스케줄링 문제 해결을 위한 지식 기반 기법과 제약 만족 기법의 비교 연구

양종윤* · 조근식**

Knowledge-Based vs. Constraints-Based Scheduling : A Case Study of Gate Allocation Problem

Jong-Yoon Yang* · Geun-Sik Jo**

요 약

다양한 산업영역에서 수행되는 스케줄링 문제를 해결하기 위하여 AI분야에서는 지식을 기반으로 한 방법이 적용되어 왔다. 그러나 최근 CSP(Constraints Satisfaction Problem) 개념이 소개되어 그 효율성이 입증되고 있으며 스케줄링 응용 문제들이 CSP로 정형화되면서부터 지식 기반 기법과 제약 만족 기법의 적용이 공존하고 있다. 지식을 기반으로 한 방법은 도메인 전문가(domain expert)의 지식을 습득하여 시스템에 반영하는데 이러한 지식은 문제해결에 중심적 역할을 수행하게 된다. 제약조건을 기반으로 한 방법은 문제를 CSP로 정형화 한 후 제약조건에 따른 일관성 유지 및 휴리스틱 탐색 방법을 적용하여 문제의 해를 효율적으로 구하게 된다. 본 연구에서는 스케줄링 문제를 해결하기 위한 지식기반 기법과 제약만족 기법을 주기장 할당 문제에 적용하여 실제 항공사의 운항 데이터를 바탕으로 실험하고 분석 및 비교를 통해 제약 만족 기법이 시스템의 유지 및 보수 측면에서 효율적이며 근사해가 아닌 최적해를 통한 문제 해결이 가능함을 보였다.

주제어: Constraints Satisfaction Problems, Knowledge-Based System, Rule-Based System

* 한국 전자통신 연구원
** 인하대학교 전자계산공학과 부교수

1. 서 론

지식 기반 시스템(Knowledge-Based System)은 오랫동안 AI 분야에서 많은 문제들을 해결하는데 사용되어져 왔다. 이러한 지식 기반의 시스템들은 일반적으로 지식 엔지니어(Knowledge Engineer)가 전문가 지식을 밝히기 위해 도메인 전문가와의 면담이나 협의의 과정을 거쳐 지식을 습득(Knowledge Aquisition)한다. 이렇게 습득된 지식은 시스템 내부적으로 규칙(Rule)의 형태로 표현되는 것이 일반적이다. 지식 습득과 지식 표현의 과정을 거쳐서 프로토타입 시스템이 구축되면 그 시스템은 전문가 수준의 성능을 가질 때까지 계속적으로 평가와 수정의 과정을 거치게 된다. 최종적으로는 전문가들이 시스템을 인정함으로서 그 시스템은 완성되는 것이다. 이러한 과정을 거쳐 완성되는 시스템은 그 개발 기간동안 전문가와의 면담과 회의의 과정을 필수적으로 거치게 되므로 비용뿐만 아니라 시간 또한 적지 않게 소비된다. 뿐만 아니라 전문가의 지식은 문제해결과정에 있어서 중추적 역할을 담당하기 때문에 지식 습득의 오류가 발생될 경우 문제 해결에 영향을 미칠 수도 있다.

그런데 최근 AI의 또 다른 분야로 제약 만족 기법에 대한 연구가 활발히 전개되고 있다. 이는 기존의 많은 AI문제들을 제약 만족 문제(CSP: Constraints Satisfaction Problems)로 정형화하여 일관성 검사 기법(Consistency Checking)이나 휴리스틱 탐색 기법(Heuristic Searching Technique)과 같이 제약조건을 위주로 한 기법들을 적용하여 문제를 해결하는 방법이다 [1][2]. 그 대표적인 응용 분야가 스케줄링(Scheduling) 문제, 할당(Allocation) 문제, 운송(Transport) 문제, 인력 배치(Crew Rotation) 문제 등이다 [3].

이 두 가지 방법 사이에는 응용 도메인에 따라 그 효용성이 부분적으로 비교되고 있다.

본 연구에서는 주기장 할당 문제라는 동일 도메인을 통하여 두 가지 방법에 의한 시스템을 모델링하고 비교해 봄으로써 문제 해결상의 중요한 영향을 미치는 몇 가지 측면의 성능을 평가해 보았다. 2장과 3장에서는 두 방법에 대한 기본적인 소개를 하고 4장을 통해 기존의 지식 기반 기법에 비교되는 제약 만족 기법을 통한 모델링과 구현을 소개할 것이다. 그리고 5장에는 구현된 결과를 가지고 두 방법을 비교하고 6장에서 결론을 맺도록 하겠다.

2. 지식 기반 기법

2.1. 도메인 지식의 이용과 표현

지식 기반 기법은 해당 도메인의 전문가 지식을 시스템에 반영해 문제를 해결하는 방법이다. 그러므로 시스템에 도메인 전문가의 지식을 표현하는 방법은 시스템 구축의 성패를 좌우하는 요소로서 가장 널리 쓰이는 것이 도메인 지식을 생성 규칙(Production Rule)의 집합으로 나타내는 것이다. 이는 때때로 규칙에서 발생하는 여러 객체를 정의하는 프레임(Frame) 시스템과 같이 사용되기도 한다.

박테리아에 감염된 환자에 대해 적절한 치료 방법을 추천하는 MYCIN 시스템은 필요한 데이터를 얻기 위해 내과의와 상호작용을 한다. MYCIN은 진단 지식의 대부분을 규칙(Rule)의 집합으로서 나타낸다. 각 규칙은 확실성 요소(certainty factor)를 가지는데 이는 어떤 규칙의 이전 사건에 의해 기술되는 증거가 주어진 규칙의 결과로 주어지는 결론을 뒷받침하는 정도를

나타낸다. 이 MYCIN을 지식 기반 시스템의 전형적 한 예로 본다.

XCON 이라고도 불리는 R1이라는 DEC VAX 시스템을 컨피규링하는 지식 기반 전문가 시스템의 경우는 MYCIN과는 다르게 확실성에 대한 수치요소를 갖지는 않는다. PROSPECTOR는 광산 답사에 대한 조언을 제공하는 지식 기반 시스템이다. PROSPECTOR에서는 각 규칙마다 두 개의 확신 평가치(confidence estimates)를 포함한다. 첫 번째 수치는 규칙의 조건 절에 기술된 증거의 존재가 규칙의 결과의 유효성을 제시하는 정도를 나타낸다. 두 번째 수치는 증거가 결과의 유효성에 필요한 정도나, 혹은 다른 말로 결과가 유효하지 않음을 나타내는 증거의 불충분 정도를 나타낸다. DESIGN ADVISER는 칩 디자인을 비평하는 프로그램이다. DESIGN ADVISER는 칩 디자이너에게 조언을 주고 디자이너는 이를 받아들일 수도 있고 거부할 수도 있다. 만약 조언이 거부되면 시스템은 회로의 모델을 보정하기 위해 정당화기반 진리 유지 시스템(Justification-based truth maintenance system)을 이용할 수 있다.

2.2. 지식을 이용한 추론

지식 기반 시스템은 지식을 기반으로 여러 추론 메커니즘을 이용해 문제를 해결한다. 규칙을 기반으로 한 이러한 프로그램들은 주로 전전 추론(forward chaining), 후진 추론(backward chaining), 또는 두 방법의 조합을 통한 추론 방법을 사용한다.

예를 들면 MYCIN은 어떤 유기체가 존재하는지를 발견하기 위해서는 후진 추론을 사용하고 그리고 나서는 유기체로부터 치료처방을 추론하기 위해서는 전진 추론을 사용한다. 반면 R1은 전진 추론을 사용한다. 전문가 시스템의 분야가 성숙되질수록 더 많은 시스템들이 개발중인 여러 가지의 다른 추론 방법을 이용한다. DESIGN ADVISER가 그러한 한 예이다. DESIGN ADVISER는 규칙을 이용하는 것에 추가로 정당화기반 진실유지 시스템(justification-based truth maintenance system)을 이용한다.

2.3. 지식 습득

지식을 기반으로 한 시스템은 전형적으로 지식 엔지니어(knowledge engineer)가 규칙으로 표현할 전문가 지식을 밝히기 위해 도메인 전문가와 인터뷰를 한다. 초기 시스템이 구축되면 그 시스템이 전문가 수준의 성능을 가질 때까지 계속적으로 개선되어야 한다. 이러한 과정은 비용도 많이 들며 시간소비도 크다. 그러므로 전문가 지식 베이스를 구축하는데 좀 더 자동화된 방법을 찾는 것이 필요하게 된다. 아직까지 완전 자동화 지식 습득 시스템이 존재하지는 않지만 전문가 지식을 좀더 효과적으로 끌어내기 위해 도메인 전문가와 상호 작용하는 프로그램들이 많이 있다. 이러한 프로그램들은 지식을 집어넣고 지식 베이스의 일관성(Consistency)을 유지하며 지식 베이스의 완전성(Completeness)을 보장하는 기능을 지원한다. 이러한 지식 기반의 시스템은 하나의 강력한 기법보다는 많은 특정 도메인 지식으로부터 지능을 얻으며 성공적으로 구축된 시스템에서 획득한 지식은 특정 분야에 대해 효율적으로 정의되며, 상식과 같은 정의하기 어려운 지식이나 그 범위가 광범위한 지식의 종류와는 대조된다.

지식 기반의 시스템은 보통 한 사람이상의 전

문가들의 도움으로 구축되며, 성공적인 시스템 구축을 위해서는 이들 전문가들이 시스템에 그들의 전문지식을 집어넣는데 많은 노력을 할 의지가 있어야 하며 이러한 지식의 전달은 점차적으로 전문가와 시스템간의 많은 상호작용을 통해 이루어진다. 전문가도 지식을 처음부터 바르고 완벽하게 얻지는 않는다. 또한 이미 구축된 시스템으로부터 특정도메인에 해당되지 않는 부분을 끌어내는 것이 가능하며 이를 새로운 도메인에서의 새 시스템의 구축에 사용하는 도구로 쓸 수 있다. 현재의 전문가 시스템이 직면하는 문제는 고도의 특정 도메인 지식을 접근해야 하므로 필요할 때 더 많은 일반 지식을 사용할 수 없다는 점이다. 예를 들면 의료 전문가 시스템에 대해 데이터 입력에 실수를 한 경우 (나이에 130을 몸무게에 10 킬로그램을 서로 바꾸어 입력한 경우) 시스템은 이를 구분할 수 없다. CYC 시스템은 특정 전문가 시스템이 구축할 수 있는 상식에 대한 지식의 토대를 제공함으로서 이러한 문제에 대처할 수 있는 한 방법을 제시한다. 이러한 지식 기반의 시스템은 문제 해결을 위한 상당히 복잡한 지식을 가지지 못한다. 보통 그들 자체의 범위와 한계를 추론할 수 없다는 점이 또 다른 어려움인 것이다. 도구의 개발에도 불구하고 지식 기반 시스템 기술을 새로운 도메인으로 적용하는데 있어 지식 습득은 여전히 커다란 병목 현상이다. 또한 이렇게 구축된 시스템의 성능을 측정하는 것은 어렵다. 왜냐하면 사용할 지식의 양을 어떻게 정하느냐를 모르기 때문이다. 시스템에 대한 형식적인 증명을 제시하는 것 또한 불가능하다. 한가지 할 수 있는 것은 실생활의 문제에서 인간 전문가에 맞서 시스템에 테스트하는 것이다. 예를 들면 MYCIN은 10개의 주어진 뇌막염 사례를 평가하는 전문가 패널에 참가해 여러 인간 전문가 보다 좋은 성적을 올렸다. 이와 같이 기존의 지식 기반 시스템들의 한계성이 밝혀지면서 다른 여러 접근 방법에 대한 연구가 이루어 졌고 그 성과로 대두되어진 것이 제약 만족 기반의 접근 방법인 것이다.

2.4. 지식 기반 기법의 비교

스케줄링 문제의 전통적 해결 방법인 정수 프로그래밍(Integer Programming)과 지식 기반 기법과의 비교 노력이 오래 전부터 시도되었다. Vasant, Nicky[16]의 논문에서는 대학의 시간표 스케줄링 문제를 대상으로 실험하여 지식 기반의 전문가 시스템이 IP 모델에 비하여 해의 도출 시간이 예측 가능하며 다중 목적(multiple objective) 함수에 관한 지식 인코딩이 쉬우며 상대적으로 문제의 표현력이 좋다는 장점을 가진다고 기술하였다.

이와 같은 노력으로 지식 기반의 전문가 시스템 구축 시도가 여러 분야에서 이루어졌고 지금도 많은 실세계의 문제들이 지식 기반 기법을 통해 해결되어지고 있다.

3. 제약 만족 기반 기법

3.1. 제약 만족 문제(CSP)

제약 만족 기반 기법의 핵심이 되는 제약 만족 문제(CSP: Constraint Satisfaction Problems) 개념은 최근 AI 분야에서 연구의 대상이 되어 왔다. CSP는 유한 이산 도메인을 가지는 변수의 집합과 이들 변수 사이의 제약조건의 집합으로 문제를 정의하는 개념이다 [6][7]. 이러한 CSP는 실

용적인 중요성을 가지는 것으로 인식되어져 왔는데 이는 OR에서의 특정한 스케줄링 문제, 타임테이블링 문제, 그리고 다른 여러 combinatorial 문제에서 발생하는 많은 문제들이 CSP로서 나타내어질 수 있기 때문이다.

문제들을 수학적 프로그래밍 문제가 아닌 CSP로서 나타내고 해결하는 이유는 크게 2가지이다. 첫 번째는 CSP의 문제 정의 방법이 보다 문제 이해에 용이하기 때문이다. 다시 말해 CSP의 변수들은 문제에서의 개체(entity)에 직접 대응되며 제약조건이 수학적 부등호로 나타내어질 필요 없이 바로 표현된다. 이는 형식적인 면에서 간단해지며 이해하기 쉬울 뿐만 아니라 좋은 휴리스틱의 선택도 직관적으로 이루어질 수 있는 것이다. 두 번째는 CSP 알고리즘 자체는 간단하지만 문제에 주어진 제약조건에 의해 축소되는 탐색공간이 많은 경우 정수 프로그래밍 방법보다 더욱 빨리 해를 찾는다.

최근 제약조건 프로그래밍 도구의 개발과 함께 문제를 CSP로 표현하고 해결하는데 더욱 용이하게 되었다. 첫 번째의 상업적 제약조건 프로그래밍 도구는 프롤로그를 확장한 CHIP(Constraint Handling In Prolog)을 기반으로 한 것이었다. Prolog III는 프롤로그를 기반으로 한 또 하나의 제약조건 프로그래밍 언어이다. 이처럼 프롤로그와 같은 논리 언어에 제약조건을 다루는 기능을 확장한 제약 논리 언어(Constraint Logic Programming)는 논리 프로그래밍 분야의 중요한 연구 영역이다.

그러나 논리 프로그래밍만이 제약조건 프로그래밍의 기본 기반만은 아니다. C++ 라이브러리인 ILOG사의 Solver가 또 다른 제약조건 프로그래밍 도구인 것이다.

이러한 도구들을 이용하여 만들어진 스케줄링 시스템중 대표적인 주기장 할당 스케줄링 시스템으로는 프랑스 코지택사에서 개발한 주기장 할당 시스템인 APACHE [8]와 ILOG사에서 개발한 CHANGI 국제공항의 주기장 할당 시스템이 있다.

3.2. CSP 모델링

제약 만족 기반의 접근 방법을 어떠한 문제를 적용하기 위해서는 해결하고자 하는 문제를 CSP로 정형화하는 것이 필수적이다. 모델링 방법은 여러 가지가 있을 수 있으며 각 방법에 따라 탐색 공간의 크기가 달라지기 때문에 문제 해결에 상당한 성능상의 차이를 발생시킬 수도 있다. 그러므로 효율적인 모델링을 위해서는 상당한 노력이 요구되어진다[13]. PERT 문제에서의 CSP 정형화를 예를 들면 PERT 문제는 우선순위 제약조건(Precedency Constraints)을 가지는 문제로서 N개의 태스크로 구성되며 각 태스크는 완료되기 위해 일정한 소요기간이 필요하며 다른 태스크에 대해 우선 순위가 존재한다. 문제를 정형화하면 도메인 변수들이 가지는 도메인의 범위는 N개의 태스크를 모두 수행하는 경우의 최대 작업기간인 각 태스크의 소요기간(D)의 합(P)이다.

$$\sum_{i=1}^{N} D(i) = P$$

도메인의 범위가 결정되고 나면 각 도메인 변수(V)의 도메인이 결정된다.

$$1 < \sum_{i=1}^{N} V(i) < P$$

다음으로 우선 순위 제약조건을 다음과 같이

지정해 주면 문제의 정형화가 이루어진다. 이 식에서 S(n)은 작업 n의 시작 시간을 나타낸다.

$$Si > Sj + D(i)$$

이 문제의 최적해는 목적함수 $\sum_{i=1}^{N} D(i)$을 최소화하는 해를 구하는 것이다.

이와 같은 CSP 모델링은 제약 만족 기법을 적용하여 시스템을 구축하는데 있어서 신중히 고려되어야 할 부분인 것이다 [9][14].

3.3. 제약 만족 기법의 비교

기존의 여러 선형 문제를 해결하기 위해 사용되어진 정수 선형 프로그래밍(Integer Linear Programming) 기법과 제약 만족 기법과의 비교 노력이 제약 만족 기법의 관심과 더불어 시도되었다. Barbara[15]의 논문에서 그 한 예를 발견할 수 있다.

논문은 문제의 특성이 선형 제약조건으로 쉽게 나타낼 수 없거나, 다른 변수의 도메인을 제약조건 전파를 통해 축소시킬 수 있는 경우에는 정수 선형 프로그래밍보다 제약 만족 기법이 문제 해결에 더 좋은 방법임을 제시하고 있다. 실험은 산업 현장의 실제적인 문제를 대상으로 실험한 것이 아니라 Progressive Party Problem이라는 이론적 문제를 대상으로 한 것이다.

이와 같은 비교 노력을 통해 제약 만족 기반 기법의 효용성이 점차로 알려지게 되어 관련 분야의 연구에 도움을 주고 있다.

하지만 지식 기반 기법과 제약 만족 기반 기법의 비교 사례는 상대적으로 적으며 이제부터 주기장 할당 문제를 대상으로 한 이 두 방법의 비교 사례를 기술하도록 하겠다.

4. 주기장 문제의 모델링 및 구현

4.1. 주기장 할당 문제

주기장 할당 문제는 자원 할당 문제의 하나로서 운항스케줄에 의해 도착하고 출발하는 항공기들을 하나의 작업(task)으로 보고 각 작업을 제한된 수의 주기장에 할당시키는 문제이다. 각 운항편은 서로 다른 시간에 도착하고 출발하며 도착부터 출발사이 시간에는 반드시 서로 중복되지 않도록 하면서 어느 주기장에든 할당되어져야 하는 것이다. 또한 항공기의 종류에 따라 어떤 주기장은 주기가 허용되지 않는다. 주기장은 크게 승객이 직접 타고 내리는 탑승교 주기장(Bridge)과 승객을 수송할 버스가 필요한 일반 주기장(Remote Spot)으로 나뉜다.

본 연구에서 적용 대상으로 한 김포 국제공항은 탑승교 20개소, 일반 주기장 60개소의 총 80개 주기장을 가지며 하루 평균 250~350편의 항공기가 운항한다. 이는 하루 평균 300편의 항공기가 운항될 때 (80!)300 가지의 노드(node)로 이루어진 탐색공간을 포함하게 된다. 따라서 방대한 탐색공간에 대한 효과적인 탐색기법 및 해결기법이 시스템 성능을 크게 좌우하게 되는 것이다.

주기장 문제에서는 여러 가지의 복합적인 제약조건이 존재한다. 하나의 운항편은 정해진 시간동안에 하나의 주기장을 점유한다. 또한 항공기를 주기장에 배정하는 경우에도 크기에 대한 제약조건이 포함된다. 이러한 크기에 대한 제약조건은 주기장의 지역에 따라 다르며 동일 지역이라 하더라도 주기장에 따라 종류가 다양하다.

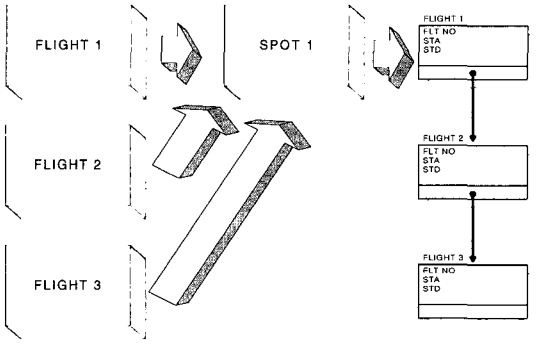
다른 스케줄링 문제와 비교하여 주기장 문제에서 특징적으로 고려해야 할 중요한 요소로는 견인(Towing)이 있다. 대부분의 스케줄링 문제에

서는 작업이 분할되는 경우가 없으나 스케줄링 문제에서는 견인을 통해 작업이 분할되는 것이다.

이러한 견인의 횟수를 줄이는 것 또한 시스템이 해를 찾을 때 고려할 대상이 될 수 있으며 모든 운항편이 배정되지 않는 열악한 환경의 경우는 미배정편의 수를 최소를 줄이는 것이 고려 대상이 된다. 이 외에도 공항의 자원이 풍족한 경우는 가능한 한 운항편을 탑승교 주기장에 배정하여 승객들이 바로 청사로 내릴 수 있도록 하는 배정도 고려 될 수 있다 [5].

4.2. 지식 기반 기법과 제약 만족 기법을 이용한 모델링

지식 기반 기법에서 사용되는 데이터 구조는 시스템에 따라 여러 형태를 가진다. 본 연구에서 개발한 주기장 할당 전문가 시스템인 RACES(Ramp Activity Coordination System)는 먼저 문제에서의 객체인 운항편과 주기장에 대한 데이터 구조를 결정하였다 [5]. 우선 각 운항편과 주기장을 하나의 객체로 보고 객체 생성



〔그림 4-1〕 RACES에서의 데이터 구조

단계를 거친다. 이렇게 생성된 운항편 객체에는 할당될 주기장 번호를 저장할 공간이 있고 주기장 객체에는 그 주기장에 할당된 운항편에 대한 정보를 유지하는 데이터 구조를 가진다. 기본적인 데이터 구조는 [그림 4-1]과 같다.

RACES 내부적으로는 각 주기장 객체에 'assign_time' 이라는 슬롯을 통해 해당 주기장에 배정되는 운항편의 정보를 저장하게 된다. [그림 1]의 경우 3개의 운항편의 출발 시간(STA), 도착 시간(STD)이 [표 4-1]과 같다면

〔표 4-1〕 운항편의 출발시간, 도착시간 예

운항편	도착시간(STA)	출발시간(STD)
Flight 1	09:00	12:00
Flight 2	07:00	08:00
Flight 3	14:00	18:00

시스템 내부의 'assign_time' 슬롯에는 다음과 같은 형태의 리스트가 구성되는 것이다.

spot1@assign_time = [[flight1, 900, 1200], [flight2, 700, 800], [flight3, 1400, 1800]]

이렇게 주기장에 할당된 운항편은 다음 운항편의 배정에 앞서 항상 시간 순서로 정렬된다. 항상 정렬된 순서를 유지함으로서 새로운 운항편의 배정 시 주기장의 빈 시간대를 찾는 도메인 검사를 효과적이고 빠르게 할 수 있도록 하였다. 하지만 이러한 데이터의 유지는 일관성이 제대로 유지되지 않는 경우 시스템의 확장이나 수정 시 상당한 노력이 요구되어 진다.

제약 만족 기법의 경우 문제를 CSP로 정형화하는 과정에서 이러한 데이터 구조가 자연스럽게 프로그래밍 도구 안에 표현되므로 프로그래머가 따로 관리할 필요가 없어진다. 이것이 두 기법간의 커다란 차이이며 제약 만족 기법이 지식

기반 기법에 비해 융통성을 발휘하는 점이다.

RACES에서 사용한 지식 표현 방법은 일반적으로 지식 기반 시스템에서 사용하는 구축방법인 규칙 기반의 방법을 채택하였다 [5]. 도메인 전문가 3사람과 약 6개월간의 면담, 회의과정을 거쳐 전문가들이 배정하는 경우 사용되는 규칙을 만들었다. 이 인간 전문가의 규칙은 사람이 읽을 수 있는 형태로 표현하면 기본 규칙 40여 개와 지역에 따른 세부 규칙 100여 개로 구성되어졌다. 다음은 이렇게 만들어진 규칙 중에 한 예를 든 것이다.

규칙 1 : 각 주기장은 주기 가능한 크기의 운항편만을 주기할 수 있다.

도메인 전문가들과 함께 만든 이러한 규칙들은 CHIP상에서는 내부적으로 다음과 같이 표현된다.

```
type_assign(Flight, Spot) :-
        get_flt_type(Flight, Type),
        get_spot_size(Spot, Size),
        member(Spot, Type), !,
        assign_flight(Flight, Spot).

type_assign(Flight, Spot) :-
        get_alternative_spot(NextSpot),
        type_assign(Flight, NextSpot).
```

RACES는 이렇게 만들어진 데이터 구조와 도메인 전문가들로부터 얻은 규칙을 이용해 추론함으로서 해를 구하게 된다. 구해진 해는 다시 전문가로부터 검증과정을 통하여 새로운 규칙을 추가하거나 수정하여 최적에 가까운 해가 되도록 시스템에 피드백(Feedback)된다. 즉, 지식 기반 시스템에서의 해는 사용자 만족도 측면에서의 근사 최적해를 구하게 되는 것이다.

제약 만족 기법을 이용한 주기장 할당 문제의 모델링에서도 시스템의 데이터 구조는 기본적으로 지식 기반 방법에서의 형태를 따르며 차이점은 운항편 객체가 도메인 변수를 포함한다는 것과 지식 기반 기법에서의 'assign_time' 같은 데이터 구조를 따로 유지할 필요가 없다는 것이다. 지식 기반 기법과 비교하여 중요한 고려 대상이 되는 것은 지식 기반 기법에서는 스케줄링 과정에서 필요에 의해 운항편을 견인시킬 수도 있는 반면 제약 만족 기법에서는 반드시 견인 제약조건을 고려하여 탐색을 시작하기 전에 계획된 운항 데이터를 가지고 견인시켜야 한다는 점이다.

지식 기반 기법에서 사용되는 규칙은 제약 만족 기법에서는 도메인 변수 사이의 제약조건 형태로 표현된다. 예를 들어 앞 절에서 사용했던 규칙 1은 제약조건에 따른 도메인 여과의 형태로 나타난다. 즉, 최초 도메인은 전체 주기장이 되어 다음과 같은 형태가 될 것이다.

```
domain(ke101, 7203, [s62, s61, s60, s55, s54,
s53, s52, s51]).
```

이 도메인은 제약조건을 적용하여 도메인내의 위배요소들을 제거해 주는 다음의 모듈을 통해 축소되는 것이다.

```
constraint_post_1(ke101, DomainList).
?- get_domina_value(DomainList, Result).
Result = domain(ke101, 7203, [s62, s61, s54, s52]).
```

위의 경우 제약조건의 부여를 통해 ke101 운항편의 도메인이 [s62, s61, s60, s55, s54, s53, s52, s51]에서 [s62, s61, s54, s52]로 축소되었음을 알 수 있다. 이 작업은 전처리 단계를 통한 도메인 여과 모듈을 통해 수행될 수도 있고 아

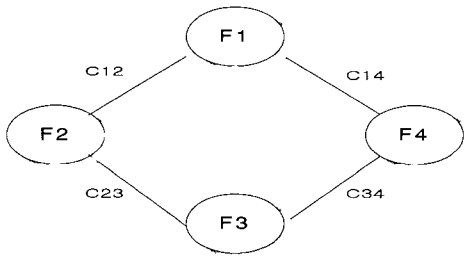
니면 객체 생성 단계에 통합될 수도 있다.

이러한 제약조건의 지정은 도메인 변수 자체에 제약이 가해지는 단항 제약조건인 경우는 주로 전처리과정을 통해 변수의 도메인을 여과함으로서 표현된다. 주기장 할당 문제에서 단항 제약조건으로 대표적인 것은 다음과 같다.

1) 각 운항편이 주기 가능한 주기장 지정.
2) 어제의 스케줄링 결과에 의해 야간 주기 된 운항편에 대한 주기장 지정.

물론 이 이외에도 견인에 대한 조건, 특수 항공기에 대한 배정 조건 등 그 공항의 사정에 따른 여러 단항 제약조건이 추가될 수 있다.

그리고 변수간 상호 관련되는 제약조건인 이항 제약조건이 표현된다. 이항 제약조건은 주기장 할당 문제에서는 비겹침 제약조건(Not-overlapping constraints)으로서 주기장 점유시간이 겹치는 운항편들은 같은 주기장을 할당받을 수 없는 제약조건인 것이다. 이는 실제적으로는 도메인 변수간의 아크 일관성 유지(arc-consistency)를 통한 제약조건 만족을 의미하는 것이다. 이를 제약조건 그래프(Constraints Graph)로 나타내면 [그림 4-2]와 같다.



〔그림 4-2〕 주기장 할당 문제의 Arc Consistency 체크를 위한 Constraints Graph

위의 그래프에서 보면 F1, F2, F3, F4는 각 운항편의 주기장 값을 가지는 도메인 변수이며 제약조건 Cij 는 각 운항편 Fi와 Fj 사이의 제약조건인 운항편의 도착시간(STA)과 출발시간(STD)이 서로 겹치지 않는다는 것을 나타낸다. 이러한 이항 제약조건은 전진 검사(forward checking)와 같은 제약 만족 기법이 적용되어 탐색 시에 도메인을 축소시킨다. 예를 들면 F1의 도메인이 [s50, s51, s52, s53]이며 F2의 도메인이 [s50, s51, s52, s53, s54, s55]인 경우 두 운항편이 서로 비겹침 이항 제약조건 관계에 있는 경우 만약 운항편 F1이 [s50]에 배정되면 그 순간 시스템은 자동적으로 F2의 도메인에서 [s50]을 제거하여 F2의 도메인은 [s51, s52, s53, s54, s55]로 축소되는 것이다.

이러한 도메인의 축소과정은 문제의 전체 탐색 공간의 크기를 급격히 작게 함으로서 시스템이 탐색 공간을 모두 탐색해 최적의 해를 구할 수 있는 방법을 제공하는 것이다. 지식 기반 시스템에서 문제의 전체 탐색 공간을 모두 탐색하지 않고 휴리스틱을 이용해 근사해를 찾는 것에 비해 사용자에게 더욱 커다란 이익을 줄 수 있는 것이다. 하지만 어떠한 문제에서 최적해의 정의는 객관적인 수치로 평가될 수 있는 것도 있으나 그렇지 않은 경우는 그 정의부터 어려운 문제도 존재한다. 본 연구에서 주기장 문제의 최적해로 객관적인 수치로 평가될 수 있는 승객수를 이용한 최적해를 구하였다. 승객을 수송할 버스가 필요한 일반 주기장의 배정을 최소화하여 수송에 필요한 자원을 절약할 수 있는 해를 최적해로 정의하였다. 그러므로 목적 함수(Objective Function)는 다음과 같다.

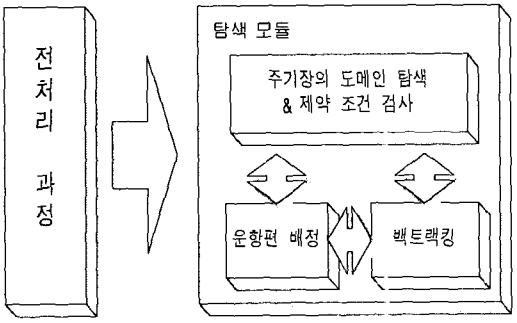
(Minimize)

$$\sum_{j=1}^{M}\left(\sum_{i=1}^{ki} Sj(i)\right)$$

M은 전체 일반 주기장의 수를 나타내며 kj는 j 번째 일반 주기장에 배정되는 운항편의 수를 나타낸다. 그러므로 Sj(i) 는 j 번째 일반 주기장의 i 번째 배정 운항편 승객수가 된다. 이 목적 함수를 최소로 하는 해를 최적해라고 정의하였다.

4.3. 제약 만족 기반 시스템의 구성

지식 기반 기법에서의 해의 탐색은 실제 코드 상에서 주기장의 시간 도메인을 체크해 가면서 운항편이 배정 가능한 지를 확인한 후 조건에 부합되는 경우 실제 배정을 수행하는 과정이다. 이러한 과정은 일단 한번 배정한 결과에 대한 백트랙킹을 코드 상으로 구현해야 하며 이는 상당한 프로그래밍 기술과 이를 위한 추가적인 데이터 구조가 요구되어진다.

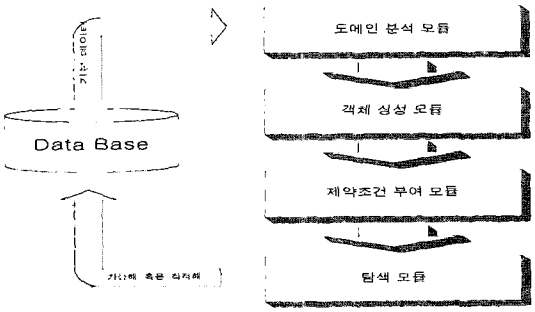


〔그림 4-3〕 지식 기반 기법에서의 시스템 구성도

[그림 4-3]에서와 같이 지식 기반 기법에서는 탐색 모듈이 주기장 도메인을 관리하면서 제약조건을 체크, 배정을 수행한다. 이는 탐색 모듈과 제약조건을 검사하는 모듈이 서로 합쳐져 있음을 의미하는데 이와 같은 형태는 시스템의 확장이나 유지 보수 시에 상당히 어려움을 겪는 원인이 된다.

반면, 제약 만족 기법에서의 해의 탐색은 제약조건을 검사하거나 도메인을 관리하는 모듈과는 독립적으로 이루어지게 된다. 즉, 시스템은 도메인 변수들의 도메인 값 중에 하나를 바인딩 함으로써 해를 탐색해 나가는 것이다. 이것은 시스템의 효율성과 확장성에 커다란 이점을 부여한다.

해를 찾는 단계에서는 휴리스틱을 부여할 수 있다. 휴리스틱의 부여는 주로 값을 바인딩 할 변수의 선택 순서와 하나의 변수 내 도메인 중에서 바인딩 할 값을 선택하는 순서에 따라 결정된다. 예를 들면 배정 우선 순위 도착시간이 빠른 것 먼저 수행한다면 변수의 선택은 도착시간이 빠른 운항편 순으로 결정될 것이다. 특정 운항편에 대해 특정 지역의 주기장으로 먼저 배정하는 것은 도메인 변수의 도메인 중 특정 지역의 주기장에 대한 값을 먼저 바인딩 하면 된다. 또한 승객 수에 따른 이윤의 극대화 및 주기장 사용의 최적화라는 관점에서 최적함수를 구성하고 이를 만족하는 최적해를 구할 수 있다.



〔그림 4-4〕 CSP 모델링 시스템의 구성도

제약 만족 기법의 적용에 있어서의 또 한가지 고려 사항으로는 시스템의 제약조건에 따라 필수적이거나 성능상의 문제로 작업을 분류하여 해를 탐색하는 과정, 즉 부분해를 구하는 과정이 필요하다는 것이다. 본 논문에서 구현한 제

약 만족 기법을 적용한 시스템도 성능상의 효율을 위해 각 운항편을 출발편, 도착편, 연결편으로 분류하여 각각의 부분 배정을 수행하는 과정을 거쳐 해를 구하였다.

[그림 4-4]는 전체 제약 만족 기법을 적용한 시스템의 구성도를 도식화한 것이다. 도메인 분석 모듈과 객체 생성 모듈은 하나로 합쳐질 수 있다. 그림에서 보듯이 제약조건을 부여하는 모듈이 탐색 모듈과 분리되어 있어 제약조건이 변경되는 경우나 시스템을 다른 공항의 환경으로 이식시키는 등의 확장성과 융통성이 제공되어지는 것이다.

4.4. 성능 향상을 위한 고려 사항

제약 만족 기법을 적용한 시스템의 성능을 향상시키기 위한 궁극적인 방법은 탐색 공간을 축소하는 것이다. 이를 위해서는 다음과 같은 3가지 원칙이 적용된다.

1) 최소의 탐색 공간을 가지는 문제의 모델링 선택
2) 모든 도메인 변수의 도메인 축소
3) 결정적인(deterministic)부분과 비결정적인(non-deterministic) 부분의 분리, 배정

최대한의 탐색 공간의 축소를 위한 모델링의 선택은 하나의 문제가 여러 가지 방법으로 모델링 가능한 경우 해당된다. 하지만 문제에 따라서는 오직 하나의 모델링 방법만이 존재하는 경우도 있으므로 모든 문제에 적용되는 것은 아니다.

모든 도메인 변수의 도메인을 축소하는 것 또한 탐색 공간을 현저히 줄일 수 있는 방법이다. 이는 전문가의 경험적 지식이 시스템에 적용되어질수록 그 효과가 크게 나타난다. 하지만 전문가의 경험적 지식으로 축소시킨 탐색 공간에 최적해가 존재한 경우, 다시 말해 전문가의 경험적 지식이 잘못된 경우에는 시스템이 찾은 해가 최적해가 아니며 이를 증명할 수 있는 방법이 없다는 것이 커다란 문제점이다.

스케줄링 문제에 따라서는 모든 작업이 도메인을 갖지 않아도 되는 문제가 있을 수 있다. 예로 주기장 할당 문제의 경우 전날 야간 주기된 운항편에 대해서는 당일 해당 운항편이 도메인을 가질 필요 없이 결정적으로 같은 주기장에 배정시킬 수 있다. 이와 같이 문제에 따라서 결정적으로 처리할 수 있는 작업들을 미리 처리한 후 비결정적으로 처리할 작업들만을 대상으로 해를 탐색하면 결정적으로 처리한 작업 수에 비례하여 탐색 공간이 크게 축소되는 것이다. 이는 지식 기반 기법과 제약 만족 기법의 혼합된 형태의 모델링이라 볼 수 있는 것이다.

5. 실험 및 평가

본 논문에서 모델링 한 두 가지의 시스템은 HP 워크스테이션 환경 하에서 CHIP(Constraint Handling in Prolog)이라는 제약 논리 프로그래밍 언어를 사용하였다. 실험에 사용된 데이터들은 1996년도에 본 연구실에서 대한항공과의 산학 협동을 통해 개발한 주기장 관리 시스템인 RACES 구축 시 전문가들을 통해 검증 받은 데이터들을 사용하였다. 데이터들은 크게 기본 항공편 계획 데이터와 전날 야간 계류 데이터, 그리고 기본적인 주기장과 비행기에 대한 데이터로 구성된다. 본 연구에서는 이 데이터를 가지고 순차적인 제약조건의 부여를 통해 실험한 결과를 토대로 하였다.

주기장 할당 문제에서 CSP 모델링을 이용하

여 구현한 프로그램은 제약조건의 지정과 탐색 모듈이 확연히 분리되어 수정 및 확장이 용이하였다. 이는 제약 만족 기반의 방법이 지식 기반 방법에 비해 문제에 대한 표현력이 뛰어나 프로그래밍 노력이 상당히 적게 든다는 것을 의미한다. [표 5-1]은 두 시스템을 비교한 것이다.

〔표 5-1〕 접근 방법에 따른 시스템의 비교

접 근 방 법		지식 기반 시스템	제약 만족 기반 시스템
데이터	운항편 수	284	
	주기장 수	80	
전문가의 도메인 지식을 문장으로 서술한 규칙수		140	
프롤로그 규칙수		268	0
제약조건수		5	155

위의 [표 5-1]에서 실험에 사용한 데이터는 운항편 수 284개와 주기장 수 80개이다. 지식 기반 시스템에서는 전문가가 문장으로 서술한 규칙을 프롤로그의 프레디킷으로 구현하였다. 표에서 나타난 바와 같이 그 수는 제약조건으로 표현한 수에 비해 상당히 많아졌고 이는 도메인 지식이 변경되는 경우 관련된 많은 부분을 수정하는 것이 불가피함을 의미한다. 반면 제약 만족 기반 시스템에서는 각 운항편의 주기장 도메인을 축소시키는 제약조건으로서 도메인 지식을 구현하기 때문에 도메인 지식의 변경에 융통성 있게 적응할 수 있는 것이다. 이는 전문가 시스템에서 필수적으로 요구되어지는 시스템의 유지 보수 및 개선의 관점에서 매우 커다란 장점인 것이다.

표에 대해 자세히 설명을 하면 배정을 위한 도메인 지식의 수는 140개이다. 여기서 도메인 지식이란 도메인 전문가로부터 지식 습득 과정을 거쳐 얻은 지식을 의미한다. 이를 지식 기반 시스템에서는 내부 규칙, 혹은 함수 268개로 표현한다. 내부 규칙으로 변환된 수가 도메인 지식의 수보다 많아지는 것은 일반적인 자연어로 표현되는 도메인 지식이 함축적이기 때문이다. 반면에 제약 만족 시스템은 위의 도메인 지식들이 모델링 시 155개의 제약조건으로 표현되며 제약조건화 되지 못하는 나머지 지식들은 5개의 내부 규칙으로 표현되어 탐색 시에 휴리스틱으로 이용된다. 이는 제약 만족 기법이 지식 기반 기법에 비해 전문가의 도메인 지식을 간결하고 효율적으로 표현할 수 있음을 의미한다.

〔표 5-2〕 리소스가 부족한 경우(김포공항 국제선 경우 - 탑승교: 16개)

운항편 종류	운항편 수	시간(ms)	최소승객수 (명)
국제선-국제선	19	9490	476
국제선-국내선	10	4990	366
국내선-국제선	10	5290	457
국제선 도착편	9	4710	0

지식 기반의 방법은 도메인 전문가의 지식을 바탕으로 근사 최적해(near optimal solution)만을 찾는 반면 제약 만족 기반의 방법은 제약조건 전파(constraint propagation)를 이용한 탐색 공간 축소를 통해 최적해를 찾는다. 이러한 최적해와 근사해의 차이는 문제에 따라 사용자에게 여러 면에서 커다란 이익의 차이를 의미하는 것이다. 그러나 본 실험에서는 김포공항의 리소스인 주기장이 매우 부족하여 지식 기반의 시스템에서는 미배정 편수를 줄이는 근사 최적해를 구한 반면 제약 만족 시스템에서는 승객 수송 횟수를 최소화시키도록 주기장을 배정하는, 다시 말해 가능한 많은 운항편을 탑승교 주기장에 배정함

으로서 일반 주기장에서 내린 승객을 버스로 수송해야하는 경우를 최소한으로 줄이는 해를 찾았다. [표 5-2]는 김포공항 국제선의 경우 최적해를 찾는 시간을 나타낸 것이다.

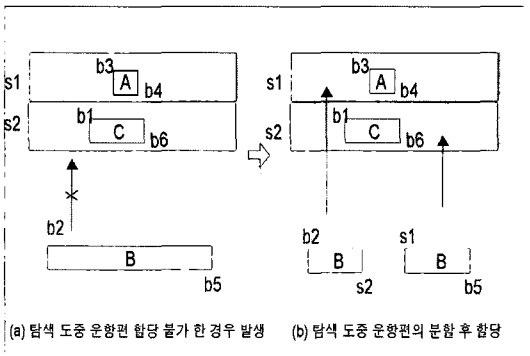
[표 5-3]은 리소스가 충분하다는 가정아래에서의 일종의 시뮬레이션 결과를 나타낸 것이다. 이렇게 탑승교 주기장에 내리는 승객 수를 최소화하는 것은 객관적인 수치가 나오기 때문에 하나의 최적해라고 할 수 있으며 리소스가 매우 부족한 상황에서는 최적해의 정의가 미배정 편수를 줄이는 것과 같이 달라지게 된다.

〔표 5-3〕 리소스가 충분한 경우(탑승교: 68개)

운항편 종류	운항편 수	시간(ms)	최소 승객수(명)
국제선-국제선	19	36120	0
국제선-국내선	10	5760	0
국내선-국제선	10	5520	0
국제선 도착편	9	4730	0

위의 두 표를 비교해서 보면 리소스가 많은, 즉 탐색공간이 큰 쪽이 최적해를 구하는데 상대적으로 많은 시간이 걸리는 것을 확인할 수 있다. 이는 후자가 탐색공간이 크기 때문에 최적해를 찾는데 더 오랜 시간이 걸리는 것으로 이와 같이 제약 만족 기법의 뛰어난 문제 표현력을 이용하여 보다 쉽게 시뮬레이션을 수행함으로서 리소스의 수를 결정하는 것과 같은 의사결정 과정에서 경제적으로 엄청난 손실을 피할 수 있는 것이다. 현재 김포공항의 운항편 수와 리소스(탑승교)를 이용한 시뮬레이션 결과를 신공항 리소스 할당 등에 적용하는 것이 그 좋은 예일 것이다.

이러한 제약 만족 기반 기법의 장점에도 불구하고 앞서 밝힌 바와 같이 김포공항의 특수한 환경으로 인해 두 방법을 통해 구한 해가 서로 상이하였다. 그 이유는 지식 기반 기법에서는 적용하려는 문제가 김포공항과 같이 해가 나오기 어려울 정도로 부족한 리소스를 가지는 경우 도메인 변수에 해당되는 운항편을 견인(towing)을 통해 분할하여 배정하기 때문이다.



〔그림 5-1〕 탐색 도중 도메인 변수의 수가 증가하는 예

[그림 5-1]을 보면 전체 리소스인 주기장이 's1', 's2' 라고 가정하면 탐색이 수행되어 운항편 'A'가 's1' 주기장에, 'C'가 's2' 주기장에 배정된 후 운항편 'B'가 배정될 수 없는 상황이 발생한다. 지식 기반의 기법을 적용한 경우 운항편 'B'를 견인이라는 작업을 거쳐 분할한 후 전반부는 's1'에 후반부는 's2' 주기장에 배정하는 것이다. 반면 제약 만족 기법에서는 이러한 방법으로의 배정을 수행할 수 없기 때문에 운항편 'B'를 's1'이나 's2' 주기장에 배정하지 못하고 미사용 주기장 같은 곳에 배정시켜 놓고 전문가가 별도로 배정시키게 된다.

6. 결 론

제약 만족 기반의 기법은 지식 기반에 비하여

다음과 같은 장점을 가진다.

- 문제에 대한 표현력이 뛰어나 제약조건의 변화에 쉽게 적응 가능하다.
- 제약조건 전파(constraint propagation)를 통해 탐색 공간을 축소하여 최적해(optimal solution)를 찾는다.

반면 다음과 같은 어려운 점을 포함한다.

- 해를 찾는 과정에서 도메인 변수의 개수가 동적으로 변하는 경우이다. 해가 나오기 어려울 정도의 리소스만이 제공되며 도메인 변수의 분할 혹은 합병이 가능한 문제 영역의 경우 전문가들의 지식 기반 해법은 해의 탐색 중간에 도메인 변수의 수를 필요에 따라 변경한다. 김포공항 주기장 문제에 있어서 배정 중간에 적절한 리소스가 할당 불가능한 운항편이 나타나게 되면 해당 운항편 하나를 2개로 분할하여 각각 배정하는 경우이다.

주기장 할당 문제에 있어서 지식 기반 기법을 이용하여 구축한 시스템을 제약 만족 기반 기법을 이용하여 구축하려고 시도하였으며 앞서 밝힌 여러 제약 만족 기반 기법의 장점에도 불구하고 김포공항의 열악한 환경으로 인해 발생하는 동적 변수로 인해 지식 기반 시스템이 찾아내는 해와 동일한 해를 구하는 시스템 구축에는 성공하지 못하였다.

제약 만족 기반 기법을 적용하려는 문제가 다음과 같은 성질을 가지고 있다면 지식 기반 기법을 이용하여 구한 해와 다른 결과를 가질 수 있다.

스케줄링 문제는 매우 광범위하기 때문에 모든 경우에 대하여 제약 만족 기법이 적용될 수 있으며 효율적이라고 말할 수는 없다. 그러므로 보다 많은 문제를 분석하고 그 특성을 파악하여 적용하는 것이 필요하며 이것은 문제 해결을 위한 보다 좋은 접근 방법의 선택에 도움을 줄 것이다.

참 고 문 헌

[1] P. V. Hentenryck, Constraint Satisfaction in Logic Programming, MIT-Press, 1989.

[2] G. S. Jo, Constraint Logic Programming Tutorial, Proceeding of '93 Korea/Japan Joint Conference on Expert System, 1993.

[3] M. Dincbas, P. V. Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. "The Constraint Logic Programming Language CHIP," In Proceedings of the International Conference on Fifth Generation Computer Systems(FGCS'88), pp 693-702, Tokyo, 1988.

[4] R. O. Brazile, K. M. Swigger, "GATES: An Airline Gate Assignment and Tracking," IEEE Expert, pp 33-39, 1988.

[5] J. J. Jung, C. Y. Yang, G. S. Jo, "The Scheduling Strategies of Ramp Activity Coordination Expert System," PACES/SPICES 97 Conference Proceedings, pp 408-414, 1997.

[6] V. Kumar, "Algorithms for Constraint Satisfaction Problem: A Survey," AI Magazine 13(1):32-44, 1992.

[7] B. M. Smith, A Tutorial on Constraint Programming, University of Leeds School of Computer Studies Research Report Series, Report 95.14, 1995.

[8] "APACHE: Constraint Logic Programming Solves Aircraft Parking Problem," The

Magazine of the Advanced Systems and Software, Vol. 7 ,No 9., September 1991.

[9] M. S. Fox, Constraint-Directed Search: A Case Study of Job-Shop Scheduling, Research Notes in Artificial Intelligence, Fitman Publishing, 1987.

[10] 오 윤상, 논리언어에서 유한이산 도메인을 위한 제약조건 해결기의 구현, 인하대 전자계산학과 석사학위 논문, 1994.

[11] P. Prosser, "Domain filtering can degrade intelligent backtracking search," International Joint Conference on Artificial Intelligence, pp. 262-267, 1993.

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

[13] H. P. Williams, Model Building in Mathematical Programming, John Wiley & Sons, 1978.

[14] S. French, Sequencing and Scheduling : An Introduction to the Mathematics of the Jop-Shop, Ellis Hoewood Limited, 1982.

[15] B. M. Smith, S. C. Brailsford, P. M. Hubbard & H. P. Williams, "The progressive party problem: integer linear programming and constraint programming compared," Principles and Practice of Constraint Programming - CP95, Springer Verlag, pp 36-52, 1995.

[16] V. Dhar, N. Ranganathan, "Inter Programming vs. Expert Systems: An Experimental Comparison," Communication of the ACM, pp 323-336, 1990.