Journal of the Korean Institute
of Industrial Engineers
Vol.22, No.2, June, 1996

단일기계 일정계획을 위한 제약조건 표현언어 및 코드 자동생성기*

Constraint Description Language and Automatic Code Generator for Single-Machine Job Sequencing Problems

이유근**, 백선덕**, 배성문**, 전치혁**, 장수영**, 최인준**

You K. Lee**, Seon D. Baek**, Sung M. Bae**, Chi H. Jun**, Soo Y. Chang**, In J. Choi**

Abstract

Scheduling problems which determine the sequence of jobs are one of the important issues to many industries. This paper deals with a single-machine job sequencing problem which has complex constraints and an objective function. To solve the problem, an expressive constraint description language and an automatic code generator are developed for our scheduling system. The user just needs to describe the scheduling problem using the constraint description language that allows to express both quantitative and qualitative constraints as well as an objective function in real world semantics. Then, a complete scheduling program based on constraint satisfaction technique is automatically generated through the code generator. Advantage of this approach is that models of the scheduling problems are easily developed and maintained because models are formulated by using the language which reflects real world semantics.

1. 서론

공정 또는 제작기계에 투입될 작업 대상재의 작업 순서를 결정하는 문제인 일정계획 문제는 대부분의 산업체에서 당면하고 있는 중요 문제 중 하나이다. 일정계획 문제에 대하여 오랜 연구[1, 9, 12, 17, 18]가 있어 왔지만 간단한 일정계획 문제에만최적해를 구

* 본 연구는 한국과학재단의 우수연구센터인 공정산업의 지능자동화센터에 의해 지원되었음

** 포항공과대학교 산업공학과

해주는 알고리듬이 존재할 뿐, 실제로 일어날 수 있는 복잡한 형태의 일정계획 문제를 위한 최적화 알고리듬은 존재하지 않는 형편이다.

본 연구에서 다루는 일정계획이란 한 대의 제작기계 또는 공정에 투입될 작업 대상재의 작업 순서를 정하는 일이다. 이러한 일정계획 문제는 작업장마다 다양한 형태를 갖는다. 수치적이거나 비수치적인 속성 정보가 존재하며, 제약조건은 논리적인 형태와 if then else의 형태도 존재한다. 이러한 형태의 일정계획 문제에 대한 연구를 살펴보면 [2, 3, 4, 5] 등이 있다.

이러한 문제를 해결하기 위한 일정계획분야의 많은 응용프로그램들에는 재사용성이 고려되어 있지 않다. 이 때문에 유사한 종류의 문제를 해결함에도 불구하고 각각의 문제를 풀기 위한 별도의 프로그램을 개발하는데 상당한 시간과 비용을 유발시킨다. 또한, 프로그램 개발을 위해서는 공정에 대한 지식과 프로그래밍에 대한 지식 모두를 필요로 한다. 즉, 하나의 일정계획 프로그램을 개발하기 위해서 보통 프로그래머와 현장 근무자가 협력하여야 한다. 이것 역시 많은 개발 시간과 비용을 초래하는 요소이다.

이러한 프로그램 개발상의 여러 문제점들을 해결하기 위한 방법론으로 제약조건 만족 패러다임(Constraint satisfaction paradigm)이 제안되었다[6, 14]. 제약조건 만족 패러다임이란 사용자가 만족시켜야 할 제약조건들을 선언적으로 표현하면, 시스템이 자신의 알고리듬을 이용하여 자동적으로 해를 찾아준다는 것이다. 이 패러다임의 기본 철학은 문제해결을 위한 알고리듬과 문제기반 지식의 분리를 통해 사용자가 쉽게 해를 얻을 수 있도록 한다는 것이다.

본 연구의 목적은 다음과 같다. 첫째, 수치적 제약조건과 논리적 제약조건을 요구하며 목적식을 가지는 일정계획 문제 해결을 위한 알고리듬의 프레임워크(Framework)를 제시한다. 둘째, 개발된 알고리듬의프레임워크를 적용한 일정계획 언어를 제약조건 만족 패러다임에 근거하여 개발한다. 셋째, 본 일정계획 언어를 이용하여 사용자가 일정계획 문제를 정의하면 자동적으로 일정계획 프로그램을 생성할 수 있는 일정계획 프로그램 생성기를 개발한다.

2. 일정계획 문제 해결을 위한 방법론

2.1 일정계획 문제 정의

본 연구에서 다루고자 하는 일정계획 문제를 요약하면 그림 1과 같다. 일반적인 관점에서 보면, 본 문제는 작업 대상재의 집합(Set)으로부터 제약조건을 만족시키며 목적식의 최적화를 추구하는 부분 시퀀스(Subsequence)를 도출하는 문제이다. 제약조건으로는 개별적 대상재에 대한 제약조건, 집합적 대상재에 대한 제약조건, 그리고 보다 효율적인 탐색을 위한 Look-ahead 제약조건이 있으며, 최적의 해를 정의하기 위한 목적식이 있다. 각각의 제약조건은 수치와 문자 자료를 모두 포함하며, 논리적인 형태로 표현된다. 또한 if then else나 AND, OR와 같이 복잡한 형태로 나타날 수도 있다.

2.2 기존의 제약조건 만족 시스템

그림을 그리거나 수정하는 시스템인 Sketch-

INPUT

A set of same entities each of which has attributes.

A set of individual or aggregate constraints among attributes of jobs.

A set of lookahead heuristic constraints among attributes of jobs.

An objective function which represents the maximization or minimization of a expression of some of the attributes of jobs.

OUTPUT

A sub-sequence of the entities which satisfies all the constraints and maximizes or minimizes the objective function.

그림 1. 일정계획 문제 정의

pad[19]이후, 많은 제약조건 만족 시스템이 개발되어 왔다. 이 절에서는 이전에 개발된 여러 제약조건 만족 시스템을 설명하고자 한다. 본 연구에서 개발하고자 하는 제약조건 만족 시스템은 일정계획분야의 것이지만, 아직까지 일정계획분야의 시스템은 개발된 것이 없기 때문에, 지금까지 개발된 여러분야의 주요 제약조건 만족 시스템에 대해 설명한다.

Sketchpad[19]는 최초의 제약조건 만족시스템으로서 선을 그리거나 수정할 때, 제약조건을 사용하여 사용자가 쉽게 점이나 선들간의 관계를 만족시키는 도형을 그릴 수 있도록 하는 시스템이다. 이것의 제약조건이나 매크로 등의 여러 기능들은 기반 환경에 비해 시대를 앞선 것이었다.

Sketchpad와 유사한 시스템을 인터액티브(interactive)한 Smalltalk-76 프로그래밍 환경하에서 구축한 제약조건 기반 시뮬레이션 도구로 ThingLab[7]이 있다. 여기서 개별적 제약조건은 이에 연동된 알고리듬을 구현하는 Smalltalk 프로시져를 통해 확장 가능하게 하였다. 하지만 제약조건의 파라미터를 Smalltalk 프로시져의 입력 및 출력 파라미터로 받아들임으로써 유연한 제약조건을 표현하는 데에는 부족하다.

TK!Solver [13]는 IBM PC를 위한 수식 문제 해결 환경이다. 여기서는 문제를 정의하는 제약조건을 수식으로 정의한다. Relax할 변수를 사용자가 지시하고 또한 짐작치를 제공하여야 하는 단점이 있다.

Juno[16]는 그림을 WYSIWYG image에디터에 묘사하기 위한 언어이다. Juno의 기여는 제약조건을 그래픽으로 표현하여, 그 그

래픽으로부터제약조건 프로그램을 자동적으로 생성하는 능력이다. 하지만, 문제의 범위가 극히 제한적이며, 가능한 데이타 오브젝트가 오직 점이며, 오직 네 가지의 제약조건만이 존재한다.

2.3 일정계획 문제 해결을 위한 프레임워크

기존의 제약조건 만족 시스템은 제약조건만을 만족하는 해를 찾아 주는 것이었기 때문에, 탐색기법[12, 15]과 Local propagation [10, 14]과 같은 제약조건 만족 알고리듬 중의 하나 또는 그 이상을 결합하여 시스템을 구성할 수 있었다. 하지만 본 연구에서 다루는 일정계획 문제는 목적식을 포함하기 때문에, 단순히 제약조건 만족 알고리듬만을 이용하면 문제를 해결할 수 없다. 따라서, 본 연구에서는 목적식과 제약조건을 가지는 일정계획 문제를 해결하기 위하여 새로운 알고리듬의 프레임워크를 제시한다. 그림 2는 일정계획 문제 해결을 위한 프레임워크를 보여준다. 본 프레임워크는 2단계로 구성되는데, 먼저 제약조건 만족 기법을 이용하여 모든 제약조건을 만족하는 하나의 가능해(Feasible solution)를 찾고, 이어서 최적화 기법을 이용하여 가능해로부터 제약조건의 만족을 유지하면서 최적해(Optimal solution)에 가까운 우수해를 도출한다.

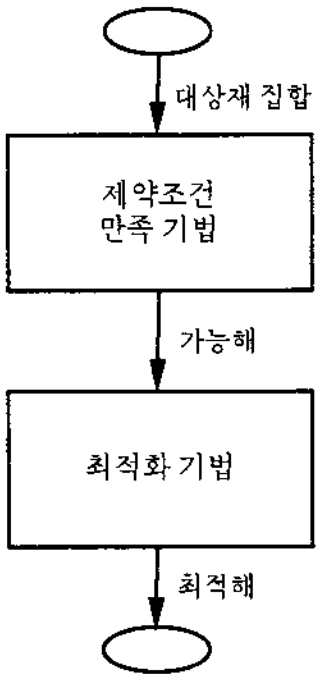

그림 2. 일정계획 문제 해결을 위한 프레임워크

2.4 제약조건 만족 기법

제약조건 만족 문제를 풀기 위해 수많은 알고리듬이 개발되어 왔다. 접근방법에 따라 크게 탐색 기법과 Local propagation 등이 있는데, 본 연구에서 다루고자 하는 방법은 탐색 기법이기 때문에 일단 Local propagation은 논외로 하겠다. 탐색 기법으로는 기본적인 Backtracking, Backjumping, Backmarking, Looking ahead, Partial looking ahead, 그리고 Forward checking 등이 있다. 여기서 기본적인 Backtracking이외의 방법들은 기본적인 Backtracking 방법에 발견적 기법들을 첨가한 형태라 볼 수 있다. 그림 3에는 서로 다른 알고리듬을 보여주는데, 아래로 내려갈수록 좀 더 많은 발견적 기법을 사용하는 것이다. 발견적 기법을 많이 사용할 수록 탐색 시간은 작아진다. 하지만 반대로 발견적 기법을 적용하는 시간이 그만큼 많아진다. 즉, 발견적 기법을 이용하는 정도에 대해서 탐색단계의 크기와 발견적 기법의 적용시간 간의 Trade-off 관계가 있음을 알 수 있다. 따라서 적용

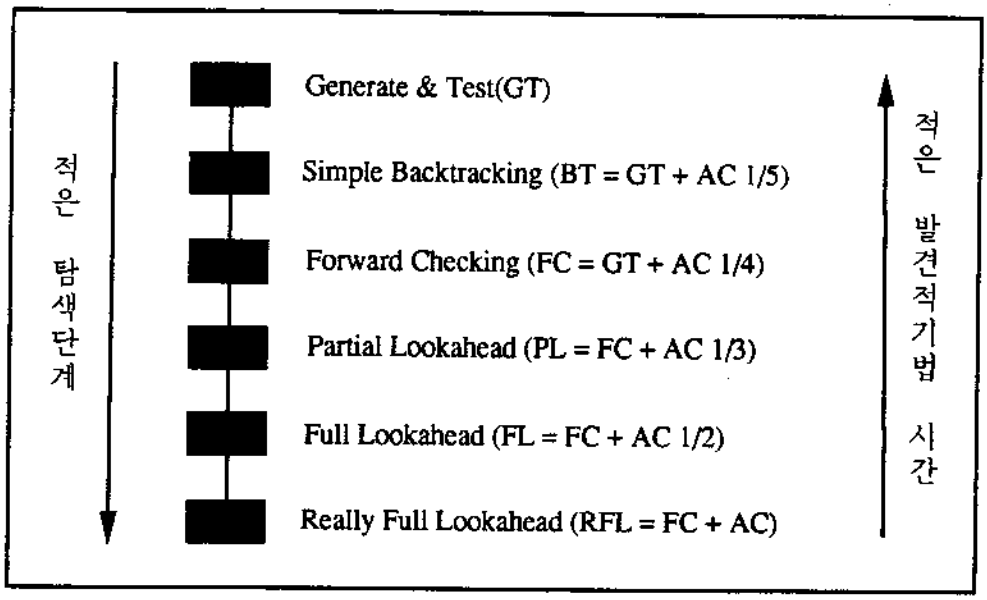

그림 3. 탐색기법의 Time Complexity

하고자 하는 문제에 따라 발견적 기법의 이용 정도가 달라질 수 있다. 실제로 N-queen 문제에 대해 각각의 알고리듬을 평가해 보았을 경우에는 Backmarking과 Forward checking이 가장 우수하다는 결과를 보였다[15].

2.4.1 발견적(Heuristic) 기법

본 연구에서는 적용하고자 하는 문제가 일정계획 문제이기 때문에, 대부분의 제약조건이 전후 대상재 사이에 작용한다는 가정하에서 Look-ahead를 활용한 발견적 기법을 포함하는 알고리듬을 개발하였다.

다음은 개발된 두 가지 발견적 기법을 설명한다. 먼저, Look-ahead는 Forward checking의 발견적 기법보다 단순한 것으로서, 아주 단순한 발견적 기법을 적용하여 적용시간을 줄이고 동시에 탐색단계도 줄이자는 것이다. 이 Look-ahead의 기능은 탐색 단계에서 부분해로부터 새로운 대상재로 진행할 때, 새로운 대상재가 과연 전체해로 도달할 수 있는지를 검사하여 불가능하면 그 대상재로 더 이상 진행하지 말자는 것이다. 둘째, 탐색 단계에서 새로운 대상재로 진행할 때, 가능하면 전체해에 도달하기 쉬운 대상재의 방향으로 진행하는 것이 탐색 시간을 좀 더 줄일 수 있다. 이와 같이 탐색을 할 때 유리한 대상재를 먼저 찾아가는 것을 대상재-Ordering이라고 한다. 대상재-Ordering은 알고리듬상에서 매우 쉽게 구현될 수 있는데, 실제로 각 탐색 때마다 대상재-Ordering을 하지 않고, 탐색 단계로 들어가기 전 단계에서 한번만 순서를 결정하면 탐색 단계에서는 매 탐색 때마다 그 순서에 따라 대상재를 적용하게 된다. Look-ahead와 대상재-Ordering을 설명한 것이 그림 4이다. 그림 4에서 대상재-Ordering 과정을 통해 미탐색 대상재들을 1,

3, 4, 2, 5, 6의 순서로 결정했으며, 현재의 탐색 단계에서 부분해와 미탐색 대상재인 2와 6과의 Look-ahead 관계를 검사해서 먼저 탐색되어야 할 대상재를 결정함을 보여준다.

우선 순위가 높은 미탐색 대상재를 선택하여, Look-ahead 휴리스틱과 일반적인 제약조건을 만족하는 지를 검사하고, 만족하면 부분해에 첨가하고, 만족하지 않으면 다음 우선 순위

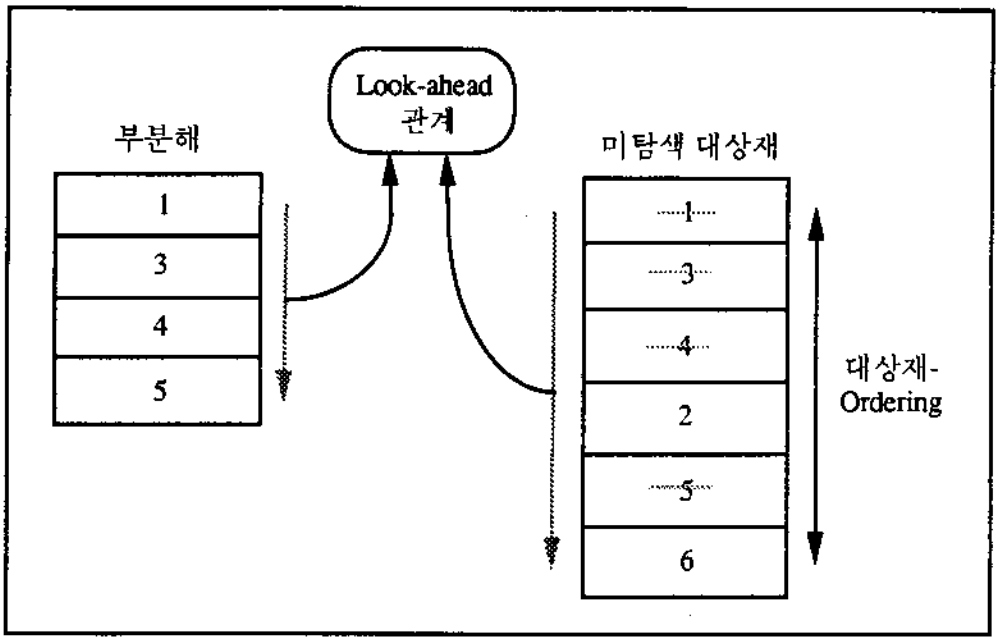

그림 4. Look-ahead관계와 대상재-Ordering

2.4.2 제약조건 만족 알고리듬

기본적인 Backtrack에 3.3.1절에서 논한 발견적 기법을 결합하여 모든 제약조건을 만족하는 가능해를 찾아주는 제약조건 만족 알고리듬을 개발하였다. 개발된 알고리듬은 먼저 탐색 전단계에서 대상재-Ordering과정을 거쳐 대상재의 탐색 순서를 결정한다. 탐색 단계는 기본적인 Backtrack 기법인 부분해로부터 새로운 대상재로의 전진과 후퇴 과정과 함께 발견적 기법인 Look-ahead 검사 과정을 추가함으로써 이루어진다. 그림 5는 탐색 단계를 설명한다. 전진은 입력 대상재 중에서 의 미탐색 대상재를 선택한다. 더 이상 선택할 대상재가 없으면, 부분해에서 가장 최근에 첨가된 대상재를 제거하여 미탐색 대상재로 돌려보낸다. 이것을 후퇴라 한다.

2.5 최적화 기법

목적식을 만족하는 최적해를 찾고자 하는 탐색 기법에 대해 많은 연구가 있었다[8, 10, 11]. 대표적인 것으로는 Hill-climbing search, Simulated annealing, Tabu search 등이 있다. 이들 세가지 탐색 기법은 모두 최적해를 찾을 가능성은 존재하나, 실제로는 최적해에 가

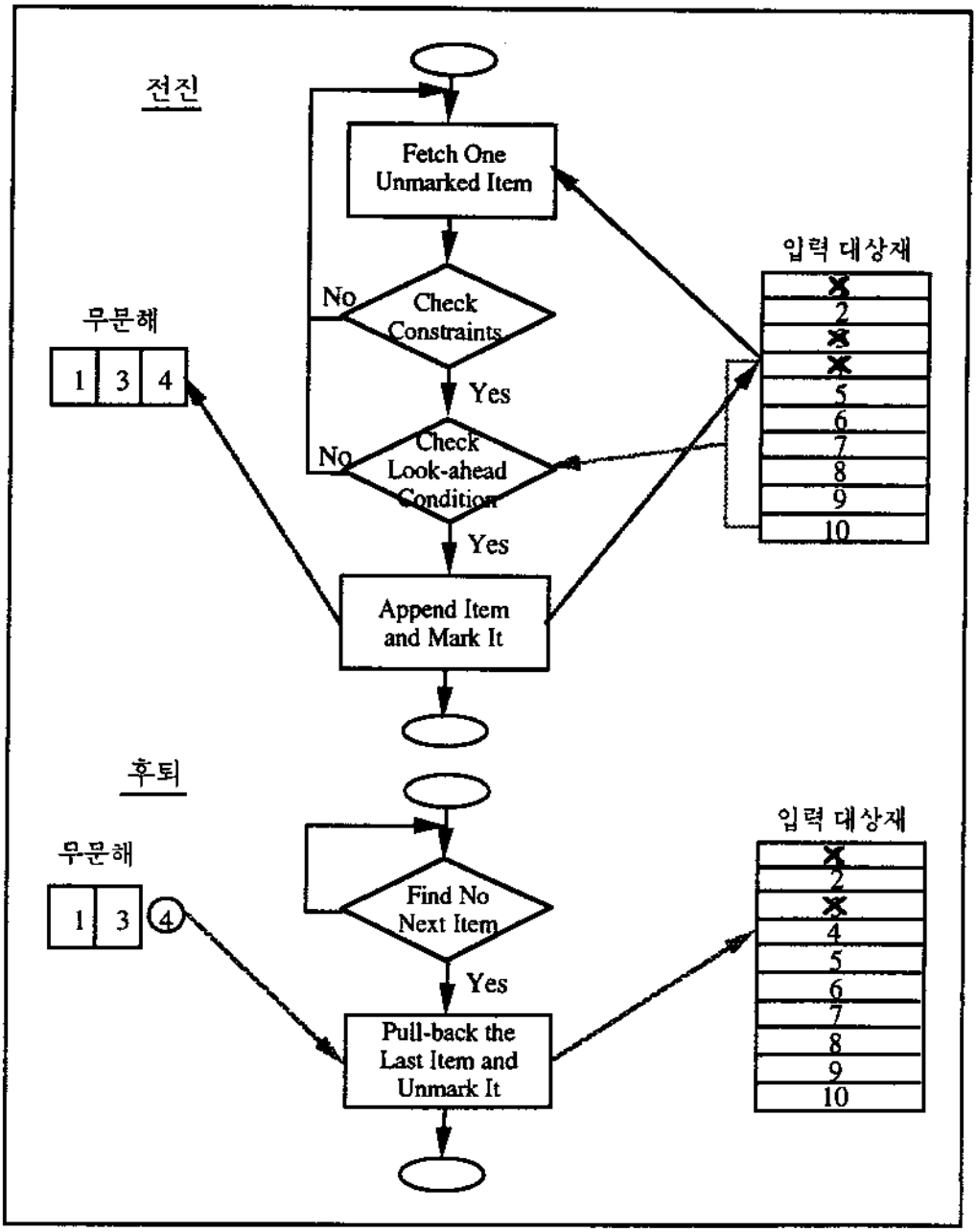

그림 5. Backtrack with Look-ahead 기법

까운 우수해만을 도출한다는 단점이 있다. 탐색 시간면에서는 일반적으로 Hill-climbing이 가장 빨리 해를 찾으며, Tabu search가 가장 늦게 해를 찾는다. 해의 우수성에서는 Simulated annealing과 Tabu search가 Hill-climbing보다 나은 경향이 있으며, 일반적으로 Tabu search는 Simulated annealing보다 비교적 해의 우수성이 안정적이라 할 수 있다. 따라서 본 연구에서는 최적해 탐색 기법으로 Tabu search를 이용한다.

3. 일정계획 언어 설계 및 프로그램 생성을 위한 방법론

3.1 알고리듬의 모듈화(Modularization)

본 연구에서 제시하는 제약조건 만족 패러다임에 근거하여 시스템을 개발하기 위해서는 알고리듬과 문제 자체가 분리되어야 한다. 알고리듬의 모듈화 과정은 이와 같이 알고리듬과 문제 자체를 분리하는 과정이다. 이를 구현하기 위하여 먼저 2절에서 제시한 알고리듬으로부터 문제기반 지식의 각 요소들을 도출하여야 한다. 그리고, 알고리듬을 문제기반 지식과의 인터페이스(Interface)를 고려하여 하나의 모듈로 작성하고, 일정계획 문제기반 지식의 각 요소들을 정해진 인터페이스의 형식에 맞추어 모듈로 구성함으로써 이루어 진다.

그림 6은 제약조건 만족 알고리듬의 모듈화 예를 보여준다. 하나의 완전한 제약조건 만족 알고리듬이 구성되기 위해서는 탐색, Data manipulation, I/O, Terminal condition, Look-ahead, Constraint 등의 모듈이 필요하다. 이러한 여러 모듈 중에서 일정계획 문제에 의존적인 부분인 Terminal condition, Look-ahead, Constraint 등의 모듈을 분리하고, 서로간의 인터페이스의 형식을 맞추어 준다. 그러면 문제가 바뀐다 하더라도 문제기반 지식인 Terminal condition, Look-ahead, Constraint 등의 모듈만을 정해진 인터페이스에 맞추어 새로이 구성하여 이미 존재하는 알고리듬 모듈과 결합하면, 하나의 완전한 제약조건 만족 프로그램이 구성된다. 그림 7은 이러한 알고리듬의 모듈화 과정을 거쳐 도출된 문제기반 지식 요소들을 보여준다. 제약조건 만족 알고리듬으로부터 개별적 제약조건, 집합적 제약조건, Look-ahead 제약조건 등이 도출되었으며, 최적화 알고리듬으로부터 개별적 제약조건, 집합적 제약조건, 목적식 등이 도출되었다.

3.2 언어 설계

알고리듬의 모듈화를 통해 도출된 문제기반 지식을 이해하기 쉽고 표현력이 높으며 모호성이 없는 문법을 설계하고자 한다. 문법을 설계하는 방법론으로 여러 가지가 개발되었는데, 본 연구에서는 표현력이 높고 프로그래밍 언어를 정의하는데 가장 널리 사용되고 있는 Backus-Naur Form(BNF)을 이용하였다. 본 연구에서 이용하고자 하는 문법 설계 및 분석 도구인 Yacc의 표현 방법이 BNF의 표현 방법을 따르기 때문에 BNF로 설계된 문법은 Yacc로 그대로 이용될 수 있다는 장점이 있다.

다음의 예는 실제로 언어의 문법을 설계하는 예를 보여준다. 이 예는 일정계획 문제 기반 지식 중에서 개별적 제약조건 내 한 문장(Statement)의 문법을 표현하는데, 하나의 문

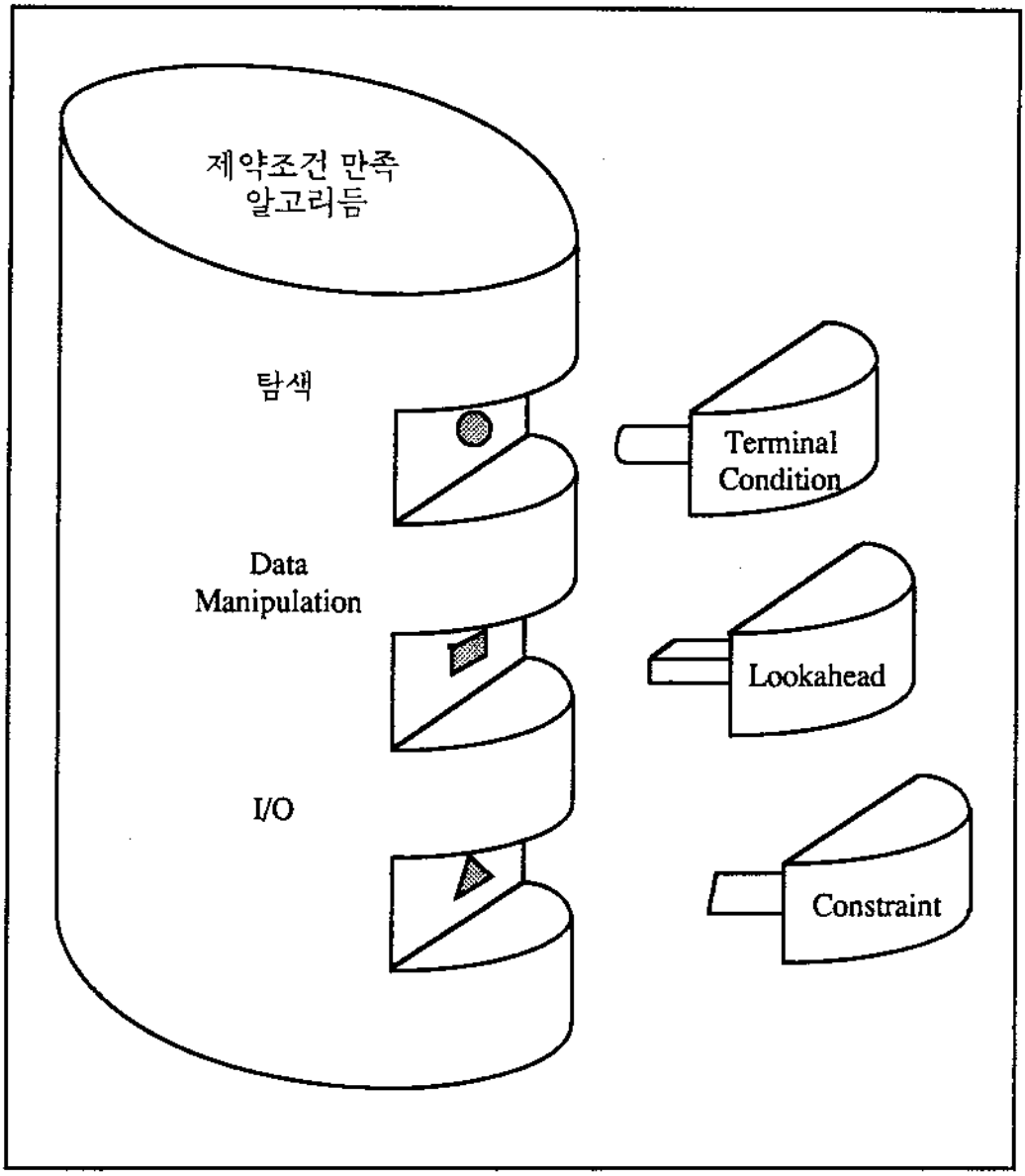

그림 6. 제약조건 만족 알고리듬의 모듈화 예

장은 조건 분기문(if then else statement)이나 비교문(comparative statement)으로 이루어지며, 괄호로 닫혀질 수도 있으며, AND나 OR로 복합문을 이룰 수 있다.

예) ⟨stmt⟩ : = (⟨stmt⟩)
| ⟨if—stmt⟩
| ⟨comp—stmt⟩
| ⟨stmt⟩ and ⟨stmt⟩

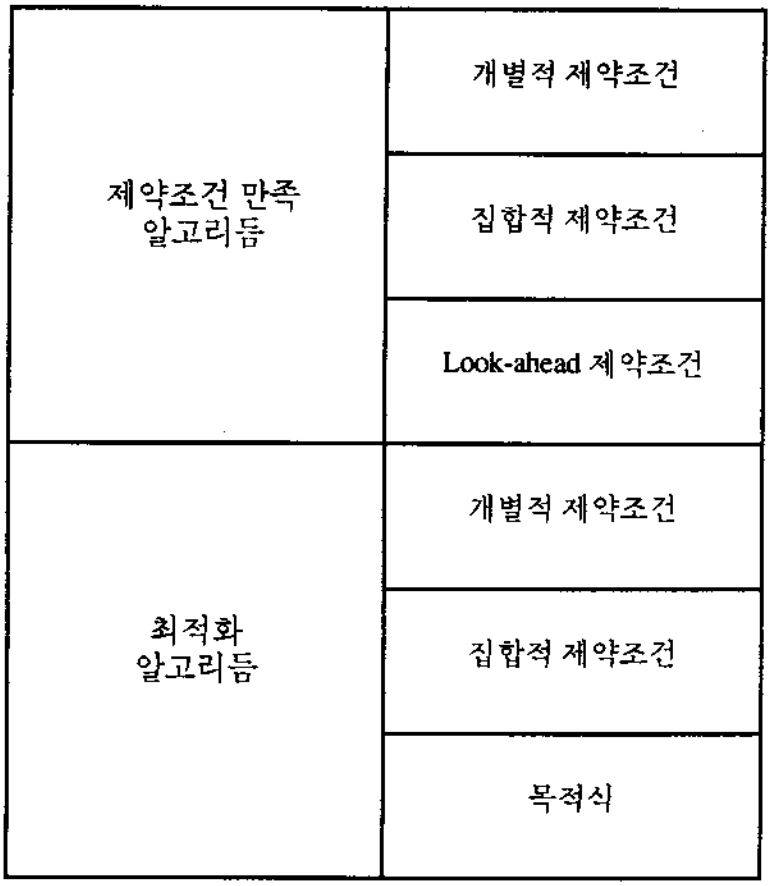

그림 7. 문제기반 지식 요소

| 〈stmt〉 or 〈stmt〉

;

위와 같은 방법론으로 표현력이 높고 문법의 모호성이 없는 언어를 설계할 수 있다.

3.3 파싱(Parsing) 및 프로그램 생성

일정계획 프로그램 생성기를 구축하기 위해서는 먼저 사용자가 입력한 제약조건들을 해석하여, 모든 정보를 다음 단계를 위한 파스 트리(Parse tree)와 자료 디렉토리(Data directory)를 구성하여야 한다. 이와 같은 작업을 파싱(Parsing)이라 하는데, UNIX의 여러 기본 도구 중 Lex와 Yacc를 이용하여 파서(Parser)를 개발할 수 있다. Lex는 문법의 토큰(Token) 분석기이고, Yacc는 생산 규칙(Production rule) 분석기이다. 먼저 토큰과 생산 규칙, 그리고 파스 트리 또는 자료 디렉토리의 생성작업을 정의하고, 이를 Lex와 Yacc를 이용해 컴파일(Compile)하면, 그 문법에 대한 파서가 생성되며, 이 파서를 통해 다음 단계를 위한 파스 트리와 자료 디렉토리를 구성할 수 있다.

일정계획 프로그램 생성의 두 번째 과정으로서, 파서를 통해 생성된 파스 트리와 자료 디렉토리를 입력받아 목적 언어의 프로그램을 생성하여야 한다. 본 연구에서는 목적 언어로서 현재 가장 널리 사용되고 있고 가장 호환성이 높은 언어인 C 언어로 정하였다. 따라서 3.1절의 알고리듬의 모듈화에 의해 정해진 알고리듬 자체와 문제기반 지식간의 인터페이스의 포맷에 따라 문제기반 지식, 즉 여러 제약조건들을 C 언어의 함수로 변환하게 된다.

파싱 과정과 프로그램 생성 과정을 예를 들어 설명하면 다음과 같다. 사용자로부터 다음과 같은 제약조건을 입력받았다고 하면,

```
if jejil[i] = 'a' then
jejil[i+1] = 'a' or jejil[i+1] = 'b' or jejil
[i+1] = 'c'
else if jejil[i] = 'b' then
jejil[i+1] = 'b' or jejil[i+1] = 'c'
else if jejil[i] = 'c' then
jejil[i+1] = 'a';
```

이 제약조건은 시퀀스의 i번째 대상재의 속성(jejil)이 'a'이면 다음 대상재의 속성은 'a' 또는 'b' 또는 'c'이어야 하고, i번째 대상재의 속성이 'b'이면 다음 대상재의 속성은 'b' 또는 'c'이어야 하고, i번째 대상재의 속성이 'c'이면 다음 대상재의 속성은 'a'이어야 한다는 것이다.

위의 제약조건을 파서에 입력하면 파스 트리가 생성된다. 그림 8은 생성된 파스 트리의 일부이다. 생성된 파스 트리의 각 노드들은 제약조건을 구성하는 기본 요소들을 생산규칙에 따라 탑-다운(Top-down) 방식으로 저장한다. 그림 8의 파스 트리를 프로그램 생성기에 입력하면 제약조건에 대한 C 코드가 생성된다. 아래의 코드는 위의 제약조건에 대해 프로그램 생성기를 통해 생성된 C 코드이다.

```
int const0(PARTIAL, LISTS[], next)
{
 if  ((LISTS[PARTIAL→seq[PARTIAL→
 no-1].jejil ==‘a’)) {
  {
   {
    if (LISTS[next].jejil == ‘a’)
       return 1;
    if (LISTS[next].jejil == ‘b’)
       return 1;
   }
   if (LISTS[next].jejil == ‘c’)
     return 1;
 }
  } else {
   if (LISTS[PARTIAL→seq[PARTIAL→
   no-1].jejil == ‘b’)) {
    {
    if (LISTS[next].jejil == ‘b’)
       return 1;
    if (LISTS[next].jejil == ‘c’)
       return 1;
    }
  } else {
   if (LISTS[next].jejil == ‘a’)
     return 1;
  }
}
```

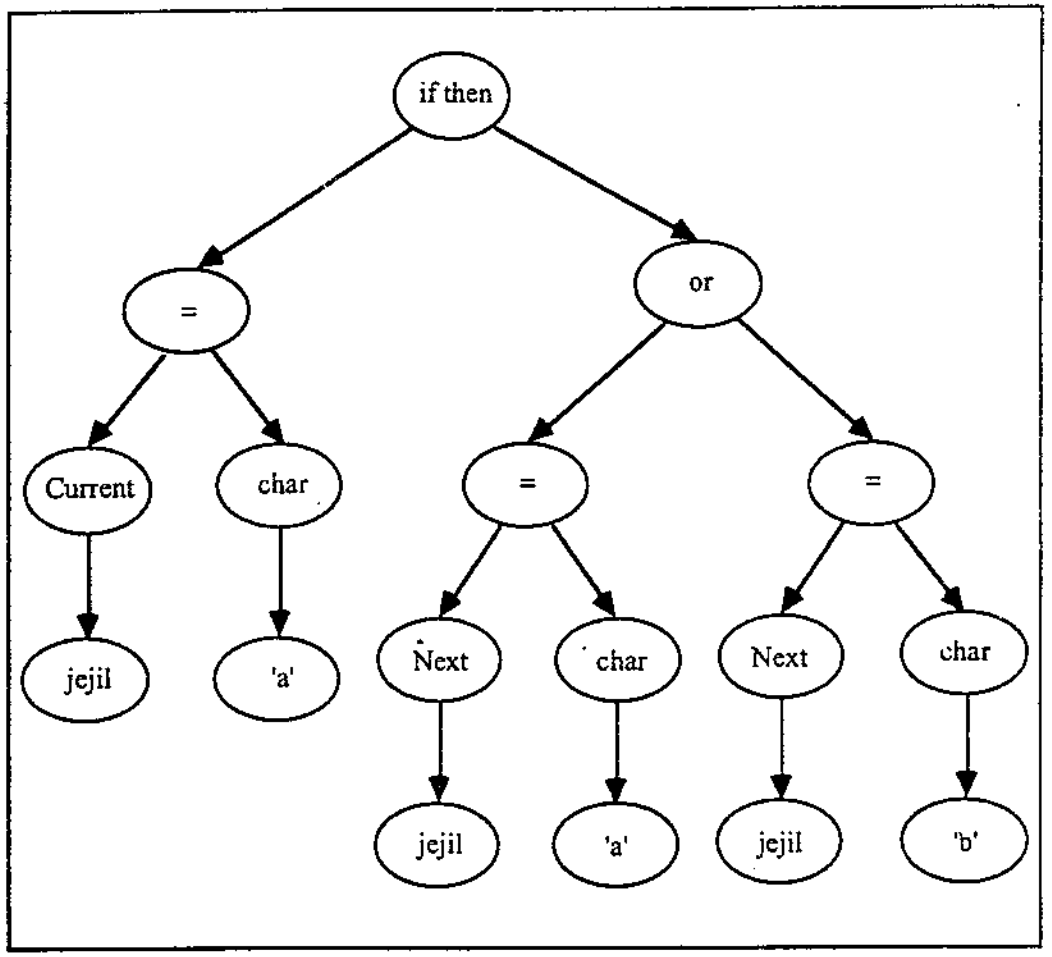

그림 8. 파스트리의 구성

```
  return 0;
}
```

4. 시스템 구현 및 적용

4.1 시스템 개요

본 연구에서는 Backus-Naur Form을 바탕으로 한 대의 기계에 대상재의 순서를 결정하는 일정계획 문제를 표현할 수 있는 일정계획 언어를 설계하였으며, UNIX의 프로그래밍 도구인 Lex와 Yacc를 이용하여 파서(Parser)와 코드 생성기를 개발하였다. 그림 9는 전체 일정계획 시스템의 개요이다. 사용자는 일정계획 문제를 표현하는데, 속성 정의, 개별적 제약조건, Look-ahead 휴리스틱, 집합적 제약조건, 목적식, 전처리 조건 등을 기술하게 된다. 그러면 일정계획 프로그램 생성기가 입력된 일정계획 문제를 해석하여 일정계획 프로그램을 생성한다. 이때 일정계획 알고리듬의 모듈은 계속되어 재사용되고, 일

정계획 문제 지식만이 새로이 코드로 생성되어 두 모듈이 결합되어 하나의 완전한 일정계획 프로그램을 이루는 것이다.

제약조건을 입력하기 위한 text editor를 제공하며 스케줄 전후의 대상재에 대한 정보도 확인할 수 있다. 이러한 기능들은 마우스를

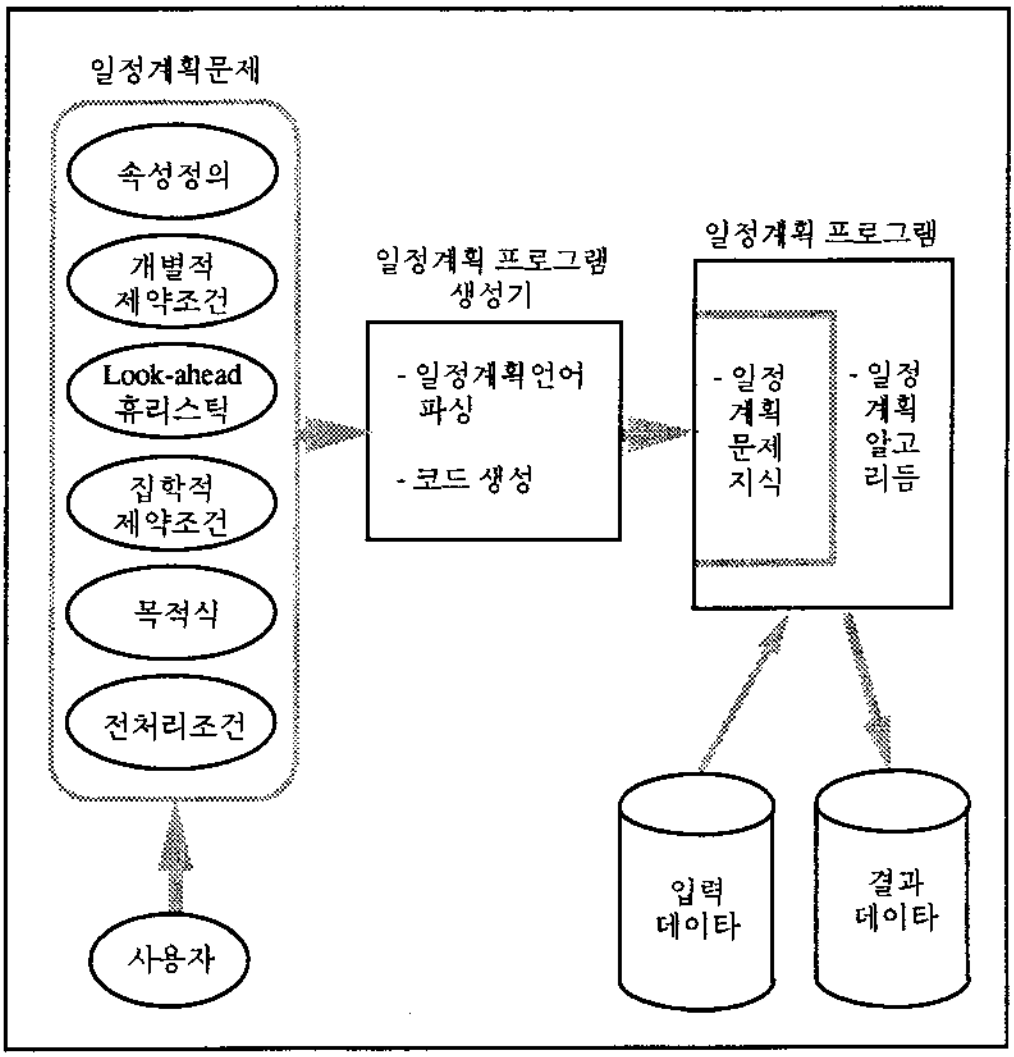

그림 9. 일정계획 시스템 개요

4.2 시각적 사용자 환경

본 시스템은 사용자의 편의를 위하여 X/Motif 환경에서 시각적 사용자 환경을 구현하였다.

이용하여 해당 버튼을 선택함으로써 호출된다.

4.3 일정계획 언어 구성요소

4.3.1 속성 정의

대상재의 속성을 정의하고, 속성의 데이타 타입과 입력화일과의 매핑을 정의한다. 입력화일과의 매핑은 그 속성의 데이타가 존재하는 시작 칼럼 수와 끝 칼럼 수로 표현된다. 속성 정의의 포맷은 다음과 같다.

type {
속성이름:데이타타입 (시작칼럼 - 끝칼럼)
...
}

가능한 데이타 타입은 정수형(integer), 실수형(float), 문자형(char), 문자열형(string)이다.

실수형 데이타 타입일 경우 추가적으로 한 가지 포맷을 더 지원하는데, 그것은 데이타 화일에 소수점이 없는 경우이다. 그것의 포맷은 다음과 같다.

속성이름:float (정수시작칼럼 - 정수끝칼럼 . 소수시작칼럼 - 소수끝칼럼)

4.3.2 개별적 제약조건

개별 대상재의 속성간의 제약조건을 표현한다. 가능한 개별적 제약조건의 표현과 그 포맷은 다음과 같다.

- 비교 제약조건(comparative constraint)
  formula boolean-operator formula
- 분기 제약조건(if-then-else constraint)
  if condition then constraint [else constraint]

또한 각 제약조건은 AND와 OR로 연결됨으로써, 복잡한 형태의 제약조건을 표현할 수 있도록 한다. 여기서, formula는 속성과 수치의 사칙연산으로 이루어 지며, boolean-operator는 =, ≤, ≥, 〈, 〉 등의 비교 연산자이다. Condition은 비교 제약조건과 똑같은 문법으로 표현되나, 이것은 제약조건이 아니라 단지 조건을 나타낸다.

4.3.3 집합적 제약조건

대상재 속성의 집합에 대한 제약조건을 표현한다. 가능한 집합적 제약조건의 표현과 그 포맷은 다음과 같다.

- 예약어(reserved word)
  SCH : 일정계획된 대상재의 집합
  BT : 일정계획할 모든 입력 대상재의 집합
- 합계 함수
  sum(SET | formula [where condition])
- 수량 함수
  count (SET [where condition])
- 집합적 비교 제약조건 : formula에 집합적 함수를 사용하는 비교 제약조건
- 집합적 분기 제약조건 : formula에 집합적 함수를 사용하는 분기 제약조건

4.3.4 목적식

최대화 또는 최소화할 목적식을 나타낸다. 목적식의 포맷은 다음과 같다.

min(or max) formula;

4.3.5 Look-ahead 휴리스틱

보다 효율적인 탐색을 위한Look-ahead 휴리스틱을 표현한다. 가능한 Look-ahead 휴리스틱의 표현과 그 포맷은 다음과 같다.

- 예약어(reserved word)
  SCH : 일정계획된 대상재의 집합
  BT : 일정계획할 모든 입력 대상재의 집합
  CS : 탐색 중 일정계획된 대상재의 집합

US : 탐색 중 아직 일정계획되지 않은 모든 대상재의 집합

LAST : CS의 마지막 대상재

- 합계 함수

  sum (SET | formula [where condition])

- 수량 함수

  count (SET [where condition])

- 휴리스틱 비교 제약조건 : condition에 CS, US, LAST 등을 사용하는 비교 제약조건

- 휴리스틱 분기 제약조건 : condition에 CS, US, LAST 등을 사용하는 분기 제약조건

4.3.6 전처리 조건

보다 효율적인 탐색을 위한 전처리 조건을 표현한다. 이것은 단순 소팅으로 대상재간의 탐색 우선 순위를 결정하는 것이다. 전처리

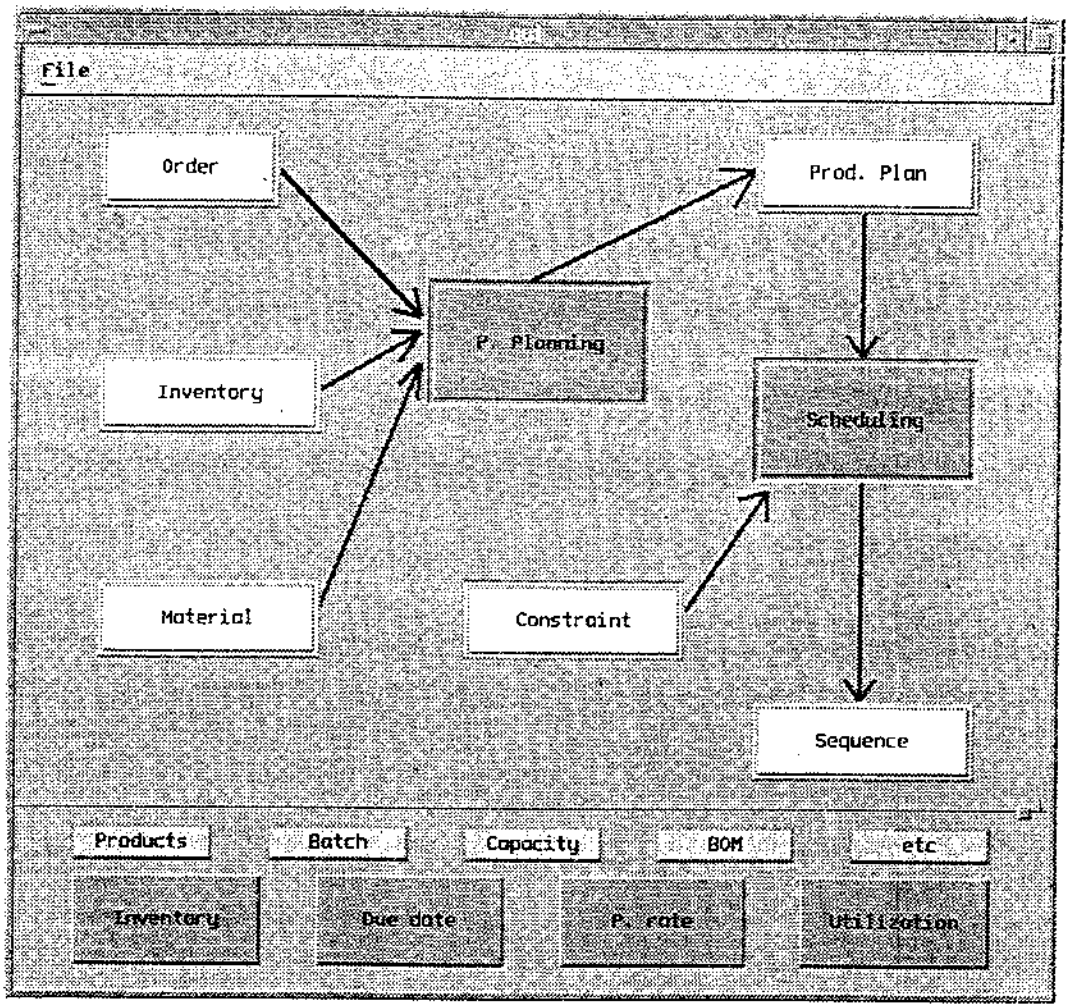

그림 10. 일정계획 시스템의 초기화면

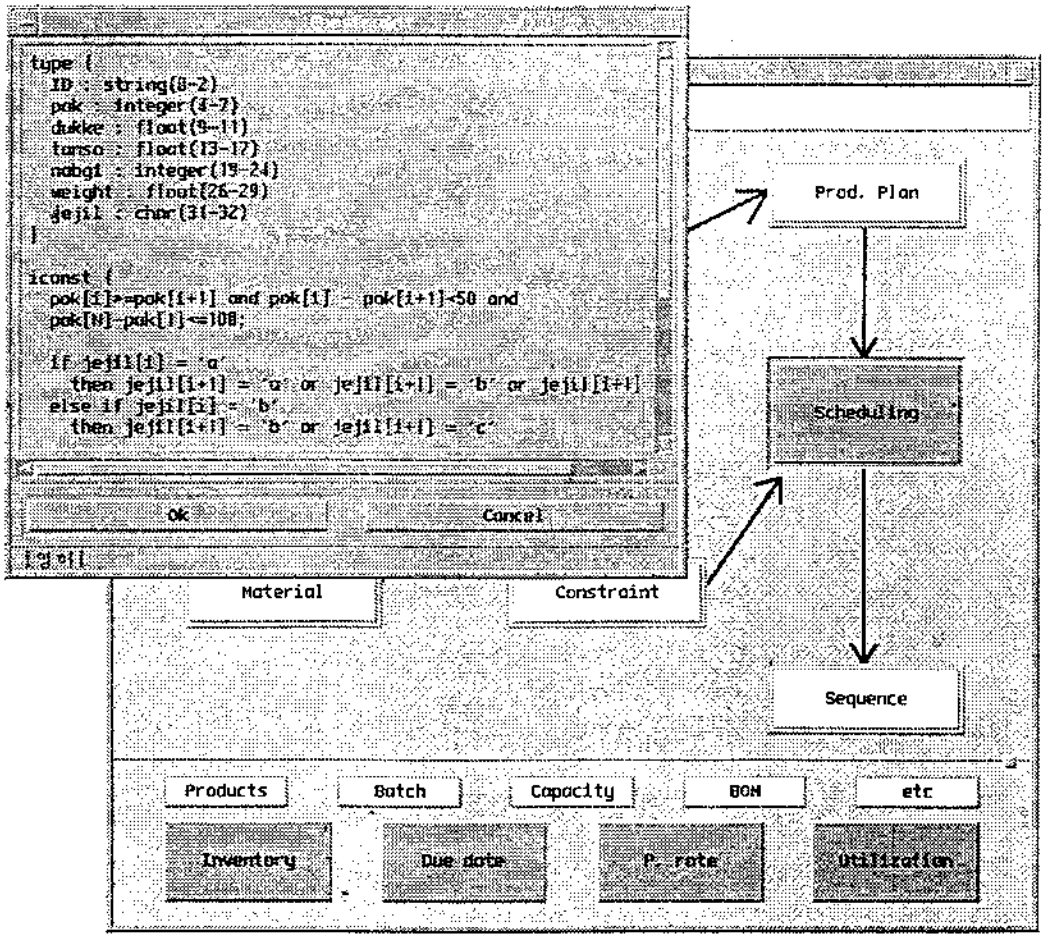

그림 11. 제약조건 입력

조건의 표현과 포맷은 다음과 같다.

ASCEND(or DESCEND) 속성이름;

여러 개의 전처리 조건이 나올 경우는 앞의 조건이 우선순위가 있으며, 뒤의 조건은 앞의 조건을 똑같이 만족시킬 경우 적용하게 된다.

4.4 적용 예

개발한 시스템을 실제 제철산업의 냉연공정에서 대상재인 코일의 작업순서를 결정하는 문제에 적용해 보았다. 제약조건에 사용되는 대상재의 속성은 폭, 두께, 탄소 함유량, 납기, 중량, 재질 등이 있다. 다음은 개별적 제약조건을 설명한다.

(i) 작업단위내 대상재의 폭은 내림차순이어야 하고, 전후 대상재간의 폭 편차는 50보다 작아야 한다. 또한 작업단위 마지막 대상재와 첫번째 대상재의 폭 편차는 100보다 작아야 한다.

(ii) 전후 대상재간의 두께 편차는 앞 대상

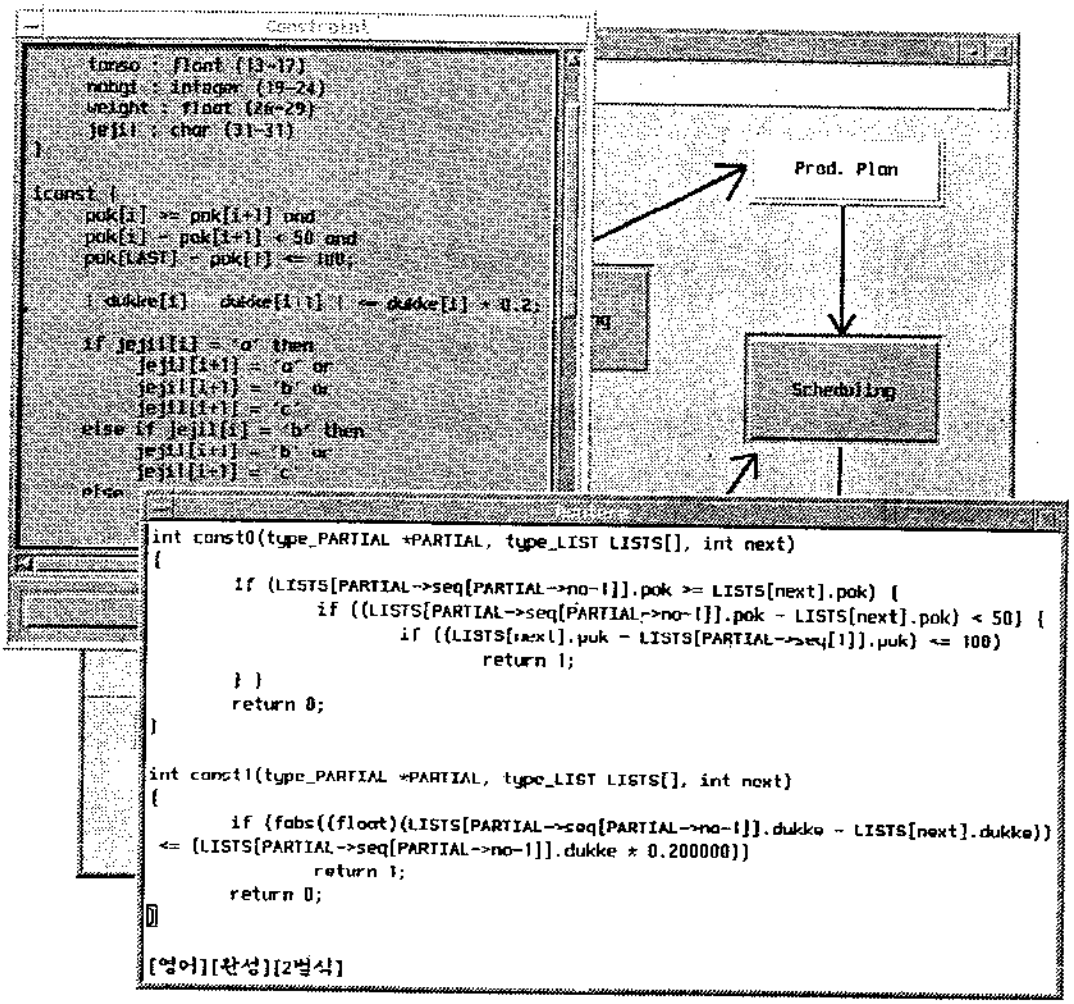

그림 11. 제약조건 입력

재의 두께의 20%이하이어야 한다.

(iii) 앞 대상재의 재질이 a이면, 다음 대상재의 재질은 a, 또는 b, 또는 c이어야 하고, 앞 대상재의 재질이 b이면, 다음 대상재의 재질은 b, 또는 c이어야 하고, 앞 대상재의 재질이 c이면, 다음 대상재의 재질은 a이어야 한다.

(iv) 전후 대상재의 폭이 같다면, 두께가 작은 대상재가 우선한다.

(v) 전후 대상재의 폭과 두께가 같다면, 탄소 함유량이 큰 대상재가 우선한다.

(vi) 전후 대상재의 폭과 두께와 탄소 함유량이 같다면, 납기가 앞서는 대상재가 우선한다.

다음은 집합적 제약조건을 설명한다.

(i) 작업단위내 모든 대상재 중량의 합이 180이상이어야 한다.

그림 10은 일정계획시스템의 초기화면이

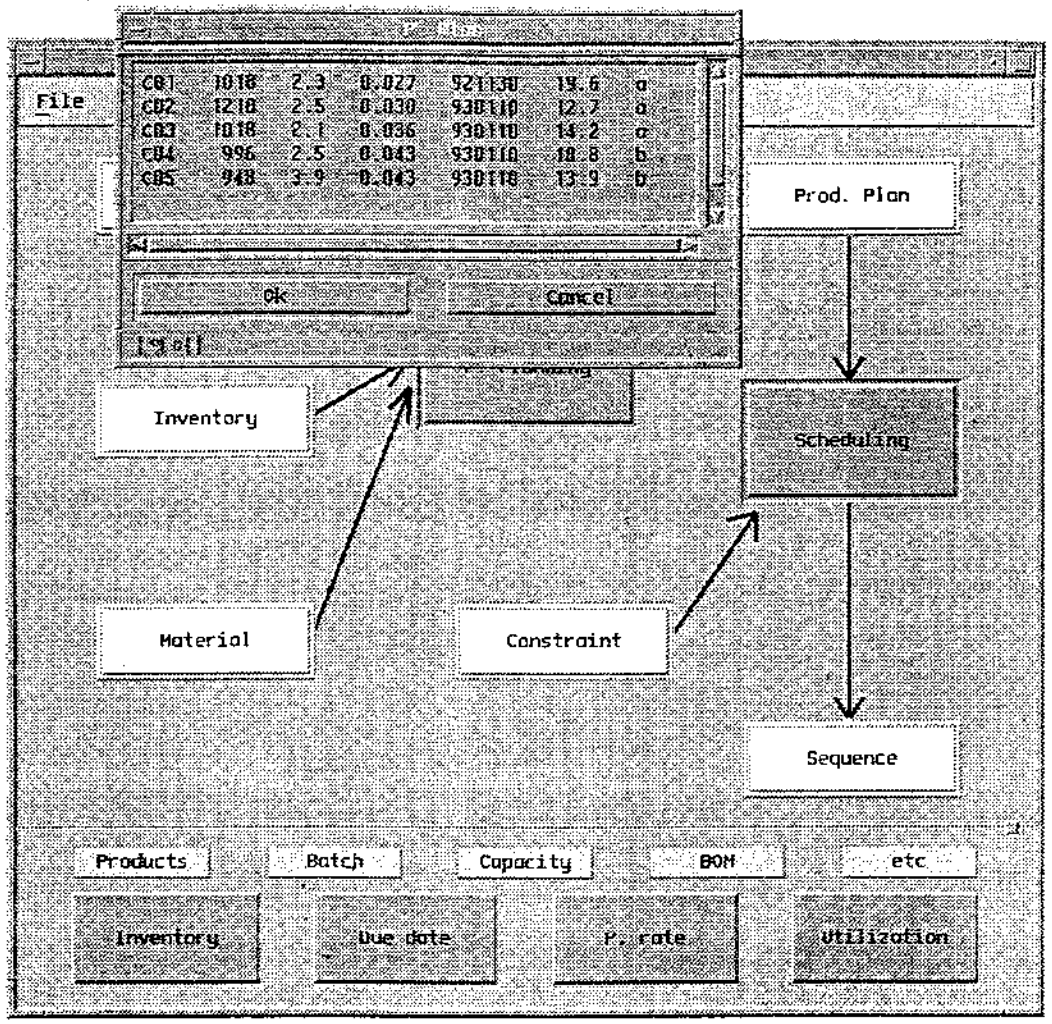

그림 12. 입력 데이타

며. 위의 예를 개발한 일정계획 언어로 표현한 것이 그림 11이다. 위의 예를 일정계획 프로그램 생성기에 입력하면, 하나의 완전한 일정계획 프로그램이 생성된다(그림 11). 이 일정계획 프로그램에 데이타 화일을 입력하면(그림 12), 일정계획 언어로 표현된 제약조건을 만족시키는 결과 데이타 화일이 출력된다(그림 13).

5. 결론 및 추후 연구방향

본 연구에서는 제약조건 만족 패러다임에 근거하여 기계가 한 대이며 다양하고 복잡한 공정제약조건과 목적식을 가지는 일정계획 문제를 해결할 수 있는 일정계획 시스템을 구축하기 위하여 모델링 능력이 뛰어난 일정계획 언어와 우수한 해를 도출할 수 있는 일

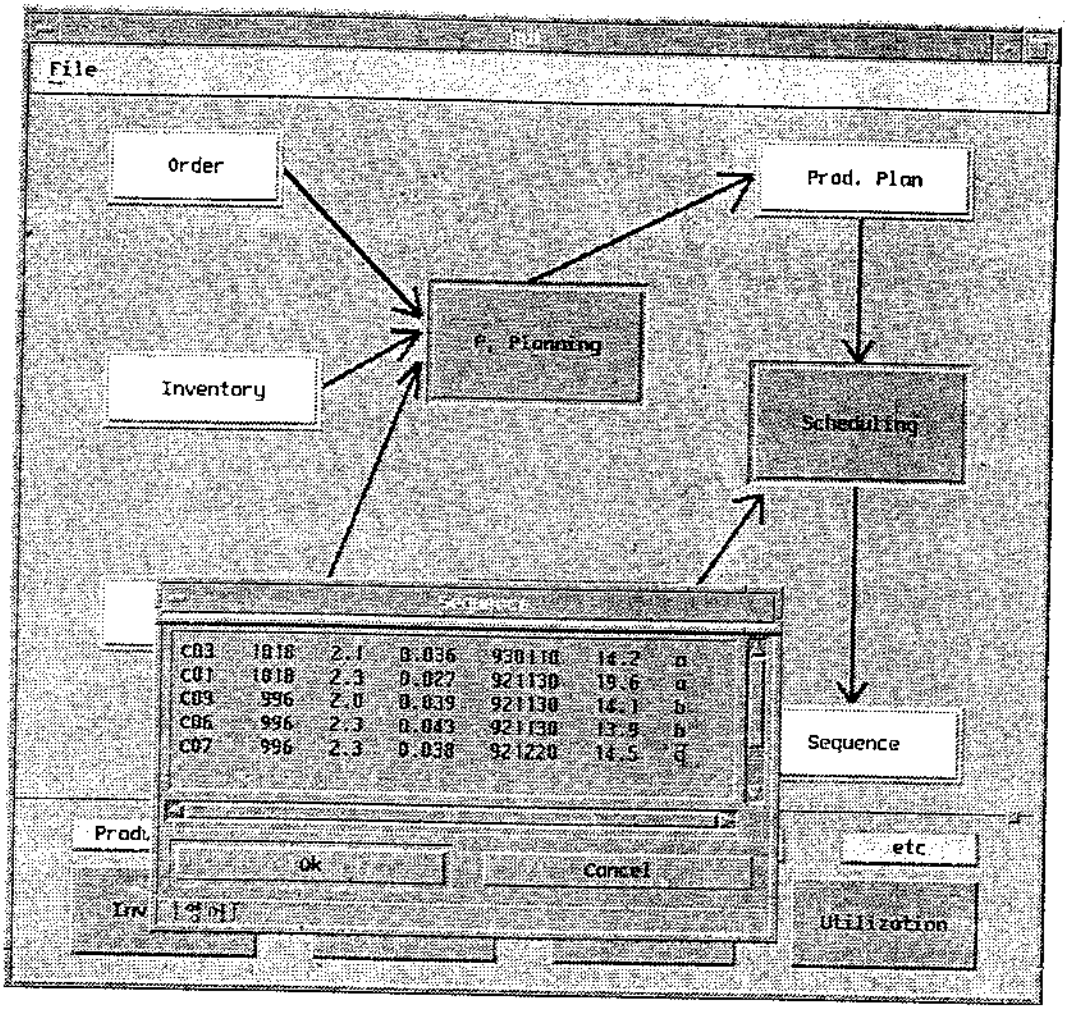

그림 13. 제약조건을 만족하는 데이타

정계획 프로그램을 자동으로 만들어 주는 일정계획 프로그램 생성기를 개발하였다. 일정계획 언어의 요소는 속성정의, 개별적 제약조건, 집합적 제약조건, 목적식, Look-ahead 휴리스틱, 전처리 조건 등이며, 사용자는 일정계획 문제를 위의 요소로 선언적으로 정의하면, 일정계획프로그램 생성기를 통해 하나의 완전한 일정계획 프로그램이 생성되게 된다. 따라서 사용자는 구체적인 알고리듬을 구축하거나 이해하지 못해도 일정계획 문제만을 기술하면, 자동적으로 그 문제를 위한 일정계획 프로그램을 만들 수 있다는 장점이 있다. 또한 대부분의 기존 제약조건 만족 시스템은 목적식이 존재하지 않는, 오직 제약조건만을 만족시키는 가능해를 도출하는 반면, 본 시스템은 목적식이 존재하는 문제도 최적해에 근사한 우수한 해를 도출할 수 있으며, 수리 계획과 비교했을 때는 표현력의

우수함으로 인해 제약조건의 의미를 전문가가 아니더라도 이해할 정도로 쉽게 전달할 수 있다는 장점이 있다. 그리고 본 일정계획 시스템을 실제로 제철산업의 냉연공정 일정계획 문제에 적용해보았다.

추후 연구방향으로는 첫째, 본 일정계획 언어의 요소 중 좀 더 효율적인 탐색을 위한 Look-ahead 휴리스틱과 전처리 조건을 사용자가 입력한 제반 제약조건들로부터 자동적으로 도출하는 규칙에 대한 연구가 필요하다. Look-ahead 휴리스틱과 전처리조건은 일반적으로 개별적 제약조건과 집합적 제약조건 간의 관계로 이루어 짐을 알 수 있었는데, 이들간의 관계를 정립함으로써 자동적인 도출이 가능하리라 본다. 둘째, 현재의 알고리듬보다 좀더 빠르고 좀더 우수한 해를 도출할 수 있는 알고리듬에 대한 연구가 필요하다. 셋째, 본 연구는 하나의 공정에 대한 일정계획 문제를 고려하였는데, 실제 현장에는 연결된 다수의 공정상에서의 대상재의 일정계획문제를 종종 접하게 되므로, 이에 대한 연구가 필요하다.

참 고 문 헌

[1] 고재석, "다단계 일관제철생산시스템에서의 생산스케줄러 개발에 관한 연구," 석사학위논문, 포항공과대학교, 1991.

[2] 김종기, "Grouping과 Sequencing이 관련된 일정계획 문제를 위한 Constraint Satisfaction Problem Technique의 응용," 석사학위논문, 포항공과대학교 산업공학과, 1992.

[3] 이유근, 전치혁, 장수영, 최인준, "연속소둔공정의 작업단위편성을 위한 발견적 기법," 대한산업공학회/한국경영과학회 '94 춘계공동학술대회 논문집, 1994. 4, pp 270-287.

[4] 전치혁, 장수영, 최인준, 이승만, 강상엽, "냉연공정에서의 작업단위 편성," 산업공학, 6, 1993, pp 117-131.

[5] 정남기, "분산관리 시스템을 위한 동적 스케줄링의 연구", 대한산업공학회지, 제21권, 제2호, 1995, pp 207-216.

[6] Ambler, A.L., Burnett, M.M., and Zimmerman, B.A., "Operational Versus Definitional: A Perspective on Programming Paradigms," IEEE Computer, Sepember, 1992, pp 28 - 43.

[7] Borning, A., "ThingLab-A Constraint-Oriented Simulation Laboratory," Xerox PARC technical report SSL-79-3, Palo Alto, Calif., 1979.

[8] Dechter, R. and Pearl, J., "Network-Based Heuristics for Constraint-Satisfaction Problems," Artificial Intelligence, 1988, pp 1-38.

[9] Fox, M. S., "Constraint-Directed Search: A Case Study of Job Shop Scheduling," San Mateo, Calif.: Morgan Kaufmann, 1987.

[10] Friden, C., Hertz, A., and D. de Werra, "Tabaris: An Exact Algorithm Based On Tabu Search for Finding a Maximum Independent Set ing a Graph," Computers Opns Res., Vol17, No.5, 1990, pp 437-445.

[11] Glover, F., "Tabu Search,"Part I. ORSA

J. Comput. 1, 1989, pp 190-206.

[12] Haralick, R.M. and Elliot, G.L., "Increasing Tree Search Efficiency for Constraint Satisfaction Problems," Artificial Intelligence, 14, 1980, pp 263-313.

[13] Konopasek, M. and Jayaraman, S., "The TK!Solver Book,"Osborn/McGraw-Hill, Berkeley, Calif., 1984.

[14] Leler, W., "Constraint Programming Languages: Their Specification and Generation," Addison-Wesley, 1988.

[15] Nadel, B.A., "Constraint satisfaction algorithms," 1989.

[16] Nelson, G., "JUNO, a Constraint-Based Graphics System,"SIGGRAPH Computer Graphics, Vol.19, No.3, 1985, pp.235-243.

[17] Posser, P., "A Reactive Scheduling Agent," In Proceedings of the 11th International Joint Conference on Artificial Intelligence, 1989, pp 1004-1009.

[18] Rit, J.F., "Propagating Temporal Constraints for Scheduling," In Proceedings of the 5th National Conference on Artificial Intelligence, 1986, pp 383-386.

[19] Sutherland , I., "SKETCHPAD: A Man-Machine Graphical Communication System," IFIPS Proceedings of the Spring Joint Computer Conference, 1963.

95년 12월 최초 접수, 96년 4월 최종 수정