IE Interfaces
Vol. 14, No. 3, pp. 218-226, September 2001.

자원 제약을 고려한 조선산업에서의 탑재 일정계획에 관한 연구

김기동[1] · 우상복[2] · 한형상[2]

[1]강원대학교 산업공학과 / [2]고등기술연구원 생산기술센터

A Study on the Erection Scheduling for Shipbuilding Considering Resource Constraints

Kidong Kim[1] · Sangbok Woo[2] · Hyungsang Hahn[2]

Scheduling for shipbuilding processes has many alternative solutions since it has long time horizon and handles many jobs. This requests the scheduling system to generate and search feasible alternative solutions in short period of time. Among shipbuilding schedules, the ship erection schedule in a dock is the most important since the dock is the most critical resource in shipyard. In this research, we model the erection scheduling problem for shipbuilding and develop a new problem solving method using CST(Constraint Satisfaction Technique) and ILOG Scheduler. Experimental results show that the proposed scheduling method outperforms the existing manual scheduling methods in terms of schedule performance and required time.

1. 서 론

조선산업은 건설과 더불어 대표적인 project shop의 특성을 가지고 있다. 고가의 단일 품목을 오랜 기간에 걸쳐 건조하기 때문에 공기단축에 의한 비용절감의 효과가 매우 크다. 따라서 이익을 극대화하기 위해서는, 공기 단축과 생산 자원(인력, 설비 등)의 효율적인 이용을 통한 생산비 절감 및 매출증대가 필수적이며, 이에는 효과적인 생산계획의 수립과 관리가 핵심을 이루고 있다. 생산계획의 수립과 관리는 선박 건조의 전 과정에서 이루어지고 있는 바, 이중 일정계획 수립은 가장 중요한 의사결정 사항 중의 하나이다.

조선산업에서의 일정계획은 계획 기간이 길고, 일정계획 수립의 대상이 되는 작업들의 종류가 다양한 관계로 매우 어려운 문제로 인식되고 있다. 게다가, 작업이 이루어지는 장소에 따라서도 주어진 일정계획 문제의 성격에 많은 차이를 보이고 있다. Yard의 경우에는 프로젝트 일정계획의 성격이 강하고, 가공 및 조립 shop의 경우에는 shop의 특성에 따라 job shop 또는 flow shop 일정계획의 성격을 보인다. 이에 더하여, 조선산업의 특성에 기인한 여러 가지 제약들(공간 제약, 인력 제약, 이종 제품 생산 등)은 조선 일정계획 문제를 보다 어려운 문제로 만들고 있다.

조선산업에서 수행되는 일정계획 수립과정을 살펴보면, 우선 가장 중요한 자원인 도크에서의 일정계획, 즉 도크에서의 선각 조립과 관련된 '탑재 일정계획'을 수립하고, 이 결과를 바탕으로 선행 탑재 일정과 도크 및 안벽에서의 의장 일정을 결정한 뒤, 선행 도장 및 의장 일정, 가공 및 조립공장에 대한 일정계획 등이 차례로 수립된다.

본 연구에서는 실제 현장의 업무규칙을 고려하여 도크에서의 탑재 일정계획 문제를 제약만족기법(CST: Constraint Satisfaction Technique)과 객체 지향 개념을 지원하는 방법론을 이용하여 풀이하고, 모형 수립의 편이성, 제시된 해의 타당성, 조선 탑재 일정 문제에 대한 적합성에 대하여 살펴보았다.

2장에서는 관련 연구 현황을 분석하고, 3장에서는 객체지향 모형화와 CST를 지원하는 도구인 ILOG Scheduler를 이용한 일정계획 모형화에 대한 개요를 설명한다. 4장에서는 조선 탑재 일정 문제의 업무규칙을 반영한 탑재 일정계획 모형을 설명하고, 5장에서는 실제 자료를 이용한 예제 풀이 결과를 제시한 뒤, 6장에서 결론 및 추후 연구 방향을 제시한다.

2. 관련 연구 현황

2.1 조선 탑재 일정 관련 연구 현황

일정계획 문제는 일반적으로 NP-hard인 문제로 분류되어 최

적해를 얻기가 매우 어려운 것으로 인식되었다(Baker, 1974). 따라서 80년대 이전에는 주로 발견적 기법에 기반한 방법론들이 현장에서 이용되었다. 그러나 80년대 말 이후 새로운 일정계획 해법의 등장, IT(Information Technology) 및 컴퓨팅 환경의 급속한 발전, 그리고 다양한 시스템 모델링 및 개발 지원도구의 활용이 가능해지면서 다양한 인공지능과 최적화 기법들이 현장에서도 이용될 수 있는 길이 열렸다.

조선 일정계획과 관련되어 보고된 연구 중 가장 많은 부분을 차지하는 분야는 탑재 일정계획의 수립과 관련된 내용이다. 일반적으로 조선 shop에서 가장 중요한 자원은 도크라고 볼 수 있다. 따라서 도크에서 수행하는 작업들에 대한 일정계획 수립은 조선 shop 전체의 생산성과 밀접한 관련이 있다.

김훈주(김훈주, 1995)는 탑재 순서(sequence) 생성과 탑재 일정 생성 문제를 다루었다. 선정된 수행도는 탑재 부하 평준화이며, 고려한 제약은 블록 간 위상(topology), 탑재 공법에 따른 탑재 순서, 탑재 작업의 기술적 제약 사항, 중간진수 범위, 크레인 일일 탑재 블록 수(탑재 횟수), 탑재 블록 간 용접장 부하 등이다. 이용한 방법론은 유전 알고리듬이다.

홍윤기 외(홍윤기 외, 1997)는 탑재 순서 생성과 탑재 초기 네트워크 구성 문제를 다루었다. 실행 가능한 모든 탑재 순서 생성을 목표로 했으며, 고려된 제약은 블록 간 위상, 탑재 공법에 따른 탑재 순서 등이다. 이용한 방법론은 backtracking을 이용한 탐색 방법이다. 탑재 일정 생성 문제는 다루지 않았다.

Choi (Choi, 1994)는 선행 탑재(PE : Pre-Erection)/도크의 탑재 중일정계획(日 단위의 일정계획 수립) 문제를 다루었다. 선정된 수행도는 전문가의 제약을 만족하는 실행 가능한 탑재 일정이다. 고려된 제약은 Goliath Crane (이후 G/C라 칭함) 부하, 탑재 전문가의 기술적 지식 등이다. 제약 조건하의 그래프 탐색 기법(constraint directed graph search)을 이용한 전문가 시스템을 이용해서 풀이했다.

민상규 외 (민상규 외, 2000)는 부하 평준화를 위한 탑재 일정을 유전 알고리듬을 이용하여 풀이했다.

그 외 보고된 자료에 의하면 탑재 관련 연구에서는 탑재 순서 생성 문제, 탑재 일정 생성 문제, 도크 내 배치 문제, PE장 배치 및 PE장의 작업일정, 가공 및 조립공장의 일정 등의 문제가 복합적으로 고려되고 있음을 알 수 있다. 방법론으로는 발견적 기법 및 탐색 기법이 주로 이용되며, 효율적인 탐색을 위해 인공지능 기법들이 이용되고 있음을 알 수 있다(대동조선, 1999; 우상복, 1999; 이경준 외, 1992).

2.2 제약만족기법

제약만족문제(CSP; Constraint Satisfaction Problem)는 주어진 제약조건을 만족하는 가능해를 찾는 것을 목적으로 하는 문제이며, 제약만족기법은 CSP의 해를 구하는 데 사용되는 방법의 총칭이다. 제약만족문제는 변수(variable), 변수에 할당 가능한 값의 집합(domain), 그리고 변수 간의 관계(제약, constraint)로 정의된다. 일정계획 문제는 변수와 제약을 이용하여 제약만족문제로 표현된다. 제약만족기법으로는 제약전파(constraint propagation), 변수정렬(variable ordering), 값정렬(value ordering), 탐색 방법 등을 들 수 있다.

제약전파는 부분해를 이용하여 다른 변수가 가질 수 있는 가능한 값의 집합을 줄여주는 기법이다. 제약전파의 기본 알고리듬인 호일관화기법(arc consistency enforcement algorithm)은 작업 순서를 정의하거나, 작업의 시작시점(또는 종료시점)을 결정하였을 때에 해당 작업과 관계를 갖고 있는 다른 작업의 작업가능시간을 조정하는 과정에 사용된다. 이외에 한정된 자원 용량을 이용한 제약전파 방법인 immediate selection이 있다. 변수정렬은 일정계획의 대상이 되는 작업 또는 공정을 선택하는 방법이며, 값정렬은 작업 또는 공정의 작업순서, 시작 시점(또는 종료시점)을 결정하는 방법이다. 탐색 방법은 가능해를 발견하지 못한 경우에 이전의 부분해로 돌아오는 backtracking 방법을 의미한다. 즉, 제약만족기법을 이용한 해법은 경영과학에서 널리 사용되어 온 분지한계 방법과 유사하다.

3. ILOG Scheduler를 이용한 일정계획 모형화

일반적으로 수식에 의한 일정계획 모형을 수립하기 위해서는 일정계획 문제의 상황을 변수를 이용하여 재해석하고, 분석하는 과정이 요구된다. ILOG Scheduler는 수식을 이용한 모형화 대신 객체를 이용해서 모형화하는 방법을 제공한다. ILOG Scheduler는 C++을 이용하여 개발되었으며, 이용법 또한 C++ 이용법과 동일하다.

ILOG Scheduler에서는 일정계획 수립의 대상이 되는 작업(탑재 일정계획에서는 블록의 탑재)을 activity라는 객체로 표현하고, 작업을 위해 이용되는 자원(탑재 일정계획에서는 G/C, PE장의 정반, 심출/취부/용접인력 등)을 resource라는 객체로 표현한다. ILOG Scheduler를 이용해서 탑재 일정계획을 모형화하기 위해서는 우선 activity와 resource를 정의하고, 이들간의 관계를 activity와 resource의 멤버 함수를 이용해서 제약 사항으로 추가한다. 3.1절과 3.2절에서는 ILOG Scheduler를 이용한 모형화 방법에 대하여 간략히 설명한다.

3.1 activity의 생성과 activity 간의 선후행 제약

ILOG Scheduler가 제공하는 activity class의 class hierarchy는 다음 <그림 1>과 같다.

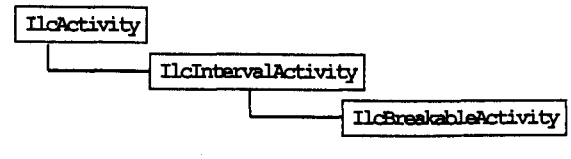

그림 1. IlcActivity의 class hierarchy.

이중 IlcIntervalActivity class는 다른 작업으로 인해 작업이 중간에 멈추는 경우가 없는 activity를 표현한다. 이 class의 생성자 중 하나는 다음과 같다.

```
IlcIntervalActivity::IlcIntervalActivity(IlcSchedule schedule,
IlcInt duration);
```

생성자의 입력 인자 중 IlcSchdule은 ILOG Scheduler를 이용해서 형성된 문제를 총체적으로 관리하며, IlcManager를 만들고 난 후, 만들어야 한다. 그리고 ILOG Scheduler에서 제공하는 class는 IlcSchedule을 만들고 난 후 사용할 수 있다. 입력 인자 duration은 해당 activity의 시작에서 끝나는 시간을 의미한다. 참고로 이 시간은 변수를 이용해서 줄 수도 있다. 이 때 이용되는 생성자는 다음과 같다.

```
IlcIntervalActivity::IlcIntervalActivity(IlcSchedule schedule, IlcIntVar
duration);
```

이를 이용해서 preJob(소요 공기는 3일)과 postJob(소요 공기 5일)이라는 activity를 생성하면 다음과 같다.

```
IlcManager m(IlcEdit);
IlcSchedule sch(m, 0, MAX_DATE); //일정계획 기간이 0부터
                                 //MAX_DATE인 객체 sch 생성
IlcIntervalActivity preJob(sch, 3); //공기가 3인 activity 생성
IlcIntervalActivity postJob(sch, 5); //공기가 5인 activity 생성
```

이렇게 생성된 activity 간에는 선후행 관계에 대한 제약을 다양하게 줄 수 있는데, 선후행 관계에 대한 제약은 IlcIntervalActivity의 상위 class인 IlcActivity의 멤버 함수를 이용해서 준다. 다음은 멤버 함수 startsAfterStart의 선언부이다. 함수 호출 결과 return되는 object는 IlcConstraint의 하위 class 형인 IlcPrecedenceConstraint이다. 다음의 멤버 함수는 act라는 activity 수행 후 delay 동안 경과 후 해당 activity를 수행할 수 있음을 나타낸다. 이 제약을 반영하려면, IlcManager의 멤버 함수 add()를 이용하여 제약 사항임을 알려줘야 한다.

```
IlcPrecedenceConstraint IlcActivity::startsAfterEnd(IlcActivity act,
IlcInt delay = 0);
```

이를 이용해서 postJob은 preJob 수행 3일 후부터 실행이 가능하다는 제약은 다음과 같이 줄 수 있다.

```
m.add(postJob.startsAfterEnd(preJob, 3));
```

이렇게 activity의 선후행 관계를 줄 수 있는 멤버 함수의 종류는 다음과 같다.

```
StartsAfterStart, StartsAfterEnd, EndsAfterStart,
EndsAfterEnd, StartsAtStart, StartsAtEnd,
EndsAtStart, EndsAtEnd
```

위에서 At의 의미는 delay 경과 후, 정확히 그 때 activity 수행이 시작되어야 함을 의미하고, After의 의미는 delay 경과 이후부터 activity 수행이 가능함을 의미한다. Activity 수행 시작 시간에 제약을 절대적 시간으로 줄 수 있는 멤버 함수들도 있다. 아래의 함수 startsBefore는 activity가 time 전에 시작되어야 함을 의미한다.

```
IlcTimeBoundConstraint IlcActivity::startsBefore(IlcInt time);
```

이렇게 activity의 시작 시각에 대한 제약을 줄 수 있는 멤버 함수의 종류는 다음과 같다.

```
StartsBefore, StartsAfter, StartsAt,
EndsBefore, EndsAfter, EndsAt
```

이를 이용해서 preJob이 스케줄링 기준 시점(0)에서 7일 후부터 실행 가능하다는 제약은 다음과 같이 줄 수 있다.

```
m.add(preJob.startsAfter(7));
```

3.2 resource 생성과 activity와 resource 관계

ILOG Scheduler에서 제공하는 resource class의 class hierarchy는 <그림 2>와 같다.

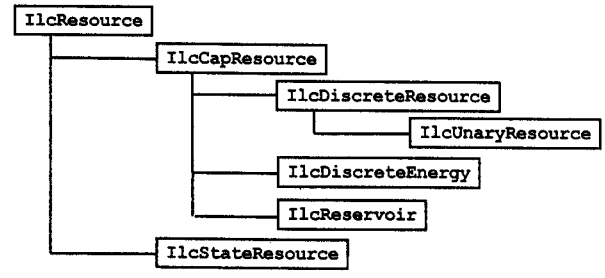

그림 2. IlcResource의 class hierarchy.

이중 IlcDiscreteResource는 단위시간당 자원의 사용 총량이 정해져 있는, 동종의 기계나 동일한 능력을 가진 인력 등을 모형화하기 적합하다. 다음은 생성자의 선언부이다.

```
IlcDiscreteResource::IlcDiscreteResource(IlcSchedule schedule,
IlcInt capacity, IlcBool timetable = IlcTrue);
```

이 생성자의 마지막 입력 인자인 IlcBool은 IlcTrue 혹은 IlcFalse 값을 갖는 자료형인데 IlcTrue이면 이 자원에 대한 timetable을 만들고, 매일 매일의 사용량에 대한 제약을 자동으로 추가한다. 이를 이용해서 용량이 4인 man이라는 자원을 만들면 다음과 같다.

```
IlcDiscreteResource manHour(sch, 4);
```

'Activity 수행을 위해 자원이 필요하다'라는 제약은 activity들이 가지고 있는 멤버 함수를 이용한다. 예를 들어, 위에서 설명한 preJob이라는 activity수행에 man이라는 자원이 3만큼 필요하다면, 다음과 같이 표현한다.

m.add(preJob.requires(man, 3));

requires라는 함수의 선언부는 다음과 같다.

IlcResourceConstraint
IlcIntervalActivity::requires(IlcCapResource resource, IlcInt capacity = 1);

이용되는 자원의 특성과 수행되는 activity의 형태에 따라 자원 요구에 대한 다양한 함수들을 제공하고 있다.

4. 제약만족기법을 이용한 탑재 일정계획 모형

제약만족기법과 객체지향 모형화를 지원하는 일반적 도구인 ILOG Scheduler를 이용하여 문제를 모형화한다. 4.1절에서는 문제의 대상이 되는 탑재 공정에 대해서 설명하며, 4.2절에서는 본 연구에서 대상으로 삼은 문제의 범위에 대해서 설명한다. 4.3절에서는 문제에서 고려해야할 현장의 업무 규칙에 대해 설명하고, 4.4절에서는 ILOG Scheduler를 이용한 탑재 일정계획의 모형을 설명한다.

4.1 탑재 공정

탑재는 G/C나 크레인을 이용하여 PE장에 위치한 블록 또는 대형블록을 도크에 세우는 작업인데, 탑재 순서는 탑재 공법(ring식, 충식, 피라미드식)에 따라 영향을 많이 받고, 기점블록 탑재(K/L: Keel Laying), 중간진수(F/O: Floating Out)와 진수(L/C: Launching) 일정에 맞추어 계획된 선각, 의장, 도장 작업 등이 수행된다(우상복, 1999).

선각작업은 일반적으로 심출, 취부, 용접 작업 순으로 이루어지는데, 심출은 탑재된 블록의 접합면에 대한 정도 측정과 다듬기 작업을 말하며, 취부는 탑재된 블록을 고정시키기 위한 1 pass 용접작업을, 용접은 탑재된 블록을 최종 결합하는 나머지 용접작업을 말한다. 이중 심출 작업은 블록의 one time setting 가능 여부에 따라 크게 두 가지의 경우로 나눌 수가 있다. One time setting이 가능한 경우에는 블록 간의 접합면에 대한 정도 측정과 다듬기 작업이 PE장에서 대부분 이루어지기 때문에, G/C가 탑재 블록을 탑재할 때 심출 작업이 간단하게 수행된다. One time setting이 불가능한 경우에는 탑재 후에 G/C 재작업(정위치 작업)이 필요하며, 이 때 심출 작업도 추가로 필요하게 된다.

탑재에 이용되는 중요 자원으로는 PE장의 정반 면적, 심출, 취부, 용접을 담당하는 인력, G/C 시간으로 볼 수 있다. 또한 탑재 블록들은 탑재 전 PE장에서의 작업을 위해 소요되는 시간, G/C 이용 시간, 탑재에 필요한 인력시수(man-hour로 표현됨)를 블록별로 가지고 있다. 또한 선행블록 및 후행블록 관계와 min pitch time이라는 정보를 가지고 있다. Min pitch time은 선행 블록의 탑재 후 특정 시간(절대적 시간)이 지난 후에야 후행 블록의 탑재가 가능한 것을 의미한다.

4.2 본 연구에서 고려한 일정계획의 범위

본 연구에서는 도크 내 탑재 작업을 대상으로 일정계획을 수립했다. 대상으로 삼은 작업의 범위는 PE장 입고에서 탑재 완료시까지 도크에서 행해지는 선각 작업이다. 대상으로 삼은 기간은 1 batch(도크 내로 해수의 유입이 이루어지는 사이 기간이며, 이 기간동안 도크 내에서 작업이 가능함) 동안이며, 탑재 네크워크에 등장하는 모든 블록과 주요 장비 설치를 일정계획 수립의 대상으로 삼았다.

4.3 탑재 일정모형의 입력, 출력, 제약사항

선박의 도크 내 탑재 일정계획 문제의 입력, 출력, 제약 사항, 일정계획의 수행도 중에서 본 연구에서 다루는 문제의 범위와 관련 있는 사항을 다음에 정리한다. 제약 사항들은 실제 조선의 업무 규칙에서 도출되었다. 다음의 제약 사항들은 D조선 생산관리 부서의 탑재 중일정계획 수립시 사용하는 내용이며, 계획에 필요한 자료는 계획 수립 전에 설계 부서나 기술 부서로부터 확보된다. 특히, 추후 모형 수립에 중요한 사항인 'one time setting 가능 여부' 역시 블록 설계가 도출된 후 기술 부서의 검토를 거쳐, 생산관리 부서의 통합 생산계획 데이터베이스에 반영되는 자료이다.

[탑재 일정계획 문제의 입력자료]
① PE장의 면적
② 선각 작업의 인력
③ 대일정 선표의 batch당 net day
④ 호선별 탑재 정보(순서, 공기, 시수), 진수까지의 탑재 블록
⑤ 블록별 PE장에서의 점유면적과 점유기간 정보
⑥ 탑재 블록별 G/C 이용 시간
⑦ 선각과 관련된 주요 장비 설치 정보

[일반적인 제약 사항]
① PE장의 정반 면적 부하
② 심출, 취부/용접의 인력부하
③ 블록별 탑재 순서와 min pitch time
④ G/C의 가용시간

[심출, 취부/용접 작업에 관한 업무 규칙]
① 심출과 취부/용접은 연속 작업으로 이루어지지 않을 수 있다.
② 취부/용접은 연속 작업으로 이루어진다.
③ 특정 블록의 취부/용접은 초기 투입된 인력이 끝까지 담당한다.
④ One time setting이 가능한 블록은 심출 작업이 탑재 당일 이루어진다.

⑤ One time setting이 불가능한 블록은 심출 작업 이 두 번 이루어진다.
⑥ One time setting이 불가능한 블록의 첫 번째 심출 작업은 탑재 당일 이루어진다.
⑦ One time setting이 불가능한 블록의 두 번째 심출 작업은 발판 설치가 끝난 날(보통 탑재 이틀 후) 이루어진다.
⑧ One time setting이 불가능한 블록의 심출 시수 는 대부분 두 번째 심출 작업에 소요된다.
⑨ One time setting이 불가능한 블록의 두 번째 심출 작업은 보통은 하루에 이루어지지만, 경우 에 따라서는 이틀 이상 소요될 수 있다.
⑩ One time setting이 불가능한 블록의 두 번째 심출 작업이 이틀 이상 소요될 경우, 두 번째 날부터는 취부/용접 작업과 병행할 수 있다.

[PE장 점유에 관한 업무 규칙]
① 어느 블록은 PE장에서 특정 기간 작업을 마치고 난 후, 도장 작업을 위해 PE장을 나왔다가 탑재되는 경우가 있다. 이때, 탑재 당일엔 PE장 점유 없이 바로 탑재된다.

[불확실한 업무 규칙에 대한 가정]
① 블록당 취부/용접의 최대 공기와 최소 공기가 표준 공기와 로직에 의해 정해질 수 있다.
② 선행블록 탑재 후 후행블록 탑재까지는 일정한 시간(min pitch time)이 필요한 경우가 있는데, 이 시간 중 선행블록의 취부/용접 공정의 진행 정도에 따라서 결정되는 시간은 취부/용접의 시수 중 일부로 표현될 수 있다.

[출력 사항]
① 각 블록별 PE장 입고 일자
② 블록별 도크 내 탑재일
③ 블록별 도크 내 심출 완료일
④ 블록별 도크 내 취부/용접 착수일, 완료일
⑤ 도크에서 이용되는 G/C 부하
⑥ 도크에서의 심출/취부/용접의 일별 시수 부하,
⑦ 도크에서의 취부/용접의 일별 시수 부하

[고려 가능한 일정계획의 수행도]
① 모든 블록의 탑재 완료 시점(진수 시점) 최소화
② 심출 + 취부/용접 부하 평준화
③ 심출 부하 평준화
④ 취부/용접 부하 평준화
⑤ G/C 부하 평준화
⑥ PE정반 면적 부하 평준화

4.4 탑재 일정모형 수립을 위한 업무 규칙 해석

ILOG Scheduler를 이용하여 일정계획을 모형화하는 과정은 activity와 resource를 정의하고 이들 관계에 대해 서술하는 것으로 볼 수 있다. 탑재 일정모형의 수립을 위해 업무 규칙을 중심으로 activity와 resource 관계를 살펴본다.

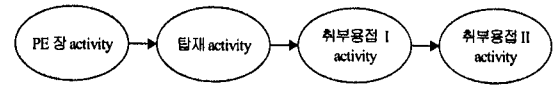

그림 3. one time setting이 가능한 블록의 sub activity 분해도.

그림 4. one time setting이 불가능한 블록의 sub activity 분해도.

탑재 공정 상황을 모형화하기 위해, 각 블록의 탑재 activity를 다음 <그림 3>, <그림 4>와 같이 sub activity로 분해했다. <그림 3>은 one time setting이 가능한 블록의 sub activity 분해도이고, <그림 4>는 one time setting이 불가능한 블록의 sub activity 분해도이다. 분해도에서 타원은 activity이고, 아크는 선후행관계를 각각 나타낸다.

이러한 선후행관계로 인한 제약은 3절에서 설명한 바와 같이 ILOG Scheduler에서 제공되는 activity class의 member function을 이용하여 표현된다. <그림 3>의 PE장 activity와 탑재 activity간의 제약을 예로 들어본다. PE장 activity가 act1이고 탑재 activity가 act2이며 탑재 activity가 PE장 activity 수행 후에 이루어지는 경우, ILOG Scheduler에서 이 관계는 m.add (act2.startsAfterEnd (act1))로 표현될 수 있다.

ILOG Scheduler를 이용하여 주요 선후행관계 제약을 구현한 내용을 아래에서 구체적으로 살펴본다.

block[][][]은 위의 각 activity를 표현한 IlcInterval Activity이며 첫번째 첨자는 호선별로 붙여지고, 두 번째 첨자는 블록별로 붙여진다. 마지막 첨자가 0은 PE장, 1은 탑재, 2는 정위치 탑재, 3은 취부용접 I, 4는 취부용접 II를 각각 표현한다. raw_block_data에는 입력 자료로부터 읽어들인 기본 자료가 저장되어있다.

```
void makePrecedenceConstraints(IlcSchedule schedule, int sid,
IlcIntervalActivity block[][MAX_BLOCK][NOSUBBLK])
{
IlcManager m = schedule.getManager();
// 동일 블럭 내에서의 선후행관계
for(int i=0; i < noBlk[sid]; i++){
  // PE장 activity 후 탑재 activity
  m.add(block[sid][i][1].startsAtEnd(block[sid][i][0], 0));
  // 탑재 후 정위치 탑재
  m.add(block[sid][i][2].startsAfterEnd(block[sid][i][1], 2));
  // 정위치 탑재가 없으면, 탑재 후 취부용접 I
  if(raw_block_data[sid][i].gc_time2 == 0)
    m.add(block[sid][i][3].startsAfterEnd(block[sid][i][1], 0));
  // 정위치 탑재가 있으면, 정위치 탑재 후 취부용접 I
  else
```

```
    m.add(block[sid][i][3].startsAfterStart(block[sid][i][2], 0));
  // 취부용접 I 후 취부용접 II
    m.add(block[sid][i][4].startsAtEnd(block[sid][i][3], 0));
}}
```

Sub activity는 각각 공기, 사용 자원, 요구되는 자원의 양을 속성으로 갖는다. 이들 activity는 시작 시각과 종료 시각을 유지하며 일정계획을 수립한 결과로 이러한 시각들이 결정된다.

PE장 activity는 탑재 일정계획 문제의 입력자료의 'PE장 점유기간' 동안 PE장 정반이라는 자원을, 입력자료의 '점유면적' 만큼 이용한다. PE장에서의 소요 기간 업무 규칙 측면에서 보면, 블록들을 두 가지 부류로 나눌 수가 있다. 그중 하나는 'PE장 점유기간'이 지난 후 PE장 밖으로 이동이 필요한 블록이고, 다른 하나는 'PE장 점유기간'이 지난 후 탑재 activity가 시작될 때까지 PE장에 계속 남아있는 블록이다. PE장 밖으로 이동이 필요한 블록은 입력자료의 'PE장 공기'와 'PE장 점유기간'을 비교해 보면 알 수 있는데, 공기가 점유기간보다 큰 블록은 이동이 필요하다. 이 경우 PE장 activity는 'PE장 점유기간' 동안에만 정반 자원을 이용한다. 그 외의 블록은 후속 activity인 탑재 activity가 발생할 때까지 정반 자원을 이용하게 된다. 또한, PE장에 입고되지 않는 단독 탑재, 주요장비설치 등에 대해서는 PE장 activity의 점유 기간을 0으로 한다.

탑재 activity는 실제 탑재가 이루어지는 상황을 표현한다. 탑재 activity의 작업 공기는 하루 미만이지만 일정계획 수립의 단위를 일로 가정했으므로 최소단위인 하루로 설정한다. 탑재 activity는 G/C, 심출 시수라는 두 종류의 자원을 이용한다. G/C 자원을 입력자료의 '탑재 블록별 G/C시수 I (단위 : 분)' 만큼 이용하고, 심출시수를 입력자료의 심출시수만큼 이용한다. 자원 이용과 관련된 제약 역시 activity class의 member function으로 표현된다. G/C를 일별 가능한 용량이 480단위(8시간 * 60분)인 goliath resource로 선언하면, 탑재 activity인 act2가 50분 동안 G/C를 이용할 경우, ILOG Scheduler에서 해당 제약은 m.add(act2.requires (goliath, 50))로 표현된다.

ILOG Scheduler를 이용하여 G/C 자원을 만들고 관련 제약을 구현한 내용을 아래에 구체적으로 나타내었다.

```
void makeGCTimeConstraints(IlcSchedule schedule,
      IlcIntervalActivity block[][MAX_BLOCK][NOSUBBLK])
{
IlcManager m = schedule.getManager();
// G/C-TIME RESOURCE 생성
IlcDiscreteResource goliath(schedule, GOLIATH_CAPA);
goliath.setName("dock1_gc");
goliath.setEdgeFinder();
// G/C RESOURCE 관련 제약 생성
for(int sid=0; sid < NO_SHIP; sid++)
  for(int i = 0; i < noBlk[sid]; i++)
    m.add(block[sid][i][1].requires(goliath,
        raw_block_data[sid][i].gc_time1));
// CLOSE THE RESOURCES
goliath.close();
}
```

단, <그림 4>와 같이 정위치 탑재 activity가 후속 activity로 존재하는 블록의 경우 대부분의 심출작업이 정위치 탑재시 이루어지므로, 탑재 activity에서의 심출시수 자원 이용은 없는 것으로 간주한다.

<그림 4>와 같이 탑재 activity 후에 정위치 탑재 activity가 있는 경우에 정위치 탑재는 탑재 2일 후에 시작되는 것으로 간주한다. 정위치 탑재 activity는 G/C, 심출시수라는 두 종류의 자원을 이용한다. G/C 자원을 입력자료의 '탑재 블록별 G/C시수 II' 만큼 이용하고, 심출시수를 입력자료의 심출시수 만큼 이용한다. 정위치 탑재의 공기는 하루이다.

취부용접 I activity와 취부용접 II activity는 후행 블록의 탑재를 고려해서 원래의 취부/용접 activity를 다시 구분한 것이다. 취부용접 I과 취부용접 II activity는 취부/용접 시수라는 자원을 작업공기 동안 균등하게 분배하여 사용한다. 작업공기는 미리 정의된 최대 작업공기(표준공기)와 최소 작업공기(표준공기-줄일 수 있는 공기) 사이의 값을 갖는 변수로 처리한다.

후행 블록과 선행 블록 간에 정의되는 min pitch time은, 선행블록의 안정성(후행블록을 올려도 될만큼의 취부/용접 진행), 선행 블록과 관련된 주요 작업의 공기 등을 고려하여 기술적으로 결정된다. 이러한 min pitch time과 관련된 항목 중 선행블록의 안정성은 일반적으로 취부/용접 시수 중 특정 부분(예를 들어 10% 공정진행)이 진행되어야 확보되는 사항인 만큼, 이를 모형에 반영하기 위해 취부/용접을 두 개의 activity로 구분했다.

다음 <그림 5>와 <그림 6>에서는 이와 관련된 제약 사항을 그림으로 나타내었다. <그림 5>는 one time setting이 이루어지는 경우를, <그림 6>은 그렇지 않은 경우를 각각 표현했다. 각 그림에서 후행블록1과 후행블록2의 정위치 탑재 activity가 점선인 것은 각 블록들이 one time setting이 가능한 경우도 있고, 그렇지 않은 경우도 있음을 의미한다.

후행블록1은 선행블록의 탑재 후 min pitch time(가접과 관련된 부분을 제외한)이 지나면 곧바로 탑재가 가능한데, 이 경우 ①의 선후행 제약이 작용한다. 즉 선행블록의 탑재 activity가 끝나면 후행블록1의 탑재가 가능한 것이다. 후행블록2는 선행블록의 탑재 후 min pitch time(취부용접 I 과 관련된 부분 제외)이 경과되고, 또한 선행블록의 취부용접 I이 끝나야만 탑재가 가능하다. Min pitch time과 관련된 제약은 ②의 제약이고 취부 관련된 제약은 ③의 제약이다.

수행도는 크게 두 가지 종류로 나누어 볼 수 있다. 하나는 여러 종류의 '자원 이용량에 대한 부하 평준화'이고, 다른 하나는 '모든 블록의 탑재 완료 시점(진수시점) 최소화'이다. 자원 이용량에 대한 부하 평준화는 일별 사용하는 자원의 양의 최대

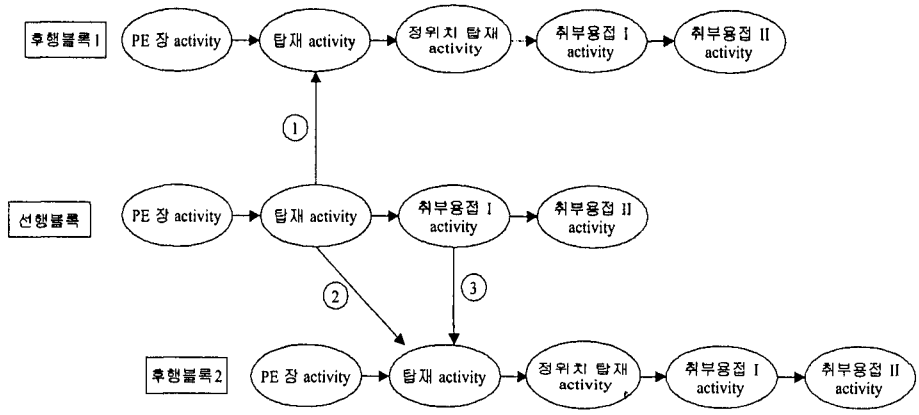

그림 5. 선행블록이 one time setting이 가능한 경우 후행 블록과의 관계.

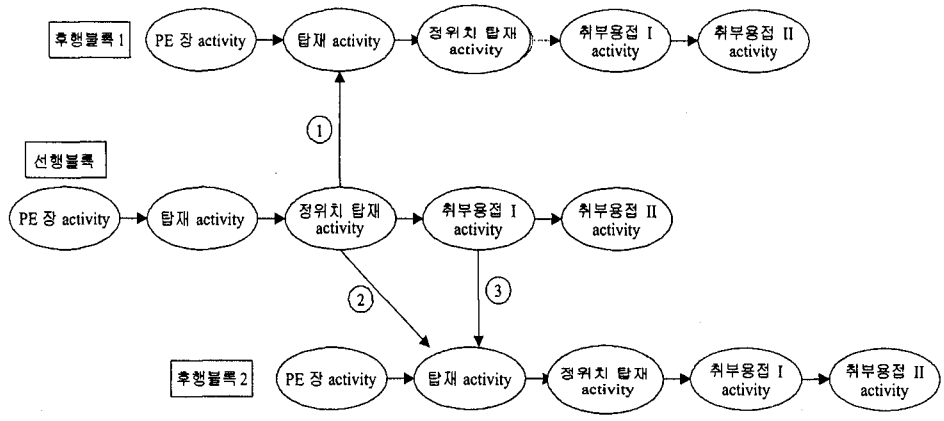

그림 6. 선행블록이 one time setting이 불가능한 경우 후행 블록과의 관계.

값을 최소화한다. 탑재 완료 시점 최소화는 각 블록의 마지막 activity의 끝나는 시점을 최소화한다.

탑재 완료 시점 최소화의 경우 ILOG를 이용한 목적함수 표현 방법에 대해 살펴본다. ILOG에서는 각 activity의 끝나는 시점을 변수로 관리하고 있는데, 이 변수값 중 최대값을 최소화한다. 이러한 내용으로 ILOG를 이용하여 목적함수를 만들고 탐색하는 주요 내용은 다음과 같이 구현되었다.

```
int main()
{....
// FOR MAKESPAN
for(sid=0; sid < NO_SHIP; sid++)
  for (int i=0; i < noBlk[sid]; i++)
    m.add(block[sid][i][4].getEndVariable() <=
        makespan);
m.add(IlcSetTimes(schedule, makespan));
m.setObjMin(makespan);
while (m.nextSolution()){
  cout << "makespan is : " << makespan.getValue() << endl;
}}
```

5. ILOG Scheduler를 이용한 예제 수행 결과

자원제약을 고려한 탑재 일정계획 모형의 검증을 위하여, D조선의 탑재 중일정계획 중에서 현재 시점 이후 수 개월 후에 진행될 특정 batch를 대상으로, 블록 및 관련 정보를 사용하여 예제를 만들고 일정을 수립하였다. 예제 풀이에 이용된 ILOG 모형은 4.3절에서 설명한 제약 사항 및 업무 규칙을 모두 반영하였다. 구체적으로, PE장 면적, 심출, 취부/용접의 가용시수, G/C의 가용시수 등의 자원 제약을 고려하였으며, 탑재 네트워크상의 선 · 후행 관계 제약과 취부/용접 공기 단축, 그리고 4.3절에서 설명한 심출, 취부/용접 작업에 관한 업무 규칙을 반영하였다.

실험에 이용된 자료에는 총 4척의 배가 탑재되는데, 이중에서 두 척은 중간진수(F/L)에서 진수(L/C)까지, 나머지 두 척은 기점 블록 탑재(K/L)에서 중간진수(F/L)까지이다. 탑재 블록의 총 수는 208개이고, 선 · 후행 관계 제약은 약 400개이며, 선정된 수행도는 모든 블록의 탑재 완료 시점 최소화이다.

예제의 풀이에 이용된 컴퓨터는 Pentium PC이며, 최적해를 도출할 때까지의 수행시간은 약 1분이었다. 예제 풀이 결과를 바탕으로 4개 호선 중 특정 호선의 탑재 일정표를 <그림 7>에

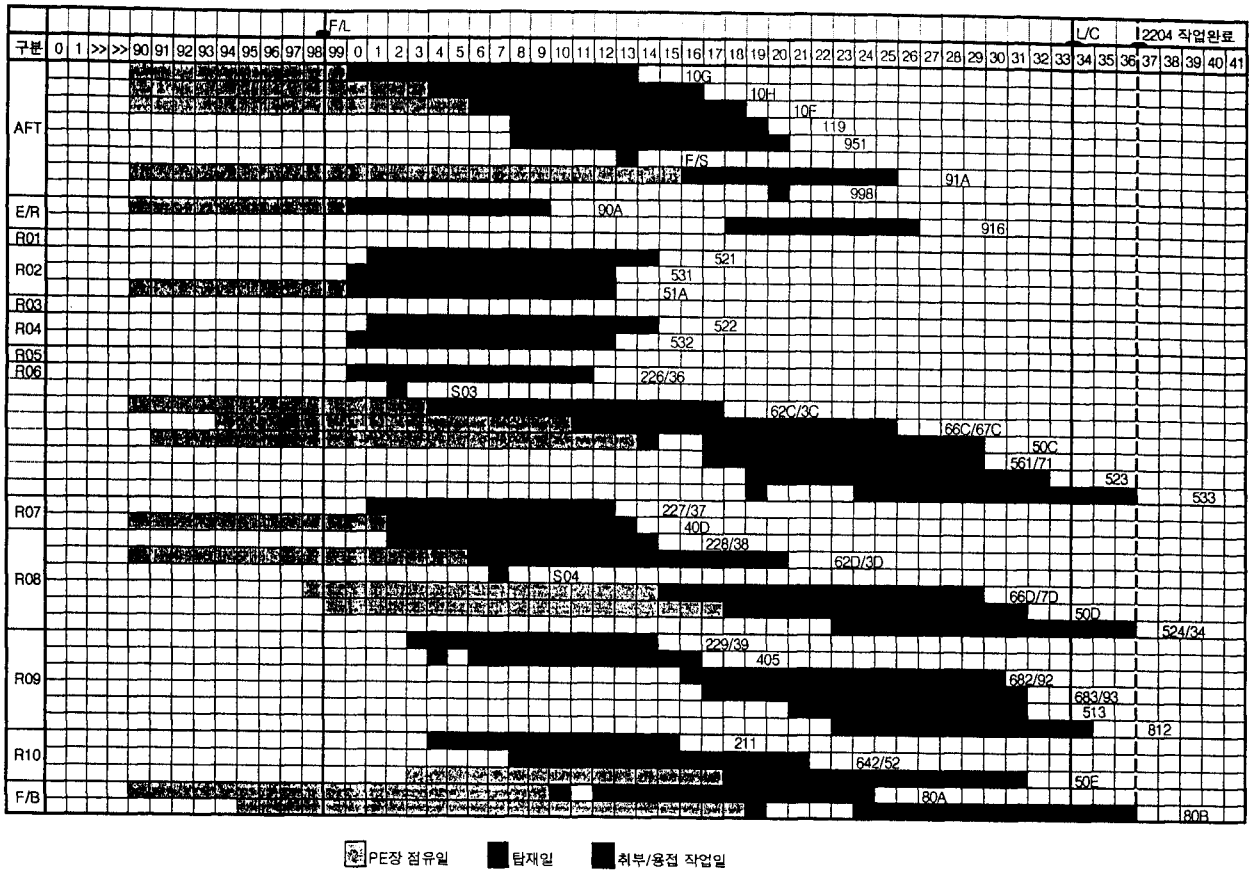

그림 7. 예제풀이 결과를 이용한 탑재 일정표(4개 호선 중 1호선).

도시했다. 그림에서 각 행은 탑재 블록을 의미하며, PE장 점유, 탑재, 취부/용접 작업이 차례로 나타나 있다.

최적해 풀이 결과를 실제 계획과 비교한 결과, 최적해에 의한 batch 기간(탑재 완료 기준)은 Net 작업일 면에서 실제 계획보다 1일 단축된 결과였으며, 전체적인 작업일정은 실제 계획과 유사한 결과를 보여주었다. 만일, 선·후행 관계와 min pitch time 등이 현재보다 정확하게 정의되고, 심출, 취부/용접의 시수와 공기 관련 자료가 엄격하게 관리된다면, 최적해의 활용도와 개선 효과는 더욱 커질 것으로 예상된다.

정리해 보면, ILOG를 이용한 최적해 접근법은 최적해를 지향하면서도 비교적 빠른 시간에 실제 계획보다 개선된 결과를 제시함으로써, 조선의 복잡한 업무규칙을 세밀하게 모델링하여 일정계획을 수립하기에 적합한 방법인 것으로 판단된다.

6. 결론 및 추후 연구 방향

조선산업은 숙련된 기술자와 고부가가치를 수반하는 첨단 설비를 총동원하는 노동 및 기술 집약적 산업이다(홍윤기 외, 1997). 이러한 현실에서 자원 활용을 극대화하기 위한 생산 계획 및 관리는 조선산업의 국가 경쟁력 제고 측면에서도 매우 중요한 사안이라고 할 수 있다. 생산 계획 및 관리의 기능 중 일정계획은, 수행되어야 할 작업들에 대한 자원의 할당과 작업의 진행 순서 결정을 담당하는 중요한 기능이다.

본 연구에서는 조선산업에서의 일정계획 중 가장 중요한 분야 중의 하나인 탑재 일정을 제약만족기법과 객체 지향을 지원하는 ILOG Scheduler를 이용하여 실제 탑재 일정과 관련된 업무 규칙 및 제약을 고려하여 모형화하였고, 예제 문제를 형성하여 풀이했다. ILOG Scheduler를 이용한 현장의 업무 규칙 모형화는 비교적 용이하며, 실제 크기의 문제에 적용한 결과, 수행도(수행 속도와 해의 우수성) 측면에서도 뛰어남을 확인할 수 있었다.

추후 연구 방향은 ILOG Scheduler를 이용한 탑재 일정모형에서, 현재는 고려되지 않았던 업무 규칙에 대해 추가적인 모형화 작업을 수행하고, 조립, PE 등 조선산업의 타 부분에 대한 일정계획 모델링과 시스템 개발을 진행할 예정이다.

참고문헌

김훈주 (1995), 유전 알고리듬을 이용한 조선 탑재 공정 계획 연구, *인하대학교 석사학위 논문*.

대동조선 (1999), PE정반 Arrangement 시스템, *조선 생산 기술연구회*.

1999년 춘계 연구발표회.
민상규 외 (2000), 조선 탑재일정의 부하 평준화를 위한 유전 알고리듬, *산업공학지*, 13(2), 225-233.
박주철, 황하룡 (1998), 조선 기준 일정계획을 위한 재계획 절차의 개발, *산업공학지*, 11(3), 129-141.
우상복 (1999), 대우조선의 제조 프로세스 및 생산관리 기술, *고등기술연구원 생산기술연구실 Technical Paper.*
이경준 외 (1992), Spatial Scheduling and its Application to Shipbuilding, *제2회 환태평양 인공지능 국제학술 회의 논문집*, 2, 1183-1189.
홍윤기 외 (1997), 조선 공정계획에서 탑재 순서 생성, *산업공학지*, 10(1), 189-207.
Baker, K. R. (1974), Introduction to sequencing and scheduling, John Wiley & Sons.
Choi, Hyung-Rim (1994), Erection Scheduling at Shipbuilding Using Constraint Directed Graph Search : DAS-ERECT, *KAIST 박사학위 논문.*
ILOG (1999a), ILOG Scheduler 4.4 Reference Manual, ILOG.
ILOG (1999b), ILOG Solver 4.4 Reference Manual, ILOG.

김 기 동
서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
서울대학교 산업공학과 박사
고등기술연구원 생산기술센터 선임연구원
현재: 강원대학교 산업공학과 조교수
관심분야: 생산정보시스템, 스케쥴링, 인공지능

우 상 복
서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
서울대학교 산업공학과 박사
현재: 고등기술연구원 생산기술센터 책임연구원
관심분야: 생산시스템, 스케쥴링, ERP/SCM/MES

한 형 상
서울대학교 산업공학과 학사
한국과학기술원 경영과학과 석사
Univ. of Wisconsin, Madison 산업공학과 박사
현재: 고등기술연구원 생산기술센터장
관심분야: 생산시스템공학, 시뮬레이션