업이 완료되면 완료 버퍼로 이동하여 랙에서 부품을 제거하고 화공처리 라인을 나가게 된다.



[그림 2-5] 사례 기업의 호이스트 생산시스템

[그림 2-5]에서 30개의 탱크는 탱크가 처리되는 순서대로 정렬되어 있다. 그러므로 화공처리라인에서 탱크의 처리 순서대로 탱크를 방문하게 될 때 로드 된 호이스트의 이동은 정렬된 순서대로 전진만 수행하게 된다. 빈 호이스트는 이동을 위해 후진을 할 수 있으나 로드 된 호이스트의 이동은 전진만 하는 제약 조건이 따르게 된다.

화공처리라인의 탱크 수는 30개로 구성되어 있으나, 하나의 공정에서 실제 처리되는 탱크 수는 8~10개 정도이다. 호이스트 생산시스템에서 부품의 다양화로 인해 16개의 다른 화공처리공정을 수행하고 있으며, 각각의 화공처리 공정은 수행하는 사업별, 사용하는 용도 그리고 사용하는 재질에 따라 다르게 구성되어 있다. 16개의 화공처리공정에서 처리되는 시간은 1시간에서 26시간으로 평균 2.5시간 정도 소요된다. 16개의 공정에서는 각각의 탱크에서 수행해야 될 탱크 처리 공정이 있으며, 각 탱크에서는 처리해야 될 하한 시간과 상한 시간이 있어 하한 시간이 경과된 후 탱크에서 이동이 가능하고 상한 시간이 경과되기 전 탱크에서 꺼내어 이동하여야 한다. <표 2-1>에는 8개의 탱크 공정을 수행하는 사례 기업의 화공처리공정의 처리 순서 예를 나타내었다.

<표 2-1> 사례 기업의 화공처리공정의 처리 순서 예

순서	탱크 번호	탱크에서 처리되는 공정	처리 시간 (하한, 상한)
1	T1	VAPOR DEGREASER	3 ~ 5 분
2	T8	ALKALINE CLEAN	10 ~ 15 분
3	T9	IMM/SPRAY RINSE	3 ~ 5 분
4	T12	DEOXIDIZE	5 ~ 15 분
5	T13	IMM/SPRAY RINSE	3 ~ 5 분
6	T20	COLORLESS CONVERSION COATING	2 ~ 3 분
7	T21	IMM/SPRAY RINSE	3 ~ 5 분
8	T30	DRY	20 ~ 35 분

호이스트 생산시스템에서는 공정의 특성으로 인해 여러 가지 제약이 발생하게 된다. 제약으로는 화공처리공정의 다양화와 전체 완료 시간과 각 탱크 별 처리 하한 시간 및 상한 시간, 한번에 이동할 수 있는 부품의 수, 그리고 수시로 발생되는 추가적인 작업 등이 있다. 그러므로 이런 여러 가지 제약조건 하에서 완료 시간을 최소화 할 수 있는 효율적인 호이스트 생산 일정을 수립한다는 것은 매우 어렵다고 할 수 있다.

사례 기업의 호이스트 생산 일정계획은 작업자의 리더에 의해 수립하는데, 시스템을 사용하는 것이 아니라 오랜 작업 경험에 의해 수작업으로 작업 순서를 결정하고 있다. 호이스트 생산시스템의 일정계획은 1일 11개의 작업을 기준으로 수립하고 있다. 화공처리라인에서 동시에 처리할 수 있는 작업은 2~3개이며, 하루에 처리할 수 있는 작업의 수는 15~20개 이지만 위와 같이 11개의 작업을 기준으로 생산 일정계획을 수립하는 것은 2가지 이유 때문이다.

첫 번째는 여러 개의 작업을 동시에 수행하려면 숙련된 작업자가 있어야 한다. 숙련된 작업자는 각각의 화공처리공정에 대해서 방문해야 되는 탱크의 순서와 탱크에서 처리해야 되는 하한 시간과 상한 시간을 알고 있으며, 각 탱크에서 다른 탱크로 이동해야 되는 호이스트 이동 시간을 알고 있다. 그리고 실시간 상황에 따라 스스로 흐름이나 시간 등을 파악하여 조정할 수 있는 능력을 보유하고 있다. 만약 비 숙련자가 여러 개의 작업을 수행하게 되면 탱크의 하한 시간과 상한 시간을 준수하지 못해 불량이 발생하게 된다. 그렇기 때문에 숙련된 작업자만이 동시에 여러 개의 작업을 수행할 수 있다.

두 번째는 수시로 추가되는 작업이 발생한다는 것이다. 기계가공공정과 판금가공공정에서 방금 완료된 부품이라도 생산 일정의 긴급 여부에 따라 바로 화공처리라인으로 투입되어야 할 경우가 수시로 발생된다. 추가로 작업이 발생되면 호이스트 생산 일정을 재 수립하여야 한다. 현재 진행되고 있는 작업과 대기하고 있는 작업에서 추가로 발생되는 작업의 수행을 위해 일정을 재 수립 한다는 것은 쉬운 일이 아니다. 추가로 발생되는 작업의 수행을 위해서

는 투입 버퍼에서 대기하고 있는 기존의 작업 중 추가되는 작업의 수만큼 다음날로 미루어야 하는 문제가 생기는데, 이때 어떤 작업을 미루어야 할 건지에 대한 선택의 문제가 발생된다. 또한 다음날의 일정계획 역시 1일전에 계획하여 생산관리 담당에게 통보하게 되는데, 이미 다음날의 일정계획도 수립되어 있으므로 다음날로 미루더라도 다음날의 생산 일정계획까지 재 수립 해야 되는 문제가 추가적으로 발생하게 된다.

이런 2가지의 문제로 인해 하루에 15~20개의 작업을 수행할 수 있음에도 불구하고 11개의 작업으로 호이스트 생산 일정을 계획하게 된다. 이렇게 하면 비 숙련자가 작업하더라도 일정계획에 맞추어 작업을 진행할 수 있으며, 추가적인 작업이 발생하더라도 숙련도에 따라 몇 개의 작업을 더 진행할 수 있게 되므로 당일의 생산 일정계획을 재 수립 할 필요가 없게 된다. 마찬가지로 다음날의 일정계획 역시 재 수립 해야 되는 문제가 발생되지 않는다. 그러므로 리더는 생산 일정계획을 재 수립하는 문제를 회피하기 위해 생산할 수 있는 작업의 수보다 훨씬 적은 작업의 수로 호이스트 생산 일정계획을 수립하고 있는 것이다.

그러나 호이스트 생산 일정계획의 재 수립 문제를 회피하기 위해 수행할 수 있는 작업의 수보다 훨씬 적은 작업의 수로 계획을 수립한다는 것은 매우 비효율적이라 할 수 있다. 하루에 15~20 개의 작업을 수행할 수 있음에도 불구하고 호이스트 생산 일정계획을 제대로 수립하지 못해 11개의 생산 일정만 계획하게 되면 하루에 수행하는 작업이 어떤 때는 계획된 11개만 수행할 수도 있으며, 추가적으로 더 작업을 진행하더라도 하루에 수행할 수 있는 작업이 15개 이하로 되는 경우가 많기 때문이다. 그러므로 이런 2가지 문제를 해결하고 호이스트 생산시스템의 완료 시간을 최소화 할 수 있는 효율적인 호이스트 생산 일정계획 방안을 수립해야 된다.

다음 절에서는 호이스트 생산 일정계획 수립을 위해 기존 연구 내용에 대한 분석을 수행하도록 하겠다.

3. 문헌 연구

본격적인 연구에 앞서 사례 기업의 호이스트 생산시스템 일정계획 수립의 문제를 해결하기 위해 기존 연구 분석을 수행하였다. 호이스트 생산 일정에 대한 연구는 Phillips and Unger(1976)가 1대의 호이스트가 설치된 단일 품목의 제품을 생산하는 전기 도금 생산라인을 분석하면서 시작되었다. 이 연구 이후로 호이스트 일정 및 생산에 대한 최적화 알고리즘을 개발하기 위해 많은 연구들이 수행되었다.

Manier and Bloch(2003)에 따르면 호이스트 일정 문제는 정적 및 동적 일정 문제의 두 가지 범주로 분류 할 수 있다고 하였다. 정적인 문제는 모든 계획된 작업은 일정계획이 수립되기 전에 시스템 내에 준비되었다고 가정하고, 준비된 작업에 대해서만 예측 가능한 일정계획을 할당 할 수가 있다. 반면 동적인 일정 문제에서는 동적으로 작업 내용이 시스템 내에서 발생될 수 있으며, 그 동적인 작업 내용으로 인해 시스템은 언제든지 일정이 재 조정될 수 있어야 한다. 이 새로운 일정계획은 조정될 계획과 현재 진행중인 계획을 수용하여 최적의 계획을 결정하여야 한다고 하였다.

Phillips와 Unger(1976)가 단일 호이스트로 전기 도금 라인을 위한 최초의 MIP(Mixed Integer Programming) 모델을 소개한 이후로 주기적인 예측 일정계획을 포함하여 지난 40년 동안 다양한 정적 호이스트 일정계획 문제에 대해 광범위하게 연구되었다(Manier and Bloch, 2003).

대다수의 연구는 호이스트가 고정된 이동 순서 또는 주기를 반복하도록 계획된 주기적 호이스트 일정계획 문제에 중점을 두었다.

Shapiro & Nuttle(1988)은 분지한계 알고리즘을 개발하여 상대적으로 작은 문제에 대한 최적의 해를 발견하였다. Lei and Wang(1989)은 모든 작업이 동일한 항목으로 구성되어 있는 가장 단순한 경우에도 호이스트 일정계획

문제가 NP-hard 임을 보여 주었다. 그 이후로 그들은 2개 이상의 작업이 한 주기에서 처리 될 수 있는 확장된 주기적 문제에 대해 분지한계 알고리즘을 개발하였다.

Chen et al.(1998)은 2단계 분지한계 알고리즘을 제안하였는데 첫 번째 단계는 탱크에 명령을 할당하는 것이며, 두 번째 단계는 호이스트 순서를 순서화하는 것으로 2단계에 의해 알고리즘을 수행하였다.

반면 Lim(1997)은 실제 크기의 문제를 해결하기 위해 유전자 알고리즘을 제안하였다. Xu and Huang(2004)은 환경 문제와 생산성을 고려한 문제에 대하여 연구하였고, Kuntay et al.(2006)은 동일한 문제를 해결하기 위해 2단계 최적화 알고리즘을 개발하였다.

Liu et al.(2012)은 생산성 극대화와 에너지 절약 및 용수 사용의 최소화에 중점을 둔 세 가지 목적 함수의 문제점을 고려하였다. 주기의 문제에서는 하나의 제품만 생산되고, 호이스트는 미리 결정된 루틴을 반복하면 문제 해결의 어려움을 크게 줄일 수 있었지만 문제의 현실감 또한 감소하게 되었다. 현실감을 높이기 위해 예측 가능한 일정계획은 주기의 문제에서 하나의 단일 제품 대신 여러 제품을 고려하게 하였다.

예측 일정계획 문제에 대해서는 Hindi and Fleszar(2004)가 하나의 호이스트 생산시스템에서 다양한 제품의 일정계획 문제를 해결하기 위해서 제약 전파 휴리스틱을 제안하여 완료 시간을 최소화 하였다.

그리고 나중에 Paul et al.(2007)은 유사한 문제에 대해 적응 시간 창 휴리스틱을 개발하였다. 그러나 이 연구에서는 새로운 일정을 생성 할 때 화공처리 라인에서 진행 중인 작업을 고려하지 않았다. 그리고 화공처리를 위해 이전 작업공정에서 완료된 여러 부품의 작업이 시스템에 임의로 도착하기 때문에 도착하는 작업들을 반영하여 호이스트 일정을 재 조정해야 한다. 그러나 이 연구에서는 호이스트 일정계획 문제에서 동적인 특성을 고려하지 않았다.

동적인 문제에 대해 Yin and Yih(1992)는 하나의 패스 휴리스틱을 제안했고 나중에 Yih(1994)는 2 단계 휴리스틱을 개발하였다. 1 단계에서는 새 작업의 투입 시간이 업데이트되어 시스템 충돌이 발생하지는 않지만, 2 단계에서는 작업 시작 시간을 업데이트 하거나 새로운 작업의 시간 창을 처리하는 유연성을 찾아봐야 하므로, 탱크에서 진행 중인 작업과 새롭게 투입되는 작업 간에 충돌이 발생되기 때문에 2단계에서 발생하는 호이스트의 충돌 문제를 다루었다.

최근, Yan et al.(2014)은 교란을 고려한 동적 호이스트 일정계획 문제에 대해 2단계 분지한계 알고리즘을 제안하였다. 이 알고리즘에서는 주어진 일정계획과 수정된 일정계획 간의 작업 완료시간 편차로 정의된 제한된 제약하에서 일정계획 조정의 완료 시간을 최소화하는 것을 목표로 하였다.

동적 일정계획과 관련된 위의 연구들은 재조정이 이루어질 때 여러 작업을 반영하지 않고, 단 하나의 새로운 작업만을 고려하였다.

동적 일정계획에 대해서 여러 연구에 대한 연구 분석 중에 사례 기업과 유사한 상황의 문제를 다룬 논문을 발견하였는데 Zhao et al.(2013)의 불확실성 하에서 다단식 자재 처리 프로세스를 위한 실시간 동적 호이스트 일정계획에 관한 연구이다.

그들은 다양한 가공공정에서 상이한 생산방식으로 제작된 여러 유형의 작업들이 생산 라인에서 동시에 지속적으로 투입되는 경우 진행중인 작업의 일정을 철저히 조사하여 작업중인 모든 작업의 운영 가능성을 보장하고 가능한 한 생산량을 극대화해야 한다고 하였다. 그리고 새로 추가되는 작업의 불확실성 즉, 도착 시간, 유형, 작업 방법 및 새로운 작업의 수가 생산 라인에 오기 전까지 완전히 알려지지 않고 예측할 수 없는 경우에는 호이스트 일정계획이 더욱 복잡해 진다고 하였다. 이러한 상황에서 일정계획의 변경은 각 작업에 대한 처리 시간 제약을 위반하지 않고 작업 중에 수행해야 하므로 일정계획 변경을 위한 계산 시간은 새 일정 도착 후 대기중인 작업과 진행중인 작업을 고

려하여 모든 작업에 실시간으로 적용 가능성을 보장할 수 있는 빠른 시간 안에 이루어져야 한다고 하였다. 그들은 이러한 불확실성 하에서 생산성 극대화를 위해 진행중인 작업을 포함하는 새로운 일정 생성을 목표로 하는 실시간 동적 호이스트 일정계획을 개발하였다.

이 연구에서 여러 유형의 다양한 작업이 생산라인에서 동시에 처리되는 경우 모든 작업의 운영 가능성을 보장하여야 한다고 하였다. 마찬가지로 본 연구의 사례기업에서 여러 가지 제약조건으로 인해서 다양한 화공처리공정의 작업들에 대한 호이스트 생산 일정계획 수립에 어려움을 가지고 있으며, 전체 완료 시간을 최소화 할 수 있는 효율적인 생산 일정계획을 수립해야 하므로 이 연구 내용과 유사한 상황의 문제를 다룬다고 할 수 있겠다.

또한 사례기업에서도 수시로 추가되는 새로운 작업에 대한 문제가 발생되어 일정계획의 재 수립에 어려움이 있는데 이 연구에서도 새로 추가되는 작업에 대한 불확실성과 일정계획의 재 조정에 대한 문제를 해결하고자 실시간 동적 호이스트 일정계획을 개발하였다.

그러므로 본 연구의 사례 기업에서 가지고 있는 상황과 유사한 문제를 이 연구에서도 가지고 있으므로 이 연구를 먼저 분석하는 것이 본 연구에 많은 도움이 될 것이라고 생각한다.

다음 절에서는 실시간 동적 호이스트 일정계획 모형에 대해 알아보고자 한다. 이 연구의 분석을 통해 RDHS 가 어떤 문제를 다루고 어떻게 구성되어 있는지 알아보고, 간단한 예제를 사용하여 결과 분석을 통해 RDHS 문제점이 무엇인지, 본 연구의 목적인 사례기업의 문제에 대해 적용할 수 있는지를 알아 보고자 한다.

4. RDHS 모형

4.1 RDHS 개요

앞서 언급한 것처럼 호이스트 생산시스템의 일정계획에 대한 연구는 Phillips and Unger(1976)에 의해 시작되었는데 그들은 사이클이라고 하는 고정된 이동 순서를 반복적으로 계획하는 순환 호이스트 일정계획(CHS, Cyclic Hoist Scheduling)을 연구하였다.

실제로 수행되는 호이스트 생산시스템의 일정계획은 CHS에서 가정하는 것보다 훨씬 더 복잡하게 이루어져 있다. 호이스트 일정계획은 새로운 작업이 임의로 시스템에 적용되는 상황을 처리할 수 있어야 하는데 이를 위해 호이스트 생산시스템은 역동적인 방식 즉, 동적 호이스트 일정계획(DHS, Dynamic Hoist Scheduling)에서 수행되어야 한다. Yin and Yih(1992) 와 Yih(1994)는 생산 라인에 있는 작업에 대한 호이스트 일정이 변경되지 않도록 연구하였는데, 이 연구에서는 기존 작업에 대한 일정은 변경되지 않으므로 새 작업에 대한 호이스트 일정만 추가하면 되었다. 이 방법은 일정 개발의 복잡성은 줄일 수 있지만 일정계획의 최적 솔루션은 보장할 수 없었다.

이후에도 여러 연구들이 진행되었지만 최적의 해를 제공해주지 못하였다. Zhao et al.(2013)은 이런 문제를 해결하기 위해 실시간 동적 호이스트 일정계획이 필요하다고 하였다. RDHS는 무작위로 생산 라인에 도착하는 다양한 방식의 여러 유형의 새로운 작업들을 동적으로 계획하여 진행중인 작업을 포함한 모든 작업 처리에 대한 일정계획의 재 조정이 이루어져야 하며, 실시간으로 적용 가능성을 보장하기 위해 새로운 일정을 최적의 상태로 신속하게 처리하고 원활하게 구현하여 생산 라인의 현재 호이스트 일정 및 처리 조건과 연결해야 한다고 하였다. 그리고 RDHS는 새로운 일정에 대해서 재 조정을 위한 솔루션의 처리 시간이 고려되어야 한다고 하였다.

이 연구에서 RDHS 의 주요 문제점을 다음과 같다고 하였다.

(a) 작업 처리방법이 다른 여러 유형의 작업 : 이전 연구의 생산 라인은 한 가지 유형의 작업만 처리하였다. RDHS는 하나의 생산 라인에서 다른 처리 방법을 사용하는 여러 유형의 작업을 처리해야 한다.

(b) 임의로 도착하는 새로운 작업 : 새로운 작업이 처리를 위해 도착하면 현재 호이스트의 일정이 업데이트 되어야 한다. 새로 투입된 작업을 포함하여 모든 작업을 실시간으로 처리하기 위해서는 새로운 일정을 생성해야 한다. 이 새 일정은 현재 일정에 원활하게 연결되어 진행되어야 한다.

(c) 실시간 적용성을 갖춘 신속한 재 조정 : 호이스트 일정의 느린 재 조정은 최적화 기회를 놓치거나 생산속도를 낮출 수 있으며, 작업 처리 시간 위반으로 인해 제품의 품질 문제를 야기할 수 있다. 재 조정을 위한 최적의 솔루션은 모든 프로세스의 제약 조건 하에서 신속하게 처리되고 원활하게 적용되어야 실시간 적용 가능성을 보장 할 수가 있다.

위의 3가지 문제가 RDHS 문제를 모델링하고 해결하는 것을 매우 어렵게 만든다고 하였다. RDHS의 주요 문제점은 예기치 않은 작업 도착 시간, 유형, 작업 방법 등의 불확실성을 효과적으로 처리하는 방법과 실시간 응용 프로그램을 위한 최적의 솔루션을 효율적으로 얻고 채택하는 방법이라고 하였다.

Zhao et al.(2013)은 새로운 작업의 불확실성과 실시간 일정계획의 최적화 및 적용 가능성을 고려한 새로운 RDHS 방법을 제안하였다. 이 연구에서 일정계획 및 재 조정은 작업 간의 원활한 연결을 위한 재 초기화 알고리즘과 최적의 호이스트 일정계획을 얻기 위해 혼합정수선형 프로그래밍(MILP, Mixed Integer Linear Programing) 모델을 포함하였다. RDHS 방법론은 다양한 작업 처리방법, 다중 작업, 다중 용량 탱크, 다양한 작업 처리 시간, 새로운 작업에 대한 최적의 처리와 같은 다단계 자재 처리 프로세스의 일정계획 문제를 해결하였고, 개발된 방법론의 효용성은 사례 연구를 통해 입증하였다.

4.2 문제 설명

금속의 도금이나 코팅 등의 표면 처리는 화공처리공정에서 다양한 작업을 처리하기 위해 다수의 탱크로 구성된 생산 라인에서 수행된다. 투입 버퍼에서 작업을 선택하여 처리 공정에 따라 해당 탱크로 이동하고 순차적으로 탱크를 방문한 뒤 마지막으로 완료 버퍼에 도착하면 작업 처리는 완료하게 된다. 그리고 새로운 작업이 도착했을 때 진행중인 작업을 고려하여 일정이 재 조정되어야 한다. 이런 RDHS 문제를 이해하기 위한 용어의 설명은 다음과 같다.

1) 작업 처리 방법

작업 처리 방법은 각 처리 단계에 대한 처리 순서 및 상주 시간 창을 포함하며 작업에 대한 처리 요구 사항을 제공하여 준다. 다양한 유형의 작업에 대한 작업 처리 방법은 각기 다르게 수행된다. (본 논문의 사례 기업은 16개의 서로 다른 화공처리공정이 존재하였다.)

2) 탱크 처리 용량

호이스트 생산라인에 존재하는 탱크에서 동시에 처리 할 수 있는 최대 작업의 수를 의미한다. 일반적인 생산라인에서 각 처리 탱크는 특정 작업을 처리할 수 있는 용량을 가진다. 대부분의 탱크는 한번에 하나의 작업을 처리 할 수 있지만 일부 탱크는 다중 작업을 처리할 수 있는 용량을 가지고 있다. 즉, 어떤 탱크는 긴 작업 처리 시간이 소요되거나 사용이 빈번하여 작업의 대기가 많이 이루어지는 등의 필요에 따라 탱크를 2개 이상의 용량으로 구성하여 여러 개의 작업을 동시에 처리 할 수 있게 된다.

3) 자유 이동, 로드 된 이동 및 유휴 대기

호이스트가 작업 없이 빈 호이스트로 이동 할 때를 자유 이동이라고 한다. 그렇지 않고 호이스트가 작업을 이동하게 되면 로드 된 이동이라고 한다. 유휴 대기란 호이스트가 아무런 활동도 하지 않고 작업이 완료될 때까지 어떤 탱크 위에 머무르는 것을 의미한다.

4) 주어진 정보

① 고정 된 처리 탱크와 하나의 호이스트를 갖춘 하나의 생산라인

② 각 처리 탱크의 작업 용량

③ 호이스트 자유 이동의 속도는 고정되어 있다. 즉, 자유 이동의 호이스트 이동 시간은 호이스트가 이동하는 공간 거리에 의해 결정된다.

④ 호이스트의 로드 된 이동 시간은 호이스트가 보유하고 있는 작업을 탱크에서 꺼내고 내리는 시간과 호이스트를 이동시키는 위치에 따라 결정된다. 즉, 로드 된 이동은 작업을 장착, 분리하는 시간이 소요되어 동일한 거리에 대한 자유 이동보다 느리다.

⑤ 호이스트 위치 및 진행중인 작업 처리 상태의 초기 상태

5) 불확실성 정보

생산 라인에서 발생하는 새로운 작업의 도착 시간, 작업 방법 및 작업의 수는 알 수 없다. 새로운 작업이 도착하면 해당 작업의 정보를 알 수 있다.

6) 결정 할 정보

① 새로 예정된 작업에 의해 일정계획이 재 조정 될 때 일정계획이 시행되어야 하는 시작 시점

② 호이스트 자유 이동, 로드 된 이동 및 유휴 대기 시간의 세부 일정

③ 각 탱크의 작업 처리 시간

④ 진행중인 모든 작업을 완료하기 위한 일정 변경의 총 시간.

7) 가정

① 새로운 작업은 적재 구역(투입 버퍼)을 통해 생산 라인에 투입하고 하역 구역(완료 버퍼)을 통해 생산 라인을 떠난다. 투입 버퍼 및 완료 버퍼는 동일한 데크를 공유한다.

② 투입 버퍼 및 완료 버퍼에서는 작업 용량의 제한은 없다.

③ 모든 작업 방법에서 각 처리 탱크의 작업 처리 제한 시간 즉, 하한 시간 및 상한 시간은 위반 할 수 없다.

4.3 수리적 모형

Tian et al. (2013)과 Feng et al. (2015)은 Zhao et al.(2013)의 RDHS 모형을 약간 수정하여 필요한 제약 식의 수를 줄일 수 있었다. 본 연구에서는 가장 진보된 형태의 모형을 소개하는데, 사용되는 기호는 기존의 연구와 다른 형태로 아래와 같이 재정의 하였다. 아래에서 기호들은 문제를 풀기 전에 이미 값이 정해져 있는 파라미터와 문제를 풀면서 값을 찾아야 하는 변수 등 두 그룹으로 나누어 설명하도록 한다.

<Parameters>

i, j : 작업을 나타내는 지표 (i, j = 1, 2, ..., n)

r, s : 탱크를 나타내는 지표 (r, s = 1, 2, ..., m)

q_i : 시간 0 에서 작업 i 의 위치 (q_i = 1, 2, ..., m)

e_i : 시간 0 의 q_i 에서 작업 i 의 경과 시간

h : 시간 0 에서 호이스트 위치 (h = 1, 2, ..., m)

c_r : 탱크 r 의 용량

$\underline{p_r}$: 탱크 r 의 처리 하한 시간

$\overline{p}_r$: 탱크 r 의 처리 상한 시간

$\alpha_{i,r}$: 탱크 r 이후에 작업 i 가 이동해야 하는 탱크

$\delta_{r,s}$: 탱크 r 에서 탱크 s 까지 빈 호이스트의 이동 시간

$\theta_{r,s}$: 탱크 r 에서 탱크 s 까지 로드 된 호이스트의 이동 시간

M : 매우 큰 양수

ε : 매우 작은 (다시 말해 0에 아주 가까운) 양수

<Decision variables>

T : 시스템의 완료 시간

$te_{i,r}$: 작업 i 를 탱크 r 에 내려놓는 시간

$ts_{i,r}$: 작업 i 를 탱크 r 에서 들어 올리는 시간

$w_{i,r,j,s}$: 호이스트가 탱크 r 에서 작업 i 를 들어올린 후 탱크 s 에서 작업 j 를 들어올리면 1, 그렇지 않으면 0

$u_{i,j,r}$: 작업 j 가 탱크 r 을 떠나기 전에 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 0

$v_{i,j,r}$: 작업 j 가 탱크 r 에 도착한 후에 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 0

$x_{i,j,r}$: 작업 j 가 탱크 r 에 머물러있을 때 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 다른 값

위 표기법을 이용하여 RDHS 모형들을 정리한 MILP 모형은 다음과 같다.

Minimize T (1)

subject to

$te_{i,m} \leq T$ for all i (2)

$te_{i,q_i} = 0$ for all i (3)

$te_{i,\alpha_{i,r}} = ts_{i,r} + \theta_{r,\alpha_{i,r}}$ for all i, r with $\alpha_{i,r} \neq 0$ (4)

$ts_{i,q_i} \geq \delta_{h,q_i}$ for all i (5)

$ts_{i,r} \geq te_{j,\alpha_{j,s}} + \delta_{\alpha_{j,s}r} - M(1 - w_{i,r,j,s})$

for all i, r, j, s with $\alpha_{i,r} \neq 0$, $\alpha_{j,s} \neq 0$ and $(\mathrm{i},\mathrm{r}) \neq (j,s)$ (6)

$w_{i,r,j,s} + w_{j,s,i,r} = 1$

for all i, r, j, s with $\alpha_{i,r} \neq 0$, $\alpha_{j,s} \neq 0$ and $(\mathrm{i},\mathrm{r}) \neq (j,s)$ (7)

$-Mu_{i,j,r} + \varepsilon \leq ts_{j,r} - ts_{i,r} \leq M(1 - u_{i,j,r})$

for all i, j, r with $\alpha_{i,r} \neq 0$, $\alpha_{j,r} \neq 0$, and $i \neq j$ (8)

$$-Mv_{i,j,r} + \varepsilon \leq te_{i,r} - ts_{j,r} \leq M(1 - v_{i,j,r})$$

for all i, j, r with $\alpha_{i,r} \neq 0$, $\alpha_{j,r} \neq 0$, and $i \neq j$ (9)

$$u_{i,j,r} + v_{i,j,r} - 1 = x_{i,j,r}$$

for all i, j, r with $\alpha_{i,r} \neq 0$, $\alpha_{j,r} \neq 0$ and $i \neq j$ (10)

$$\sum_{j \neq i, \alpha_{j,r} \neq 0} x_{i,j,r} \leq c_r - 1 \text{ for all } i, r \text{ with } \alpha_{i,r} \neq 0 \text{ and } i < r < m \quad (11)$$

$$\underline{p}_{q_i} \leq ts_{i,q_i} - te_{i,q_i} + e_i \leq \overline{p}_{q_i} \text{ for all } i \text{ with } q_i > 1 \quad (12)$$

$$\underline{p}_r \leq ts_{i,r} - te_{i,r} \leq \overline{p}_r \text{ for all } i, r \text{ with } r \neq q_i \text{ and } \alpha_{i,r} \neq 0 \quad (13)$$

$w_{i,r,j,s}$: 모든 i, r, j, s 에 대해 0/1 정수

$u_{i,j,r}$, $v_{i,j,r}$: 모든 i, j, r 에 대해 0/1 정수

이 모델에서 식 (1)은 이 문제의 목적이 완료 시간을 최소화 하는 것이라는 걸 보여주고 있다. 식 (2)는 모든 작업의 완료 시간과 전체 작업의 완료 시간 사이의 관계를 보여준다. 식 (3)에서는 변수 $te_{i,r}$의 값을 초기화 해준다. 식 (4)는 하나의 작업을 시작한 시간($ts_{i,r}$)과 그 작업을 완료하는 시간($te_{i,\alpha_{i,r}}$)과의 관계를 보여준다. 식 (5)는 변수 $ts_{i,r}$의 초기 조건을 지정해준다. 식 (6)과 식 (7)은 식 (4)와 반대로 하나의 작업을 마친 시간($te_{j,\alpha_{j,s}}$)이 주어졌을 경우 그 후속 작업을 시작할 수 있는 시간($ts_{i,r}$)을 제한하고 있다. 식 (8)에서 (10) 은 $x_{i,j,r}$ 값을 정해주기 위한 식들이다. 이 식들에 의하면 $u_{i,j,r}$ 와 $v_{i,j,r}$ 가 모두 1이어야 $x_{i,j,r}$ 값이 1이 되는데, 그 의미는 작업 j 가 탱크 r 에 머물러 있을 때 작업 i 가 탱크 r 에 도착할 경우에만 $x_{i,j,r}$ 값이 1이라는 것이다. 이 $x_{i,j,r}$ 값을 이용해 식 (11)에서 각 탱크에 대한 용량을 제약하였다. 식 (12)와 식 (13)은 각 작업이 각 탱크에서 머무는 시간에 대한 제약을 보여준다.

이제 이 모형을 사용해 문제를 해결하기 위해 간단한 예제를 사용하기로 한다. 이 예제는 Zhao et al.(2013)의 예제를 기초로 하고 처리 시간 및 호이스트 이동 시간 등의 수치들은 본 연구의 사례 기업의 상황을 반영하여 현실적으로 수정한 것이다. 사용되는 데이터는 다음과 같다.

m = 8, n = 5, h = 1, q_i = (7, 6, 4, 3, 1), e_i= (33, 3, 1, 4, 12),

c_r= (∞ , 1, 1, 1, 2, 1, 1, ∞), $\underline{p}_r$= (0, 3, 15, 1, 40, 5, 30, 0),

$\overline{p}_r$= (0, 10, 25, 5, 60, 15, 50, 0),

$$\alpha_{i,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 5 & 6 & 7 & 8 & 0 \\ 0 & 0 & 4 & 5 & 6 & 7 & 8 & 0 \\ 2 & 3 & 4 & 5 & 8 & 0 & 0 & 0 \end{bmatrix},$$

$$\delta_{r,s} = \begin{bmatrix} 0 & 0.3 & 0.6 & 0.9 & 1.2 & 1.5 & 1.8 & 2.1 \\ 0.3 & 0 & 0.3 & 0.6 & 0.9 & 1.2 & 1.5 & 1.8 \\ 0.6 & 0.3 & 0 & 0.3 & 0.6 & 0.9 & 1.2 & 1.5 \\ 0.9 & 0.6 & 0.3 & 0 & 0.3 & 0.6 & 0.9 & 1.2 \\ 1.2 & 0.9 & 0.6 & 0.3 & 0 & 0.3 & 0.6 & 0.9 \\ 1.5 & 1.2 & 0.9 & 0.6 & 0.3 & 0 & 0.3 & 0.6 \\ 1.8 & 1.5 & 1.2 & 0.9 & 0.6 & 0.3 & 0 & 0.3 \\ 2.1 & 1.8 & 1.5 & 1.2 & 0.9 & 0.6 & 0.3 & 0 \end{bmatrix}, \quad \theta_{r,s} = \begin{bmatrix} 0 & 2.3 & 2.6 & 2.9 & 3.2 & 3.5 & 3.8 & 4.1 \\ 2.3 & 0 & 2.3 & 2.6 & 2.9 & 3.2 & 3.5 & 3.8 \\ 2.6 & 2.3 & 0 & 2.3 & 2.6 & 2.9 & 3.2 & 3.5 \\ 2.9 & 2.6 & 2.3 & 0 & 2.3 & 2.6 & 2.9 & 3.2 \\ 3.2 & 2.9 & 2.6 & 2.3 & 0 & 2.3 & 2.6 & 2.9 \\ 3.5 & 3.2 & 2.9 & 2.6 & 2.3 & 0 & 2.3 & 2.6 \\ 3.8 & 3.5 & 3.2 & 2.9 & 2.6 & 2.3 & 0 & 2.3 \\ 4.1 & 3.8 & 3.5 & 3.2 & 2.9 & 2.6 & 2.3 & 0 \end{bmatrix}.$$

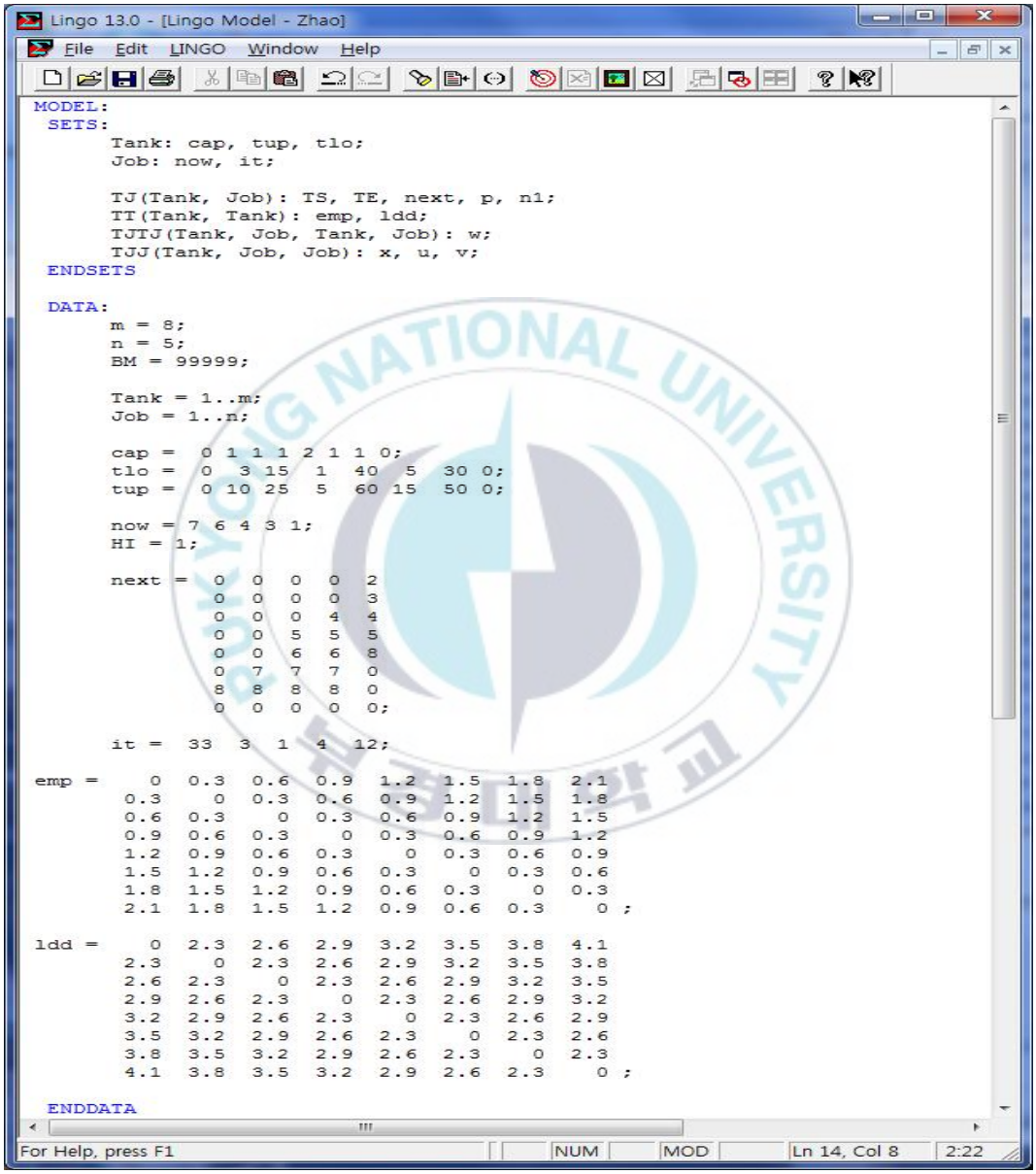
이 예제의 데이터를 설명해보면 대상 시스템은 버퍼를 포함해 8개의 탱크로 되어있는데 투입 버퍼 1개와 완료 버퍼 1개 그리고 6개의 화학처리 탱크로 이루어져 있고, 계획 시점 현재 5개의 작업이 시스템에 존재한다. 호이스트는 현재 1번 탱크(투입 버퍼) 위에 위치하고 있다. 탱크의 용량은 두 버퍼(1번과 8번 탱크)의 경우 무한하고, 5번 탱크(투입 버퍼를 제외하면 실제로는 4번 탱크)는 2개, 나머지 5개의 탱크는 모두 1개이다. 각 작업의 현재 위치를 보면 1, 2, 3, 4 번 작업은 공정에 투입되어 7, 6, 4, 3 번

탱크에서 각각 처리 중이고, 5번 작업은 아직 공정에 투입되지 않고 1번 탱크인 투입 버퍼에서 대기 중이다. 각 작업이 현재의 탱크에 도착 한지는 각각 33분, 3분, 1분, 4분, 12분이 경과하였다. 각 탱크에서의 처리 시간은 $\underline{p}_r$와 $\overline{p}_r$에 주어져 있다. 예를 들어 5번 탱크 작업은 하한 시간과 상한 시간이 각각 40분과 60분이므로 40분이 지나면 탱크에서 꺼낼 수 있으며 60분이 지나기 전에는 탱크 작업이 완료되어야 함을 알 수 있다. $\alpha_{i,r}$ 행렬을 보면 각 작업의 남은 공정들을 알 수 있다. 예를 들어 3번 작업은 현재 위치한 4번 탱크의 작업이 끝나면 5, 6, 7 번 탱크를 거쳐 마지막으로 완료 버퍼인 8번 탱크로 이동하게 된다. 마지막으로 $\delta_{r,s}$ 및 $\theta_{r,s}$ 행렬을 보면 빈 호이스트의 이동 시간과 로드 된 경우의 호이스트 이동 시간을 알 수 있다. 호이스트 이동 시간은 빈 이동일 경우 바로 옆 탱크로 이동하는데 18초(0.3분)가 걸리고, 로드 된 호이스트 이동의 경우에는 장착 및 분리 작업을 위해 2분이 추가되어야 한다. 즉, 바로 옆 탱크로 빈 호이스트 이동 시는 18초가 소요되고 로드 된 이동의 경우 장착, 이동 및 분리에 소요되는 시간은 2분 18초가 된다는 것이다.

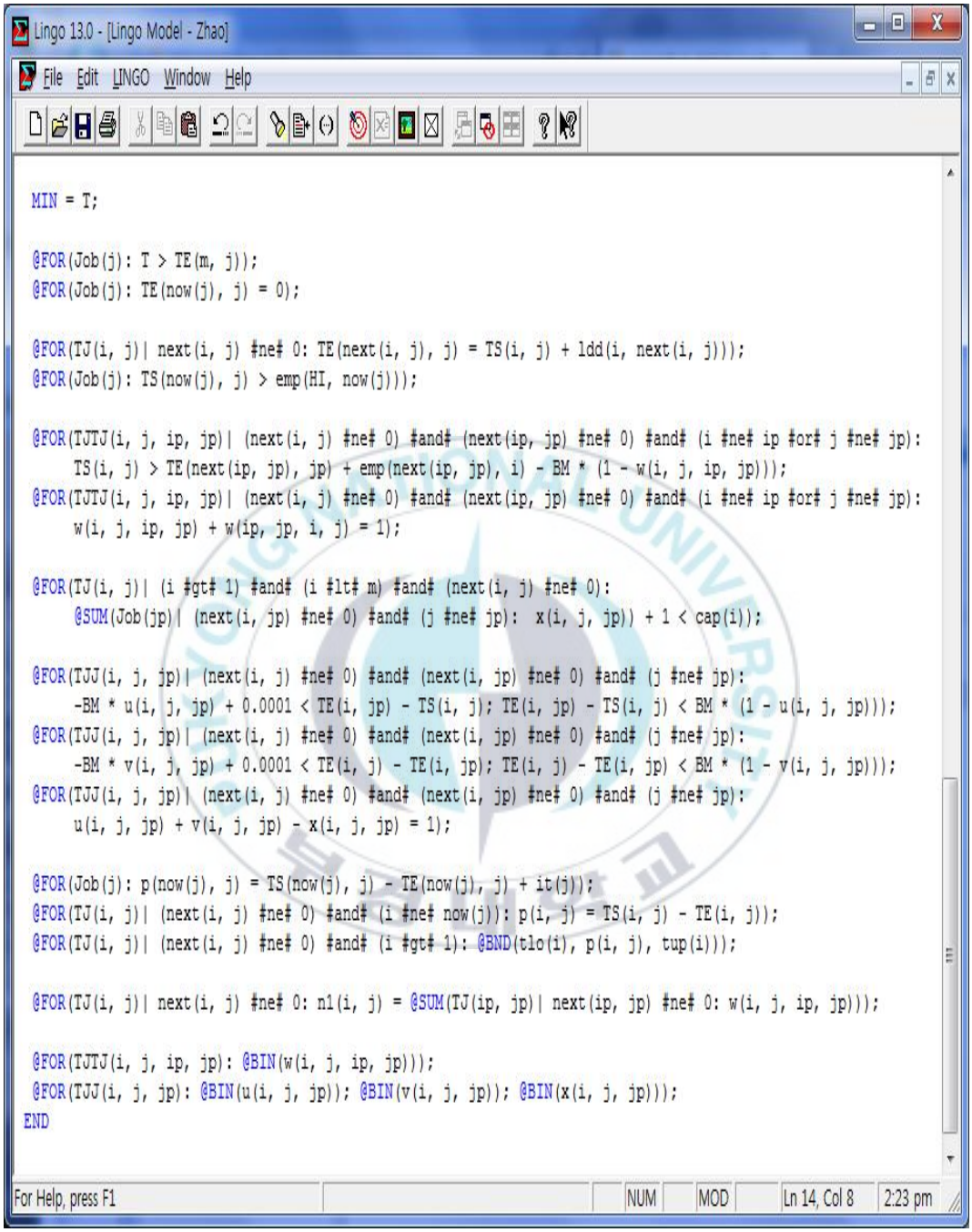
이 문제의 해는 $\alpha_{i,r}$ 행렬에서 0 이 아닌 값을 갖는 17 개의 요소들에 대한 처리 순서로 표시할 수 있다. 요소들을 살펴보면 1번 작업이 1개, 2번 작업은 2개, 3번 작업은 4개, 4번 작업은 5개, 5번 작업은 5개가 포함되어 있다. 따라서 가능(feasible) 및 불가능(infeasible) 해를 포함하여 해의 총수는 이 17개의 작업 번호 (1, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5)를 일렬로 세우는 방법의 수인 17! / 1! 2! 4! 5! 5! 개가 된다. 즉, 문제의 난이도는 $\alpha_{i,r}$ 행렬에 0 이 아닌 요소가 몇 개가 있는지에 따라 결정되는 것이다.

최적화 소프트웨어 LINGO를 사용해 MILP 모형을 이 예제에 적용하여 [그림 2-6]과 [그림 2-7]와 같이 최적 해를 구해보았다. 4GB RAM과 2.60GHz G620 CPU가 장착된 PC에서 최적 해를 구하는데 소요된 시간은 0.01초 미만으로 이 예제의 경우 매우 빠른 시간에 해를 구할 수 있었으며,

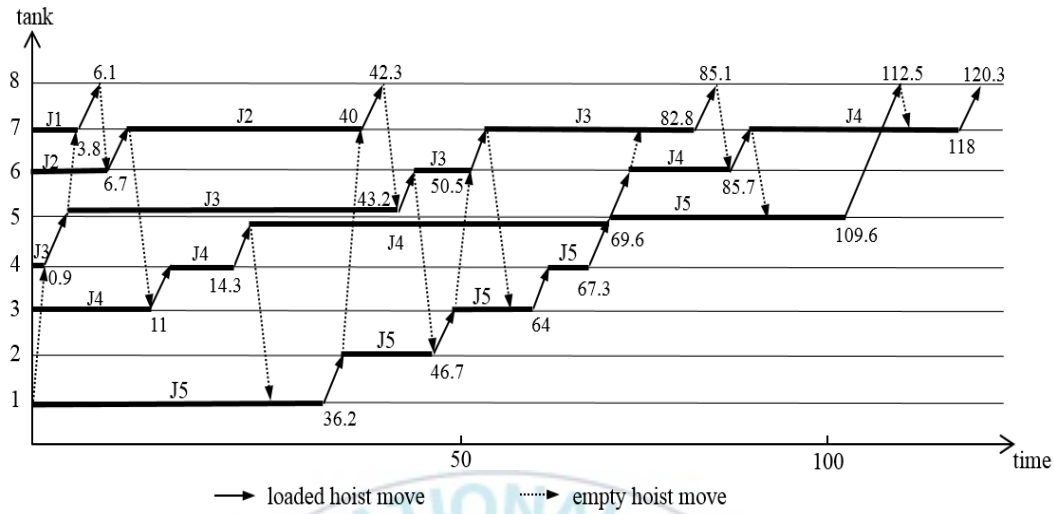
그 결과 최적의 작업 처리 순서는 [그림 2-8]과 같이 (3, 1, 2, 4, 4, 5, 2, 3, 5, 3, 5, 5, 4, 3, 4, 5, 4)이었다. 그 때의 완료 시간은 120.3 분이었다.



[그림 2-6] 예제의 LINGO 프로그램 데이터 부분



[그림 2-7] 예제의 LINGO 프로그램 모형 부분



[그림 2-8] 예제 문제의 최적 일정

RDHS 모형이 현실 문제에 어느 정도 적용 가능한지를 조사하기 위해 탱크의 수가 12인 경우(두 개의 버퍼와 10개의 화학 처리용 탱크)에 대한 예제를 만들어 LINGO를 사용해 계산하여 보았다.

그 결과 작업의 수가 7개인 경우는 8초 만에 최적해를 구할 수 있었고, 작업의 수가 8개인 경우에는 3분 9초, 9개인 경우는 1시간 55분 17초가 소요되었다. 그리고 작업의 수가 10개인 경우에는 시스템을 강제로 종료하기까지인 5시간 동안 해를 구하지 못하였다.

따라서 RDHS 방법론으로 작은 크기의 문제에 대해 해를 찾을 수 있었으나 탱크의 수가 사례기업과 같이 큰 크기의 문제에 대해서는 해를 찾지 못하였다. 그래서 30개의 탱크로 구성된 사례 기업의 호이스트 생산시스템에서는 적용할 수가 없어, 해의 품질이 어느 정도 저하되더라도 현실적인 시간 안에 해를 찾을 수 있는 호이스트 생산시스템 일정계획 방법론의 개발이 필요하게 되었다.

5. 요약

이 장에서 연구한 Zhao et al.(2013)의 실시간 동적 호이스트 일정계획은 생산라인에서 연속적으로 임의로 도착하는 여러 가지 유형의 작업을 신속하게 처리할 수 있도록 최적의 일정계획을 실시간으로 생성할 수 있었다. 그들은 신규로 들어오는 작업의 불확실성과 실시간 일정계획의 최적화 및 적용 가능성을 고려한 새로운 최적화된 RDHS 방법론을 개발하였다. RDHS는 임의로 도착하는 작업의 도착 시간, 처리 유형, 작업 방법 및 작업 수와 같은 불확실성으로 인해 발생할 수 있는 복잡한 상황에 대해 대처할 수 있었다. RDHS 방법론은 다양한 작업처리 방법, 다중 용량 탱크, 다양한 작업 처리 시간, 새로운 작업에 대한 최적의 처리와 같은 다단계 자재 프로세스의 일정계획 문제를 해결할 수 있었으며, 개발된 방법론의 효용성은 다양한 사례 연구를 통해 입증하였다.

이 연구가 본 연구의 사례 기업에서 가지고 있는 여러 가지 제약 조건과 유사한 상황의 문제에 대해 연구한 사례였다. 여기에서는 8개의 탱크와 5개의 작업으로 비교적 크기가 작은 문제에 대해 사례 분석을 실시하여 빠른 시간 안에 확실한 해를 보장할 수 있었다. 그러나 이 모형은 현실적인 규모인 10개 이상의 탱크와 10개 이상의 작업에 대한 문제에 대해서는 적정 시간 내에 해를 제시해주지 못하였다. 본 연구의 사례 기업은 30개의 탱크에서 16개의 다른 작업 유형으로 20개 정도의 작업을 수행하고 있으므로 RDHS 방법론은 사례 기업에 적용할 수가 없었다.

따라서 다음 장에서는 이 연구와 유사한 상황의 문제에 대해서 사례 기업과 같은 현실적인 규모의 경우에도 빠른 시간 안에 양호한 품질의 해를 제공해 줄 수 있는 분지한계 기반의 휴리스틱 방법론을 개발하여 호이스트 생산시스템의 효용성을 증대하고자 한다.

III. 분지한계 기반의 휴리스틱 일정계획

1. 개요

본 장에서는 Zhao et al.(2013), Tian et al.(2013), Feng et al.(2015) 등이 MILP 형태로 제시한 RDHS 모형을 좀 더 현실적인 시간 안에 해결할 수 있는 휴리스틱 알고리즘을 제안하였다. 앞서 제 2장의 말미에서 지적한 것처럼 MILP 모형은 현실적인 규모의 문제에 대해서는 적정시간 내에 해를 찾지 못하였다.

사례기업과 같이 30개의 탱크에서 16개의 서로 각기 다른 화공처리공정이 존재하고 하루에 15~20개 정도의 작업을 처리하는 현실적인 규모의 문제를 고려할 때 합리적인 계산시간에 최적의 해를 제공해 줄 수 있는 분지한계 기반의 휴리스틱 일정계획 방법을 제안하려고 한다. 제안하는 방법이 현실적인 규모의 문제를 해결할 수 있다면 호이스트 생산시스템의 생산 효율성은 증대될 것이라 생각된다.

본 장의 나머지 부분은 다음과 같이 구성하였다. 제 2절에서는 휴리스틱 방법론의 연구에 앞서 대상 문제에 대한 명확한 정의를 내리고자 한다. 그런 다음 제 3절에서는 본 연구에서 제안하고자 하는 휴리스틱 방법론에 대해 소개한다. 휴리스틱 방법론에는 분지 정책과 하한 계산하는 방법, 그리고 알고리즘으로 나누어 설명하였다.

제 4절 수치실험에서는 제안된 휴리스틱 방법의 성능을 평가하기 위해 2장의 예제로 실험을 수행하고, 최적화 소프트웨어 LINGO와 휴리스틱의 실험 결과를 비교 분석해 보았다. 그리고 현실적인 규모인 탱크 30개와 처리 대상 작업의 수를 10개로 설정하여 휴리스틱 방법으로 실험을 수행하고 분석해 보았다. 마지막으로 제 5절에서는 이 장을 요약하고 추후 연구과제를 제시하였다.

2. 문제 정의

앞에서 설명하였듯이 본 장에서는 2장의 실시간 동적 호이스트 일정계획과 같은 문제를 다룬다. 문제를 명확히 하기 위해 2장 4.2절에서 설명한 RDHS 용어를 바탕으로 필요한 핵심 가정들을 다음과 같이 정리하였다.

(1) 처리대상 작업의 수(n)는 유한하며 알려져 있다. 이 중 일부는 이미 생산 라인에 투입되어 공정이 진행 중이고 나머지는 투입 버퍼에서 처리 대기 중이다.
(2) 탱크 수(m)는 유한하다. 첫 번째 탱크는 투입 버퍼이고, m 번째 탱크는 완료 버퍼이다.
(3) 각 작업을 처리하기 위한 공정 순서와 탱크들은 정해져 있으며, 각 작업들의 공정 순서와 처리하는 탱크는 서로 다를 수 있다.
(4) 각 작업이 각 탱크에서 처리되는 시간은 탱크와 작업에 따라 달라지는 지정된 시간 창 내에 있어야 한다. 즉, 각 작업에 대한 탱크 처리 시간은 하한 시간과 상한 시간 사이에 있어야 한다.
(5) 각 탱크(혹은 버퍼) 사이의 물자이동은 하나의 작업만 처리할 수 있는 한 대의 단일 호이스트에 의해 수행된다.
(6) 호이스트는 한 번에 하나의 작업만을 처리하며, 일단 하나의 목적지를 향해 이동을 시작하면 중간에 멈추지 못한다.
(7) 문제의 목적은 시스템에 존재하는 모든 처리대상 작업을 완료 버퍼로 이동하기까지의 전체 완료 시간을 최소화하는 것이다.
(8) 하나의 탱크에서 일단 하나의 액침 작업이 시작되면 그 작업이 종료되기 전에는 중단할 수 없다.
(9) 각 탱크와 호이스트는 고장이 없고 액침 공정의 셋업 시간은 따로 고려하지 않는다.

(10) 각 탱크 사이의 호이스트 이동 시간은 빈 상태와 로드 된 상태의 두 가지 형태로 나누어지며 그 값은 알려진 상수로 나타낸다.

(11) 각 탱크는 동시에 처리 가능한 작업 수로 표시되는 용량을 가지며 그 값은 상수로 나타낸다. 액침 작업을 수행하지 않는 투입 버퍼와 완료 버퍼는 그 값이 무한대이다.

(12) 일정계획은 새로운 작업이 도착할 때마다 수행된다. 그때의 일정 시간은 0으로 설정한다.

다음절에서는 이 절에서 정의된 가정을 바탕으로 휴리스틱 방법론에 대해서 알아보도록 하겠다.

3. 휴리스틱 방법론

본 연구에서 제안하는 휴리스틱은 분지한계법에 기초를 두고 있다. 2장에서 사례 기업의 현실적인 문제에 대해서 RDHS 모형이 적용될 수 없음을 확인한 다음 유전 알고리즘의 적용 가능성을 검토하였다. 2장 4.3절에서 제시한 예제의 경우 17개의 작업 번호에 대한 최적 순서를 결정하는 문제이므로 일반적인 순서 결정 문제와 유사하게 유전 알고리즘을 적용해 보았으나 랜덤으로 생성한 해가 실행가능해일 확률이 1% 미만으로 너무 낮아서 유전 알고리즘으로는 도저히 해를 구할 수가 없었다.

두 번째로 시도한 방법은 분지한계법을 사용한 최적해 탐색이었다. 그러나 이 방법도 문제의 크기가 커지면 RDHS 모형보다 오히려 더 긴 시간이 소요되었다. 따라서 이 분지한계법을 간략화하여 소요시간을 단축하고 비교적 우수한 해를 찾으려는 시도를 하게 되었다. 이 절차를 분지 정책과 하한 계산의 두 단계로 나누어 설명하고 분지한계 알고리즘에 대해 알아보도록 하겠다.

3.1 분지 정책 (Branching policy)

본 휴리스틱의 핵심은 현재의 부분 일정(Partial schedule)에 새로운 일정을 추가할 때 필요한 가지(Branch)의 수를 줄이는 것이다. 2장 4.3절의 예제를 사용해 설명하도록 한다. 현재 고려대상의 부분 일정이 (5, 3, 1) 라고 하자. 이 부분 일정의 의미는 가장 먼저 5번 작업을 1번 탱크(투입 버퍼)에서 2번 탱크로 옮기고, 다음으로 3번 작업을 4번 탱크에서 5번 탱크로 옮긴 다음, 1번 작업을 7번 탱크에서 8번 탱크로 옮기는 순으로 첫 3개의 작업이 이루어진다는 것이다. 이 부분 일정에 다음 작업을 추가하기 위해서는 5개 작업 전체에 대해 가능성을 타진해야 하므로 5개의 가지가 필요해진다. 그러나 본 연

구에서는 문제에 대한 약간의 관찰을 통해 가능성이 적은 가지를 제외하고자 한다. 그 절차는 다음과 같다.

(1) 이미 완료된 작업은 분지 대상에서 제외한다. 부분 일정 (5, 3, 1)의 경우 1번 작업은 이미 완료 버퍼로 이동하여 작업 처리가 종료되었으므로 분지 대상에서 제외하여야 한다.
(2) 후속 탱크의 용량을 고려하여 이동 불가 작업은 제외한다. 부분 일정 (5, 3, 1)의 경우 5번 작업의 현 위치는 2번 탱크이고 다음 탱크는 3번이다. 그런데 3번 탱크의 용량은 1이고 현재 4번 작업이 처리 중이므로 이동이 불가능하게 된다. 또한 5번 탱크의 3번 작업도 마찬가지로 6번 탱크에서 작업 중이므로 이동이 불가능하다. 따라서 5번과 3번 작업을 분지 대상에서 제외할 수 있다.
(3) 호이스트 대기시간을 줄여주는 작업을 선택한다. 부분 일정 (5, 3, 1)의 경우 이미 3개의 작업을 분지 대상에서 제외 하였으므로 남은 두 후보는 2번 및 4번 작업이다. 이제 호이스트가 이 두 작업을 처리할 경우 발생하는 대기시간을 계산한다. 이 계산을 위해서는 최초의 호이스트 위치로부터 현 부분 일정을 수행하면서 변경되는 시간 값들을 계속 보유하고 있어야 한다. 예제의 경우 호이스트가 1번 작업을 처리한 직후의 상황을 보면, 일단 호이스트의 위치는 8번 탱크 위에 있고, 2번 작업은 6번 탱크에 있는데 그 탱크에서의 경과 시간은 11.1 분이며, 4번 작업은 3번 탱크에서 12.1 분 동안 작업이 진행된 상황이다. 2번 작업은 6번 탱크에서 처리 시간이 (5, 15) 구간에 있어야 하므로 호이스트가 현 위치에서 6번 탱크로 이동하는 0.6분 후 대기 없이 바로 이동 작업을 시작할 수 있다. 반면 4번 작업은 3번 탱크에서 처리 시간이 (15, 25) 인데 빈 호이스트 이동 시간이 1.5 분이므로 호이스트의 3번 탱크 도착 시점이 13.6 분이 되고 따라서 (5, 3, 1) 다음에 4번 작업을 처리하려면 호이스트가 3번 탱크 위에

서 1.4분(= 15 - 13.6) 동안 대기하여야 한다. 이렇게 각 분지 후보 작업을 현재의 부분 일정 바로 다음에 처리할 경우 호이스트의 대기시간이 얼마나 발생하는지를 조사하여 대기가 없는 작업은 모두 분지 대상으로 포함시키고, 대기가 발생하는 경우에는 대기시간이 가장 짧은 작업 2개만을 분지 대상으로 한다.

(4) 문제의 크기가 커지면 분지 수를 2개로 한정하여도 알고리즘 수행시간이 길어지게 된다. 또한 작업의 수가 많으면 대기시간이 없는 작업들의 수도 늘어나 분지의 수가 더 늘어나게 된다. 따라서 알고리즘 초반에는 다양한 가능성을 열어두기 위하여 대기시간이 없는 작업들을 모두 분지 대상에 포함시키고 그 숫자가 2개 이상일 경우에는 대기시간이 짧은 작업 2개까지 분지대상으로 한 뒤, 알고리즘 후반에 가서는 분지의 수를 무조건 1개로 줄이기로 하였다. 그 방법은 미리 정해진 BRATIO(비율) 값을 알고리즘 수행 시 파라미터로 사용하는 것인데, 예를 들어 BRATIO 값이 1/3 이었다면 위의 예에서 부분 일정의 길이가 1/3 이하일 경우에는 분지의 수를 2개 이상으로 설정하고, 1/3 이상일 경우에는 분지의 수를 1개로 설정하게 된다. 이 BRATIO 값을 이용하면 알고리즘 수행 시간과 해의 품질을 적절히 조정할 수 있다. 즉, 문제의 크기가 커서 알고리즘 수행시간이 길어지면 0 에 가까운 BRATIO 값으로 분지의 수를 1로 하여 계산 시간을 줄이고, 문제의 크기가 작아서 알고리즘 수행 시간에 대한 부담이 없는 경우에는 1 에 가까운 BRATIO 값으로 분지의 수를 2개 이상으로 하여 해의 품질을 높일 수 있게 되는 것이다.

분지 정책에 따라 분지가 이루어지면 그 다음 생성된 부분일정에 대한 가능해 여부를 확인하고 하한 계산을 수행하여야 한다.

3.2 하한(Lower bound) 계산

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

가능해 여부를 확인한 다음에는 그 부분 일정의 최적해 하한을 계산하는 것이 필요하다. 계산하는 방법은 현재까지의 경과된 시간에 다음 3개 항목 중 최대값을 더하는 것이다. 부분 일정 (5, 3, 1)의 경우, 마지막 작업인 1번 작업의 이동이 끝난 시점이 8.1분(= 2.3 + 0.6 + 2.3 + 0.6 + 2.3)이므로 이 8.1분에 다음 3개의 값 중 최대값을 더한다.

(1) 완료되지 않은 각 작업에 대해 시스템 내에 그 작업만 존재한다는 가정하에서 완료 시간을 계산하고 가장 큰 값을 찾는다. 부분 일정 (5, 3, 1)이 실행된 상황에서는 미완료 작업이 2번, 3번, 4번, 5번 4개이다. 이 중 5번 작업의 완료 시간을 계산해보자. 5번 작업은 현재 2번 탱크에 있고 경과시간은 5.8분이다. 현재 호이스트의 위치가 8번 탱크에 있으므로 빈 호이스트가 2번 탱크로 가면 1.8 분이 경과되어 처리 경과 시간은 7.6분(= 5.8 + 1.8)이 된다. 5번 작업의 2번 탱크 처리 하한 시간은 3분이므로 바로 3번 탱크로 이동할 수 있고 그 이동 시간은 2.3분이다. 3번 탱크에서 최소 처리 시간(15분) 동안 작업한 후 4번 탱크로 이동(2.3분)하고, 4번

탱크에서도 최소 처리 시간(1분) 동안 작업한 후 5번 탱크로 이동(2.3 분)한다. 5번 탱크에서 최소 처리 시간(40분) 작업한 후 완료 버퍼로 이동(2.9 분)하면 5번 작업의 처리가 완료되고, 총 소요시간은 67.6분(= 1.8 + 2.3 + 15 + 2.3 + 1 + 2.3 + 40 + 2.9)이 된다. 2번, 3번, 4번 작업에 대해서도 같은 방법으로 완료 시간을 계산하여 가장 큰 값을 찾고 그 결과를 MX_1 이라고 하자.

(2) 버퍼를 제외한 각 탱크에 대해 그 탱크를 필요로 하는 작업들의 총 처리 시간을 계산한다. 앞선 예제에서는 버퍼를 제외한 6개의 탱크(2번~7번) 각각에 대해 그 탱크의 처리를 완료하기 위한 최소 시간을 계산하면 된다. 부분 일정 (5, 3, 1) 상황에서 5번 탱크를 대상으로 계산해보면, 1번 및 2번 작업은 5번 탱크와 관계가 없으므로 무시한다. 3번 작업은 현재 5번 탱크에 있고 그 처리 경과 시간이 2.9분(= 0.6 + 2.3)이므로 처리 시간 하한(40분)까지 37.1분이 필요하다. 또한 처리 후 다음 탱크로 이동까지 2.3분이 필요하므로 5번 탱크와 관련하여 3번 작업을 처리하기 위해 필요한 시간은 39.4분이다. 다음으로 4번 작업의 처리를 위해서는 4번 탱크에서 가져와(2.3분) 하한 시간 동안의 작업(40분) 후 다음 탱크로 이동(2.3분)하기 위해 총 44.6분이 필요하다. 마지막으로 5번 작업을 가져와서(2.3분) 하한 시간 동안 작업(40분)을 하고 다음 순서인 8번 탱크로 이동(2.9분)하는데 45.2분이 필요하다. 따라서 부분 일정 (5, 3, 1) 상황에서 남아있는 모든 작업이 5번 탱크를 통과하는데 최소한 129.2분이 필요하다. 나머지 탱크들(2, 3, 4, 6, 7번)에 대해서도 같은 방법으로 최소 필요 시간을 구한 다음 이 값들 중에서 최대값을 찾아 MX_2 라고 하자.

(3) 남아 있는 작업들을 모두 처리하기 위한 호이스트의 최소 이동 시간을 계산한다. 다시 한 번 부분 일정 (5, 3, 1) 상황을 보면, 이 부분 일정은 $\alpha_{i,r}$ 행렬에서 0 이 아닌 값을 갖는 17 개의 작업 중에서 현재까지 3개가 처리되었다는 의미이므로 앞으로 14 개의 작업이 남아있게 되고 이 작업들

을 처리하기 위해 로드 된 상태의 호이스트 이동 시간을 모두 더하면 앞으로 호이스트가 수행해야 할 최소한의 시간을 계산할 수 있다. 이 값을 MX_3 라고 하자.

이제 이 3개 값을 사용해 각 부분 일정이 가질 수 있는 완료 시간 하한을 계산할 수 있다. 부분일정 (5, 3, 1)의 하한은 8.1+ $max\{MX_1, MX_2, MX_3\}$로 계산할 수 있다.

3.3 알고리즘

본 연구에서 제안하는 단순화된 분지한계 알고리즘은 다음과 같이 정리될 수 있다.

단순화된 분지한계 알고리즘

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

단계 1 : 첫 번째 분지한계

단계 1.1 : 실행 가능한 모든 작업 즉, 다음 순서 탱크가 비어있는 작업에 대해 분지한다.

단계 1.2 : 각 분지가 불가능 해이면 고려대상에서 제외한다.

단계 1.3 : 각 분지의 부분 일정에 대한 경과 시간을 계산한다.

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

단계 3 : 분지한계

단계 3.1 : 현재 경과 시간이 최대인 분지를 찾아 제 3.1절 정책에 따라 분지한다.

단계 3.2 : 각 분지가 불가능해이면 고려대상에서 제외한다.

단계 3.3 : 각 분지의 부분 일정에 대한 경과 시간을 계산한다.

단계 3.4 : 각 분지의 부분 일정이 완료 일정인 경우

단계 3.4.1 : 경과시간 < CurMin 이면 CurMin = 경과시간

단계 3.4.2 : 그 분지는 고려대상에서 제외한다.

단계 3.5 : 완료 일정이 아닌 경우

단계 3.5.1 : 그 분지에 대한 하한을 계산한다.

단계 3.5.2 : 하한이 CurMin보다 크면 그 분지는 고려대상에서 제외한다.

단계 4 : <단계 2>로 간다.

이 알고리즘을 앞서 LINGO 시스템을 실행한 컴퓨터에서 C++ 언어로 구현하고 같은 예제를 계산해 보았다. 이 문제는 크기가 작아서 BRATIO 값을 1로 두었다. 즉, 길이에 관계없이 모든 부분일정에서 분지의 수를 2개 이상으로 설정하였다.

그 결과 알고리즘 수행 시간은 앞선 LINGO 에서 계산된 최적해와 마찬가지로 0.01초 미만이었으나 해는 약간 달랐다. 작업 처리 순서는 (3, 1, 2, 4, 4, 2, 5, 3, 5, 3, 5, 5, 4, 3, 4, 5, 4)로 나왔고, 그 때의 완료 시간은 123.7분으로 계산되었다. 앞서 2장의 최적해 완료 시간 120.3분에 비해 2.8% 정도 부족한 결과를 보여 주었다.

사례 기업과 같이 탱크 30개 작업 20개의 문제에서 최적 해를 현실에 적용할 수 있는 짧은 시간 내에 계산할 수 있다면 최적화된 호이스트 생산시스템 일정계획을 수립할 수 있을 것으로 생각한다.

4. 수치 실험

본 절에서는 앞 절에서 소개한 분지한계 기반의 휴리스틱 방법의 성능을 평가하기 위해 탱크가 12인 좀 더 현실적인 크기의 문제를 사용해 최적해와 비교해 보고자 한다. 앞서 제 2장에서 언급하였던 것처럼 n이 10인 경우 LINGO를 통해서는 최적 해를 구할 수가 없었으므로 n이 10보다 작은 경우에 대해서 최적해와 비교하였다. 비교하는 방법은 탱크의 크기를 12로 고정한 상황에서 n이 7, 8, 9 3개의 값을 가질 때 LINGO를 통해 최적 해를 구하고 그 결과와 휴리스틱의 결과를 비교하였다. 휴리스틱은 BRATIO 값을 0, 0.25, 0.5, 0.75, 1.0 등 5개의 값으로 변화시키면서 실행하였고, 결과로 도출된 작업 전체 완료 시간과 결과를 도출하기 위해 소요된 계산 시간을 각각 비교하였다. 그 결과를 <표 3-1>에 정리하였다.

<표 3-1> 휴리스틱 해와 최적의 해의 성능 비교 (m=12)

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

표에 제시된 수치를 이해하기 위해 n이 7인 경우에 대한 도출 과정을 설명한다. 우선 최적의 완료 시간이 100이라고 표시되어 있는데 실제 계산 값은 1,189로서 비교를 위해 기준 값을 100으로 설정한 것이다. 휴리스틱에 의한 완료 시간은 BRATIO 값에 따라 각각 1,197, 1,197, 1,196, 1,196, 1,196 이 도출되었고, 최적해 1,189를 100으로 놓았을 때 최적 해를 기준으로 이 값들은 각각 100.67, 100.67, 100.59, 100.59, 100.59으로 변환되었다. 즉, 이 결과는 최적 해에 비해 각각 0.67%, 0.67%, 0.59%, 0.59%, 0.59%만큼 나쁜 결과를 보여주고 있다. 계산 시간 측면에서는 n이 7인 경우 최적해 도출 시간이 8초였는데 휴리스틱 수행 시간은 BRATIO 값에 관계없이 모두 1초 미만이었다.

n이 8 및 9인 경우에도 비슷한 방법으로 표의 항목들을 구할 수 있었다. 다만 BRATIO가 0일 때는 휴리스틱이 해를 찾지 못하여 해당 란은 공란으로 비워져 있다. 또한 앞서 언급한 것처럼 n이 10인 경우에는 LINGO를 통해 최적 해를 구할 수 없어서 이 표에는 n 값을 9 까지만 정리하였다. 그러나 휴리스틱은 n이 10일 때도 BRATIO 값이 0이 아니면 해를 구할 수 있었으며, 소요시간은 BRATIO 값이 0.25, 0.5, 0.75, 1.0일 때 각각 1초 미만, 59초, 722초, 893초였다. 특이한 점은 여기서 BRATIO 값이 0.75 및 1.0일 때의 소요시간 722초 및 893초로 n이 9일 때보다 작았다는 것이다. 이것은 일반적인 현상이라기 보다는 본 예제의 특수성에 기인한 것으로 보인다.

하나의 예제로 일반적인 특성을 논하기는 어렵지만 <표 3-1>을 잘 관찰해 보면 본 연구에서 제안하는 휴리스틱의 대략적인 특성을 파악할 수 있다. 우선 해의 품질을 보면 휴리스틱의 완료 시간은 최적 해에 비해 최악의 경우 5.17% 나쁜 결과를 보여주었고 최선의 경우 최적해와 동일하였다. 대부분의 경우 최적해와 1% 안쪽의 차이를 보여주었으므로 휴리스틱의 성능이 매우 뛰어나다는 것을 알 수 있다. 계산시간 측면에서도 BRATIO 값만 잘 선택한다면 1분 이내에 결과를 도출할 수 있으므로 현장에서 실제로 사용하는데 아무

런 문제가 없을 것으로 생각된다. 다만 BRATIO 값에 따라 해를 구하지 못하기도 하고 소요시간에도 차이가 있다는 것이 문제점으로 지적될 수 있다. 하지만 사례 기업의 상황에서 탱크의 수는 30개로 이미 정해져 있고 동시에 처리할 작업의 수를 5개 정도로 본다면 대략적인 BRATIO 값을 사전 테스트를 통해 알 수 있으므로 실제 사용에 큰 문제가 없을 것으로 판단된다.

사실상 <표 3-1>은 최적해와 비교할 수 있는 가장 큰 사이즈의 문제를 다루고 있다. 앞서 언급한 것처럼 본 연구의 동기가 된 사례 기업의 경우 30개의 탱크를 사용하고 있는데, 이런 규모의 문제는 LINGO로 최적 해를 구할 수 없으므로 휴리스틱의 성능 검증용 문제로는 사용할 수가 없었다.

그래서 제안된 휴리스틱을 사례 기업에 적용하기 위해 실제 자료와 아주 유사한 데이터를 사용한 예제를 구성해 보았다. 탱크는 30개이고 처리대상 작업의 수는 10개로 하였다. 실제로 이 기업의 경우 동시에 처리하는 작업의 수는 통상 5개 미만인데 예제에서는 이것보다 충분히 큰 수를 사용한 것이다. 또한 실제와 같이 각 작업이 거쳐야 할 탱크의 수는 8~10개 정도로 하였다. BRATIO 값을 0부터 0.1씩 증가시키며 알고리즘을 테스트하였는데 0.5까지는 해를 찾지 못하였고 0.6 이상에서는 해를 구할 수 있었다. 해를 구한 경우 소요된 계산 시간은 모두 1초와 2초 사이였다. 또한 해의 품질을 보면, BRATIO 값이 0.6일 경우의 완료 시간 값을 100 이라고 했을 때 0.7일 때는 97.1, 0.8일 때는 95.4, 그리고 0.9와 1.0일 때는 95.1 이었다.

여기서 한 가지 의문은 문제의 크기가 훨씬 큼에도 불구하고 탱크 30개인 문제의 계산시간이 앞선 예제보다 훨씬 짧았다는 점이다. 이것은 알고리즘 수행과정을 보면 알 수 있는데, 뒤쪽 예제의 경우 매우 많은 분지들이 불가능해로 결정되어 고려대상에서 제외되었다는 것을 알 수 있었다. 불가능해로 결정되는 것은 일부 탱크의 처리 시간이 상한시간을 초과하는 경우인데, 이 예제의 경우 각 탱크의 처리 시간 하한과 상한의 차이가 이전 예제에 비해 작았고 이로 인해 불가능해가 많이 생성되었던 것이 원인이었다고 생각된다.

5. 요약

이 장에서는 금속 부품의 화공처리 작업을 수행하는 단일 호이스트 생산시스템의 일정계획 문제를 다루었다. 문제의 핵심은 이 생산시스템에서 물자의 이동을 담당하는 호이스트의 효율적인 사용을 위해 완료 시간을 최소화 할 수 있는 일정계획을 수립하는 것이다. 연구 대상의 호이스트 생산시스템은 관련 연구들 중에서 가장 일반적이고 현실적인 상황을 다루고 있는데, 그 특징으로 1) 다양한 처리 조건을 갖는 다양한 작업을 처리하고 있고, 2) 이미 생산시스템에 투입되어 진행중인 작업들도 계획 대상에 포함시켰으며, 3) 작업 탱크들은 각각의 처리용량을 가질 수 있다는 것이다.

수리적 모형을 이용한 최적해는 문제의 크기가 현실적으로 커질 경우 적정 시간 안에 해의 도출이 불가능하였으므로 본 연구에서는 분지한계 기반의 휴리스틱을 개발하였다. 최적 해를 구할 수 있는 크기의 문제를 통해 휴리스틱의 결과와 LINGO의 최적 해를 비교해 본 결과 해의 품질에 큰 차이가 없었고, 소요시간 측면에서는 문제의 크기가 커질 경우 휴리스틱이 훨씬 빨리 해를 도출해 주었다. 따라서 본 연구에서 제안하는 휴리스틱은 문제의 크기가 클 경우에 매우 활용도가 높을 것으로 기대된다.

다음 장에서는 사례 기업의 단일 호이스트 생산시스템에서 다양한 작업을 처리하기 위한 일정계획 수립 방법을 알아보고자 한다. 사례 기업에서는 호이스트 일정계획을 어떤 수립 방법과 규칙으로 계획하는지를 알아보고 이 장에서 사용한 동일 예제를 가지고 사례 기업의 호이스트 일정계획 수립 규칙에 따라 실험을 실시해보고자 한다. 그리고 그 실험 결과를 분지 한계 기반의 휴리스틱 일정계획에서 실험한 결과와 비교하여 어느 정도의 효율성이 있는지 분석해 보고자 한다.

IV. 현실적 제약이 고려된 상황에서의 접근

1. 개요

항공산업의 화공처리공정은 호이스트를 사용한 생산시스템으로 다양한 작업을 처리하고 있다. 호이스트 생산시스템에서 다양한 작업을 처리하기 위해서 작업들의 처리 및 운송을 고려하는 호이스트 생산 일정계획이 매우 중요하다. 본 연구에서는 완료 시간을 최소화 할 수 있는 효율적인 단일 호이스트 생산시스템의 일정계획 방안을 연구하고 있다. 2장에서 연구한 실시간 동적 호이스트 일정계획은 크기가 작은 문제에 대해서는 최적 해를 제시하였으나, 사례기업과 같이 크기가 큰 문제에 대해서는 해를 제시하지 못하였다. 그리하여 3장에서는 문제의 크기가 현실적인 규모에도 최적의 해를 빠른 시간에 계산할 수 있는 분지한계 기반의 휴리스틱 일정계획을 개발하였다.

본 장에서는 사례 기업의 호이스트 일정계획 수립방법에 대해 알아보고, 3장 실험에서 사용한 동일 예제로 사례 기업에서의 일정계획 방법으로 구현했을 경우 어떤 결과가 나오는지 탐색하고자 한다. 사례 기업의 결과를 가지고 3장의 연구 결과와 비교해보면 분지한계 기반의 휴리스틱 일정계획의 결과가 어느 정도 효용성이 있는지 확인할 수 있을 것이라 생각한다.

본 장의 나머지 부분은 다음과 같이 구성하였다. 다음 절에서는 사례 기업의 호이스트 생산시스템에 대해 알아보고, 호이스트 생산시스템의 일정계획 수립방법에 대해 설명하고자 한다. 그리고 제 3절에서 사례기업의 일정계획 수립규칙에 대해 알아보고, 제 4절에서는 3장에서 설명한 예제를 사용하여 사례기업의 일정계획 방법과 규칙에 따라 실험을 해보고자 한다. 마지막 제 5절에서는 3장 분지한계 기반의 휴리스틱 일정계획의 결과와 사례 기업의 실험 결과를 비교하여 휴리스틱 일정계획이 어느 정도 효용성이 있는지 분석해 보고, 이 장을 요약하였다.

2. 사례 기업 연구

2.1 사례 기업의 호이스트 생산시스템

사례 기업에서는 세계 최대 여객기 제작회사에 날개 구조물, 동체, Door 등의 기체 구조물을 제작하여 납품하고 있으며, 국방관련 사업으로 회전익(헬기)과 고정익(전투기, 수송기) 및 무인기 등의 기체 구조물과 항공기를 제작하여 납품하고 있다.

여객기와 헬기, 그리고 전투기, 수송기, 무인기 등 항공기의 종류나 사용 용도에 따라 항공기 기체 구조물은 알루미늄 합금, 티타늄 합금, 스테인리스 등 여러 가지의 금속 재질을 사용하여 제작하게 된다. 금속 부품의 재질이 다르게 되면 금속 표면에 적용하는 피막과 화공처리 방법이 달라지게 된다. 그리고 동일 금속이라 할지라도 합금되는 재질에 따라서 적용하는 화공처리 공정도 차이가 나게 된다. 즉, 표면 처리를 수행하는 화학 재료와 처리 공정도 달라지게 되어 재질이나 합금의 종류에 따라 여러 가지 화공처리공정이 존재하게 된다.

또한 고객이 다르거나 민간항공기와 군용항공기 등 항공기의 사용되는 용도와 목적, 그리고 수행되는 사업에 따라 동일 공정이라 할지라도 제작사의 기준과 생산 방법에 따라 탱크에서 처리되는 시간이 다르거나 처리되는 화공처리공정의 차이가 있어 방문하는 탱크의 수가 달라질 수 있다. 사례 기업에서는 [그림 2-5]와 같이 30개의 탱크로 구성된 1개의 화공처리라인에서 8개의 화공처리공정이 있으며, 탱크에서 처리하는 시간이 다르거나 방문하는 탱크의 수가 달라서 세부적으로 16개의 각기 다른 화공처리공정이 존재하고 있다. 즉, 다양한 부품으로 인해 16개의 서로 다른 화공처리공정에 대해서 호이스트 생산 일정계획을 수립하여야 한다는 것이다.

호이스트 생산시스템에서 하루에 처리하는 작업 양은 부품의 크기와 수행되

는 공정에 따라 다르지만 하루에 수백 개에서 수천 개의 부품들이 화공처리공정에서 처리된다. 호이스트 생산시스템이 있는 화공처리공정은 기계가공공정과 판금가공공정에서 완료된 금속부품들이 필수적으로 거쳐가야 되는 공정이다. 그러므로 여러 개의 기계가공장비와 판금가공장비에서 완료된 금속 부품들이 모두 화공처리공정으로 집결되어 화공처리공정에서는 항상 병목현상이 발생되게 된다. 또한 신규로 들어오는 작업들은 도착하는 시기와 화공처리 작업방법 그리고 작업의 수 등을 알지 못하는 불확실성과 진행중인 작업을 고려하여 실시간으로 일정을 계획해야 한다는 복잡한 문제를 가지게 된다.

사례 기업의 호이스트 생산시스템에서는 동시에 2~3개의 작업을 화공처리라인에서 처리할 수 있으며, 하루 최대 15~20개 정도의 작업을 수행 할 수 있다. 그러나 실제로 사례 기업에서는 하루에 11개의 작업을 기준으로 호이스트 생산 일정계획을 수립한다.

그 이유는 첫 번째로 동시에 2~3개의 작업을 수행하려면 숙련된 작업자가 있어야 된다는 것이다. 숙련된 작업자는 각각의 화공처리공정에 대하여 하한 시간과 상한 시간을 다 알고 있으며, 실시간 상황에 따라 스스로 흐름이나 시간 등을 파악하여 조정할 수 있는 능력을 보유하고 있다. 숙련되지 않은 작업자는 2~3개의 작업을 동시에 처리하는데 어려움이 있으며 그 결과로 부품의 불량을 발생시키게 된다.

두 번째는 수시로 추가되는 작업이 발생하게 된다는 것이다. 기계가공공정과 판금가공공정에서 추가로 발생되는 작업의 수행을 위해서는 호이스트 생산 일정계획을 재 수립하여야 한다. 이때 추가 작업의 발생으로 대기중인 작업과 진행중인 작업을 고려한 일정계획을 재 조정하여야 하는 문제가 발생한다. 또한 기존의 작업 중 추가되는 작업의 수만큼 다음날로 미루어야 하는 문제가 생기는데, 이때 어떤 작업을 미루어야 할 건지에 대한 선택의 문제가 발생한다. 또한 다음날의 일정계획도 1일 전에 수립되어 있으므로 다음날로 미루더라도 다음날의 생산일정계획 역시 재 수립해야 하는 문제가 추가적으로 발생

하게 된다.

이런 2가지의 문제들을 해결하기 위해 사례 기업에서는 하루에 15~20개의 작업을 수행할 수 있음에도 불구하고 11개 작업을 기준으로 호이스트 생산시스템의 일정계획을 수립하고 있다. 그러나 숙련되지 않은 작업자나 추가되는 작업들 때문에 이런 생산일정계획을 수립한다는 것은 매우 비효율적이다. 이렇게 계획하면 계획된 11개의 작업만 수행할 수도 있고 추가적인 작업으로 인해 15~20 개의 작업을 수행할 수도 있겠지만 대부분 하루에 처리되는 작업 양은 15개 이하로 수행되는 경우가 많기 때문에 생산성의 효율이 향상되지 못하고 오히려 작업할 수 있는 능력보다 더 적은 작업을 수행하게 된다.

2.2 사례 기업의 호이스트 생산시스템 일정계획 수립 방법

사례 기업에서 어떻게 호이스트 생산시스템 일정계획을 수립하는지 알아보아야 그 문제점에 대해 분석할 수 있으며, 효율적인 호이스트 생산시스템 일정계획 방안을 수립할 수 있다. 여기서 사례 기업의 일정계획 수립 방법에 대해 알아보고자 한다.

사례 기업의 호이스트 생산시스템에 기계가공공정과 판금가공공정에서 완료된 부품들이 도착되면 생산 일정계획을 수립하기 위해 먼저 처리되는 화공처리공정 별로 부품들을 분류하게 된다. 즉, 16개의 화공처리공정 별로 Batch 작업이 이루어지므로 16개의 공정 별로 부품의 분류 작업을 수행하게 된다.

이렇게 화공처리공정 별로 분류작업이 완료되면 화공처리공정을 수행할 작업(Batch)을 선택하게 된다. 사례 기업에서는 하루에 11개의 작업을 기준으로 생산 일정계획을 수립하므로 11개의 작업을 선택하여야 한다. 이때 여섯 가지 순서에 의해 일정 계획을 수립하게 된다.

첫 번째로 부품의 긴급도 여부이다. 사례 기업에서 생산되는 항공기 부품들

은 조립공정에서 조립 후 고객에게 납기일에 맞추어 납품을 해야 한다. 화공처리공정은 기계가공공정과 판금가공공정의 완료 후에 수행하게 된다. 화공처리 공정 진행 후 마지막으로 페인트 작업을 수행하게 되면 부품 제작이 완료하게 되며, 완료된 부품들은 조립공정으로 이동하여 조립 후 고객에게 납품되어야 한다. 가공공정에서 생산 일정이 지연된 부품들이 화공처리공정으로 도착하는 경우가 빈번이 발생하게 되며, 부품 제작 완료 후 조립공정을 수행해야 하므로 생산 일정이 많이 지연된 부품들은 대기 없이 긴급히 화공처리공정에서 작업을 완료하여야 한다. 그러므로 생산 일정이 지연된 순서에 따라 긴급 여부를 결정하여 긴급한 부품들을 먼저 작업에 투입 시켜야 한다.

두 번째로 월간 납품하는 납품 대수이다. 민간 항공기의 경우 항공제작사에서 제작하는 항공기는 보통 월간 2~10대 정도이지만 많이 생산되는 것은 월간 40~60대까지 생산이 이루어지기도 한다. 사례 기업에서 제작되는 작업공정서의 부품 수 즉, 1개의 작업공정서로 작업할 수 있는 부품의 수는 1, 3, 6, 9, 12, 18, 24, 36 개 순으로 정해지게 된다. 이 부품의 수는 월간 납품하는 납품 대수나 부품의 크기에 의해 결정된다. 즉, 부품의 크기가 작거나 월간 납품 대수가 많으면 작업공정서의 부품 수는 많아지게 되며, 부품의 크기가 크거나 월간 납품 대수가 적으면 부품의 수는 작아지게 된다. 그러므로 월간 납품하는 대수가 많으면 수행해야 하는 부품 수가 많아지게 되므로 화공처리공정을 수행해야 하는 작업도 많아지게 된다. 그래서 작업이 많은 부품들은 일정한 수량에 대해 매일 작업이 이루어져야 생산 일정에 맞추어 조립공정에 부품을 제공할 수가 있다. 납품이 적은 부품들만 선택하게 되면 납품이 많은 부품들은 일정이 늦어지게 된다. 그러므로 납품하는 대수에 따라 적절하게 작업을 선택하여야 한다.

세 번째는 랙에 고정되는 부품의 수이다. 크기가 큰 부품들은 화공처리공정에서 한번에 3개 정도 밖에 수행하지 못한다. 또한 크기가 작은 부품들은 한번에 200~300개도 수행할 수 있다. 크기가 큰 작업만 선택하면 하루 처리하

는 부품의 수는 작아지게 되며, 크기가 작은 부품들만 선택하게 되면 랙에 부품을 고정하는 시간이 많이 소요되어 랙 장착 작업이 완료되지 않아 화공처리 라인에서 작업을 수행하지 못하고 랙 작업이 완료될 때까지 대기하여야 하는 문제가 발생하게 된다. 크기가 큰 부품들은 랙에 장착하는 작업 시간이 짧으므로 랙에서 장착하는 작업 시간이 길게 소요되는 작업을 수행하는 동안 먼저 작업에 투입할 수가 있다. 그러므로 크기가 큰 부품과 크기가 작은 부품들을 적절히 선택하여야 호이스트 생산시스템의 작업 효율을 향상시킬 수 있다. 크기가 큰 제품들은 하루에 처리할 수 있는 작업이 제약되므로 동체나 날개의 외피(Skin)와 같이 큰 부품들이 많이 소요되는 사업들의 부품들은 하루에 작업할 수 있는 양을 제한하는 대신 매일 작업이 이루어져야 한다.

즉, 부품의 납품 대수와 크기에 따라 매일 공정에 반영해야 되는 작업들이 존재하게 된다. 사례 기업에서는 하루에 계획되는 11개의 작업 중 4개의 작업은 매일 화공처리공정을 수행해야 되는 작업이다. 여기서 1개의 화공처리공정 즉, 1개의 작업에 처리되는 부품의 수는 랙에 고정할 수 있는 부품의 크기로 결정된다. 그러므로, 1개의 작업을 처리할 수 있는 부품의 크기와 랙에 고정하는데 소요되는 시간을 잘 판단하여 수행해야 할 부품의 수를 결정하여야 한다.

네 번째로 화공처리공정에 소요되는 전체 소요 시간과 처리 하한 시간 및 상한 시간에 따라 적절히 작업을 선택해야 된다. 사례 기업에서 수행하는 16개의 화공처리공정은 짧게는 1시간에서 길게는 26시간 정도 걸리며, 평균 2.5시간 정도 처리 시간이 소요된다. 화공처리공정이 짧은 작업 즉, 탱크에서 수행되는 하한 시간과 상한 시간이 짧은 공정들은 그 시간 동안 다른 작업을 수행하기 위해 이동해야 되는 이동 시간보다 하한 시간과 상한 시간이 짧기 때문에 다른 탱크로의 이동에 제약이 발생하게 되어 다른 작업을 수행할 수 없게 된다. 그러므로 화공처리공정 시간이 짧은 작업들만 선택하면 탱크에서 처리되는 시간이 너무 짧아 동시에 여러 개의 작업 수행이 어려워지므로 작업의

효율성은 떨어지게 된다. 그리고 화공처리공정이 긴 작업 즉, 탱크에서 수행되는 하한 시간과 상한 시간이 긴 공정들은 그 공정을 수행하는 동안 다른 작업의 이동은 가능해지나, 해당 탱크에서 수행하고 있는 작업이 완료되기 전까지는 다른 작업을 수행할 수 없게 되므로 후속 작업의 대기 시간이 증가하게 된다. 이럴 경우에도 마찬가지로 화공처리 시간이 긴 작업들만 선택하게 되면 하루에 수행할 수 있는 작업의 수는 작아지게 된다. 그러므로 화공처리공정을 수행할 작업 선택 시 처리 시간이 짧은 작업과 긴 작업을 적절히 선택하여야 작업 시간이 긴 공정이 이루어지는 동안 작업 시간이 짧게 이루어지는 공정의 작업을 수행할 수 있게 된다.

다섯 번째는 16개의 화공처리공정 중 동일하게 사용되는 탱크들이 많이 겹치지 않는 작업들을 선택해야 한다. 사례 기업의 호이스트 생산시스템은 16개의 공정을 모두 처리할 수 있도록 30개의 탱크로 구성되어 있으나, 각 공정들은 8~10개의 탱크 처리 공정에서 수행되어지기 때문에 같은 탱크를 사용하는 공정도 있고 다른 탱크를 사용하는 공정도 있다. 그러므로 동시에 처리하는 작업이 다른 탱크를 사용하는 화공처리공정으로 구성되어 있다면 사용되는 탱크가 겹치지 않으므로 1개의 작업이 그 탱크에서 수행되고 있더라도 그 작업이 완료되기 전에 후속 작업을 다음 순서의 탱크로 이동 할 수 있게 된다. 만약 사용되는 탱크가 동일하다면 그 탱크에서 한 작업이 수행되는 동안 다음 작업은 탱크로 이동되지 못하고 대기하고 있어야 하므로 작업을 수행할 수 없게 되어 생산 효율이 떨어지게 된다. 즉, 작업 선택 시 동일하게 사용되는 탱크들이 많이 겹치지 않는 작업들로 선택하게 되면 동시에 여러 개의 작업을 수행할 수 있게 될 것이다.

설명한 5가지에 의해 수행해야 될 작업이 선택되면 마지막 여섯 번째로 선택된 작업에 대해 화공처리공정의 작업 순서를 결정하여야 한다. 작업 순서는 세 번째와 네 번째와 다섯 번째에 설명한 랙에 고정되는 부품의 수와 화공처리공정에서 소요되는 시간 그리고 사용되는 탱크들이 겹치지 않는 작업들을

적절히 배치하여 최적의 호이스트 생산 일정을 수행할 수 있도록 계획하여야 한다. 이 3가지 사항을 최적의 상태로 배치하게 되면 화공처리공정에서 하루에 처리할 수 있는 작업의 수가 증가되므로 호이스트 생산 일정을 최적화 할 수 있게 된다.

그러나 이런 생산 일정계획을 사람이 매일 수작업으로 수립하여야 하는데 하루에 처리하는 수백, 수천 개의 부품에 대해서 위의 6단계로 작업을 분류하고 판단하여 최적의 호이스트 생산 일정을 계획한다는 것은 거의 불가능하며, 계획한다고 해도 소요되는 시간과 분석할 수 있는 능력에 한계가 따르게 된다. 그리고 사례 기업에서 생산되는 부품의 수는 만5천개 정도 되는데, 일정계획을 수립하려면 부품의 긴급 여부와 월간 납품하는 납품 대수, 랙에 고정되는 부품의 수, 화공처리공정에서 수행되는 시간, 동일하게 사용되는 탱크들이 많이 겹치지 않는 부품들의 정보를 알고 있어야 한다. 작업자가 만5천개의 부품에 대한 정보를 다 알지 못하므로 수작업으로 6가지 단계에 의해 최적화된 조건으로 일정계획을 수립한다는 것은 거의 불가능한 일이라고 생각된다.

그러므로 사례 기업에서는 해당 화공처리공정 별로 작업을 분류를 한 뒤 생산관리에 의해 긴급하다고 통보된 부품들과 매일 작업이 이루어져야 되는 작업을 선택하고, 부품의 수가 적은 작업과 많은 작업을 적절하게 결정하여 화공처리공정을 수행할 작업들을 선택하게 된다. 그리고 수행할 작업에 대해 작업자의 경험으로 작업 순서를 결정한다. 숙련된 작업자가 아니면 계획을 효율적으로 수립 할 수 없으므로 화공처리공정의 생산 일정계획은 화공처리공정의 제일 선임인 리더만이 수립할 수 있으며, 그렇지 않은 작업자가 계획을 수립하면 효율적인 생산 일정계획을 수립할 수 없게 된다.

사례 기업에서 호이스트 생산 일정계획 수립을 리더의 경험에 의해서 수행하고 있지만 사례 기업의 일정계획 방법의 실험을 위해서는 일정계획 수립 규칙이 있어야 한다. 다음절에서는 사례 기업의 호이스트 생산 일정계획 수립 규칙에 대해서 알아보고자 한다.

3. 사례 기업의 일정계획 수립 규칙

사례 기업에서 수행하고 있는 호이스트 생산시스템 일정계획의 수립 방법으로 분지한계 기반의 휴리스틱 일정계획에서 사용된 예제를 적용하여 실험을 해 보기 전에 먼저 사례 기업의 호이스트 생산시스템의 일정계획 수립 규칙부터 정의하여야 한다.

호이스트 생산시스템 일정계획의 문제 해결에 필요한 가정들은 사례 기업의 호이스트 일정계획 수립 방법을 바탕으로 정의하였다.

사례 기업의 호이스트 생산시스템 일정계획 수립 시에는 화공처리공정에 대기 중인 부품의 수가 수백에서 수천 개에 이르므로 하루에 모든 대기 부품에 대해 화공처리공정을 수행할 수가 없다. 그러므로 먼저 화공처리공정 별로 부품을 분류한 다음 투입될 작업들을 부품의 긴급도 여부, 월간 납품하는 대수, 랙에 고정되는 부품의 수, 화공처리공정에서 소요되는 시간, 화공처리공정 중 동일하게 사용되는 탱크들이 많이 겹치지 않는 작업 등의 기준에 따라서 선택하여야 한다. 그리고 선택된 작업에 대해 화공처리공정의 작업 순서를 결정하여야 한다.

그러나 화공처리공정 대상 작업을 선택하는 문제에 대해서 위의 기준과 같이 작업을 선택하는 것이 고려되어야 하는데, 위의 5가지 조건에 대한 데이터가 너무 방대하여 기준에 따라 작업을 선택하기에 어려움이 있다. 예를 들면 부품의 크기 같은 경우 항공기에 사용되는 부품은 복잡한 기하학적 형상으로 되어 있어 단순히 가로, 세로, 높이로 부품의 크기를 파악할 수가 없다. 적용하려면 복잡한 기하학적 형상을 가로, 세로, 높이의 크기로 변환하여야 하는데, 사례 기업에서 현재 가공하는 부품의 수가 약 만5천개 정도이므로, 자료를 변환하기가 너무 어렵고 시간이 많이 소요된다. 그리고 사례 기업에서 수행하는 사업이 약 50여개 정도이며, 각 사업의 월간 납품 대수가 2~60대 정도이어서 각 부품의 납품 일정 별로 납품 대수에 대해 데이터를 수집하게 되

면 데이터가 너무 많이 존재하게 된다. 이 방대한 데이터들을 수집하기에도 어려움이 있지만 수집된 많은 데이터를 판단하기 위해 분석하는 것은 여간 어려울 뿐만 아니라 너무 복잡하게 된다.

그래서 본 연구에서는 작업을 선택하는 부분은 제외하고 선택된 작업에 대해 화공처리공정에서 작업 순서를 결정하는 문제에 대해서만 다루고자 한다. 즉, 호이스트 생산시스템의 일정계획은 작업을 대기하는 투입 버퍼에서 각 탱크를 방문하여 정해진 탱크들의 처리공정을 각각 수행한 뒤 완료 버퍼로 도착하기까지의 문제로 정의를 하고자 한다.

일정계획에 대한 문제의 정의는 3장 2절에 정의된 가정과 동일하다. 다음은 사례 기업에서 호이스트 일정계획을 수립하는 규칙을 정리한 것이다.

사례 기업의 일정계획 수립 규칙

<규칙 1> 완료 버퍼에 도착한 작업은 일정계획에서 제외한다.

<규칙 2> 투입 버퍼와 완료 버퍼를 제외한 모든 탱크에서 수행하는 작업은 4개를 초과할 수 없다.(사례기업에서는 2~3개 정도를 처리함)

<규칙 3> 각 탱크의 진행 시간을 확인해서 하한 시간이 경과된 후 다음 탱크로 이동할 수 있으며, 상한 시간이 도달하기 전에 탱크에서 꺼내어 이동하여야 한다.

<규칙 4> 탱크의 하한 시간이 경과되었고, 상한 시간이 도달될 때까지 다음 탱크에서 다른 작업을 진행하고 있어 이동할 수 없다면, 투입 버퍼로 이동하여 다음 탱크가 비워질 때까지 대기하여야 한다.

<규칙 5> 탱크의 하한 시간이 경과된 작업이 2개 이상 존재 할 때는 상한 시간과 이동 시간을 계산하여 먼저 도달하는 작업부터 이동하여야 한다.

<규칙 6> 호이스트가 이동할 작업이 없을 때는 다음 도달되는 작업의 탱크 위치로 이동하여 탱크의 하한 시간이 경과될 때까지 대기한다.

<규칙 7> 모든 작업이 완료 버퍼로 이동하게 되면, 호이스트의 일정계획을 종료한다.

다음 절에서는 3장의 분지한계 기반의 휴리스틱 일정계획에 사용된 동일 예제로 사례 기업의 호이스트 생산시스템의 일정계획 수립 규칙에 의거 계획을 수립하였을 때에 어떤 결과가 나오는지 실험을 통해 분석하고 그 결과를 3장의 분지한계 기반의 휴리스틱 일정계획과 비교하여 사례 기업의 일정계획 수립 규칙에 의해 계획된 결과보다 분지한계 기반의 휴리스틱 일정계획이 어느 정도의 효율성이 있는지 분석해 보고자 한다.

4. 사례 기업의 일정계획 실험

이 절에서는 분지한계 기반의 휴리스틱 일정계획 연구에서 사용된 예제로 사례 기업의 호이스트 생산시스템 일정계획 방법에 따라 실험을 해보고자 한다. 사례 기업에서 투입되는 작업의 순서를 결정할 때 랙에 고정되는 부품의 수, 화공처리공정에 소요되는 시간 그리고 사용되는 탱크들이 겹치지 않는 작업들을 적절히 배치하는 방법으로 작업 순서를 결정하나, 여기에서는 선택된 작업에 대해 사례 기업의 일정계획 수립 규칙에 의거 일정계획을 수립한다.

호이스트 생산시스템의 일정계획은 작업을 대기하는 투입 버퍼에서 각 탱크를 방문하여 정해진 탱크들의 화공처리공정을 완료한 뒤 완료 버퍼로 도착하기까지의 문제이다. 예제를 통해 사례 기업의 일정계획 수립 방법을 이용하여 해를 찾아보도록 한다. 이 실험을 수행하는 목적은 사례 기업의 사례를 통해 분석된 결과와 분지한계 기반의 휴리스틱 일정계획이 얼마나 차이가 나는지 즉, 연구결과가 어느 정도 효율성이 있는지를 비교 분석하기 위해서이다.

실험은 2장 4.3절의 예제를 사용하였다. 예제에서 5개의 작업이 7번, 6번, 4번, 3번, 1번 탱크에서 진행되고 있다. 현재 호이스트는 1번 탱크에 위치하고 있다. 1번 작업은 7번 탱크에서 작업을 시작한지 33분이 경과되어 하한 시간 30분을 경과하였으며 상한 시간 50분이 도달하기 전까지 17분 이내에 8번 탱크로 이동해야 한다. 2번 작업은 6번 탱크에서 작업을 시작한지 3분이 경과되었고 하한 시간이 5분 상한 시간이 15분이므로 2분에서 12분 사이에 다음 탱크인 7번 탱크로 이동하여야 한다. 3번 작업은 4번 탱크에서 작업을 시작한지 1분 경과되었고 하한 시간이 1분, 상한 시간이 5분 이므로 지금 바로 이동하던지 4분이내 5번 탱크로 이동되어야 한다. 4번 작업은 3번 탱크에서 작업을 시작한지 4분이 경과되었으며 하한 시간이 15분, 상한 시간이 25분 이므로 11분에서 21분 사이에 4번 탱크로 이동되어야 한다. 5번 작업은 1번 탱크에서 대기를 한지 12분이 경과 되었다.

<표 4-1> 진행중인 작업의 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	7	8	30	50	33	+3	-17	1.8	2.3
2	6	7	5	15	3	-2	-11	1.5	2.3
3	4	5	1	5	1	0	-4	0.9	2.3
4	3	4	15	25	4	-11	-22	0.6	2.3
5	1	2	0	0	12	0	0	0	2.3

<표 4-1>은 5개의 작업에 대한 현재 진행 중인 작업의 진행현황이다. 현재 5개의 작업 중 완료 버퍼에 도달한 작업은 없으며, 투입 버퍼를 제외한 4개의 작업이 탱크에서 진행 중이므로 5번 작업은 제외하도록 한다. 4개의 작업 중 이동 하한 시간이 경과하여 바로 이동할 수 있는 작업은 1번과 3번 작업이다.

1번 작업은 아직 이동 상한 시간이 17분 남았고 3번 작업은 이동 상한 시간이 4분 남았다. 1번 작업 이동 후 3번 작업을 선택하면 1번 작업의 7번 탱크로 이동하는 시간 1.8 분과 다음 8번 탱크로 이동 시간 2.3 분 그리고 3번 작업의 4번 탱크로 이동 시간 1.2 분 총 5.3 분이 소요되어 3번 작업의 상한 시간 4분을 초과하게 되므로 작업자는 3번 작업을 선택하게 된다. 3번 작업의 이동을 위해 1번 탱크에서 4번 탱크로 호이스트 이동 시간은 0.9 분이며, 탱크에서 부품을 장착하여 꺼내고 5번 탱크로 이동하여 탱크에 담그는 시간은 2.3분이 소요되므로 3번 작업을 이동하는데 소요되는 총 시간은 3.2 분이다.

<표 4-2>는 3번 작업 이동 후 진행현황이다. 3번 작업 이동 후 이동할 수 있는 작업은 1, 2, 5번 작업이다. 투입 버퍼의 5번 작업은 제외하고 진행 중인 1, 2번 작업에서 작업자는 선택 할 것이다. 2번 작업은 다음 탱크가 7번 이나 7번 탱크에서는 이미 1번 작업이 수행 중이므로, 1번 작업이 완료되지 않으면 이동할 수 없다. 그러므로 작업자는 1번 작업을 선택하여 이동할 것이다.

<표 4-2> 작업 3 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	7	8	30	50	36.2	+6.2	-13.8	0.6	2.3
2	6	7	5	15	6.2	+1.2	-8.8	0.3	2.3
3	5	6	40	60	0	-40	-60	0	2.3
4	3	4	15	25	7.2	-7.8	-17.8	0.6	2.3
5	1	2	0	0	15.2	0	0	0.9	2.3

1번 작업을 위해 7번 탱크로 이동 시간은 0.6 분이고 부품을 꺼내고 8번 탱크로 이동 후 호이스트를 내리는 작업까지 2.3 분이 걸리므로 작업을 완료하는데 총 2.9분이 소요되게 된다.

그 다음은 2번 작업을 7번 탱크로 이동할 수 있으므로 2번 작업을 수행하게 될 것이다. 호이스트를 8번 탱크에서 6번 탱크로 이동하는 시간은 0.6 분, 탱크에서 꺼내고 7번 탱크로 이동하여 탱크에 내려놓는 시간은 2.3 분이 걸리므로 총 2.9 분이 소요되게 된다.

<표 4-3> 작업 1, 2 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	7	8	30	50	0	-30	-50	0	2.3
3	5	6	40	60	5.8	-34.2	-54.2	0.6	2.3
4	3	4	15	25	13	-2	-12	1.2	2.3
5	1	2	0	0	21	0	0	1.8	2.3

<표 4-3>은 작업 1, 2 번 이동 후 진행 현황이다. 1번 작업이 완료 버퍼로 이동하여 작업이 완료되어 현재 탱크에 진행 중인 작업은 2, 3, 4 번 3개의

작업이다. 진행중인 작업에서 바로 이동할 수 있는 작업은 없고 진행중인 작업은 3개 이므로 이동 가능한 작업은 투입 버퍼에 대기하고 있는 5번 작업이다. 5번 작업의 이동을 위해 7번 탱크에서 1번 탱크까지 호이스트 이동 시간은 1.8 분이다. 부품을 들고 다음 2번 탱크로 이동하는 시간은 2.3 분 걸리므로 총 4.1 분이 소요된다.

5번 작업 이동 후 4번 작업의 하한 시간이 2.1 분 경과 되었으므로 4번 작업의 이동을 위해 3번 탱크로 이동할 것이다. 2번 탱크에서 3번 탱크로 이동 시간은 0.3 분이며, 부품을 꺼내어 다음 탱크인 4번 탱크로 이동 시간 2.3분 걸리므로 총 2.6 분이 소요된다.

<표 4-4> 작업 5, 4 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	7	8	30	50	6.7	-23.3	-43.3	0	2.3
3	5	6	40	60	12.5	-27.5	-47.5	0.6	2.3
4	4	5	1	5	0	**-1**	-5	0	**2.3**
5	2	3	3	10	2.6	**-0.4**	-7.4	**0.6**	**2.3**

<표 4-4>는 작업 5, 4 이동 후의 진행현황이다. 이제 진행 할 수 있는 작업은 없다. 앞으로 0.4분 뒤에 5번 작업의 하한 시간이 완료되며, 1분 뒤에 4번 작업이 완료된다. 작업자는 4번 작업의 다음 방문 탱크인 5번 탱크에서 3번 작업이 진행 중이지만, 5번 탱크는 용량이 2이므로 이동할 수 있다. 5번 작업의 상한 시간이 7.4 분 남았으므로 4번 작업을 1분 대기하였다가, 5번 탱크로 이동 후에도 5번 작업의 상한 시간을 경과하지 않으므로 작업자는 호이스트를 이동하지 않고 4번 탱크에서 1분을 대기하였다가 4번 작업을 이동할 것이다. 대기시간은 1 분이고 이동 시간은 2,3 분 걸리므로 총 3.3 분이 소요된다.

4번 작업 이동 후 5번 작업의 하한 시간이 2.9분 경과 되었으므로 5번 작업을 이동하여야 한다. 5번 탱크에서 2번 탱크로 이동 시간 0.9 분과 2번 탱크에서 3번 탱크로 이동 시간 2.3 분 걸리므로 총 3.1 분이 소요될 것이다.

<표 4-5> 작업 4, 5 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	7	8	30	50	13.2	-16.8	-36.8	1.2	2.3
3	5	6	40	60	19	-21	-41	0.3	2.3
4	5	6	40	60	3.2	-36.8	-56.8	0.3	2.3
5	3	4	15	25	0	**-15**	-25	0	**2.3**

<표 4-5>는 작업 4, 5 이동 후의 진행현황이다. 현재 진행 중인 작업은 2, 3, 4, 5 번 4개의 작업이며, 하한 시간이 경과된 작업이 없으므로 이동할 수 있는 작업은 존재하지 않는다. 4개의 작업 중 하한 시간이 먼저 도달되는 작업은 5번 작업이므로 작업자는 3번 탱크에서 5번 작업의 하한 시간 15 분을 대기한 후 4번 탱크로 이동 할 것이다. 대기시간 15분과 4번 탱크로 이동 시간 2.3 분 걸리므로 총 17.3 분이 소요될 것이다.

5번 작업 이동 후 2번 작업의 하한 시간이 0.5분 경과되었다. 2번 작업의 이동을 위해 4번 탱크에서 7번 탱크로 이동 시간 0.9 분 7번 탱크에서 8번 탱크로 이동 시간 2.3 분 걸리므로 총 3.2 분이 소요될 것이다.

<표 4-6>은 작업 5, 2번 이동 후 진행현황이다. 2번 작업이 완료 버퍼로 이동 하였으므로 계획을 해야 되는 작업은 3, 4, 5 번 3개의 작업이다. 이제 5번 작업의 하한 시간이 지났으므로 8번 탱크에서 4번 탱크로 이동하여 5번 작업을 5번 탱크로 이동 하여야 한다. 그러나 5번 탱크에는 3번, 4번 작업이 진행하고 있으므로 5번 탱크로 이동 할 수 없다. 3번 작업의 하한 시간이 0.5분

남았으나 3번 작업을 이동하면 5번 작업의 상한 시간이 경과하므로 5번 작업의 상한 시간을 초과하지 않도록 5번 작업을 1번 투입 버퍼로 이동하여 3번 작업이 이동 될 때 까지 대기하여야 한다. 4번 탱크로 이동 시간은 1.2 분이며, 4번 탱크에서 1번 탱크로 이동 시간은 2.9 분으로 총 4.1 분이 소요된다.

<표 4-6> 작업 5, 2 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	8	-	0	0	0	0	0	0	0
3	5	6	40	60	39.5	-0.5	-21.5	0.9	2.3
4	5	6	40	60	23.7	-16.3	-36.3	0.9	2.3
5	4	5	1	5	3.2	+2.2	-1.8	1.2	2.3

3번 작업의 하한 시간이 경과 되었으므로 1번 탱크에서 5번 탱크로 이동하여 3번 작업을 6번 탱크로 이동시킬 것이다. 5번 탱크로 이동 시간은 1.2 분 5번 탱크에서 6번 탱크로 이동 시간은 2.3 분 총 3.5 분이 소요될 것이다.

3번 작업 이동 후 5번 탱크가 비워졌으므로 5번 작업을 5번 탱크로 이동할 수가 있다. 6번 탱크에서 1번 탱크로 이동 시간은 1.5 분 5번 탱크로 이동 시간은 3.2 분 걸리므로 총 4.7 분이 소요될 것이다.

<표 4-7> 작업 5, 3, 5 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	8	-	0	0	0	0	0	0	0
3	6	7	5	15	4.7	-0.3	-10.3	0.3	2.3
4	5	6	40	60	36	-4	-24	0	2.3
5	5	6	40	60	0	-40	-60	0	2.3

<표 4-7>은 작업 5, 3, 5 이동 후 진행현황이다. 하한 시간이 경과된 작업은 존재하지 않으며 0.3 분 뒤에 3번 작업의 하한 시간이 경과되므로 6번 탱크로 이동하여 3번 작업을 7번 탱크로 이동할 것이다. 6번 탱크 이동 시간이 0.3 분이며 이동 후 3번 작업의 하한 시간이 경과되므로 대기 없이 7번 탱크로 이동시키면 된다. 대기 시간 0.3분과 이동 시간 2.3분 걸리므로 총 2.6 분이 소요된다.

이제 4번 작업의 하한 시간이 1.4 분 남았다. 1.4 분 대기 동안 7번 탱크에서 5번 탱크로 이동 할 것이다. 이동 후 5번 탱크에서 0.8 분 대기하고, 6번 탱크로 이동 할 것이다. 4번 작업의 대기시간 1.4 분과 6번 탱크로 이동 시간 2.3 분 걸리므로 총 3.7 분이 소요된다.

이동 후 4번 작업 6번 탱크의 하한 시간이 5분으로 3번, 5번 작업의 하한 시간 보다 빨리 도달하게 된다. 그러므로 6번 탱크에서 5 분 대기 후 4번 작업을 7번 탱크로 이동하여야 한다. 그러나 7번 탱크의 작업이 아직 26.3 분 남았고 6번 탱크의 상한 시간 15 분 뒤에도 7번 탱크는 작업 중이므로 4번 작업은 하한 시간 5분 후 1번 탱크로 이동하여야 한다. 대기시간 5 분에 1번 탱크 이동 시간 3.5 분 걸리므로 총 8.5 분이 소요된다.

<표 4-8> 작업 3, 4, 4 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	8	-	0	0	0	0	0	0	0
3	7	8	30	50	12.2	-17.8	-37.8	1.8	2.3
4	1	7	0	0	0	0	0	0	3.8
5	5	6	40	60	14.8	-25.2	-45.2	1.2	2.3

<표 4-8>은 작업 3, 4, 4 이동 후 진행현황이다. 이제 3번 작업이 17.8 분 남았으므로 3번 작업 대기 시간 동안 호이스트는 7번 탱크로 이동하게 될 것이다. 작업시간 17.8 분 경과 후 3번 작업은 8번 탱크로 이동할 것이다. 대기시간 17.8 분과 이동 시간 2.3 분 걸리므로 총 20.1 분이 소요될 것이다.

3번 작업 이동 후 4번 작업을 1번 탱크에서 7번 탱크로 이동하여야 하므로 8번 탱크에서 1번 탱크로 이동 시간은 2.1 분 그리고 7번 탱크로 이동 시간은 3.8 분 걸리므로 총 5.9 분이 소요될 것이다.

이제 5번 작업의 하한 시간이 경과 되었으므로 5번 작업을 위해 5번 탱크로 이동 할 것이다. 5번 탱크로 이동 시간은 0.6 분, 6번 탱크로 이동 시간은 2.3 분 걸리므로 총 2.9 분이 소요될 것이다.

<표 4-9> 작업 3, 4, 5 이동 후 진행 현황

작업	현재 탱크	다음 탱크	하한	상한	투입 시간	이동 하한	이동 상한	현 탱크 이동시간	다음탱크 이동시간
1	8	-	0	0	0	0	0	0	0
2	8	-	0	0	0	0	0	0	0
3	8	-	0	0	0	0	0	0	0
4	7	8	30	50	2.9	-27.1	-47.1	0.3	2.3
5	6	7	5	15	0	-5	-15	0	2.3

<표 4-9>는 작업 3, 4, 5 이동 후 진행현황이다. 3번 탱크가 완료 버퍼로 이동 하였으므로 이제 진행 중인 작업은 4, 5 번만이 남았고, 하한 시간이 경과된 작업은 없다. 5분 후에 5번 작업의 하한 시간이 도달하지만 다음 탱크인 7번 탱크에 4번 작업이 작업 중이다. 상한 시간 15 분 후에도 7번 탱크를 사용 중이므로 5번 작업을 6번 탱크에서 5 분 대기 후 1번 탱크로 이동할 것이다. 6번 탱크에서 대기시간 5 분과 1번 탱크로 이동 시간 3.5 분 걸리므로 총 8.5 분 소요될 것이다.

5번 작업이 1번 탱크로 이동 후 4번 작업의 하한 시간이 18.6 분 남았으므로 18.6 분을 대기하여야 한다. 대기하는 동안 7번 탱크로 호이스트를 이동할 것이며 7번 탱크에서 작업 경과 후 8번 탱크로 이동 할 것이다. 4번 작업의 대기시간 18.6 분과 8번 탱크로 이동 시간 2.3 분 걸리므로 총 20.9 분이 소요될 것이다.

4번 작업이 완료 버퍼로 이동하고 이제 진행 중인 작업은 5번 밖에 없다. 5번 작업을 1번 탱크에서 7번 탱크로 이동시키기 위해 1번 탱크로 이동 시간 2.1 분 그리고 7번 탱크로 이동 시간 3.8 분 걸리므로 총 5.9 분이 소요될 것이다.

5번 작업만이 남았으므로 5번 탱크에서 30 분 대기 후 8번 탱크로 이동하면 대기 시간 30 분 이동 시간 2.3 분 걸리므로 총 32.3 분 소요되게 된다.

5번 작업의 완료로 5개의 작업이 모두 완료 버퍼에 있으므로 이제 일정을 계획할 작업이 존재하지 않아 호이스트 일정계획은 종료하게 된다.

호이스트 생산시스템에서 다양한 작업을 처리하기 위해 3장 분지한계 기반의 휴리스틱 일정계획에 대한 연구에서 사용한 예제와 동일한 예제로 사례 기업의 호이스트 생산시스템의 생산 일정계획 수립 방법 및 규칙에 의거 일정계획을 수립해 본 결과, 5개의 작업을 모두 진행하는데 총 166.3 분이 소요되었으며, 작업 처리 순서는 (3, 1, 2, 5, 4, 4, 5, 5, 2, 5, 3, 5, 3, 4, 4, 3, 4, 5, 5, 4, 5, 5)로 나타났다.

이 결과는 2장의 RDHS 모델을 적용하여 실험한 최적해 120.3분 보다 38%로 증가된 수치이며, 3장의 분지한계 기반의 휴리스틱 알고리즘의 해 123.7분 보다 34.4% 증가된 결과를 나타내었다. 즉, RDHS의 최적해와 휴리스틱의 해가 사례기업의 일정계획 수립 방법 및 규칙에 의거 실험한 일정계획의 결과보다 훨씬 더 우수한 해를 나타내었다.

5. 요약

이 장에서는 호이스트 생산시스템 일정계획에 대해 분지한계 기반의 휴리스틱 일정계획에서 사용한 예제와 동일한 예제로 이 논문의 사례 기업인 항공산업의 단일 호이스트 생산시스템에서 다양한 작업을 처리하기 위한 일정계획을 수립하는 방법과 규칙으로 실험을 실시하였다.

사례 기업에서는 화공처리공정을 수행하기 위해 호이스트 생산시스템 일정계획을 수작업으로 작업자의 리더에 의해 계획되고 있었는데, 하루에 수행되는 수백에서 수천 개의 부품을 대상으로 16개의 화공처리공정에 대한 일정계획은 6가지 단계를 거쳐 수립하고 있었다.

첫 번째는 부품의 긴급도 여부 이었으며, 두 번째는 월간 납품하는 납품 대수, 세 번째는 랙에 고정되는 부품의 수, 네 번째는 화공처리공정에서 소요되는 전체 소요 시간이고 다섯 번째는 화공처리공정 중 동일하게 사용되는 탱크들이 많이 겹치지 않는 작업의 선택이었다. 다섯 번째까지는 화공처리공정에서 수행될 작업을 선택하기 위해 판단하는 단계였으며, 마지막 여섯 번째는 부품의 수와 화공처리공정에서 소요되는 시간 그리고 사용되는 탱크들이 겹치지 않는 작업들을 적절히 배치하여 최적의 호이스트 생산 일정을 수행할 수 있도록 작업 순서를 결정하는 것이다.

그러나 이것을 사람이 매일 6단계에 의해 최적의 호이스트 생산 일정을 수립한다는 것은 거의 불가능하며, 계획한다고 해도 소요되는 시간과 분석할 수 있는 능력에 한계가 따르게 된다. 이 6단계 중 작업 대상을 선택하기 위해 1~5단계가 고려되어야 하나, 이 5가지 조건에 대한 데이터가 너무 방대하여 수집하는데 어려움이 있을 뿐 아니라 판단을 하기 위해 분석하는 것이 너무 복잡하여 당장 실행하지는 못하였다. 그래서 본 연구에서는 작업을 선택하는 부분은 제외하고 선택된 작업들에 대해 화공처리공정에서 작업 순서를 결정하는 문제에 대해서만 다루었다.

3장 분지한계 기반의 휴리스틱 일정계획의 연구에서 사용된 문제의 정의와 예제를 가지고 사례 기업의 호이스트 생산시스템 일정계획 수립 규칙에 따라 실험을 해 보았다.

실험 결과 작업 처리 순서는 (3, 1, 2, 5, 4, 4, 5, 5, 2, 5, 3, 5, 3, 4, 4, 3, 4, 5, 5, 4, 5, 5)로 나왔으며, 이 때의 완료 시간은 166.3분이 소요되었다. 3장의 분지 한계 기반의 휴리스틱 일정계획은 시스템의 작업 완료 시간은 123.7분으로 나타났었다.

두 결과를 비교해 보면 사례 기업의 작업 완료 시간이 휴리스틱 일정계획보다 34.4% 증가되었음을 알 수 있었다. 즉, 휴리스틱 일정계획의 해가 사례 기업의 결과보다 25.6% 더 나은 효과를 나타내었다. 이는 Kumar(1994)가 호이스트 일정계획에 따라 평균 작업 대기시간이 20%까지 줄어들 수 있다고 한 연구 결과보다 더 나은 효과를 볼 수 있었다.

그리고 호이스트 작업 처리 완료 시간의 차이뿐 아니라 현장에서 수작업으로 계획을 수립하는데 숙련된 작업자가 몇 시간이 소요되었다. 그리고 작업자가 수립한 계획은 최적의 호이스트 생산시스템 일정계획으로 수립되었다고 보장하기도 어려웠다. 그러나 분지 한계 기반의 휴리스틱으로 사례 기업과 같은 현실적인 크기의 문제에 대해 생산 일정계획을 수립하면 해의 품질은 최적해와 차이가 없었으며, 소요시간 측면에서는 문제의 크기가 커질 경우 휴리스틱이 훨씬 빨리 해를 도출할 수 있었다. 그러므로 본 연구에서 제안하는 휴리스틱은 문제의 크기가 커질 경우에도 최적의 효율적인 호이스트 생산시스템의 일정계획을 수립할 수 있을 것이라 생각한다.

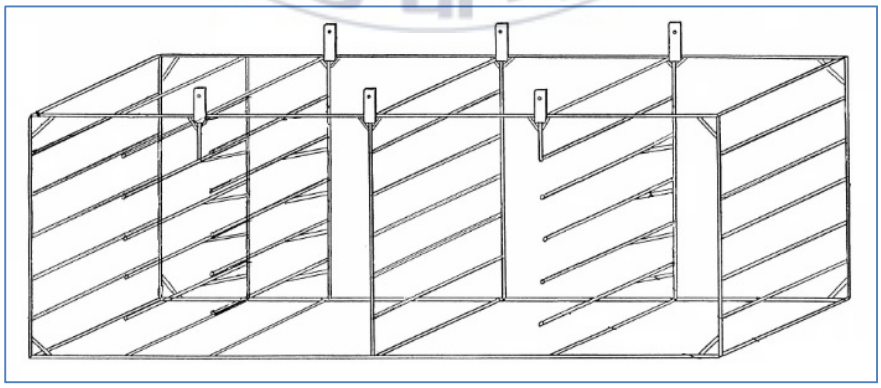
다음 장에서는 2장의 RDHS 수리모형과 3장의 분지한계 기반의 휴리스틱 일정계획에 4장에서의 확인된 랙이라는 현실적인 문제에 대한 최적화 모형 및 휴리스틱 알고리즘에 대해 알아보도록 하겠다.

V. 현실적 문제에 대한 최적화 모형 및 휴리스틱 알고리즘

1. 개요

본 장에서는 2장에서 관찰하였던 RDHS 수리모형과 3장의 분지한계 기반의 휴리스틱 일정계획에 4장에서 확인한 제약조건을 추가한 확장형 수리모형을 제안하고자 한다. 추가된 제약조건은 랙에 대한 문제이며 랙 제약조건을 반영한 수리모형에 대하여 예제를 통해 유효성을 살펴보고, 그 다음 3장에서 제안하였던 분지한계 기반의 휴리스틱 모형을 이 모형에 적용해 보고자 한다. 그리하여 사례기업과 같은 현실적인 규모의 문제를 고려할 때 합리적인 계산시간에 최적의 해를 제공할 수 있는 휴리스틱 알고리즘을 제안하려고 한다.

화공처리공정의 구성은 [그림 1-1]과 같이 탱크, 호이스트, 투입 버퍼, 완료 버퍼와 랙으로 이루어진다. 화공처리공정은 화학처리용 액체가 채워진 탱크에서 작업을 처리하게 되는데 각 부품이 필요한 탱크들을 순서대로 방문하여 탱크 별로 정해진 시간 동안 부품을 탱크에 담그고 꺼내어 다음 탱크로 이동하는 공정으로 이루어진다. 이 때 부품을 호이스트로 이동하기 위해 필요한 장비가 랙이다. [그림 5-1]은 부품을 고정하는 랙을 그림으로 나타낸 것이다.



[그림 5-1] 부품을 고정하는 랙

랙이 필요한 이유는 첫 번째 부품을 호이스트로 이동할 수 있게 해준다. 호이스트로 이동하려면 호이스트에 부품을 걸 수 있어야 하는데 항공산업의 부품들은 크기 및 형상이 다양하여 부품 자체로는 호이스트에 걸 수 있는 방법이 없다. 그렇기 때문에 호이스트에 걸 수 있는 장비인 랙이 필요한 것이다.

두 번째는 호이스트 이동 시 부품이 떨어지는 것을 방지한다. 호이스트로 이동 시 호이스트의 움직임으로 인해 흔들림이나 진동이 발생된다. 부품이 고정되어 있지 않으면 호이스트 이동 중 호이스트의 흔들림이나 진동으로 인해 부품이 떨어질 수 있는데 부품이 떨어지면 부품 손상으로 결함이 발생되거나 화학 탱크에 빠져 부품을 찾지 못하게 된다. 그러므로 호이스트 이동 시 부품이 떨어지는 것을 방지하기 위해 부품을 고정할 수 있는 랙이 필요한 것이다.

세 번째는 부품들의 간격을 유지할 수 있게 해주어 부품에 화학처리 용액들이 잘 침투될 수 있도록 하여준다. 부품들은 화공처리를 위해 화학처리용 액체가 채워져 있는 탱크에 담겨지게 되는데, 부품들이 일정한 간격을 유지하지 않고 서로 겹쳐 있으면 겹쳐진 부위에 화학처리 용액들이 잘 침투되지 않아 부품 표면에 화공처리가 제대로 수행되지 않으므로 결함이 발생하게 된다. 그렇기 때문에 부품을 일정한 간격을 두고 부품간에 서로 간섭이 일어나지 않게 하기 위해 부품을 고정할 수 있는 랙이 필요한 것이다.

네 번째는 한번에 많은 부품을 처리하기 위해 랙을 사용한다. 화공처리는 공정 별로 이루어지게 되는데 사례기업의 경우 30개의 탱크로 16개의 각기 다른 화공처리 공정으로 이루어져 있다. 이 화공처리공정은 짧게는 1시간에서 길게는 26시간이며 평균 2.5시간 정도 수행시간이 소요되며, 하루에 11개 정도의 작업을 수행할 수 있다. 화공처리 공정의 수행 시간이 길고 항공기의 부품의 수가 많으므로 한번 작업할 때 동일한 공정의 부품을 최대한 많이 처리할 수 있어야 한다. 랙에 부품을 크기 별로 고정할 수 있게 제작을 하면 한번에 처리할 수 있는 부품의 수를 많이 고정시킬 수 있게 된다. 사례기업의 경우 랙은 대, 중, 소 크기의 부품을 고정 시킬 수 있게 3가지 타입의 랙을 가

지고 있으며, 가장 효율적으로 공정을 처리할 수 있도록 Batch를 구성하여 화공처리 작업을 수행하게 된다. 이 Batch의 부품 수는 랙에 고정 할 수 있는 부품의 수로 결정되게 된다.

랙 이라는 현실적인 문제는 2장에서 관찰하였던 RDHS 모형에는 반영되어 있지 않았다. 이 장에서는 2장 4.3절의 수리모형에 랙 이라는 제약조건을 추가한 확장형 수리모형을 제안하고자 한다.

Zhao et al.(2013)은 서로 다른 화공처리공정을 수행하는 다양한 작업이 시스템에 무작위(또는 동적)로 도착하고, 작업과 함께 재조정 되어야 하는 화공처리시스템의 일정계획 문제에 대해 MILP 모델을 제안하였다.

Tian et al.(2013)은 Zhao et al.(2013)의 모델에서 모델의 제약과 변수의 수를 줄이기 위해 몇 가지 제약을 재 구성하였다. Feng et al. (2015)는 Zhao et al.(2013) 와 Tian et al.(2013)이 발표한 모델을 수정하여 연구하였다.

그러나 랙 제약조건은 호이스트 생산시스템에 존재하는 상당히 일반적인 문제이지만 기존 연구에서 이 문제를 해결하기 위한 연구 내용은 찾을 수가 없었다. 그래서 이 장에서는 랙 제약 조건을 고려하여 Zhao al al.(2013), Tian et al.(2013), Feng et al.(2015)의 모델을 수정한 확장된 새로운 MILP 모델을 제안하고자 한다. 또한 사례 기업과 같은 현실적인 산업 규모의 문제를 고려할 때 MILP 모델이 합리적인 계산 시간에 최적의 해답을 제공 할 수 없으므로 짧은 계산 시간 내에 해답을 찾기 위한 휴리스틱을 제시하였다.

본 장의 나머지 부분은 다음과 같이 구성하였다. 제 2절 문제설명 및 모델링에서는 고려된 문제와 MILP 모델에 대해 설명을 하였다. 그리고 랙 제약 조건에 대한 가정과 변수를 정의하였으며, 확장된 MILP 모델을 예제를 통하여 실험해 보았다. 그 다음에 제 3절에서는 휴리스틱 알고리즘의 분지정책과 하한계산 그리고 축소된 분지한계 알고리즘에 대해 설명하였다. 제 4절에서는 휴리스틱 알고리즘의 성과를 평가하기 위해 수치 실험을 소개하고 실험의 결과를 제시하였다. 마지막으로 제5절에서 이 장을 요약하였다.

2. 문제 설명 및 모델링

앞에서 언급한 바와 같이 호이스트 생산시스템의 일정계획에서 랙 제약조건을 본 연구에서 고려하였다. 작업은 제한 없이 투입 버퍼에 도착하지만, 투입 버퍼나 완료 버퍼에 빈 랙이 없으면 랙에 부품을 장착할 수 없으므로 탱크 라인으로 진행할 수 없게 된다.

부품이 장착된 랙의 작업이 액침공정을 완료하게 되면 랙이 완료 버퍼로 이동하게 되고, 부품은 완료 버퍼에 있는 랙에서 분리되게 된다. 부품이 분리된 빈 랙은 새로운 작업이 필요할 때까지 완료 버퍼에서 남아있게 되며, 새로운 작업이 필요할 때 투입 버퍼로 이동하게 된다.

랙 제약 조건을 포함한 시스템의 특징은 몇 가지 가정에 의해 설명 될 수 있다. 다음은 3장 2절의 가정에 랙 제약조건으로 인해 수정되거나 추가되는 가정들만 설명한 것이다.

(1) 탱크 수는 유한하다. 첫 번째 탱크(탱크 1)는 투입 버퍼이고, 두 번째 탱크(탱크 2)는 작업이 랙에 설치되는 공간이다. 탱크 3에서 m-1은 액침공정의 실제 탱크이며, 마지막 탱크(탱크 m)는 작업이 랙에서 분리되고 랙이 새로운 작업을 기다리는 완료 버퍼이다.

(2) 탱크에서 작업을 처리하는 시간은 탱크와 작업에 따라 달라지는 지정된 시간 창 내에 있어야 한다. 특히, 탱크 1의 경우 처리 시간은 작업이 라인으로 진행할 때까지의 대기 시간을 의미하고 시간 창은 [0, ∞] 이다. 탱크 2의 처리 시간은 부품 설치 시간과 대기 시간을 의미하며, 시간 창은 [설치 시간, ∞] 이다.

여기서 설치 시간은 작업에 대해 주어진 상수로 간주된다. 완료 버퍼의 작업은 제거 작업이 완료 되자마자 시스템을 나가기 때문에 처리 시간의 하한과 상한은 동일하며 그 값은 단지 또 다른 상수로 가정하는 작업을

제거하는 시간뿐이다.

(3) 완료된 작업을 배출한 빈 랙은 완료 버퍼에 있으며, 투입 버퍼의 작업이 라인으로 진행할 준비가 완료되면 탱크 2로 이동하게 된다. 빈 랙이 탱크 2에 도착하면 투입 버퍼에 있는 새 작업의 부품이 작업자에 의해 랙에 장착되게 된다.

(4) 탱크 1에서 탱크 2로의 실제 호이스트 이동은 없지만, 빈 랙의 운반을 고려하기 위해 탱크 1에서 탱크 2로의 로드 된 호이스트 이동 시간은 완료 버퍼에서 탱크 2로의 로드 된 호이스트 이동 시간이라고 가정한다.

(5) 문제의 목적은 호이스트 생산라인에서 모든 n개의 작업이 완료 버퍼에 도착하고 랙에서 각 작업의 제거가 완료되는 시간으로 정의되는 전체 완료 시간을 최소화하는 것이다.

이 연구에서 표기법은 2장 4.3절에서 설명한 변수에 랙 제약 조건에 따른 몇 가지 새로운 변수를 채택하고 일부 변수는 버리고 일부 변수는 수정하였다.

<Parameters>

i, j : 작업을 나타내는 지표 (i, j = 1, 2, ..., n)

r, s, g : 탱크를 나타내는 지표 (r, s, g = 1, 2, ..., m)

q_i : 시간 0 에서 작업 i 의 현재 위치 (q_i = 1, 2, ..., m)

e_i : 시간 0 의 현재 위치에서 작업 i 의 경과 시간

h : 시간 0 에서 호이스트의 현재 위치 (h = 1, 2, ..., m)

c_r : 탱크 r 의 용량

R : 시스템의 총 랙 수

$\alpha_{i,r}$: 탱크 r 이후에 작업 i 가 이동해야 하는 탱크

$\underline{P_{i,r}}$: 탱크 r 에서 작업 i 의 처리 시간의 하한

$\bar{P}_{i,r}$: 탱크 r 에서 작업 i 의 처리 시간의 상한

$\theta_{r,s}$: 탱크 r 에서 탱크 s 까지의 로드 된 호이스트 이동 시간

$\delta_{r,s}$: 탱크 r 에서 탱크 s 까지의 빈 호이스트 이동 시간

M : 최대 공배수

ε : 최소 공약수

<Decision variables>

T : 시스템의 완료 시간.

$t_{i,r}$: 작업 i 가 탱크 r 을 벗어날 때.

b_i : 작업 i 에 랙이 할당 된 시간.

$w_{i,r,j,s}$: 호이스트가 탱크 r 에서 작업 i 를 선택한 후 탱크 j 에서 작업 j 를 선택하면 1, 그렇지 않으면 0.

$u_{i,j,r}$: 작업 j 가 탱크 r 에 도착하기 전에 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 0.

$v_{i,j,r}$: 작업 j 가 탱크 r 에 도착한 후에 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 0.

$x_{i,j,r}$: 작업 j 가 탱크 r 에 머물러있을 때 작업 i 가 탱크 r 에 도착하면 1, 그렇지 않으면 0.

$\lambda_{i,j}$: 작업 j 가 다른 랙에서 해제되기 전에 랙이 작업 i 에 할당된 경우 1, 그렇지 않으면 0.

$\mu_{i,j}$: 작업 j 에 다른 랙이 할당된 후 랙이 작업 i 에 할당된 경우 1, 그렇지 않으면 0.

$z_{i,j}$: 작업 j 가 다른 랙을 차지할 때 랙이 작업 i 에 할당된 경우 1, 그렇지 않으면 0.

위 표기법을 이용해서 2장 4.3절에 제안한 MILP 모형에 랙 제약조건의 문제를 추가하여 다음과 같이 확장된 MILP 모델로 나타낼 수 있다.

Minimize T (1)

subject to

$\underline{P}_{i,q_i} \le t_{i,q_i} + e_i \le \bar{P}_{i,q_i}$ for all i with $q_i > 1$ (2)

$t_{i,q_i} \ge \delta_{h,q_i}$ for all i with $q_i < m$ (3)

$\underline{P}_{i,r} \le t_{i,r} - t_{i,s} - \theta_{s,r} \le \bar{P}_{i,r}$ for all i, r, s with $\alpha_{i,s} = r$ (4)

$t_{i,m} = P_{i,m} - e_i$ for all i with $q_i = m$ (5)

$t_{i,m} \le T$ for all i (6)

$t_{j,s} - t_{i,r} \ge \theta_{r,\alpha_{i,r}} + \delta_{\alpha_{i,r}s} - M(1 - w_{i,r,j,s})$

for all i, r, j, s with $\alpha_{i,r} \ne 0$, $\alpha_{j,s} \ne 0$ and $(i,r) \ne (j,s)$ (7)

$w_{i,r,j,s} + w_{j,s,i,r} = 1$

for all i, r, j, s with $\alpha_{i,r} \ne 0$, $\alpha_{j,s} \ne 0$ and $(i,r) \ne (j,s)$ (8)

$-Mu_{i,j,r} + \varepsilon \le t_{i,s} + \theta_{s,r} - t_{j,r} \le M(1 - u_{i,j,r})$

for all i, r, j, s with $\alpha_{j,r} \ne 0$, $\alpha_{i,s} \ne 0$ and $i \ne j$ (9)

$-Mv_{i,j,q_j} + \varepsilon \le -t_{i,s} - \theta_{s,q_j} \le M(1 - v_{i,j,q_j})$

for all i, j, s with $\alpha_{i,s} = q_j$ and $i \ne j$ (10)

$-Mv_{i,j,r} + \varepsilon \le t_{j,g} + \theta_{g,r} - t_{i,s} - \theta_{s,r} \le M(1 - v_{i,j,r})$ for all i, r, j, s, g

with $q_j \ne r$, $\alpha_{j,r} \ne 0$, $\alpha_{j,g} = r$, $\alpha_{i,s} = r$ and $i \ne j$ (11)

$2x_{i,j,r} \le u_{i,j,r} + v_{i,j,r} \le 1 + x_{i,j,r}$

for all i, j, r with $\alpha_{i,r} \ne 0$, $\alpha_{j,r} \ne 0$ and $i \ne j$ (12)

$\sum_{j \ne i, \alpha_{j,r} \ne 0} x_{i,j,r} \le c_r - 1$ for all i, r with $\alpha_{i,r} \ne 0$ and $i < r < m$ (13)

$b_i = 0$ for all i with $\alpha_{i,1} = 0$ (14)

$b_i = t_{i,1}$ for all i with $\alpha_{i,1} > 0$ (15)

$-M\lambda_{i,j} + \varepsilon \le b_i - t_{j,m} \le M(1 - \lambda_{i,j})$ for all $i \ne j$ (16)

$-M\mu_{i,j} + \varepsilon \le b_j - b_i \le M(1 - \mu_{i,j})$ for all $i \ne j$ (17)

$2z_{i,j} \le \lambda_{i,j} + \mu_{i,j} \le 1 + z_{i,j}$ for all $i \ne j$ (18)

$$\sum_{j\neq i} z_{i,j} \leq R-1 \text{ for all } i \quad (19)$$

$w_{i,r,j,s}$: 모든 i, r, j 및 s에 대한 0/1 정수

$u_{i,j,r}$, $v_{i,j,r}$, $x_{i,j,r}$: 모든 i, j, r 에 대한 0/1 의 정수

$\lambda_{i,j}$, $\mu_{i,j}$, $z_{i,j}$: 모든 i와 j에 대한 0/1 정수

이 모델에서 식 (1) ~ (13)은 Zhao et al.(2013), Tian et al.(2013)와 거의 동일한 의미를 갖는다. 식 (1)은 이 문제의 목적이 완료 시간을 최소화 하는 것이라는 걸 보여주고 있다.

식 (2)에서, e_i는 현재 탱크 q_i에서의 작업 i의 경과 시간이고, t_{i,q_i}는 작업이 탱크를 떠날 때의 시간이므로, $t_{i,q_i}+e_i$ 는 탱크에서 작업의 총 처리 시간을 의미한다. 식 (3)은 작업 i가 현재 탱크 q_i를 떠나는 시간이 현재 위치에서 탱크까지 빈 호이스트 이동 시간보다 빠를 수 없음을 보여준다.

식 (4)에서 $t_{i,s}+\theta_{s,r}$는 탱크 r에서의 작업 i의 도착 시간이기 때문에 r, $t_{i,r}-t_{i,s}-\theta_{s,r}$은 탱크에서 작업의 처리 시간이다.

작업이 시간 0 ($q_i=m$)에서 완료 버퍼에 머무르는 경우, 작업의 완료 시간은 식 (5)에 의해 계산 될 수 있다. 식 (6)은 모든 작업의 완료 시간과 전체 완료 시간 사이의 관계를 보여준다. 식 (7)과 식 (8)은 두 개의 로드 된 호이스트 이동 사이에서 호이스트의 적절한 이동 시간을 보장하여야 한다.

식 (9)에서 (12)는 j가 탱크 r에 머물러 있을 때 즉, 작업 i가 탱크 r에 도착할 때 변수 $x_{i,j,r}$가 값 1을 갖는다고 설정한다. 즉, 탱크 r에 작업 j의 도착 시간(=$t_{j,s}+\theta_{s,r}$) ≤ 탱크 r에서의 작업 i의 도착 시간(=$t_{i,s}+\theta_{s,r}$) ≤ 탱크 r에서의 작업 j의 출발 시간(= $t_{j,r}$) 이다. 식 (13)은 각 탱크에 대한 용량 제약을 나타낸다.

랙 제약을 표현하기 위해 식 (14)에서 (19)를 새로이 연구에 포함시켰다. 식 (14)로부터 랙이 작업 i에 할당된 시간은 작업이 시간 0 에서 처리 중이면

0 으로 간주된다. 반면, 식 (15)는 작업이 투입 버퍼에 시간 0 인 경우 랙 할당 시간이 $t_{i,1}$이라는 것을 의미한다. $t_{i,1}$은 작업 1이 탱크 1을 떠날 때를 의미하지만 호이스트는 탱크 1로 돌아가지 않는다. $t_{i,1}$의 실제 의미는 완료 버퍼에서 작업 i에 대한 빈 랙의 출발 시간이다.

t_{jm} 은 식 (9)에서 (12)와 유사하게 작업 j가 랙에서 분리되는 시간이기 때문에 식 (16)에서 (18)은 랙이 할당된 경우에만 변수 $z_{i,j}$의 값이 1임을 보장한다. 작업 j가 다른 랙을 차지할 때 작업 i로, 다시 말해 작업 j (=b_j) ≤ 랙 i에 작업 시간 i(=b_i) ≤ 작업 j의 랙 해제 시간(=$t_{j,m}$)에 시간을 할당하는 랙이다. 식 (19)는 랙이 작업 i에 할당 될 때 작업에 할당된 랙의 총 수가 R 보다 적음을 나타낸다.

이제 모델의 유효성을 검사하기 위해 랙 제약조건이 추가된 간단한 예제 문제를 살펴보도록 하겠다. 사용되는 데이터는 다음과 같다.

m = 8, n = 5, R = 3, h = 4, q_i = (8, 6, 4, 1, 1), e_i= (15, 5, 1, 12, 0),

c_r= (∞, 2, 1, 1, 2, 1, 1, ∞),

$$\alpha_{i,r}=\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&7&8&0\\0&0&0&5&6&7&8&0\\2&4&0&5&6&7&8&0\\2&3&4&5&8&0&0&0\end{bmatrix},\quad \underline{P}_{i,r}=\begin{bmatrix}0&0&0&0&0&0&0&30\\0&0&0&0&0&15&10&30\\0&0&0&10&20&15&10&30\\0&30&0&10&20&15&10&30\\0&30&10&10&20&0&0&30\end{bmatrix},$$

$$\bar{P}_{ir}=\begin{bmatrix}0&0&0&0&0&0&0&30\\0&0&0&0&0&30&30&30\\0&0&0&25&40&30&30&30\\\infty&\infty&0&25&40&30&30&30\\\infty&\infty&25&25&40&0&0&30\end{bmatrix}$$

$$\delta_{r,s}=\begin{bmatrix}0&1&2&3&4&5&6&7\\6&0&1&2&3&4&5&6\\5&1&0&1&2&3&4&5\\4&2&1&0&1&2&3&4\\3&3&2&1&0&1&2&3\\2&4&3&2&1&0&1&2\\1&5&4&3&2&1&0&1\\0&6&5&4&3&2&1&0\end{bmatrix},\quad \theta_{r,s}=\begin{bmatrix}0&11&7&8&9&10&11&12\\6&0&6&7&8&9&10&11\\7&6&0&6&7&8&9&10\\8&7&6&0&6&7&8&9\\9&8&7&6&0&6&7&8\\10&9&8&7&6&0&6&7\\11&10&9&8&7&6&0&6\\12&11&10&9&8&7&6&0\end{bmatrix}$$

예제를 보면 투입 버퍼와 랙 설치 공간 및 완료 버퍼를 포함한 탱크 수는 8 개이다. 새 작업과 진행중인 작업을 포함한 작업 수는 5 개이다. 시스템에 3 개의 랙이 있으며 호이스트는 시간 0 상태로 탱크 4에 있다.

q_i 와 e_i의 데이터는 각 작업이 머무르는 곳과 시간이 0 일 때 머물렀던 시간을 보여준다. 예를 들어 작업 1은 일정이 잡히면 15 분 동안 탱크 8에 머물려 있어야 한다.

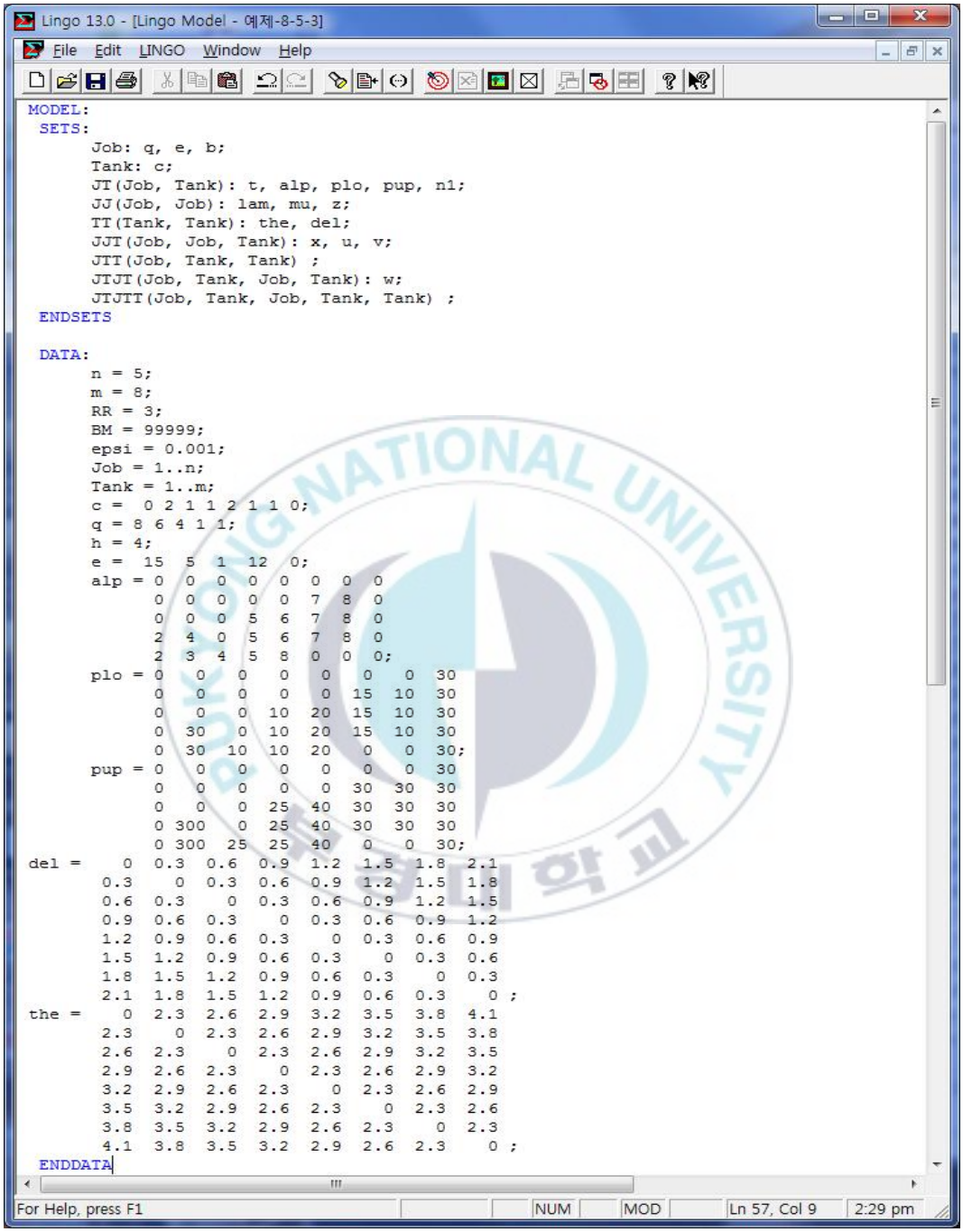
c_r의 데이터는 투입 버퍼와 완료 버퍼가 무한대의 용량을 가지고 있으며, 랙에 2 개의 작업만 동시에 장착할 수 있다는 것을 보여주고 있다. 화학 처리가 이루어지는 5개 탱크의 용량은 각각 1, 1, 2, 1 및 1이다.

$\alpha_{i,r}$ 행렬은 작업에 대한 액침순서를 나타낸다. 예를 들어 행렬의 마지막 행은 작업 5의 액침순서가 1-2-3-4-5-8 임을 나타낸다. 다음 두 행렬 $\underline{P}_{i,r}$ 와 $\bar{P}_{i,r}$ 은 각 탱크의 각 작업에 대한 시간 창을 표시한다.

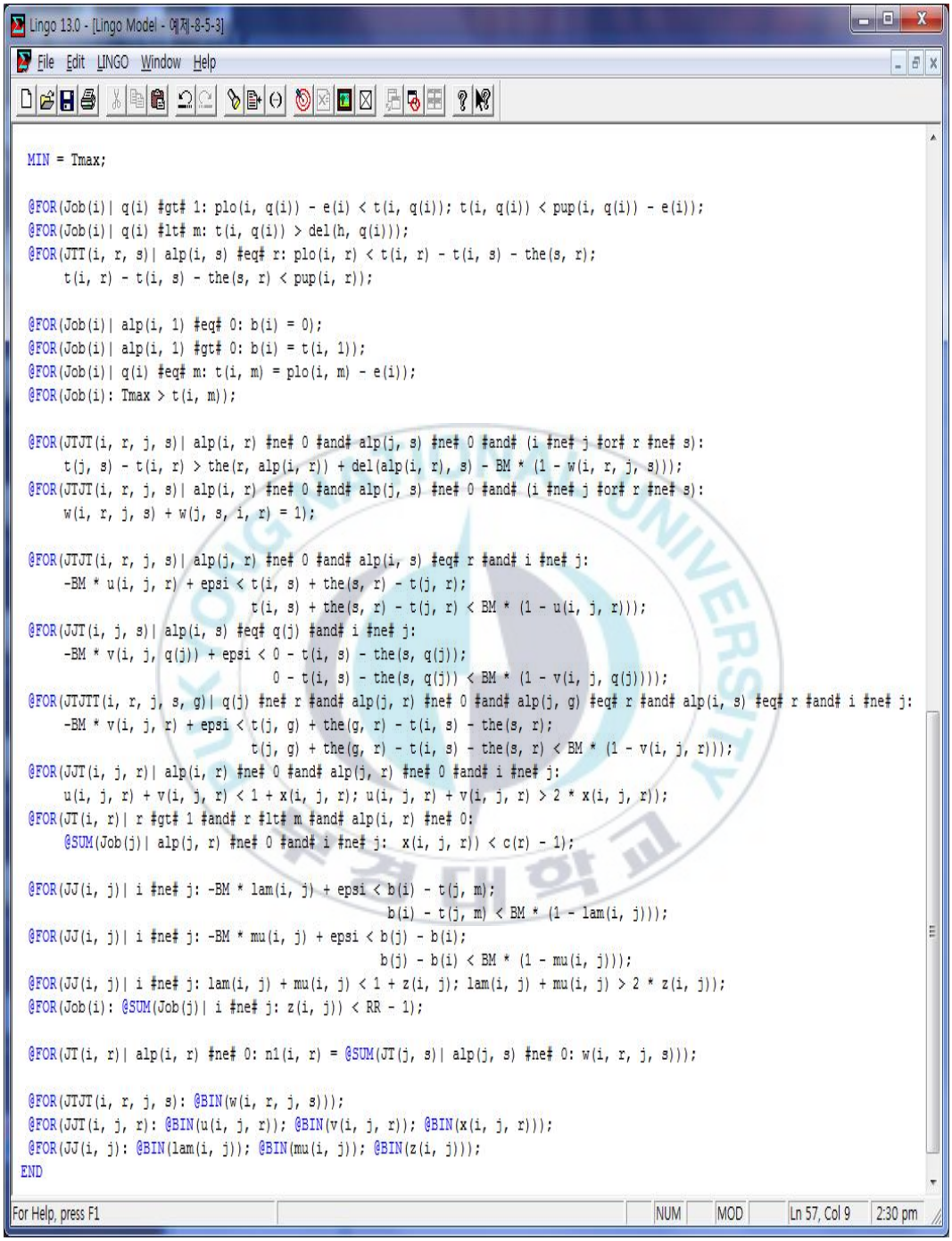
$\delta_{r,s}$ 행렬은 빈 호이스트 이동 시간을 나타낸다. 이 행렬의 첫 번째 열은 마지막 열과 같다. 이는 빈 호이스트가 탱크 1로 이동한다는 것은 실제로 빈 랙을 가져가기 위해 빈 호이스트가 탱크 8로 이동했기 때문이다.

한편 부품이 장착된 호이스트 이동 시간을 나타내는 행렬 $\theta_{r,s}$ 에서 요소 (1, 2)는 탱크 8에서 탱크 2로 빈 랙을 가진 호이스트의 이동 시간을 의미하기 때문에 요소 (8, 2)와 동일하게 된다.

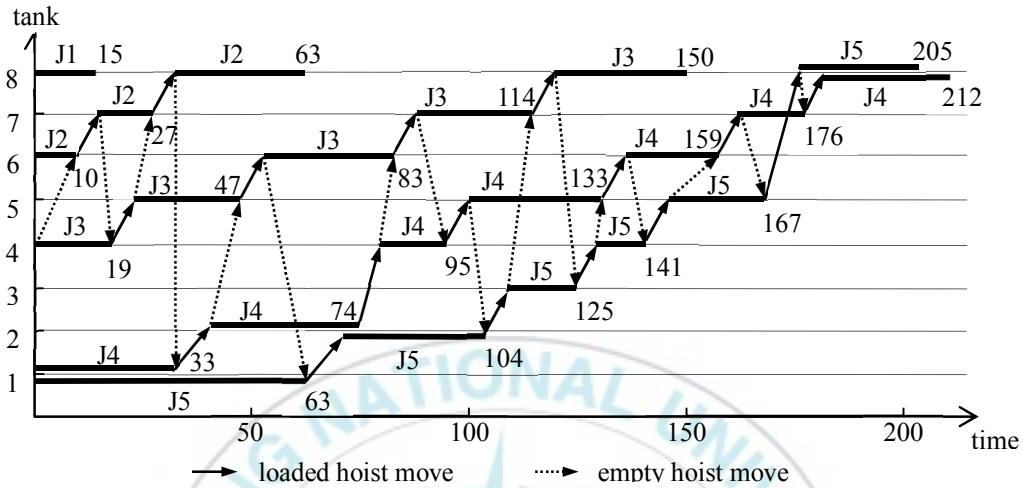
모델과 예제 데이터를 최적화 소프트웨어 LINGO에서 [그림 5-2]와 [그림 5-3]과 같이 적용하면 2.6GHz PC에서 매우 짧은 계산 시간(0.01초 미만)으로 최적의 해를 구할 수 있다. [그림 5-4]는 예제 문제의 호이스트 최적 작업 순서가 (2, 3, 2, 4, 3, 5, 4, 3, 4, 5, 3, 5, 4, 5, 4, 5, 4) 이라는 것을 보여주고 있으며, 이때의 완료 시간은 212 이다..



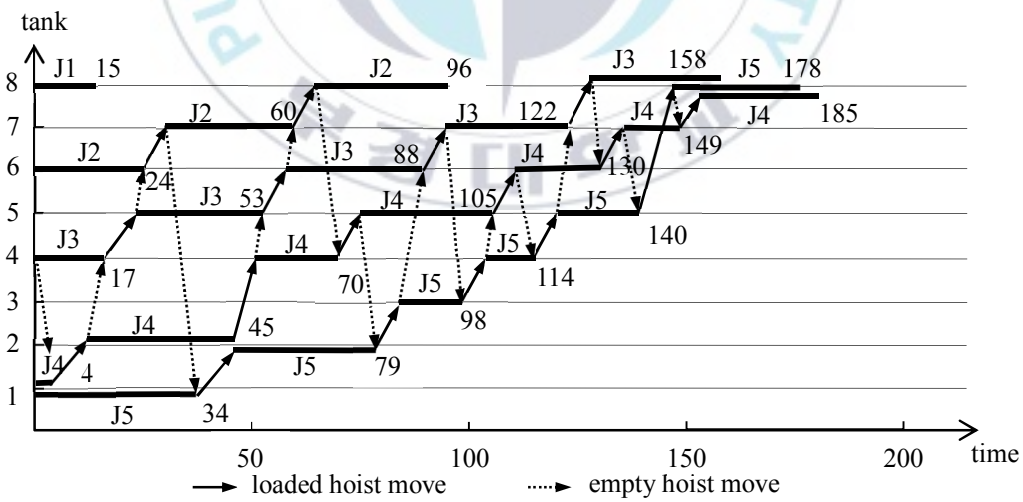
[그림 5-2] 예제의 LINGO 프로그램 데이터 부분



[그림 5-3] 예제의 LINGO 프로그램 모형 부분



[그림 5-4] 예제 문제의 최적 일정



[그림 5-5] 랙 제약이 없는 최적의 일정

[그림 5-4]에서 탱크 1~7 에 표시된 숫자는 호이스트 작업의 시작 시간 즉, $t_{i,r}$ 이다. 반면 탱크 8의 숫자는 작업의 완료 시간이다. 여기서 호이스트는 독립적으로 이동하는 조건이다. 그림은 또한 시스템의 완료 시간이 212 임을 보여준다. 또한 이 그림에서 결합 된 랙의 수 또는 2~8 번 탱크에 동시에 머무르는 작업의 수는 항상 R(=3)의 값보다 작거나 같음을 알 수 있다.

랙 제약 조건의 영향을 평가하기 위해 랙 제약이 없는 상태에서 동일한 문제에 대해 실험을 실시해 보았다. 결과는 [그림 5-5]에 나타나 있으며, 호이스트 작업의 최적 순서는 변경되었고 이때의 완료 시간은 185 이었다. 랙 제약(R=3) 이 있을 때의 212 보다 작다는 것을 확인 할 수 있었다.

[그림 5-5]에서 시간 4에 작업 4가 탱크 1에서 탱크 2로 이동되었음을 알 수 있다. 이 시간 이후 작업 1이 시스템을 떠날 때까지 생산 라인의 랙 수는 시간이 15까지는 4가 된다. 이 두 수치를 비교해 보면 랙 제약 조건의 영향과 이 연구가 필요한 이유를 쉽게 이해할 수 있다.

예제 문제의 결과는 앞서 2장에서 설명한 것처럼 $\alpha_{i,r}$ 행렬에 있는 0이 아닌 요소의 순열로 문제의 해를 나타낼 수 있음을 보여준다. 즉, 이 예제에서는 작업 2에 대해 두 개의 0이 아닌 요소가 있고, 작업 3에는 4개, 작업 4에는 6개, 작업 5에는 5개의 요소가 있다. 따라서 실행 불가능한 해와 가능한 해를 포함하여 해의 총 수는 17개의 작업 번호(2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5)를 일렬로 세우는 방법의 수인 17! / 2! 4! 6! 5! 개가 되는 것이다.

이 예제와 같이 소규모의 문제는 LINGO와 같은 최적화 프로그램으로 쉽게 최적해를 찾을 수 있었으나, Lei and Wang(1989)에서 보여지듯이 단일 항목 호이스트 생산계획 문제 조차도 NP-hard 에 속하는 문제이므로 문제의 크기가 커지면 짧은 시간에 최적해를 구할 수가 없었다.

이러한 사실을 실험해 보기 위해 본 연구에서는 탱크의 수가 12개인

문제를 만들어서 LINGO로 최적해를 구해보았다. 그 결과 7개의 작업에 대해 8초 내에 최적의 해를 찾아냈으며, 8개의 작업에 대해서는 189초 내에, 그리고 9개의 작업에 대해 6,917초 내에 최적의 해를 찾아 낼 수 있었다.

작업 수가 10 개로 늘어나면 시스템은 3시간 이내에 최적의 해를 찾아내지 못했다. 앞서 언급했듯이 이 연구에 동기를 부여한 사례 기업의 탱크 수는 30개이므로 MILP 모델은 사례 기업에 적용할 수가 없었다.

따라서 이 연구에서는 합리적인 계산 시간에서 일정계획 문제에 대한 최적의 해를 얻을 수 있는 휴리스틱 알고리즘 개발하는데 중점을 두었다.

3. 휴리스틱 알고리즘

이 장에서 제안된 휴리스틱 방법은 단순화된 분지한계 알고리즘으로, 아래에 설명된 것처럼 분지의 수를 감소시키는 것이 목적이다.

3.1 분지 정책

사례 기업의 문제에 대한 분지한계 알고리즘에서 노드는 복제된 작업 집합의 순서가 지정된 하위 집합으로 정의된 부분 일정을 의미한다. 예를 들어, [그림 5-4]의 최적 해에서 호이스트를 3회 로드 한 후의 부분 일정은 (2, 3, 2)가 된다. 분지한계 알고리즘의 노드 또는 부분 일정에서 일반적으로 n 개의 분지가 필요하게 된다.

그러나 이 연구에서 일부 분지는 알고리즘을 단순화하기 위해 추가 고려 사항에서 제외하였다. 자세한 절차는 다음과 같다.

(1) 이미 완료 버퍼에 도착한 작업은 분지 후보에서 제외하였다. 이전 예제에서 부분 일정계획 (2, 3, 2)의 노드에 있는 경우, 작업 1, 2는 분지에서 제외 되어야 하며 이 노드의 분지 수는 작업 3, 4, 5로 3으로 줄어들게 된다.

(2) 후속 탱크의 용량을 고려할 때 다음 탱크로 갈 수 없는 작업은 분지 대상에서 제외하였다. 이전 예제에서 부분 일정 (3, 4) 노드에 있는 경우 작업 3의 다음 탱크인 탱크 6의 용량이 1이고 작업 2가 탱크 6에 남아 있기 때문에 작업 3을 분지에서 제외해야 한다.

(3) 사용 가능한 랙이 완료 버퍼에 없으면 투입 버퍼의 작업이 분지 대상 후보에서 제외된다. 앞의 예제에서 R이 3 일 때 부분 일정 (2, 3, 2, 4)의 노드에 있다면 작업 2, 3 및 4가 이미 3 개의 랙을 차지하고 있으므로 작업 5는 분지에서 제외되어야 한다.

(4) 호이스트가 이동하려는 다음 작업을 들어올리기 위해 오래 기다려야 하는 경우, 작업을 분지 후보 대상에서 제외하는 것이 좋다. 이전 예제에서는 부분 일정 (4, 3)의 노드에 있다고 가정하였다. [그림 5-5]는 탱크가 작업 2에 의해 점유되어 작업 3이 다음 탱크로 갈 수 없다는 것을 보여준다. 따라서 분지 작업의 후보는 작업 2, 4 및 5가 되게 된다.

이제 호이스트 대기 시간이 호이스트의 다음 작업을 위해 각 작업이 선택되었다고 가정해야 한다. [그림 5-5]는 부분 일정 (4, 3)이 시간 23 (= 17 + 6)에서 수행되는 것을 보여주고 있다. 다음 작업에 대해 작업 2가 선택되면 호이스트가 탱크 6으로 갈 수 있는 시간을 계산해야 하며 부분 일정 (4, 3) 이후의 호이스트 위치는 탱크 5이기 때문에 시간 24 (= 23 + 1)가 된다. 탱크 6에서 작업 2의 경과 시간은 29 (= 5 + 24)이며, 하한 시간이 15이므로 대기시간 없이 즉시 작업 2를 들어 올릴 수 있다. 동일한 절차를 사용하여 작업 4와 5에 대한 호이스트 대기 시간을 찾을 수 있으며 결과는 각각의 대기시간이 19와 0이 되게 된다.

호이스트 대기 시간이 계산되면 대기 시간이 0 인 모든 작업이 분지를 위해 선택되게 된다. 대기 시간이 0 인 작업 수가 2보다 크거나 같으면 분지 작업이 더 이상 고려되지 않는다. 그러나 대기 시간이 0 인 작업이 없거나 하나의 작업인 경우 두 개의 분지가 두 대기 열에 대해 대기 시간이 0 이므로 대기 시간이 가장 짧다. 예제에서 부분 일정 (4, 3)의 대기 시간이 0 인 분지 후보(즉, 작업 2와 5)와 대기 시간이 양수인 분지 후보(즉, 작업 4)가 두 개 있으므로 두 개의 노드가 부분 일정 (4, 3, 2) 및 (4, 3, 5)를 생성하게 된다.

(5) 위에서처럼 분지의 수가 2로 제한되어 있지만 문제의 크기가 클 경우 알고리즘의 계산 시간이 매우 길어진다. 따라서 분지 수를 줄여야 한다.

호이스트 대기 시간이 0 인 분지 후보의 수가 2보다 작으면, (0, 1) 간격으로부터 소정의 변수로서 정의되는 BRATIO 값을 사용하여 분지의

수를 제어할 수 있다. 이렇게 하려면 현재 부분 일정에 있는 작업 수와 호이스트가 수행해야 하는 총 작업 수의 비율을 계산해야 된다. 앞의 예에서 현재 부분 일정이 (4, 3, 2)이면 비율은 3/17로 계산됨을 알 수 있다. 계산 후 비율이 BRATIO 값보다 작으면 분지 수는 2개로 설정되며, 그렇지 않고 비율이 BRATIO 값보다 크면 분지 수는 하나로 된다.

예를 들어 호이스트 대기 시간이 0 인 작업이 하나 있고, 호이스트 대기 시간이 양의 다른 작업이 있는 경우 대기 시간이 0 인 작업과 대기 시간이 가장 짧은 작업에 대해 분지가 두 개 생성되게 된다. 이 비율이 BRATIO 값보다 크면 BRATIO 값과 같거나 호이스트 대기 시간이 0 인 분지 하나가 생성되게 된다.

BRATIO 값을 사용하면 알고리즘의 계산 시간뿐 아니라 해결하려는 해의 품질도 제어 할 수 있다. 크기가 큰 문제의 경우 상대적으로 작은 BRATIO 값을 사용하여 계산 시간을 단축 할 수 있게 된다.

반면에 큰 BRATIO 값은 상대적으로 크기가 작은 문제에 사용되어 더 넓게 해답을 탐색하고 더 나은 일정을 찾게 된다. 실제 어플리케이션에서 알고리즘은 다양한 BRATIO 값으로 테스트 할 수 있으며 테스트를 통해 실제 일정 수립에 적합한 값을 선택할 수 있게 된다.

3.2 하한 계산

위에서 언급한 분지 정책에 의해 새로운 노드가 생성된 후에는 각 새 노드에 대한 부분 일정의 실행 가능성을 확인하여야 한다.

이렇게 하려면 부분 일정을 수행한 후 현재 탱크의 모든 작업에 대한 경과 시간을 계산하여 탱크의 작업 처리 시간의 상한과 비교해야 한다.

이전 예제에서 일정 (4, 3)에서 분지된 부분 일정 (4, 3, 4)을 고려해보자.

부분 일정 (4, 3, 4)이 수행 된 후 탱크 6의 작업 2가 호이스트의 다음 작업인 경우 호이스트는 탱크 6으로 넘어가야 한다.

빈 호이스트가 탱크로 갈 수 있는 가장 빠른 시간은 탱크 2에 있는 작업 4의 완료 시간이 45이고, 작업 4의 탱크 2에서 탱크 4까지의 로드 된 호이스트 이동 시간이 7이고, 작업 4의 탱크 4에서 탱크 6으로의 빈 호이스트 이동 시간은 2이다. 그러나 이때의 탱크 6에서 작업 2의 경과 시간은 처리 시간 상한인 59(= 5 + 54) 이다.

이 간단한 계산을 통해 부분 일정 (4, 3, 4)이 실행 불가능한 것으로 판명되었으며 노드를 추가 고려 대상에서 제외 할 수 있게 된다.

노드의 실행 가능성을 확인한 후에는 각 생존 노드에 대한 하한 값을 계산해야 하며, 그 값은 LB1, LB2, LB3가 정의되고 각 부분 일정에 대해 다음과 같이 계산된 MAX{LB1, LB2, LB3}이 되게 된다.

(1) 아직 완료되지 않은 각 작업에 대해 작업의 가장 빠른 완료 시간은 다른 작업을 고려하지 않고 계산된다. 그리고 나서 LB1이 그 중 최대 값이 되도록 해야 한다. 이전의 예제에서, 부분 일정 (4, 3, 2)이 방금 수행된 시간인 시간 30에서 4 개의 완료되지 않은 작업 즉, 작업 2, 3, 4 및 5가 있었다.

여기서 작업 3의 완료 시간은 다음과 같이 계산된다. 먼저 빈 호이스트는 탱크 7에서 탱크 5로 이동하여 시간 32(= 30 + 2)에 작업 3을 선택한다. 그러나, 탱크 5에서 작업 3의 최소 처리 시간이 20 분이고, 시간 23의 탱크에 이 작업이 도착했기 때문에 호이스트는 시간 43에서 작업을 선택할 수 있으므로 작업 3은 시간 49에 탱크 6에 도달 할 수 있다(= 43 + 6). 작업 3은 탱크 6의 최소 처리 시간(15 분) 후, 탱크 7의 이동 시간(6 분), 탱크 7의 최소 처리 시간(10 분) 및 탱크 8의 이동 시간(6 분) 까지의 시간 86 (= 49 + 15 + 6 + 10 + 6)에 탱크 8에 도착할 수 있다.

탱크 8에서 작업 3은 랙에서 30분 동안 해제되고 마지막으로 작업은

시간 116에서 시스템을 나갈 수 있다. 이 절차를 사용하면 다른 작업 (2, 4 및 5)의 가장 빠른 완료 시간을 마지막으로 부분 일정 (4, 3, 2)에 대한 LB1 값을 찾을 수 있게 된다.

(2) 남아있는 모든 작업이 탱크에서 처리 될 수 있는 가장 빠른 시간은 단지 하나의 용량을 가진 각 탱크에 대해 계산된다. 그리고 LB2가 그 중 최대 값이 되도록 해야 한다. 이전 예제에서 부분 일정 (4, 3, 2)으로 탱크 7을 고려하였다.

탱크에서 처리될 나머지 작업은 작업 2, 3 및 4 이다. 부분 일정 (4, 3, 2)가 수행되고 최소 처리 시간이 10 분인 경우 작업 2가 탱크 7에 막 도착 했으므로 나머지 작업 2에서 탱크 7의 최소 처리 시간은 10 분이다. 작업 2를 탱크 7에서 다음 탱크로 이동하는 호이스트 시간을 포함하여 탱크 7에서 작업 2를 완료하는 가장 빠른 시간은 46 (= 30 + 10 + 6) 이다.

작업 3과 4를 처리하는데 필요한 시간은 22이다. 즉, 탱크 7에 로드 된 호이스트 이동 시간(6분), 최소 처리 시간(10분) 및 탱크 7에서 로드 된 호이스트 이동 시간(6분) 이다. 따라서 부분 일정 (4, 3, 2)이 수행 될 때 탱크 7에서 작업 2, 3 및 4를 처리하는 가장 빠른 시간은 90(= 46 + 22 + 22) 이다.

유사한 절차를 사용하여 다른 단일 용량 탱크 (탱크 3, 4 및 6)의 나머지 작업을 처리하는데 필요한 가장 빠른 시간을 계산할 수 있으며, 마지막으로 부분 일정에 대한 LB2 값을 찾을 수 있다.

(3) 이제 모든 나머지 작업을 수행하는 데 필요한 최소 총 로드 된 호이스트 소요 시간이 계산되어야 한다. 그리고 완료 버퍼에 남아있는 부품을 제거하는 작업의 최소 처리 시간도 있다. 그러므로 LB3는 계산된 두 개의 시간 값과 부분 일정의 현재 시간의 합으로 정의된다. 부분 일정 (4, 3, 2)을 다시 한번 고려하면 17 개의 작업 중 3개가 이미 수행되었으므로

모든 작업을 완료하려면 14 개의 로드 된 호이스트 이동이 필요하게 된다. 이 14 개의 작업에 대한 총 로드 된 호이스트 이동 시간은 92(= 11 x 6 + 7 + 8 + 11)가 된다. 남아있는 모든 작업에 대한 완료 버퍼의 처리 시간은 30 분이고 현재 시간은 30이므로 LB3의 값은 부분 일정 (4, 3, 2)에 대해 처리 시간은 152(= 92 + 30 + 30)가 된다.

3.3 알고리즘

앞에서 설명한 것처럼 분지 수는 일반적인 분지한계 알고리즘보다 훨씬 줄어들게 된다. 따라서 본 논문에서 제안한 휴리스틱 알고리즘을 축소된 분지한계 알고리즘(Reduced Branch and Bound Algorithm) 이라 하며 알고리즘의 세부 절차는 다음과 같다.

축소된 분지한계 알고리즘

단계 0. 루트 노드에서 CurMin의 값은 시스템의 최소 임시 저장 기간의 현재 값으로 정의되며 ∞로 설정된다.

단계 1. 첫 번째 분지한계

단계 1.1 완료 버퍼에 도착하지 않은 모든 작업에 대해 분지를 만든다.

단계 1.2 탱크 용량 제약 조건 또는 랙 제약 조건을 위반하는 분지를 제외한다.

단계 1.3 분지의 부분 일정에 대한 각 탱크의 작업 경과 시간을 계산한다. 경과 시간이 상한을 초과하면 분지를 제외한다.

단계 1.4 분지의 부분 일정을 수행하는데 필요한 시간을 계산한다.

단계 2. 분지 할 노드가 없으면 현재 CurMin 값으로 알고리즘을 중지해야 한다. CurMin = ∞의 경우는 알고리즘이 실현 가능한 해결책을

찾지 못했음을 의미하는 것이다.

단계 3. 분지한계

단계 3.1 부분 일정을 수행하는데 필요한 최대 시간 값을 갖는 노드를 선택하고, 제 3.1 절의 정책을 사용하여 노드에서 분지한다.

단계 3.2 각 분지에서 작업 경과 시간을 확인하고 경과 시간이 상한을 초과하면 분지를 제외한다.

단계 3.3 분지의 부분 일정을 수행하는데 필요한 시간을 계산한다.

단계 3.4 분지의 부분 일정이 완료 일정 인 경우,

단계 3.4.1 전체 일정의 완료 시간을 계산한다.

단계 3.4.2 완료 시간이 CurMin 값보다 작으면 CurMin 값을 완료 시간으로 설정한다.

단계 3.4.3 분지를 더 이상 고려하지 않는다.

단계 3.5 분지의 부분 일정이 완료 일정이 아닌 경우,

단계 3.5.1 제 3.2절의 절차를 사용해 분지에 대한 하한을 계산한다.

단계 3.5.2 하한이 CurMin 값보다 큰 경우, 분지를 더 이상 고려하지 않는다.

단계 4. [단계 2]로 이동한다.

알고리즘을 검증하기 위해 C++ 언어로 프로그램을 작성하였고, 2절에서 사용된 것과 같은 예제 문제를 가지고 테스트를 수행하였다. 문제가 매우 작기 때문에 BRATIO 값을 1로 설정하였다. 즉, 모든 노드에 대한 분지의 수가 2보다 크거나 같아야 한다.

결과적으로 계산 시간은 0.01초 미만이었고, 해는 최적해와 정확히 동일했다. BRATIO 값을 0으로 설정하게 되면, 완료 시간은 223이 되어 최적해 212보다 5.2%나 더 나쁘게 나타났다. 그러나 예제 문제가 너무 작기 때문에 BRATIO 값을 1로 설정하여 테스트를 수행 할 수 있었다.

4. 수치 실험

이 논문에서 제안된 휴리스틱 알고리즘의 성과를 평가하기 위해 이전 예제 문제보다 더 큰 문제를 사용하여 수치 실험을 수행하였다. 이 문제에서 버퍼를 포함한 탱크 수는 12, 15, 18로 고정하였고, 작업 수는 6에서 8까지 설정하였다. 9개 이상의 작업은 포함하지 않았는데, 왜냐하면 LINGO 소프트웨어가 최적의 해를 찾지 못했기 때문이다.

이러한 문제를 사용하여 휴리스틱 알고리즘의 결과를 최적의 해와 비교하였다. 이를 위해 LINGO를 사용하여 각 문제에 대한 최적 해를 먼저 찾은 다음 5 개의 BRATIO 값, 즉, 0, 0.25, 0.5, 0.75 및 1.0에 대해 휴리스틱 알고리즘을 적용하여 같은 문제의 해를 구하였다. 각 문제에 대해 LINGO의 결과와 5 가지 휴리스틱 결과에 대해 완료 시간 및 계산 시간을 비교하였다. 그 결과를 <표 5-1> <표 5-2> <표 5-3>에 나타내었다.

<표 5-1> 휴리스틱 해와 최적 해의 성능 비교 (m=12)

n			6	7	8
number of nonzero elements in $\alpha_{i,r}$ matrix			28	30	36
LINGO (Optimal)		Makespan	100	100	100
		Com. Time	2	5	31
Heuristic	*BRATIO* = 0.0	Makespan	103.3	104.5	110.2
		Com. Time	<1	<1	<1
	BRATIO = 0.25	Makespan	101.7	102.3	107.5
		Com. Time	< 1	< 1	<1
	BRATIO = 0.5	Makespan	100.8	101.9	102.8
		Com. Time	< 1	<1	<1
	BRATIO = 0.75	Makespan	100.0	100.6	101.4
		Com. Time	<1	<1	<1
	BRATIO = 1.0	Makespan	100.0	100.2	100.8
		Com. Time	<1	<1	1

<표 5-1>은 탱크의 수가 12 일 때의 결과값이다. 문제의 복잡성은 문제의 행렬에서 0이 아닌 요소의 수에 크게 의존하기 때문에 표는 제 2 행의 정보를 나타내고 있다.

표를 더 잘 이해하기 위해 n = 6에 대한 열의 값을 얻는 데 사용 된 절차를 설명한다. 첫 번째로 문제의 최적 해를 LINGO를 이용하여 찾았다. 계산 시간은 2초내 이었으며 그 때의 완료 시간은 763 이었다.

다음으로 5 개의 BRATIO 값, 즉, 0, 0.25, 0.5, 0.75 및 1.0에 대해 휴리스틱 알고리즘을 실행하였다. 완료 시간은 각각 788, 776, 669, 763 및 763 이었다. 휴리스틱 알고리즘의 계산 시간은 모든 경우에 1 초 미만이었다. 최적의 완료 시간은 문제에 대해 763 이지만 비교를 표준화하기 위해 표에 100으로 표시하였다.

따라서 5 가지 BRATIO 값에 대한 완료 시간은 103.3(= 788 / 763), 101.7(= 776 / 763), 100.8(= 769 / 763), 100.0(= 763 / 763) 및 100.0(= 763 / 763) 이다. n = 7과 8에 대한 열도 동일한 절차를 사용하여 계산하였다.

<표 5-2> 및 <표 5-3>은 탱크의 수가 각각 15 및 18 일 때의 결과를 나타내고 있다. 이 표에서 행렬이 0이 아닌 요소의 수가 <표 5-1>보다 작음을 알 수 있다. 이것은 LINGO가 <표 5-1>과 숫자가 같을 때 최적의 해를 찾을 수 없었기 때문이다. <표 5-2>와 <표 5-3>의 값을 얻는 절차는 <표 5-1>과 거의 유사하다.

이 논문에서 연구된 문제의 구조는 매우 복잡하며 예제 문제는 임의로 생성할 수가 없었다. 그래서 유용한 예제를 만들기 위해 데이터를 신중하게 만들고 LINGO로 먼저 해결하여 실현 가능한 해가 있는지 여부를 확인하여야 했다. 따라서 예제가 부족하여 많은 실험을 수행한 결과를 제공할 수가 없으므로 본 논문에서 제안한 휴리스틱의 보편적인 특성을 설명하기에 어려움이 있다고 생각한다.

<표 5-2> 휴리스틱 해와 최적 해의 성능 비교 (m=15)

n			6	7	8
number of nonzero elements in $\alpha_{i,r}$ matrix			26	28	35
LINGO (Optimal)		Makespan	100	100	100
		Com. Time	3	7	290
Heuristic	*BRATIO* = 0.0	Makespan	104.2	108.4	114.7
		Com. Time	<1	<1	<1
	BRATIO = 0.25	Makespan	103.0	106.3	112.3
		Com. Time	< 1	< 1	<1
	BRATIO = 0.5	Makespan	101.6	104.9	108.4
		Com. Time	< 1	<1	1
	BRATIO = 0.75	Makespan	100.3	100.9	101.0
		Com. Time	<1	<1	2
	BRATIO = 1.0	Makespan	100.0	100.6	100.4
		Com. Time	<1	<1	4

<표 5-3> 휴리스틱 해와 최적 해의 성능 비교 (m=18)

n			6	7	8
number of nonzero elements in $\alpha_{i,r}$ matrix			22	24	30
LINGO (Optimal)		Makespan	100	100	100
		Com. Time	2	11	2,133
Heuristic	*BRATIO* = 0.0	Makespan	103.0	105.3	114.2
		Com. Time	<1	<1	<1
	BRATIO = 0.25	Makespan	101.4	103.0	110.2
		Com. Time	< 1	< 1	1
	BRATIO = 0.5	Makespan	100.5	101.2	105.4
		Com. Time	< 1	<1	9
	BRATIO = 0.75	Makespan	100.0	100.2	103.0
		Com. Time	<1	<1	12
	BRATIO = 1.0	Makespan	100.0	100.0	101.6
		Com. Time	<1	<1	18

그럼에도 불구하고 표는 매우 일관된 결과를 보여주기 때문에 표를 검토하여 다음과 같이 휴리스틱의 일부 특성을 얻을 수가 있었다.

1) 해의 품질 측면에서 볼 때 휴리스틱의 완료 시간은 상당히 좋아 보였다. 표는 최악의 결과가 최적 해보다 14.7 % 나 더 나쁜 결과를 보여주었으며 최상의 결과는 최적 해와 같았다. 계산 시간을 고려할 때 휴리스틱 방법은 MILP 모델보다 훨씬 실용적이었다. (m, n) = (18, 8)의 경우 MILP는 최적 해를 2,133 초에 찾았지만 휴리스틱은 BRATIO가 1 일 때 조차도 18 초 내에 해를 찾아내었다.
2) LINGO에서 주어진 m 값에 대해 최적 해를 찾는데 걸리는 시간은 n 값이 커질수록 빠르게 증가하였다. 그래서 큰 m과 n의 문제에 대해 최적의 해를 찾을 수는 없었다.
3) n 값이 증가하면 휴리스틱으로 계산한 해의 품질이 나빠졌다. 다시 말하면, 휴리스틱의 완료 시간이 증가하였고 휴리스틱의 계산 시간이 길어졌다.
4) 이 문제의 경우에 휴리스틱에서 사용하는 BRATIO 값이 클수록 발견 할 수 있는 더 나은 해(더 작은 완료 시간)를 찾아내었으며 더 긴 계산 시간이 소요되었다.

그러므로 계산 시간이 너무 길어서 문제에 대한 해답을 찾을 수 없다면 비교적 작은 BRATIO 값을 사용하여 합리적인 계산 시간에 문제의 해를 찾을 수 있었다. 실제로 <표 5-1>에서 <표 5-3>까지는 최적 해를 찾을 수 있는 가장 큰 문제를 다루었다.

그러나 이전에 언급했듯이 이 연구에 동기를 부여한 사례 기업은 30개의 탱크로 구성된 호이스트 생산라인을 갖추고 있으므로 LINGO는 현실적인 규모의 실제 문제에 대해서 최적의 해를 찾을 수 없었다.

이러한 상황에서 휴리스틱 알고리즘을 테스트하기 위해 탱크 수와 작업 수가 각각 30과 10 인 실제 문제를 기반으로 한 예제를 만들었다.

휴리스틱은 BRATIO 값이 0 또는 0.1 인 경우 실현 가능한 해법을 찾지 못했지만, BRATIO 값을 0.2, 0.3, 0.4 및 0.5로 설정하면 계산 시간은 2, 25, 197 및 977 초에 해결 방법을 찾아내었다.

BRATIO가 0.5 에서 계산 시간이 977 초로 실제 상황에서 사용하기에는 너무 길기 때문에 BRATIO 값이 0.5보다 큰 경우는 테스트에 사용하지 않았다.

0 ~ 1.0의 BRATIO 값에 대해 완료 시간을 정규화하면 BRATIO 값 0.2, 0.3 및 0.4에 대해 각각 100.3, 100.0 및 100.0으로 나타낼 수 있었다. 이 비교에서 얻어진 결과로 BRATIO 가 0.2, 0.3, 0.4 일 때 완료 시간은 차이가 별로 없음을 알 수 있었다.

그러므로 비교적 작은 BRATIO 값은 작업 또는 탱크의 수가 매우 클 때 분지의 수를 작게 하여 계산 시간을 줄일 수 있었으므로, 문제의 크기가 큰 상황에서 작은 BRATIO 값을 사용하면 효율적인 계산 시간을 얻을 수 있을 것이라고 생각된다.

5. 요약

이 장에서는 랙 제약이 있는 동적 단일 호이스트에서 다양한 작업을 처리하기 위한 호이스트 생산 일정계획에 대한 문제를 분석하였다.

이 연구는 사례 기업에서 발생된 호이스트 생산시스템의 실용적인 일정계획 수립 문제로 인해 유발되었으며, 이 문제는 부품을 호이스트로 이동할 수 있게 하고, 부품의 분실 방지와 화학처리를 잘 수행할 수 있도록 하는 랙이라는 제약에 의해서 발생되었다. 이 일반적인 랙 제약은 이전 연구에 반영되어 있지 않은 새로운 제약조건 이었다.

그래서 2장에서 제안된 MILP 모델에 새로운 랙 제약조건을 반영하여 확장한 MILP 모델을 개발하였다. 이 모델을 최적화 소프트웨어 LINGO에 적용하면 비교적 짧은 시간에 상대적으로 작은 문제에 대한 최적의 해를 찾을 수 있었다. 그러나 현실적인 산업 규모의 문제의 경우 MILP 모델은 합리적인 계산 시간에 최적의 해를 찾을 수 없었으므로 이러한 문제를 해결하기 위해 분지한계 기반 휴리스틱 알고리즘을 제안하였다.

그리고 휴리스틱의 성능을 평가하기 위해 큰 예제 문제를 생성하고 알고리즘 결과를 MILP 모델에서 얻은 최적의 해와 비교하였다. 사용 가능한 예제 문제를 만드는데 어려움이 있었기 때문에 충분히 많은 문제를 테스트 할 수는 없었지만, 실험을 통해 계산해 본 결과 알고리즘은 매우 우수한 특성을 나타내었다. 확인된 완료 시간의 품질이 좋고 계산 시간이 최적 해였다. 따라서 실제적인 산업 규모의 문제를 해결하는데 휴리스틱을 사용할 수 있다고 결론을 내릴 수 있었다.

본 연구에서 개발된 휴리스틱 방법론은 아주 빠른 시간 안에 해를 찾을 수 있었고, 현실적인 문제에 대해 최적의 해를 제시하여 주었다. 사례 기업에서 개발된 휴리스틱 방법론으로 호이스트 생산 일정계획을 적용하게 되면 호이스트 생산시스템의 효율성을 증대하는데 아주 유용할 것이라고 생각된다.

VI. 결론

1. 연구 의의 및 요약

본 연구는 항공산업에서 금속 부품의 화공처리 작업을 수행하는 단일 호이스트 생산시스템의 사례 문제를 다루었다. 문제의 핵심은 이 생산시스템에서 물자의 이동을 담당하는 호이스트의 완료 시간을 최소화 하기 위해 효율적인 일정계획을 수립하는 것이었다. 연구 대상의 호이스트 생산시스템은 가장 일반적이고 현실적인 상황을 다루고 있는데, 그 특징으로는 1) 다양한 처리 조건을 갖는 다양한 작업을 처리하고 있고, 2) 이미 생산시스템에 투입되어 진행중인 작업들도 계획 대상에 포함시켰으며, 3) 작업 탱크들은 각각의 처리용량을 가질 수 있다는 것 등이다.

본격적인 연구에 앞서 제 2장에서는 기존 문헌 연구 분석을 하였으며, 사례 기업의 문제와 유사한 상황의 문제를 다룬 Zhao at. al.(2013)의 불확실성 하에서 다단식 자재 처리 프로세스를 위한 실시간 동적 호이스트 일정계획에 관한 연구를 집중적으로 분석하였다. 그들은 8개의 생산라인에서 5개의 작업으로 사례 분석을 실시하여 빠른 시간 안에 확실한 해를 보장할 수 있다고 하였다. 그래서 예제를 생성하여 최적화 소프트웨어 LINGO를 통해 실험해 본 결과 RDHS는 빠른 시간 내에 최적의 해를 찾을 수가 있었으나 현실적인 규모(10개 이상)에서는 RDHS 방법론은 적정 시간 안에 최적의 해를 제시하지 못하였다.

그리하여 제 3장에서는 RDHS 방법론에 대해 현실적인 규모의 경우에도 빠른 시간 안에 양호한 품질의 해를 제공해주는 분지한계 기반의 휴리스틱 방법론을 개발하였다. 최적 해를 구할 수 있는 크기의 문제를 통해 휴리스틱 결과와 최적 해를 비교해 본 결과, 해의 품질에 큰 차이가 없었지만 소요시간 측면에서는 문제의 크기가 커질 경우 휴리스틱이 훨씬 빨리 해를 도출해 주었다.

따라서 문제의 크기가 현실적으로 커질 경우에도 분지한계 기반의 휴리스틱 방법론은 훨씬 빨리 해를 도출할 수 있었으므로 매우 활용도가 높을 것으로 기대된다.

그리고 제 4장에서 사례 기업의 호이스트 생산시스템의 일정계획 방법에 대해 알아 보았다. 3장의 분지한계 기반의 휴리스틱 일정계획과 동일한 예제를 사용하여 사례 기업의 일정계획 수립 방법 및 규칙으로 실험해 본 결과 완료시간은 166.3분으로 나타났다. 이 방법은 작업자가 사례 기업의 일정계획 수립 규칙에 의거 직접 수작업으로 계획을 하여 나타난 결과였다.

2장의 RDHS 모형으로 최적화 소프트웨어 LINGO를 사용해 최적 해를 구했을 때 완료 시간은 120.3분으로 사례 기업보다 28% 정도 향상된 결과를 보였다. 3장에서 제안한 분지한계 기반의 휴리스틱 알고리즘을 통해 해를 구할 때 완료 시간은 123.7분으로 사례 기업보다 25.6% 정도 일정을 단축할 수 있었다. 그러므로 RDHS 모형과 휴리스틱 알고리즘으로 찾아낸 최적의 해는 사례 기업의 호이스트 생산시스템 일정계획 수립 방법보다 훨씬 더 좋은 결과를 제공해 준다는 것을 알 수 있었다.

그 다음 연구로 제 5장에서 랙 제약이 있는 동적 단일 호이스트에서 다양한 작업을 처리하기 위한 호이스트 생산 일정계획에 대한 문제를 분석하였다. 이 연구는 사례 기업에서 발생하는 실용적인 일정 수립 문제로 인해 유발되었으며, 이 문제는 부품을 호이스트로 이동할 수 있게 하고, 부품의 분실 방지와 화학처리를 잘 수행할 수 있도록 하는 랙 이라는 제약에 의해 발생되었다. 이 일반적인 랙 제약은 이전의 연구에 반영되어 있지 않은 새로운 제약조건 이었다.

그래서 새로운 랙 제약조건을 반영하여 2장에서 제안된 MILP 모델을 확장한 MILP 모델을 개발하였다. 개발한 모델을 최적화 소프트웨어 LINGO에 적용하면 비교적 짧은 시간에 상대적으로 작은 문제에 대해 최적의 해를 찾을 수 있었다. 그러나 현실적인 산업 규모의 문제의 경우 MILP 모델은 합리적인

계산 시간에 최적의 해를 찾을 수 없었으므로 이러한 문제를 해결하기 위해 분지한계 기반의 휴리스틱 알고리즘을 제안하였다. 제안된 휴리스틱 알고리즘의 계산 시간은 0.01초 미만이었고 해는 최적해와 동일하여 아주 우수한 결과를 나타내었다.

그리고 휴리스틱의 성능을 평가하기 위해 사례 기업과 유사한 크기의 큰 예제 문제를 생성하고 알고리즘 결과를 MILP 모델에서 얻은 최적의 해와 비교하였다. 사용 가능한 예제 문제를 만드는 데 어려움이 있어 충분히 많은 문제를 테스트 할 수는 없었지만, 실험 결과 제안된 휴리스틱 알고리즘은 매우 우수한 특성을 나타내었다. 즉, 확인된 완료 시간의 품질이 좋고 계산 시간이 최적 해로 나타났다.

따라서 사례 기업과 같이 실제적인 산업 규모의 문제를 해결하는데 분지한계 기반의 휴리스틱 알고리즘을 사용하면 완료 시간을 최소화 할 수 있는 효율적인 일정계획을 수립할 수 있다고 결론을 내릴 수 있었다.

2. 연구의 한계점과 향후 연구과제

본 연구의 한계점으로 첫 번째 단일 호이스트 생산시스템의 문제만을 다루었다. 그러나 실제로는 단일 호이스트 생산시스템만 존재하는 것이 아니라 호이스트가 2대 이상인 다중 호이스트 생산시스템의 경우도 적용할 수가 있어야 한다. 본 연구의 휴리스틱 알고리즘을 호이스트의 수가 2대 이상인 경우에 대해서도 확대하여 적용할 필요성이 있다고 생각한다.

두 번째 탱크에서 처리를 할 경우 고정된 하한 시간과 상한 시간에 대해서만 적용하였다. 그러나 실제로는 동일한 탱크에서 처리되는 하한 시간과 상한 시간은 공정 순서에 따라 가변적으로 변하는 작업이 존재한다. 그러므로 다음 연구에서는 처리되는 하한 시간과 상한 시간이 가변적인 작업이 존재하는 상황에 대해 적용할 필요성이 있다고 생각한다

세 번째 호이스트 생산시스템에 투입 버퍼와 완료 버퍼만으로 구성된 탱크 라인에 대해서만 적용하였다. 그러나 실제 작업의 효율성을 높이기 위해 화공처리 라인 중간에 대기 탱크 즉, 중간 버퍼가 존재하여야 한다. 중간 버퍼가 있을 경우 다음 작업의 탱크가 작업 중일 때 대기해야 하는 작업을 투입 버퍼나 완료 버퍼로 이동하는 것보다 중간 버퍼로 이동하면 이동 시간이 단축될 뿐 아니라 짧은 이동 시간으로 인해 다른 작업의 이동을 수행할 수 있으므로 훨씬 더 유연한 생산 일정계획을 수립할 수 있다. 그러므로 중간 버퍼가 존재하는 호이스트 생산시스템에 대한 연구도 필요할 것이라 생각한다.

본 연구의 한계점으로 서술한 3가지 상황과 또 다른 여러 가지 생산 환경에 알맞게 본 연구의 휴리스틱을 수정하여 적용하는 연구를 수행할 예정이다.

향후 여러 가지 환경에 맞게 호이스트 생산시스템의 일정계획을 최적화 할 수 있다면 호이스트 생산시스템의 효율성은 더욱 더 증대될 수 있을 것이라 생각한다.

참고문헌

이정구, 김정배, 고시근 (2016), 단일 호이스트 생산시스템에서 다양한 주문을 처리하기 위한 분지한계 기반의 휴리스틱 일정계획, 대한산업공학학회지, 제42권, 제3호, pp.173~181.

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

Feng, J., Che, A., & Chu, C. (2015). Dynamic hoist scheduling problem with multi-capacity reentrant machines: A mixed integer programming approach. Computers & Industrial Engineering, 87, 611-620.

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

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

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

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

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

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

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

Manier, M.A., & Bloch, C. (2003). A classification for hoist scheduling problems. International Journal of Flexible Manufacturing Systems, 15(1), 37-55.

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

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

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

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

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

Yan, P., Che, A., Cai, X., & Tang, X. (2014). Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance.

Computers & Operations Research, 50, 128-140.

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

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

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