Journal of the Korean Institute
of Industrial Engineers
Vol.20, No.3, September., 1994

생산일정계획 성능향상을 위한 2 단계 휴리스틱 방법의 개발[1)]

Development of a Two-Phase Heuristic Method for the Enhancement of a Production Schedule

홍윤호*, 김승권*, 김선욱**, 이준열*

Yun Ho Hong*, Sheung-Kown Kim*, Sun Uk Kim**, Jun Youl Lee**

Abstract

There are so many situations in which production schedules need to be rescheduled, due to the uncertainty or dynamic characteristics existing in manufacturing environment. Under the circumstances, it is necessary to build a system which helps a schedule manager understand the dynamic situations and reflect his insight into the system easily.

We developed a system that can improve a production schedule. The system consists of a schedule editor and a schedule generator. The schedule editor, a rule-based system, always keeps a schedule feasible by keeping track of local changes in the revised one. The schedule generator consists of a two-phase heuristic method. In the 1st phase it generates a schedule by a priority rule with Giffler and Thompson algorithm then it improves the performance in terms of the makespan in the 2nd phase. Experimental results show that, most of the times, the system generates a better schedule than the one with ordinary priority rules only.

1. 서론

최근 점차적으로 소비자의 요구가 다양해지고 있으며 기업체도 제품의 종류와 기능의 다양화를 통해 소비자의 요구에 적극 대응하여 시장을 넓히고 이익을 극대화하려는 움직

1) 사용자 중심의 지식기반 시뮬레이션을 활용한 통합생산관리업무의 의사결정지원 시스템중 일부

* 이 논문은 1993년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

* 고려대학교 산업공학과

** 단국대학교 산업공학과

임을 보이고 있다. 그런데 이는 짧은 기간내에 다양한 수요를 만족시켜야 하는 과제를 낳게 되므로 효과적이고 철저한 생산일정관리의 필요성이 더욱 커졌다고 볼 수 있다.

생산현장은 계획수립시에 이용할 수 있는 정보의 정확도 및 거발주된 자재의 정확한 입고 여부, 기계고장, 예상보다 과도한 불량발생, 규격 및 생산량의 변경 등 미리 예측하기 어려운 여러 요인들이 존재하는 동적 특성을 가지고 있다. 이러한 동적 특성 때문에 제한된 가정하에서 수립된 일정계획은 진행과정에서 차질이 생기게 되는데[7,9], 이를 빠른 시간안에 적절히 조정해 줄 수 있어야만 안정된 관리상태에서 생산을 해 나갈 수 있다. 그렇지 않다면, 많은 비용을 들여 구축한 생산일정계획 시스템이라 하더라도 제구실을 해주지 못하고 생산관리자로 하여금 과거의 비공식적 시스템에 의존하게 한다. 이러한 상황에서 생산일정계획 자체를 문제상황 발생시에 부분적으로 수정되더라도 전체 일정은 큰 무리가 없도록 일정계획을 수립하여 그대로 지켜가는 방식(robust schedule)을 생각할 수 있다. 그렇지만 변동상황을 흡수할 수 있는 정도도 한계가 있으므로 일정을 실천해 가는 과정에서 피할 수 없는 문제가 발생하였을 때에 이를 대처하고 실시간(real time)으로 재일정을 수립하여 실천할 수 있도록 해주는 생산통제 시스템이 필요하다. Graves는 일정의 'Robustness'에 대해 논하면서 "일정계획문제는 없다. 다만 재일정 문제가 있을 뿐이다"라고까지 기술해 재일정 문제의 중요성과 어려움에 대해 강조한바 있다[9].

생산통제 시스템은 계층적 계획 시스템(Hierarchical planning system)에서 제일 하부의 모듈이다[1]. 생산통제 시스템은 크게 두가지 기능으로 이용되고 있다. 첫째, 상위 계획 단계에서 수립한 계획된 활동들을 실행할 것을 명령하고, 둘째, 비정상적인 조업을 발견하여 수립된 생산일정을 크게 흐트림이 없이 일정을 조정하기 위한 감독기능을 수행한다[2]. 즉, 개별 작업장에 주어진 생산목표를 달성하기 위해 어떤 기계에서 언제 어떤 작업물이 가공되어야 하는가에 대한 일정계획(predictive shop floor scheduling)을 수립하는 것과 생산환경변화에 대해 신속하고 효과적인 대응 (reactive control)을 수행하는 것이다[7,9].

작업장내에 동적인 특성이 심해지면 초기 일정수립과 재수립의 구분은 없어지고 실시간 일정계획만이 필요하다[5]. 실시간 일정계획은 짧은 시간에 좋은 일정을 수립해야 하는 것으로서 문제의 복잡도 및 시간제약으로 인해 해결하기 어려운 문제이다. 그런데, 일정계획이론(scheduling theory)에서 개발된 알고리즘들은[8] 비현실적인 가정과 현실 문제를 풀기에는 계산시간이 너무 길다는 단점이 있고, 우선순위규칙을 이용할 경우 그 사용방법이 단순하며 시스템의 구조와 상관없이 유연하게 적용할 수 있어 실용적이지만, 이것으로 만들어진 일정은 성능 향상의 여지가 남아 있다. 따라서 우선순위 규칙의 사용과 아울러 우수한 일정을 실시간에 제공해 주는 일정계획 기법 개발이 필요하다.

과거에는 데이타의 획득과 처리의 어려움으로 생산관리에 부분적으로 수리적 기법을 이용하는 것으로 만족하였다. 그런데 급속도로 정보관리 기법이 발달됨에 따라 현재에는

생산시스템 전체를 총체적으로 관리하고 통제 할 수 있게 되어 FMS 및 CIM 시스템의 도입으로 이어 지고 있다. 문제가 잘 정의될 수 있는 상황에서는 수리적 기법을 이용하지만 수리적 기법의 실용성이 떨어질 경우에는 인공지능이나 전문가 시스템을 이용하는 방안들이 강구되고 있다[10]. 생산통제의 경우도 의사결정문제 자체를 잘 정의하기는 쉽지 않다. 따라서 수리적기법을 이용하기 보다는 대량의 정보처리 후 수립된 계획을 실천하는 과정에서 필요한 여러 가지 의사결정들을 지원해 주기 위한 의사결정지원 시스템을 개발할 필요가 있다. 이러한 의사결정지원 시스템에서 수리적으로 모형화하기 어려운 일정계획 문제에 대해서는 인공지능을 이용 할 수 있다. 인공지능을 이용하는 취지는 일정계획문제를 해결하는데 있어서 인공지능의 지식표현기법을 통해 현실제약과 일정계획 지식을 구체적으로 표현하고 지식을 이용해서 탐색범위를 줄이는 알고리즘을 적용함으로써 현실적으로 실행가능한 좋은 해를 찾으려고 하는 것이다. 인공지능을 응용하는 것은 OR기법의 자연스러운 확장이라고 볼 수 있는데, 인공지능 분야에서는 분지한계법과 같은 OR 기법을 확장한 간단한 휴리스틱보다는 좀 더 정교하고 효율적인 탐색기법을 개발해 왔고 휴리스틱 추론과 표현에 주로 기여해 왔다. 인공지능이나 OR 둘 다 탐색기법에 관련된 것이긴 하지만 인공지능은 탐색방향을 결정해 주는 지식을 강조한다는 점이 차이가 있다 [4].

Ammons et al.[3]은 고도로 자동화된 수준의 FMS에서 인간의 역할을 감독자로 규정하고 인간의 역할을 FMS 설계시에 소프트웨어나 하드웨어 설계와 마찬가지로 설계해야 한다고 주장한다. 일정계획과 통제를 무인자동화하는 것이 현실적으로 한계를 드러낼 수 밖에 없으며, 따라서 자동화가 어떠한 수준이던 간에 결국 운영에 대한 책임을 인간이 질 수 밖에 없음을 주장하고 인간의 역할을 고려한 FMS 설계 방침으로서, 'supervisory control paradigm'을 제안하였다.

이러한 맥락에서 본 연구에서는 생산통제 시스템을 규칙기반시스템기법과 heuristic을 이용하여 구현하였고 인간이 생산일정에 개입할 수 있도록 하여 생산일정의 현실성을 높이고 일정의 신속한 수립과 재수립을 꾀하고 있다.

본 연구에서 목표로 하는 생산통제 시스템은 다음과 같은 특성을 가진다.

1. 사용자 인터페이스를 통한 일정정보 검색과 편집기능
2. 자동일정계획기능

사용자 인터페이스를 통해 자동일정계획 기능으로 처리할 수 없는 예외를 처리하고, 주어진 정보로 고려할 수가 없었던 생산현장 상황을 반영시켜 주어 생산일정을 현실화시켜 주는 등, 사용자의 지식과 의사를 적극 반영할 수 있도록 하였다.

본 연구에서는 사용자와 대화를 하도록 사용자 인터페이스 환경을 구현하였는데, 생산일정이 그래픽화면에 간트 도표 형식으로 제시되며, 필요한 정보를 마우스와 아이콘을 이용하여 대화식으로 검색해 볼 수 있다. 그리고 작업의 이동, 교환, 추가 및 삭제 등의 편집기능을 이용하여 일정을 변경시켜 줄 수

있도록 하였다. 또한 사용자가 일정을 변경하였을 때 파급되는 효과를 눈으로 쉽게 확인할 수 있다. 본 연구의 또다른 중요한 기능인 자동일정계획모듈은 일반적인 job shop 문제에 대해 makespan 측면에서 좋은 일정을 수립해 주는 기능을 수행한다. 생산통제시스템 구현에는 C++ 언어를 이용하였다.

2. 사용자 편의의 일정편집기능

편집기능이 작동하기 위해서는 작업의 선후행관계들을 지속적으로 유지시켜야 한다. 선후행관계 제약은 규칙(rule)으로서 함축되어 표시되며, 일정이 흐트러졌을 때 일정이 제약조건을 만족하도록 제약을 전파시킨다. 제약이 어겨졌을 때 그 제약조건을 추적하여 일정을 실행가능하게 유지시켜 주는 기능을 규칙기반 시스템이 추론을 통해 수행해 준다. 일정편집시 이용되는 규칙기반 시스템 모듈은 일정편집시 뿐만 아니라 자동일정계획 기능 수행시에 좋은 일정을 찾기 위해 반복되는 일정변경에 대한 실행가능성을 유지하도록 하는데에도 사용된다.

본 시스템에서는 편집기능을 이용하여 사용자가 일정편집기와 대화를 통해 일정계획의 성능을 향상시키거나 문제상황 발생시 재일정수립을 수동으로 시도해 볼 수 있다. 따라서 일정계획수립시 사용자의 지식과 경험을 직접 활용할 수 있다. 이 기능으로 사용자의 일정계획에 대한 이해를 도울 수 있고 정보의 불확실성에 의한 잘못을 그래픽을 통해 확인할 수 있다. 이는 사용자가 시스템을 사용하면서 성능향상을 위한 여러가지 경우를 발견하여 지식을 획득하고 규칙베이스(Rule Base)화 시킬 수 있는 여지를 남겨둔 것이기도 하다.

작업교환, 작업이동, 작업삭제, 추가 기능, 작업경로 파악, 작업정보 검색 기능이 일정편집기능으로 제공되며, 모두 아이콘과 마우스를 이용하여 편집을 할 수 있도록 되어 있다. 작업이동의 예를 보면 먼저 작업이동 아이콘을 누른 후 이동시킬 작업물을 선택하여 마우스 버튼을 누른 채로 옮기고자 하는 위치로 옮기면 된다.

편집기의 기본적인 기능은 다음과 같다. 사용자가 편집을 하였을 때 편집된 작업이 제약을 만족하지 않을 경우 다른 작업의 가공시간대를 변경시켜 실행가능성을 타진하게 되는데, 제약을 만족하지 않음으로써 다른 작업에 미치는 여파를 추적한다. 결국 사용자가 취한 작업변경이 받아들여 질 수 있는가를 추론하여 받아들여 질 수 있을 경우에는 일정을 수정해주고, 그렇지 않을 경우에는 편집을 기각시킨다. 그런데 이 파급효과를 추적하는데 있어서 규칙기반 시스템을 이용할 경우 제어의 흐름을 일일이 고려하여 코딩시켜 주지 않아도 되므로 이러한 기능을 구현하는데 있어서 문제의 복잡도를 줄일 수 있다.

편집기능이 공통으로 거치게 되는 과정을 작업이동의 예를 들어 설명하면 다음과 같다. 마우스로 아이콘을 누른 후 작업물을 선택하고 이동시키면 이것은 사용자 인터페이스를 통하여 추론기관에 정보로 전달된다. 전달된 정보는 작업메모리(working memory)에 입력되고 추론기관으로 하여금 추론을 시작하도록 한다. 이동된 작업과 옮겨진 상태를 인식하고 제약조건들이 만족되는가를 검토한다.

인식과정은 옮겨진 작업이 위치한 시간대에서 다른 작업과 함께 배열된 양태를 파악하는 것이다. 인식이 끝나면 다음과 같은 제약조건들을 검토한다.

① 작업가능한 기계인가?
② 선행공정의 종료시간을 앞지르는가?
③ 선행작업이 없다면 원재료가 있는가?
④ 후행작업 또는 옮겨지는 작업물이 처리되는 기계에서 시간적으로 다음에 오게 되는 작업의 처리시간을 침범하는가?

처음 세가지 조건들이 가능하지 않다면 원상태로 돌리고 마지막 조건의 경우에는 후행작업이 뒤로 밀리면서 실행가능한 일정이 되는지 추론을 계속하게 된다. 후행 작업 또는 같은 기계에서 가공되는 작업물의 작업시간을 침범하면 선택된 현 작업물의 정보를 Stack에 넣게 된다. 이 과정을 되풀이 한 후, 더 이상 고려해야 될 문제가 없으면 Stack에서 정보를 꺼내서 같은 기계에서 가공되는 작업물을 고려하였는가를 확인한다. 고려하였으면 Queue에 넣고, 그렇지 않으면 Stack에 다시 넣고 앞에서 수행했던 과정을 반복한다.

편집기능은 어겨진 제약이 파급된 효과를 추적하기 위해 Stack, Queue를 사용하는데 그림 1에 묘사하였다.

객체지향언어인 C++로 구현하므로서 절차식언어인 FORTRAN, C를 이용한 재래식 프로그래밍이나 다른 지식표현방법과는 달리 [6] 규칙을 명확히 다른 원시코드와 분리하여 표현할 수 있으므로 규칙을 읽을 때 그리고 추가삭제시킬 때 비교적 수월하게 되어

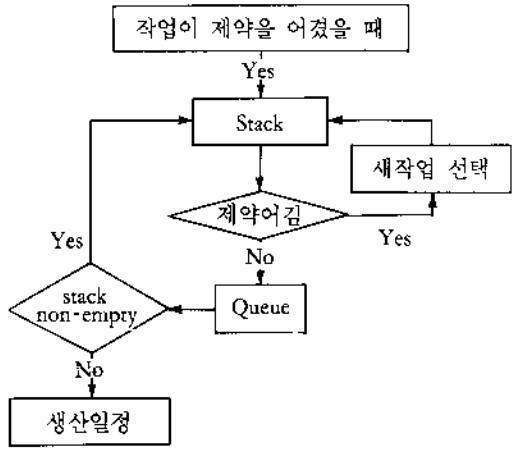

그림 1. 제약이 위반 되었을 때의 파급효과추적

있다. 규칙 표현 방법을 보기 위해 규칙을 표현하는 소스 코드의 일부를 보면 다음과 같다.

예) 작업교환의 경우
IF 선택된 작업들이 앞당겨질때 밀리는 것을 파악하는 단계 AND 작업 1의 시작 시간 〈 작업 2의 시작시간
THEN 앞으로 당겨지는 작업 = 작업2
고려중인 작업 = 작업2
뒤로 밀리는 작업 = 작업1
단계 = 작업의 앞당겨질 수 있는 시간을 결정하는 단계

```
// RULE
rule = new Rule;
rule->IF(EQUAL, STAGE, DETERMINE_MOVING_DIREC-
TION);
rule->AND(LESS_THAN, JOB1_STARTTIME, JOB2_START-
TIME);
rule->THEN(EQUAL, JOB2, MOVING_FORWARD);
```

```
rule->THEN_AND(EQUAL, CURRENT_JOB, JOB2);
rule->THEN_AND(EQUAL, JOB1, MOVING_BACKWARD
);
rule->THEN_AND(EQUAL,STAGE,
                DETERMINE_MOVING_EARLY_
                TIME);
rule_set->Append_rule(rule);
```

여기서 IF(), AND(), THEN(), THEN_AND()는 class Rule의 member function이다. 이런 식으로 규칙이 표현되고 프로그램 시작시에 생성시켜 rule-set에 저장한다. rule-set은 class RuleBase로 선언된 변수이다. IF(), AND(), THEN()의 인수로 전달되는 것은 내부에서 class Predicate로 저장되며 IF(relation, subject, object)로 표현된다. 편집을 위한 규칙은 모두 22개이다. 이 규칙들은 모두 규칙베이스를 표현하는 객체에 저장된다.

3. 일정수립 기능

3-1 일정수립 방법

생산통제 시스템에서의 일정수립 기능은 상위 계획단계에서 수립된 계획을 실천하기 위해 초기 일정계획수립시와 일정진행 과정에서의 재일정계획시에 이용된다. 본 연구에서 제시하는 일정계획 편집기는 기존의 우선순위 규칙을 이용하여 수립된 일정계획보다 나은 일정을 실시간에 제공할 것을 목적으로 하고 있다. 본 기능은 사용자 대화환경에서 손쉽게 이용할 수 있으며 결과를 바로 화면에서 확인할 수 있다. 본 연구에서는 이를 위해 일정수립을 2단계로 나누어 실시하는 방안을 고안했다.

단계 1에서는 우선순위 규칙을 이용하여 Giffler & Thompson 알고리즘으로 'active schedule'을 생성시켜 준다[8]. Giffler & Thompon 알고리즘에서 사용되는 우선순위 규칙으로는 'Maximum Flow Time'에 대해서 가장 좋은 수행도를 보여 준다고 알려진 MWKR(Most Work Remaining) 규칙을 사용한다. 'Maximum Flow Time'은 모든 job의 가공가능시점이 동일할 경우 makespan과 동일한 수행척도가 된다. 단계 2에서는 단계 1에서 수립된 일정을 대상으로 makespan측면에서 개선이 될 수 있도록 탐색을 통한 일정변경을 모색하게 된다. 탐색은 특별히 makespan을 줄이기 위한 탐색전략에 의해 탐색방향과 깊이가 결정된다. 단계 1에서 생성된 생산일정은 하한이 되며 본 시스템에서는 최소한 단계 1에서 생성된 일정 수준의 수행도를 가지는 일정을 제공하게 된다. 단계 1의 Giffler & Thom-pson Algorithm은 참고문헌을 참조하기 바란다[8].

단계 2의 일정개선 알고리즘은 다음과 같다. 일정개선을 위한 알고리즘의 기본 내용은 makespan과 가공 종료시간이 일치하는 공정을 찾아 이를 앞당길 수 있도록 조치를 취해 주는 것이다. 단계 1을 거친 일정에서 makespan을 줄이기 위해 공정의 착수시점을 앞당기려 할 때, 이 공정이 처해 있는 상황은 선행작업에 막혀 있거나 같은 기계에서 가공되는 다른 작업에 의해 막혀 있는 등 두 가지 경우가 있을 수 있다. 같은 기계에서 가공되는 또 다른 작업에 의해 막혀 있는 경우에는 가공되는 기계에서의 유휴시간을 줄여 줄 수 있도록 그 기계에서

가공되는 작업들의 착수 시점을 조정해 주거나 makespan에 주는 영향을 고려하여 작업을 교환하여 착수시점을 변경하도록 한다. 선행작업에 의해 막혀 있는 경우에는 선행작업들을 추적하여 선행작업을 앞당길 수 있도록 가능성 있는 선행작업을 선택하여 앞에서와 같은 방법으로 착수시점을 조정 및 변경시킨다. 알고리즘에서 작업을 앞당기거나 뒤로 미룰 때 작업의 선후행관계 제약을 유지시켜 주기 위해서 2장에서 설명한 편집규칙을 이용한 추론기능이 이용된다.

단계 2에서의 알고리즘은 두가지 탐색전략과 종료기준을 사용한다.

1. Front_탐색

탐색기준 공정과 같은 기계에서 가공되는 공정들 중 탐색기준 공정 앞쪽에 있는 공정들의 작업착수시점을 조정하므로서 탐색기준 공정을 앞당기는 시도를 하는 procedure

2. Level_탐색

탐색기준 공정의 선행공정을 추적하여 앞당겨질 가능성이 있는 작업에 대해 'Front_탐색'을 시도하도록 해주는 procedure

3. 종료기준

이 휴리스틱의 종료기준은 주 모듈의 반복 횟수를 미리 정해 놓고 이 횟수까지 실행시키는 것이다. 경험적으로 실행횟수는 대략 160회 정도가 적당하고 생각하였으며, 3-2에서 실험을 할 때 반복횟수를 160회로 제한하였다. 또한 알고리즘 진행중에 makespan에 걸리는 공정이 가공되는 기계에서 유휴시간이 없는 경우에 종료한다.

단계 1, 2에서 사용된 탐색전략은 일종의 분지전략이다.이를 구체적으로 다음과 같이 자세하게 가상코드로 묘사하였다.

● 기호 정의

schedule_list = 생산일정, 그일정에 적용되어야할 탐색 전략, 탐색기준 작업을 하나의 노드에 저장하는데 이 노드들을 저장하는 리스트(저장하고 있는 노드들을 makespan의 오름차순으로 정렬)

current_schedule = 현재 일정

idle_interval_list = 특정기계에서의 유휴시간 구간을 저장하는 리스트

basis_operation = 탐색기준 공정

candidate_list = 'Front_탐색'에서 유휴시간 안쪽으로 당겨 질 수 있는 공정들을 저장하는 리스트

mode = 탐색전략

precedent_list = 'Level_탐색'시에 앞당겨 질 가능성이 있는 공정을 저장하는 리스트

● 휴리스틱 의사코드(pseudo code)

(1) Main

```
Main Module
{
   schedule_list 생성
   current_schedule = 단계 1에서 생성된 일정
   mode = '탐색방향_결정'
```

```
basis_operation = makespan에 걸리는 공정
schedule_list에 (current_schedule, mode, basis_
                operation)추가
while(schedule_list가 비어 있지 않으면)
    {
   종료기준 평가
   schedule_list 정리
   schedule_list의 첫번째 노드 선택
   current_schedule = 노드에 저장되어 있는 일
                정
   basis_operation = 노드에 저장되어 있는 basis
                _operation
   mode = 노드에 저장되어 있는 mode schedule
                _list의 첫번째 노드 삭제
switch(mode)
{
  case 1: '탐색방향결정'
        basis_operation이 선행작업에 막혀 있는
        지 같은 기계의 바로 앞작업에 막혀 있
        는지 파악
        if(선행작업에 막혀 있다)
          Level_탐색 실행
        if(같은 기계의 바로 앞 작업에
          막혀 있다)
        Front_탐색 실행
  case 2: 'Level_탐색'
        Level_탐색 실행
  case 3: 'Front_탐색'
        Front_탐색 실행
  }
}
```

(2) Procedure

```
Front_탐색
{
  basis_operation 앞쪽의 기계유휴시간(idle time)을
  검색, idle_interval_list에 저장

  if(idle_interval_list가 비어 있지 않다)
  {
   유휴시간 이후의 작업들 중 선행작업의 종료시
   간을 고려하여 유휴시간 안으로 당겨질 수 있는
   것들을 candidate_list에 저장
}
else
{
  if(basis_operation == makespan에 걸리는 공정)
   {
     최적일정이다.
     종료
  }
  basis_operation과 앞공정과 교환해 본다
  if(test해서 좋으면)
 {
  basis_operation의 후행작업들을 거슬러 올라 가
  면서 'Front_탐색'
  basis_operation = makespan에 걸리는 공정
  mode = '탐색방향 결정'
  schedule_list에 (current_schedule, mode, basis_
  operation)추가
  }
}

if(candidate_list가 비어 있지 않다)
{
  가장 앞으로 당겨질 수 있는 작업을 선택하여 앞
  으로 당긴다.
```

```
    if(앞으로 당겨 지면)
    {
    일정전체에 대해 작업순서를 바꾸지 않는 범위내
    에서 유휴시간 줄이기
    if(basis_operation이 makespan에 걸리는 작업이
      아님)
    {
      basis_operation의 후행작업들을 거슬러 올라 가
      면서 'Front_탐색'
    }
      basis_operation = makespan에 걸리는 공정
      mode = '탐색방향 결정'
      schedule_list에 (current_schedule, mode, basis_
      operation)추가
    }
    else
    {
     basis_operation 앞과 유휴시간 사이의 첫번째 공
     정부터
     while(현 대상공정 ≠ basis_operation)
     {
     basis_operation = basis_operation 앞과 유휴구
     간 사이의 공정
     mode = 'Level_탐색'
     schedule_list에 (current_schedule, mode, basis_
     operation)추가
     다음 공정으로
     }
    }
}
Level_탐색
{
    basis_operation의 선행작업들을 추적, 선후행작업
    들간의 간격이 있는 것들을 precedent_list에 저장
     precedent_list의 첫번째 공정부터
     while(precedent_list의 끝이 아님)
     {
     basis_operation = precedent_list의 공정
     mode = 'Front_탐색'
     schedule_list에 (current_schedule,
     mode, basis_operation)추가
     precedent_list의 다음 노드
     }
}
```

3-2 일정개선 휴리스틱의 성능평가실험

가. 실험목적 및 문제생성

본 연구에서 제안한 일정개선 기능의 목표는 앞 절에서도 언급하였다시피, 일정개선 알고리즘으로 초기에 우선순위 규칙 MWKR을 이용하여 생성된 일정을 개선시켜 SPT나 FCFS와 같은 단순우선순위규칙만을 사용하였을 때보다 나은 일정을 아주 빠른 시간에 얻고자 하는 것이다. 따라서 본 실험을 통하여 일정개선 알고리즘이 초기 일정을 어느 정도로 개선시켜 주는지를 파악하고자 하였다.

이를 위해 여러 크기의 문제를 임의적으로 발생시킨 다음 이를 본 연구에서 제안한 방식으로 일정을 수립한다. 이때 단계 1의 초기일정이 단계 2를 통해 개선되는 빈도를 알아 낸다. 또한 대표적인 우선순위 규칙인 FCFS, SPT를 동일한 문제에 대해 Giffler & Thompson의 알고리즘을 적용하여 생성시킨 일정과 본 연구에서 제안한 방식으로 생성시킨 일정을 makespan 측면에서 수행도를 비교한다. 한편 일정을 빠른 시간에 찾아 주는 것

도 또 하나의 목표이므로 수행시간도 함께 측정한다.

기계와 Job의 갯수가 주어져 있을 때, Job의 공정갯수는 기계댓수와 동일하게 두었으며 가공시간을 U(1800,5400)인 일양분포를 따르게 하였다. 가공경로(routing)를 결정할 때 Job이 거쳐가는 기계를 일양분포로 선택되게 하였고 같은 기계를 2번 이상 거치지 않도록 하였다.

실험은 문제크기를 기계댓수/job수로 표시하였을 때, 5/5, 7/7, 9/9, 10/10에 대해서 각각 30문제씩 발생시켜 SPT, FCFS 그리고 본 연구에서 제안한 알고리즘을 이용하여 각각에 대해 일정을 생성시켜 보았다. 일반적인 Job Shop의 경우에는 문제가 10/10인 경우에도 최적해를 구하는 것이 무척어려운 것으로 보고되고 있어 앞에서 열거한 정도의 문제는 일반적인 Job Shop에서 다루는 문제크기와 비교할 때 적당한 것이라 생각한다[1,8].

나. 실험결과 및 분석

표 1은 본 연구에서 제시한 알고리즘이 기존의 알고리즘(단계 1)을 거쳐 생성된 초기일정을 개선시키는데에 성공한 빈도와 평균계산시간을 요약해 놓은 것이다.

표 1에서 ①은 주어진 문제에서의 공정의 갯수를 나타낸다. ②는 30번 실행한 결과 초기일정을 향상시킨 빈도수를 나타낸다. ③은 각 문제 크기별로 생성된 예제들의 초기일정을 개선시키는데 소요된 시간의 평균이다. ④는 공정수에 자연로그를 취한 값이고 ⑤는 계산시간에 자연로그를 취한 값이다.

공정수와 계산시간의 관계를 그림 2에 나타내었다.

표 1. 초기일정개선 빈도와 평균계산시간

문제	① 공정수	② 초기일정 향상빈도	③ 계산시간 (sec)	④ ln(공정수)	⑤ ln(계산시간)
5/5	25	24(30)*	3.206	3.2188	1.1650
7/7	49	28(30)	9.130	3.8918	2.0224
9/9	81	29(30)	23.365	4.3944	3.1512
10/10	100	28(30)	34.905	4.6051	3.5526

* ( )안은 실행 횟수

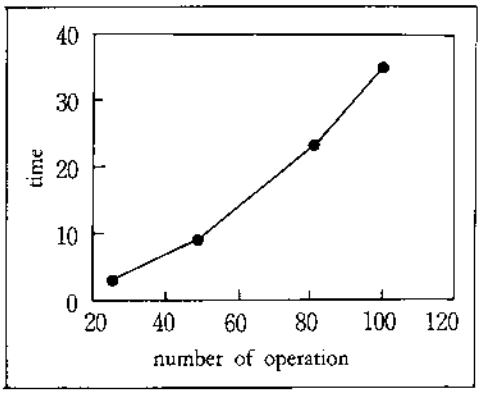

그림 2. 공정수 변화에 따른 계산시간변화

본 연구에서는 공정수를 문제의 복잡도를 나타내는 기준으로 삼았다. 공정수가 증가함에 따른 계산시간 증가 정도를 파악하기 위해 공정수와 계산시간에 로그를 취하고 이들의 관계를 회귀분석한 후 이를 근거로 계산시간을 공정수의 함수로 표현하였다.

그림 3은 공정수와 계산시간에 자연로그를 취해 나타 낸 것이다.

T = 계산시간, N = 공정수, y = log T, x = log N으로 기호를 정의할 때, 공정수와 계산시간간의 관계를 단순회귀모형으로 모형화시키면 y = a + bx 이 된다. 여기서 a, b

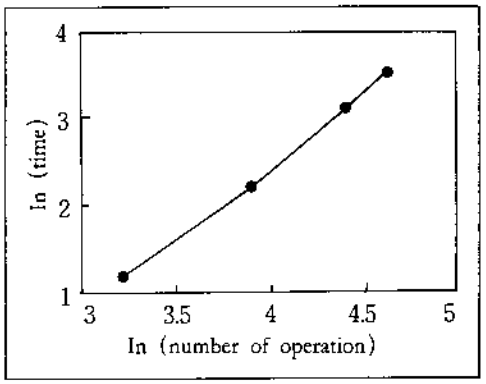

그림 3. 공정수와 계산시간 각각에 자연로그를 취했을 경우의 공정수변화에 따른 계산시간 변화

는 회귀계수이다.

회귀분석결과 회귀계수 a = -4.421, b = 1.723으로서 회귀분석식은 y = -4.421 + 1.723x, 결정계수 R^2가 0.988으로 나타났다. 결정계수(Cofficient of determination)는 0과 1 사이의 값을 가지는데 1에 가까울수록 회귀분석식이 타당하다고 볼 수 있다. 비록 Sample Data의 규모는 작지만 실험결과에서 구해진 값을 보면 상당히 높은 결정계수값을 가지므로 결과로서 얻어진 관계식이 타당할 것이라고 조심스럽게 결론지을 수 있다.

회귀분석식으로부터 $T = e^{-4.421}$, $N^{1.723} = 0.012 \times N^{1.723}$을 유도할 수 있다. 따라서 계산시간 T가 공정수 N의 다항식으로 표현됨을 알 수 있다. 위의 단순회귀분석결과가 유효한 제한된 범위내에서는 공정수의 증가에 따른 계산시간의 증가를 감내할 수 있을 것으로 판단된다. 표 2는 각각의 규칙으로 만들어진 일정들을 makespan 측면에서 30번 비교하였을 때 각 일정생성 방법별로 가장 좋은 일정을 생성시킨 경우의 수를 나타낸다. 표에서 알 수 있듯이 대부분의 경우에는 본

표 2. 규칙기반 일정계획수립기와 우선순위규칙에 의한 일정계획 성능비교

문제	가공공정수 (operation)	SPT	FCFS	본연구에서의 방법
5/5	25	0	3	27
7/7	49	0	4	26
9/9	81	0	2	28
10/10	100	0	1	29

연구에서 제안한 방법이 가장 좋은 것으로 나타났으나 FCFS의 경우에는 적은 빈도수이지만 매 문제크기마다 가장 좋은 일정을 제공해 주는 경우가 있는 것으로 나타나 본 연구에서 제안한 방법이 규칙을 사용하는 경우보다 항상 좋다라고 말할 수는 없다. 그러나 대부분의 경우에는 본 연구에서 제안한 방법이 우세하므로 단순규칙을 사용하는 것보다 나은 일정을 제공한다고 볼 수 있다.

5. 결론 및 향후 연구과제

생산통제는 상위 계획 단계에서 결정된 생산계획을 실천하기 위해 운영적 수준의 생산일정을 수립하고 생산일정진행시 재일정이 요구되는 상황에서 신속히 재일정을 수립해주는 기능을 수행한다. 일정계획문제는 수리적으로 모형화시켰을 경우, 많은 제약과 변수로 인해 해의 공간이 크고 복잡하다. 더구나 생산통제의 경우 작업장의 동적특성으로 인해 재일정수립요구가 잦아, 많은 노력을 들여 일정을 재수립한다는 것이 시간제약으로 인해 허락되지 않고 실시간으로 빨리 좋은 일정을 수립하는 것이 중

요하다. 따라서 생산통제를 위한 의사결정 지원 시스템으로 이러한 업무들을 손쉽게 하고자 함이 본 연구의 목적이며, 이를 위해 사용자 편의의 편집환경과 자동재일정 기능을 갖춘 생산통제용 일정계획 편집기를 개발하였다.

편집기능은 편집시 일정변경에 대해 일정의 실행가능성을 유지시켜 주는 기능은 규칙기반 시스템이 추론을 통해 은연중에 제약으로 형성하여 전파시킴으로서 전체 일정이 선후행 제약을 만족시켜 주도록 일정을 조정해 준다.

본 연구에서는 일반적인 job shop형태의 작업장의 생산통제를 다루고 있으며, 일정은 makespan으로 평가된다. 일정수립기능은 2단계 방법이 적용되며, 초기일정은 'Giffler & Thopmson의 알고리즘'에 우선순위규칙을 적용하여 얻어진다. 본 연구에서는 단계 1에서 사용하는 규칙으로, 수행평가척도가 maximum flow time일 경우 좋은 일정을 제공해 준다고 보고된 MWKR 규칙을 사용하였다. 단계 2에서는 단계 1에서 수립된 일정을 받아서 일정개선 알고리즘으로 개선시켜 준다. 일정개선 알고리즘은 makespan에 걸리는 공정을 앞당기기 위한 두가지 탐색전략에 의해 해의 공간을 탐색하여 더 나은 일정을 찾아 준다. 임의적으로 발생시킨 다양한 크기의 예제들에 대해 실험을 한 결과 SPT, FCFS와 같은 우선순위규칙을 이용해서 일정을 생성한 각각의 결과보다 대개의 경우 더 나은 일정을 생성시켜 줌을 알 수 있었다. 일정개선 알고리즘에서 탐색시에 일정을 변경시켜주는 시도들이 반복되는데, 이때 일정의 실행가능성을 유지시켜 주기 위해 사용자 편집시와 마찬가지로 규칙기반 시스템에 의한 추론이 작동된다.

본 시스템의 편집기에서는 생산일정정보를 간트차트 형식으로 나타내줌으로써 사용자가 쉽게 일정을 이해할 수 있다. 또한 사용자와의 대화를 통해 생산일정을 변경할 수 있으며 국부적으로 변경된 일정이 전체에 어떻게 영향을 주는지 즉시 보여줌으로써 사용자가 생산일정에 대한 이해와 통찰력을 가질 수 있고, 정보의 불완전성을 발견하고 고칠 수도 있다. 이것으로 생산통제시스템의 실용성과 생산일정계획의 수행도 향상을 기대할 수 있다. 또한 계획에서 벗어나는 이상 상황에 대해 전체적으로 다시 일정계획을 세우지 않고도 짧은 시간 안에 재 일정계획을 수립할 수 있다.

기업에서 생산성향상을 위해 필요에 따라 자동화를 하는 것이 당연한 과제로 받아들여 지고 있으나 자동화 시스템을 도입할 경우에도 자동화된 생산 통제기능이 처리하지 못하는 예외상황처리와 자동화 시스템의 효과와 안정성을 위하여 인간의 개입이 불가피하므로 일정계획과 통제를 완전 무인자동화하기 보다는 인간의 역할을 생산통제 기능에 활용할 수 있도록 시스템 설계시에 고려해 주는 것이 바람직 하다. 그러므로 감독자로서의 인간의 역할을 효과적으로 수행할 수 있도록 하는, 사용자와 생산시스템간의 인터페이스 기능이 중요하다. 이러한 측면에서 사용자와 생산현장을 친밀하게 연결시켜 주면서 좋은 일정을 수립해 주는 생산통제 시스템 개발에 본 연구에서 개발된 시스템이 하나의 모형이 될

수 있을 것이다.

일정계획수립시에 실제 문제에 있어서는 다양한 요구조건이 존재하는데 이들은 상호마찰을 일으킬 수 있다. 향후 연구과제로서 이를 다양한 평가기준으로 설정하고 이들을 동시에 만족시키면서 좋은 일정을 제공해 줄 수 있도록 하는 연구로 확장이 가능하다. 그리고 본 연구에서의 탐색방법은 문제영역에 대한 좀더 자세한 연구를 통해 얻어진 지식을 탐색전략에 반영하므로서 좀더 정교하고 효율적일 수 있는 여지가 남아 있다.

6. 참고 문헌

[1] Amoldo C. Hax and Dan Candea, *Production and Inventory Management*, Prentice-Hall, 1984.

[2] James L. Riggs, *Production Systems: planning, analysis and control*, 4th Ed. John Wiely & Sons, 1987.

[3] Jane C. Ammons, T. Govindaraj, C.M. Mitchell,"A supervisory control paradigm for real-time control of flexible manufacturing systems," Annals of Operations Research,Vol.15, pp315-335., 1988, J. C. Baltzer A, G. Scientific Publishing Company.

[4] Jeff Rickel, "Issues in the design of scheduling systems", *Expert systems and Intelligent Manufacturing*, edited by Michael D. Oliff.

[5] Khosrow Hadavi, Wen-Ling Hsu, Tony Chen, and Cheoung-Nam Lee, "An Architecture for Real Time Distributed Scheduling," *Artificial Intelligence Applications in Manufacturing*, Edited by A. (Fazel) Famili Dana S. Nau, Steven H. Kim.

[6] Robert I. Levine, Diane E. Drang, Barry Edelson, *AI and Expert Systems, A Comprehensive Guide, C language*, second edition, McGraw-Hill, 1991.

[7] Roger Kerr, *Knowledge-Based Manufacturing Management*, Addison Wesley, 1991.

[8] Simon French, B. A., M. A. & D. Phil., *Sequencing and Scheduling : An Introduction to the Mathematics of the Job-Shop*, John Wiley & Sons, 1982.

[9] Stephen C. Graves, "A Review of Production Scheduling," Operation Research , Vol. 29, No. 4, July-August 1981.

[10] 홍유신, 성덕현, 박기진, "전문가 시스템 및 인공지능을 이용한 생산관리를 위한 기초조사", Journal of the Korean Institute of Industrial Engineers, Vol. 16, No.1, June, 1990.