메타프로그래밍 제어를 통한 제약 중심의 코스 스케줄링에 관한 연구

정종진 · 조근식

Constraint Directed Course Scheduling in Meta-Programming

Jong-Jin Jung and Geun-Sik Jo

요 약

전통적으로 스케줄링 문제를 해결하기 위해 LP(Linear Programming) 기법이 주로 적용되어 왔으나, 스케줄링 문제의 많은 자원과 지식, 제약조건의 복잡한 상관 관계를 LP 기법으로 표현하고 처리하기가 쉽지 않다. 따라서 최근에는 AI 기법을 스케줄링 문제에 많이 적용하고 있고, AI 기법은 지식 표현및 휴리스틱을 다루기에 효과적이므로 문제를 모델링하고 해결하는데 용이하다 할 수 있다. 본 논문에서는 AI 기법을 기반으로 하여 스케줄링에 적합한 휴리스틱 및, 탐색 기법, 지식 표현 방법등을 연구하고, 이를 바탕으로 코스 스케줄링 시스템을 구현하였다. 먼저 시스템은 전체적으로 메타프로그래밍을 통하여 초기 스케줄링(initial scheduling)과 동적 재스케줄링(reactive scheduling)을 수행하도록 하였다. 메타프로그램이 초기 스케줄링을 수행할때에는 휴리스틱과 자체적인 도메인 여과 기법을 적용하여 탐색 공간의 불일치 요소(inconsistency)를 제거시킴으로써 백트랙킹의 발생을 최소화시켰다. 또한 초기 스케줄링의 결과를 가지고 메타프로그램이 동적 재스케줄링을 수행할때에는 제약조건을 통한 휴리스틱을 이용하여 초기해에 대한 조정을 최소화할 수 있는 메카니즘을 제시하였다. 이에 대한 적용 결과는 실험을 통하여 기존의 논리 언어가 제공하는 탐색 알고리즘과 비교하고 분석하였다.

I. 서 론

스케줄링은 제조업이나, 항공 산업, 또는 경영 과학등의 다양한 영역에서 응용되는 핵심 기술로서 각각의 적용 영역의 특성에 따라 작업과, 자원의 형태, 그리고 스케줄링 결과에 대한 만족도가 달라질 수 있다. 따라서 어떠한 시스템을 개발하였다 하여도 그

*인하대학교 대학원 전자계산공학과

개발 기법에 종속되어 시스템은 특정 도메인에 한정될 가능성이 많고, 시스템의 용도에 따라 그 효용 가치를 상실하는 경우가 발생할 수 있는 것이다. 스케줄링 문제를 해결하기 위한 연구로서, 기존에는 OR(Operations Research) 기법을 많이 적용했다. OR적 관점에서 Linear Programming은 문제에 대한 제약조건을 =이나 ≤, ≥의 관계로 표현하고, 이에 대한 목적함수로서 최적해를 구한다.[3,7] 그러나, 이 접근 방법은 수학적인 프로그래밍 기법에 기반을 두고 있기 때문에 스케줄링 문제에 포함되어 있는 많은 제약조건을 표현하기가 어렵고, 오랜 경험에 의한 전문가의 도메인 지식(domain knowledge)을 적용하기가 쉽지 않다. 따라서 문제에 대한 모델링(modeling)이 어렵고 복잡해지며, 모델링한 후 수정이 용이하지 않다는 한계를 내포하고 있다.[3,4] 뿐만 아니라, 이 접근 방법에 의해 구해진 해를 사용자가 알아 보기 힘들고, 주변 상황의 변동에 의해 시스템의 수정이 불가피해져서 재스케줄링해야 하는 경우에는 이에 대한 조정이 힘들어지게 된다. 이런 문제점을 해결하기 위해 최근에는 스케줄링 문제에 인공지능(Artificial Intelligence)기법을 적용하는 연구가 활발히 진행되기 시작했다. OR 분야에서의 접근 방법들이 수학적 연산을 처리하기에 쉽고, 특별한 알고리즘을 사용해서 하나 또는 여러 개의 목적함수를 사용할 수 있게 하는 반면에, 인공지능 분야에서는 주로 기호적 관계에 기반을 두고 경험이나 직관에 의한 문제 해결법을 다루기 위해 추론 방식의 접근 방법(inference-based approaches)을 많이 사용한다.[9,15] 따라서 여러 기호적 제약조건이 복잡하게 연관되어 있고, 전문가의 경험에 의한 지식을 필요로 하는 많은 스케줄링 문제의 경우 인공지능 기법을 적용하여 효율적으로 해결할 수 있다.

본 논문에서는 이러한 관점에서 효과적인 스케줄링을 수행하기 위해 필요한 휴리스틱과, 탐색 기법, 그리고 지식 표현 방법등을 연구하여 코스 스케줄링 문제라는 어플리케이션을 대상으로 적용하고, 이에 대한 시스템을 구현하였다. 본 논문의 구성을 보면, 먼저 2장에서 시스템이 적용한 코스 스케줄링 문제에 포함되는 자원들과 여러 가지 제약조건에 대해 살펴보고, 3장에서는 이러한 문제를 가지고 시스템이 메타프로그래밍의 제어를 통하여 초기 스케줄링(initial scheduling)과 동적 재스케줄링(reactive scheduling)을 수행하는데 있어서 적용한 기법들을 설명한다. 또한 4장의 실험 결과를 통하여 적용 기법들을 비교하고 분석하며, 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해서 논하도록 한다.

II. 코스 스케줄링 문제

본 연구에서 구현한 시스템이 다루고 있는 문제는 인하 대학교 전자계산공학과의 수업 환경에 기반을 두고 있다. 이 과에서는 모든 과목이 스케줄링을 하기 전에 각각의 교수들에게 미리 배정되어 있고, 한 과목에 대해서 두 명 이상의 교수가 설강을 할 수 없도록 되어 있다. 또한 수강하는 학생들의 수와 교수들이 설강해 놓은 과목의 수는 많은 반면에, 수업을 하기 위한 강의실의 수는 상대적으로 적다는 여건을 가정으로 하고 있다. 코스 스케줄링 문제는 구현 관점에서 볼 때 교수와, 교수들이 설강해 놓은 과목, 학생, 강의실이라는 네 가지의 자원들 사이에서 교수의 과목을 교수와 학생이 가지고 있는 제약조건을 고려하여 강의실에 할당하는 문제라고 볼 수 있다. 이 자원들은 상호 연결되어 있으며, 또한 그들 사이에는 복잡한 제약조건이 관계되어 있다. 교수와 학생, 그리고 강의실에 대한 각각의 제약조건은 다음과 같다.

1) 교수에 대한 제약조건

1. 모든 교수에 대해서 적용되어야 할 제약조건

1.1 모든 교수들은 일주일에 1회씩 모두 모여서 학과 회의 시간을 갖는다.

1.2 모든 교수들은 하루 총 8교시의 수업 시간중 4교시 시간에 점심 시간을 갖는다.

2. 각각의 교수에 대해서 적용되어야 할 제약조건

2.1 각 교수는 최소한 일주일에 1일의 연구일을 갖는다.

2.2 각 교수는 개인적인 업무에 필요한 시간을 가능한 한 많이 갖도록 한다.

3. 각 교수는 같은 시간에 두 과목을 동시에 강의할 수 없다.
4. 각 교수는 하루에 5시간 이상의 수업 시간을 갖지 않도록 한다.
5. 각 교수는 일주일에 3시간 이상 1교시 수업을 갖지 않는다.
6. 각 교수는 3학점 짜리 과목에 대해서 하루에 3시간 연강을 하지 않는다.

2) 학생에 대한 제약조건

1. 2학년 학생은 교양과목 이수를 위하여 1학기에 3학점의 "공업 수학" 이란 과목을 수강해야 한다.
2. "공업 수학" 과목에 대한 강의실이 다른 건물에 있으므로 시간표에서 이 시간의 바로 앞뒤 시간에는 다른 강의시간이 배정되어 있으면 안된다.
3. 3학년 학생은 교양과목 이수를 위하여 1학기에 3학점의 "공업 경영" 이란 과목을 수강해야 한다.
4. "공업 경영" 과목에 대한 강의실이 다른 건물에 있으므로 시간표에서 이 시간의 바로 앞뒤 시간에는 다른 강의 시간이 배정되어 있으면 안된다.
5. 각 학년 학생은 동시에 두개의 필수과목을 수강할 수 없다.
6. 1학년을 제외한 모든 학년의 학생은 교양과목을 포함하여 총 18 학점의 전공과목을 이수해야 한다.
7. 한 학년에 대해 설강된 필수과목중에서 같은 시간에 다른 강의실에 동시에 배정되어 있는 과목이 존재해서는 안된다.

3) 강의실에 대한 제약조건

한 강의실에서 이루어지는 수업은 1교시에서 8교시로 제한한다.

이상의 제약조건을 고려하여 시스템은 어떠한 충돌도 발생시키지 않고 스케줄링을 수행해야 한다. 즉, 각각의 교수에 대한 과목들을 강의 시간 테이블상에 배치하는데 있어서 위에서 언급한 모든 제약조건을 만족시켜야만 하는 것이다.

III. 메타프로그래밍 제어하의 스케줄링

3-1. 시스템의 구조

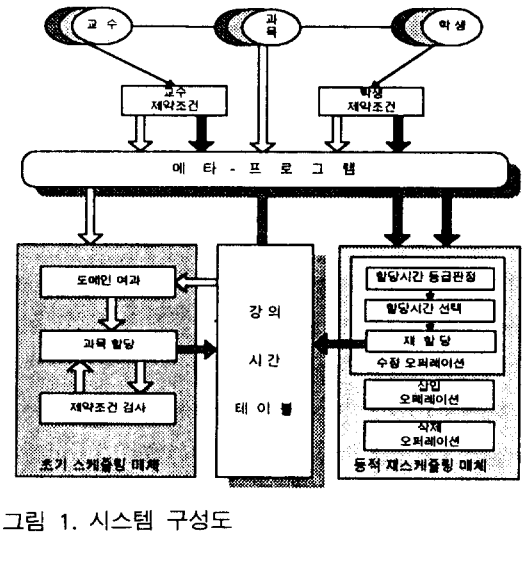

그림 1. 시스템 구성도

본 연구에서는 코스 스케줄링 문제에 대한 시스템을 구현하는데 있어서, 시스템이 스케줄링을 수행하는

방법은 우선 시스템을 메타프로그램의 제어하에 초기 스케줄링 매체(initial scheduling agent)와 동적 재스케줄링 매체(reactive scheduling agent)로 분리해서 수행하도록 하였다. 위의 (그림 1)에서 흰색 화살표는 초기 스케줄링 매체의 수행에 관련된 부분을 나타내고, 검정색 화살표는 동적 재스케줄링 매체의 수행에 관련된 부분을 나타낸다. 또한 회색 화살표는 구하려는 강의 시간 테이블의 변화 과정을 보여주고 있다. 그림에서 알 수 있듯이, 메타프로그램은 전체적인 스케줄링 과정을 관리하고 제어한다. 먼저 메타프로그램은 초기 스케줄링 매체로부터 교수와 학생, 과목등의 자원을 가지고 초기 스케줄링을 수행한다. 이로부터 구해진 초기해는 메타프로그램의 제어하에 다시 동적 스케줄링 매체의 입력 부분이 되어 재스케줄링된다. 물론 시스템은 외부 변화가 발생하지 않을 때에는 초기 스케줄링 매체의 수행으로서 작업을 끝내고 초기해를 출력할 것이다.

3-2. 초기 스케줄링 매체

1) 제약조건 관리 기법

본 연구에서는 코스 스케줄링 문제를 기술하는데 있어서 기호적인 접근 방법을 사용하였다. 강의 시간 할당 테이블상의 모든 시간은 의미 있는 숫자들의 리스트로 표현된다. 즉, 각 숫자는 강의실과, 요일, 강의 시간에 대한 정보를 내포하고 있다. 예를 들면, 만약에 어떤 수업이 화요일 1교시에 320호 강의실에서 이루어지도록 스케줄되어 있다면, 이에 상응하는 리스트의 숫자는 321이 된다. 여기서 3은 320호 강의실을 의미하고, 2는 화요일을 의미하며, 1은 1교시를 의미한다. 이와 같은 강의시간 리스트를 바탕으로 하여 본 연구에서 구현한 시스템에서는 대부분의 제약조건을 리스트에서 할당할 수 없는 시간들을 제거하는 방식으로 표현하였다. 이렇게 표현된 제약조건은 모두 프롤로그 데이타 베이스에 명시적으로 선언되어 스케줄링시에 처리된다. 또한 문제에 포함된 자원에 강요되는 다양한 종류의 제약조건을 효과적으로 다루기 위하여 제약조건을 몇개의 그룹으로 분리하여 처리하였다. 즉, 시스템은 문제를 표현하는데 있어서 기호적인 방법을 사용하기 때문에, 제약조건에 대한 처리는 도메인 여과(domain filtering)를 통해 이루어진다. 따라서 제약조건은 그 성격에 따라서 처리되는 순서가 구분되어서, 이들이 스케줄링 전에 처리될 수도 있고, 스케줄링 도중에 처리될 수도 있는 것이다. 다음의 (그림 2)에서는 스케줄링시 제약조건이 처리되는 과정을 보여주고 있다.

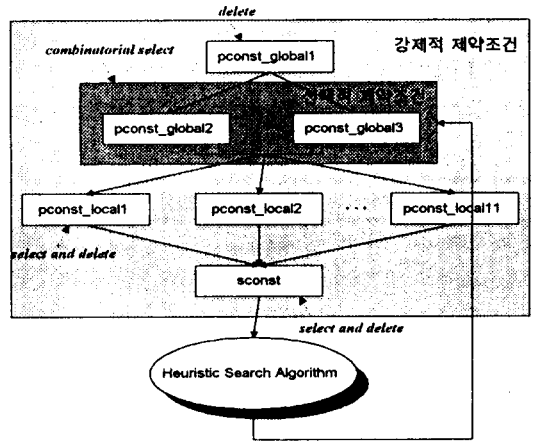

그림 2. 제약조건의 처리과정

위의 (그림 3)에서 pconst-global1은 모든 교수에 해당하는 제약조건으로, 스케줄링을 시작하기 전에 처리(*delete*)된다. 다시말해서, 시스템은 대부분의 제약조건을 도메인으로부터 제약조건에 위배되는 요소들을 제거하는 방식으로 처리하므로 모든 교수들에 대해서 만족되어야 할 pconst-global1에 위배되는 시간 요소들을 강의 시간 테이블에서 미리 제거할 수 있는 것이다. 만약 이러한 제약조건들을 스케줄링 전에 미리 처리하지 않고, 스케줄링 도중에 처리한다면, 시스템이

백트랙킹을 발생시킬때마다 이 처리 과정을 반복 수행하여 시스템의 수행 능력을 크게 저하시킬 것이다. pconst-locali(i=1,2,…,11)와 sconst는 각각의 교수와 학생에 대해서 만족되어야 할 제약조건으로 스케줄링 수행 도중에 해당될때마다 선택되어 처리(*select and delete*)된다. 이상의 제약조건들은 강제적 제약조건(mandatory constraints)이라 불리우는 그룹에 속하는 것으로서, 강의 시간표를 작성하기 위해 반드시 만족되어야만 한다. 예를 들면, 교수가 설강해 놓은 과목중 어떤 것도 동시에 같은 시간에 할당되면 안된다는 제약조건이 이에 포함된다. 즉, 한 교수의 과목이 어떤 시간에 이미 할당되어 있다면, 다른 과목들은 다른 시간에 할당되어야만 하는 것이다.

또한 pconst-global2와 pconst-global3는 선택적 제약조건(optional constraints)이라는 그룹에 속하는 것들로서, 반드시 만족되어야 하지만 주어진 여건 속에서 가능한 한 최적의 해를 구하기 위해 유연성을 발휘할 수 있는 제약조건이다. 이들은 데이타 베이스에 선언할 수 없고 술어절(predicate)로 표현되어 스케줄링 도중에 검사된다. 예를 들어, 일주일에 각 교수가 갖을 수 있는 1교시 수업 시간의 최대수나, 각 교수에 대해 할당된 하루의 최대 수업 시간수 등이 이런 제약조건에 속한다. 분리된 제약조건을 바탕으로 시스템이 스케줄링을 수행함에 있어서는 강제적 제약조건을 우선적으로 처리한 후에 선택적 제약조건을 처리한다. 또한 선택적 제약조건은 선호도(preference)를 가지고 있어서 일차적으로 선호도가 높은 제약조건에 대해 시스템이 할당을 시도해 보고, 해를 구할 수 없을 경우에는 조금 낮은 선호도를 갖는 제약조건에 대해 차례로 할당(*combinatorial select*)한다. 따라서 시스템은 최적해에 가까운 해를 구할 수 있는 것이다.

이와 같이 제약조건을 비슷한 성질을 갖는 그룹들로 분리하므로써 본 연구에서는 스케줄링 문제를 좀더 쉽게 묘사할 수 있고, 또한 시스템으로 하여금 실제적인 수행 과정을 조정할 수 있도록 하였다. 그러나 이런 방법을 기존의 프롤로그 언어가 가지고 있는 백트랙킹 기법으로 수행한다면, 시스템의 수행 시간이 훨씬 늘어날 것이다. 따라서 효과적인 탐색 기법의 적용이 요구되어진다.

2) 도메인 여과를 통한 제약 만족

본 연구에서는 시스템이 탐색 공간을 줄이면서 스케줄링을 효과적으로 수행할 수 있도록 하기 위하여 자체적인 휴리스틱 탐색 기법(heuristic searching technique)을 구현하였다. 탐색 공간을 줄이기 위해 기존의 여러 기법들이 연구되어 왔고, 또한 이러한 기법들은 많은 시스템에 적용되었다. (4)와 (13)의 논문과 같은 경우에는 각각 리스트 커미트먼트(least commitment)기법과 포워드 체킹(forward checking)기법을 사용하였다. 이들은 각각 방법론적인 차이를 가지고 있으나 의미론적으로 볼때에는 모두 어떤 값을 할당하기 전에 탐색 공간을 미리 검사하여 탐색 공간의 크기를 줄이도록 하는 방법들로 간주할 수 있다.

본 연구에서는 이런 맥락에서 기존의 프롤로그 탐색 기법을 개선한 것으로 도메인 여과 기법을 사용하였다. 도메인 여과 기법은 스케줄링을 시작하기 전에 일관성

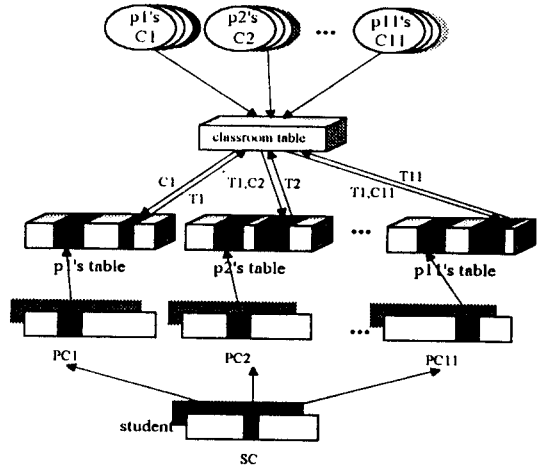

그림 3. 도메인 여과에 의한 제약 만족

검사를 수행하므로써 변수의 도메인중에서 불일치되는 부분을 제거하여 탐색 공간을 줄이는 방법으로 위에서 언급된 기법과 유사하게 작동되나, 코스 스케줄링 문제의 경우 주어진 제약조건이 탐색 공간을 반복적으로 찾게 하는 특성을 갖고 있기 때문에 더욱 적당하게 이용될 수 있다. 도메인 여과에 의한 제약만족의 수행 방법은 다음의 (그림 4)와 같다.

시스템이 각각의 교수와 학생에 대해서 만족되어야 할 제약조건을 고려하면서 주어진 과목을 강의 시간 할당 테이블에 할당을 시도해 나갈 때, 기존의 할당 결과는 나머지 할당해야 할 과목의 실패 가능성을 추론하는데 도움을 줄 수 있다. (그림 5)에서 보여지듯이, 먼저 각각의 교수들은 원래의 시간 테이블로부터 복사된 가상의 시간 테이블을 소유한다. 첫 번째 교수를 p1, 이 교수가 가지고 있는 제약조건을 PC1이라 하고, 두 번째 교수와 제약조건을 p2, PC2, i번째 교수와 제약조건을 각각 pi, PCi라 한다면, 교수 pi는 pi's table을 갖는다. 그래서 선택된 첫 번째 교수 p1이 PC1을 만족시키면서 자신의 시간 테이블 p1's table에 과목 C1을 할당하고 나면, 시스템은 C1이 할당된 시간들인 T1을 나머지 교수들이 소유하는 가상의 시간 테이블에서 제거해서 T1에 다른 교수의 과목들이 중복되어 할당되지 않도록 한다. 즉, C1의 관점에서 보면, T1에 이미 C1이 할당되었으므로 다른 교수들의 시간 테이블의 T1 시간대에 다른 과목이 할당되어서는 안되는 것이다. 또한 첫 번째 교수의 과목이 강의 시간 테이블의 어떤 시간에 할당되면, 그 다음에 나머지 9명의 교수들의 시간 테이블에 대해서 이미 할당된 시간들을 제거하고 남은 시간들을 가지고 할당할 수 있는지를 각각 검사한다. 뿐만 아니라 시스템이 새로운 교수들의 과목에 대해서 할당을 시도할 때에는 할당 가능한 총 시간수와 교수의 과목수를 비교해 봄으로써 재차 할당 실패 가능성을 검사하게 된다. 그래서, 만약 다른 교수들의 과목중에서 각각 가지고 있는 제약조건을 만족시키면서 할당할 수 없는 과목이 존재하게 된다면, 시스템은 할당한 시간들을 취소시키고 다른 시간들에 재차 할당하게 된다.

결과적으로 이러한 할당 실패 가능성을 미리 조사해 봄으로써 시스템이 실제 해를 구하는데 있어서 계산 시간을 가능한 한 줄일 수 있는 것이다.

3-3. 동적 재스케줄링 매체

대부분의 다른 스케줄링 문제에서와 마찬가지로 코스 스케줄링 문제를 해결하는데 있어서 유동적으로 변하는 환경에 대처하여 재스케줄링하는 능력은 시스템이 갖추어야 할 중요한 요소중의 하나이다. 코스 스케줄링 문제에 있어서 교수의 개인적인 사정이나, 강의실 조건, 또는 각종 교내 행사 등에 의하여 기존의 작성된 스케줄링 결과가 변경되어야 하는 상황은 항상 발생할 수 있다. 이 경우 시스템은 변화에 능동적으로 대처하기 위해서 기존의 배정된 자원중에서 변경 사항에 영향을 받지 않는 부분은 그대로 보존하고, 가능한 한 변경 내용과 관련된 부분에 대해서만 재스케줄링하므로써 이미 작성된 스케줄링 결과의 조정량을 최소화하도록 하는 능력이 필요하다. 기존의 다른 스케줄링 시스템에서 보여지듯이 동적 재스케줄링의 수행은 로드 밸런스(load balancing)나, 제약조건 완화(constraint relaxation), 기존 스케줄의 부분적 교환 등과 같은 도메인 휴리스틱 기법(domain specific heuristics)을 사용하여 효과적으로 행할 수 있다.

(6)의 시스템에서는 환경 변화에 따른 내용 변경을 현재의 스케줄링에서의 제약조건 위배로 간주하고 위배된 부분에 대해서 분석하고 탐지하여 전체적인 탐색 과정에서 문제를 일으킨 탐색 단계로 가서 재스케줄링하는 방법을 취하고 있다.

또한, (11)에서는 변화에 따른 제약조건의 충돌이 발생할 경우 이를 인지하여 충돌의 종류에 따라서 자원-중심 스케줄링(resource-based scheduling), 주문-중심 스케줄링(order-based scheduling)등의 몇

가지 스케줄링 방법중에서 가장 합당한 방법을 선택하여 재스케줄링을 한다.

(12)의 논문과 같은 경우에는 제약조건 기록(constraint recording) 방법을 근간으로 하는 재스케줄링 에이전트가 스케줄링을 수행하면서 스케줄의 진척 상황을 기록해 놓는다. 그래서 재스케줄링해야 할때가 되면 이 기록 내용을 참조하여 불일치(inconsistency)를 일으킨 장소로 백트랙킹해서, 그 시점부터 다시 스케줄링하는 방법을 사용하고 있다.

본 연구에서는 3-1절에서 설명한대로 전체적인 스케줄링 시스템을 메타프로그램의 제어하에 초기 스케줄링 매체와 동적 스케줄링 매체로 분리해서 수행하도록 한다. 그래서 초기 스케줄링 매체로부터 구해진 해와, 자원, 제약조건을 토대로 동적 재스케줄링 매체가 재스케줄링을 수행할때 변화에 따른 자원 및 제약조건의 충돌이 감지되면, 메타프로그램이 이 충돌이 영향을 미치는 다른 제약조건을 인과 관계의 집합으로 서로 연결시키고, 조정하도록 하는 것이다. 그러나 어떻게 변화에 영향을 받는 부분에 대해서만 재스케줄링을 하여 초기해의 조정량을 최소화할 것인지는 어려운 문제이다. 경우에 따라서는 계산 시간상으로 시스템이 환경 변화의 내용으로부터 바뀌어진 제약조건을 가지고 초기 스케줄링 매체가 스케줄링을 다시 수행하는 것보다 더 오래 걸릴 수도 있기 때문이다. 따라서 시스템은 재스케줄링을 할때 각각의 제약조건이 갖는 특성으로부터 구할 수 있는 휴리스틱을 이용하도록 한다. 또한 초기 스케줄링 매체의 수행과정에서 사용했던 휴리스틱 탐색 기법을 이 부분에서도 적용하여 탐색 공간을 줄이면서 스케줄링하도록 한다. 먼저 메타프로그램은 외부 변화에 의해 변경된 내용을 입력받아 초기 스케줄링 매체로부터 구해진 해와 비교한다. 만약 변화된 내용이 구하려는 해에 영향을 미치지 않을때에는 물론 초기해를 최종적인 결과로 출력할 것이다. 그러나 변경 내용을 초기해에 비추었을때 불일치가 발생했을때에는 변경 내용과 초기해를 동적 재스케줄링 매체의 입력 부분으로 보내고 재스케줄링해서 불일치되는 부분을 수정하도록 한다. 외부 변화에 의한 동적 재스케줄링 과정은 그 특성상 다음과 같은 3가지의 오퍼레이션으로 구분할 수 있다.

1. 삭제 오퍼레이션(deletion operation)
2. 삽입 오퍼레이션(addition operation)
3. 수정 오퍼레이션(modification operation)

먼저 삭제 오퍼레이션의 수행은 부분적인 제약조건 완화(constraint relaxation)에 상응하는 것으로, 교수의 과목을 삭제하는 경우와 강의 시간 할당 테이블상의 시간 요소를 삭제하는 경우가 있다. 이 중 교수의 과목이 삭제되는 경우는 단순히 초기해로부터 삭제된 과목의 배정이 있는지를 검사하여 제거하면 된다. 그러나 테이블의 시간 요소가 삭제되는 경우는 만약 삭제하려는 시간 요소에 이미 특정 과목이 배정되어 있다면, 이 과목에 대해서 다시 배정시켜 주어야 하므로 복잡해진다. 그런데 시간 요소의 삭제에 대한 수행은 결국 삭제되는 시간을 다른 변수로 인식해서 수정 오퍼레이션을 수행하는 것과 같은 방식이라 할 수 있다.

삽입 오퍼레이션은 과목 추가의 경우로 볼 수 있으며, 이 경우에는 먼저 강의 시간 테이블중에서 남아 있는 시간에 할당할 수 있는지를 검사하고, 실패할때 역시 다른 변수를 가진 수정 오퍼레이션을 수행하도록 한다.

수정 오퍼레이션의 수행은 의미적으로 볼때 (Px, Cy, Tz)→(Pa, Cb, Tc)의 연산이라 할 수 있다. 즉 이 오퍼레이션은 초기 스케줄링 매체로부터 구해진 강의 시간 테이블에서, 사용자의 요구에 의해 교수 Px의 과목 Cy에 대해 배정된 시간 Tz : (Px, Cy, Tz)를 교수 Pa에 대해 설강된 과목 Cb가 배정되어 있는 시간 Tc : (Pa, Cb, Tc)로 옮기는 과정인 것이다. 시스템이 이 연산을 수행하는데 있어서 백트랙킹의 발생을 적게 하고, 초기해에 대한 최소한의 부분 교정이 이루어지

도록 하기 위해서는 재스케줄링 과정 동안에도 2)에서 설명한 도메인 여과 기법을 적용하고, 또한 제약조건에 대한 휴리스틱을 이용하도록 한다. (Px, Cy, Tz)→(Pa, Cb, Tc) 연산은 (그림 6)에서와 같은 5가지의 경우로 고려해 볼 수 있다.

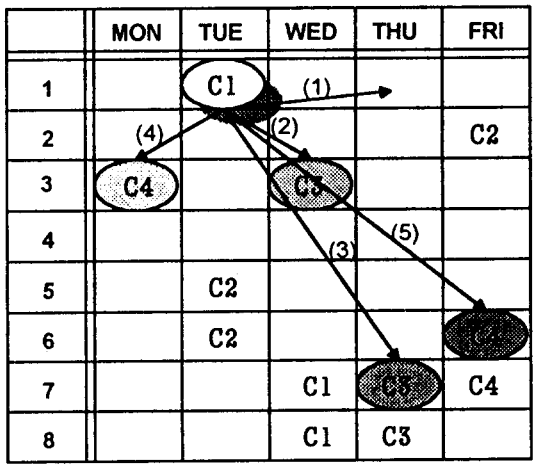

그림 4. 수정 오퍼레이션의 수행

위 그림의 테이블은 초기 스케줄링 매체를 통해 얻은 결과로서, p1 교수의 과목 c1과, p2 교수의 과목 c2, p3 교수의 과목 c3, p4 교수의 과목 c4가 각각 제약조건에 합당한 시간에 할당되어 있다고 가정하자. 이때 외부로부터의 환경 변화에 의해 c1 과목의 배정이 제약조건에 위배되어 시스템이 재스케줄링해서 c1에 대해 할당되어 있는 시간을 다른 시간으로 옮겨야 하고, 이에 따라 나머지 세 과목 역시 부분적으로 다시 배정되어한다고 하면, 다음과 같은 5가지의 가능한 연산이 수행될 수 있다.

첫번째, (1)은 (p1, c1, Tz)→(p1, c1, Tz') 연산으로서 할당되고자 하는 시간이 초기 스케줄링의 수행 후 남은 시간 요소 중의 하나인 경우이다. 이 경우에는 백트랙킹이 전혀 발생하지 않고, c1에 대해 초기 스케줄링 결과로서 할당되어 있는 시간 Tz를, 할당되지 않고 남은 시간인 Tz'로 단순히 교체시키는 한 번의 동작으로 수행된다. 따라서 가장 바람직한 경우라 할 수 있다.

두번째로, (2)는 (p1, c1, Tz)↔(p3, c3, Tz1') 연산으로 p1과 p3간의 제약조건이 서로 만족되어 기존의 c1과 c3에 대해 할당되어 있는 시간들인 Tz와 Tz1'를 서로 교환하는 형식이다. 이 경우 역시 백트랙킹의 발생 없이 Tz와 Tz1'를 서로 교환하는데 필요한 두 번의 동작으로 수행된다.

세번째로, (3)의 경우는 (p1, c1, Tz)↔(p3, c3, Tz') 연산으로서 (2)의 경우와 비슷하나 Tz를 이미 할당되어 있는 2개짜리 연강 시간중에서 하나인 Tz2'와 교체하는 형태이므로 한 과목에 대한 강의 시간이 분산되어 교수의 선호도 제약조건을 충분히 만족시키지 못하게 된다. 따라서, (2)의 경우보다는 좋지 않고 가능하면 피해야 한다.

(4)와 (5)는 각각 다음과 같은 연산으로서, p1과 p4간의 제약조건이 서로 충돌을 일으켜 Tz를 Tz1'(또는 Tz2')로 변경시키고, Tz1'(또는 Tz2')는 또 다른 시간 Tz'로 변경시켜야 할 경우이다.

(4) (p1, c1, Tz)↔(p4, c4, Tz1'),
(p4, c4, Tz1')↔(Px, Cy, Tz")

(5) (p1, c1, Tz)↔(p4, c4, Tz2'),
(p4, c4, Tz2')↔(Px, Cy, Tz')

다시말하면, (2)와 (3)의 경우처럼 c1에 대한 할당 시간인 Tz와 c4에 대한 할당 시간인 Tz1'(또는 Tz2')를 서로 맞바꾸려는데 있어서 Tz1'에 대해서는 p1의 제약조건이 만족되어 Tz를 Tz1'로 옮길 수가 있으나, Tz에 대해서는 p4의 제약조건이 위배되어 Tz1'를 Tz로 옮기지 못하고 다시 제약조건에 맞는 임의의 다른 시간인 Tz"로 이동시켜야 하는 것이다. 따라서 이 경우는 원래의 c1에 대한 가정과 마찬가지로 Tz"를 어떻게 선택할 것인지를 결정하는데 있어서 위의 5가지 연산을 다시 고려해주어야 하므로 좋지 못하다. 또한 (5)는 (3)과 비슷한 경우로서 가장 좋지 않은 경우이다. (4)와 (5)에서는 Tz"를 결정하는 방법에

따라 백트랙킹의 발생 횟수와 계산 시간상의 상당한 차이가 있을 것이다.

메타프로그램은 이때 휴리스틱을 적용하여 가능한 한 제약조건이 적게 작용하는 시간 요소로 옮기도록 한다. 즉 강의 시간 테이블상의 모든 시간 요소들은 기존의 교수와, 학생, 강의실 제약조건과 상호 연관 관계가 있다. 따라서, 관련된 제약조건 정도에 비례하여 교수와, 학생, 강의실별 제약조건을 각 시간 요소 단위의 제약조건으로 변환시킬 수 있다. 그래서 재할당 가능성에 대한 등급을 책정하고, 등급이 가장 좋은 시간 요소를 우선적으로 선택하여 할당을 시도하는 것이다. 시간 요소별로 등급을 책정할 경우 위의 5 가지의 경우에 대한 할당 시도 순위는 다음과 같다. (1)>(2)>(3)>(4)>(5)

결과적으로, 동적 재스케줄링 매체가 재할당을 수행할 때는 제약조건수가 가장 적은 교수에 대해서 가장 적은 제약조건을 갖는 학년의 과목에 할당되어 있는 시간 요소를 우선적으로 선택하여 재할당을 시도한다. 그리고 실패할 경우에는 자동적으로 그 다음 등급을 갖는 시간 요소에 재차 시도한다. 이와 같은 수행 방법은 재할당 실패 가능성을 가능한 한 줄이도록 할 것이다.

Ⅳ. 실험 및 결과

본 연구에서 구현한 시스템은 인하대학교 전자계산공학과의 수업 환경에 기반을 두고 있다. 처음 개발된 시스템은 PC/386 시스템환경하의 프롤로그 언어로 구축되었다. 시스템을 논리 언어의 환경하에서 제약 중심 스케줄링 방식으로 구현함으로써 문제에 대한 모델화나 제약조건의 처리시 기존의 LP 방법이나 다른 기법들보다 효율을 기할 수 있었다. 즉, LP 방법은 문제에 포함된 기호적 제약조건을 표현하는데 있어서 방대한 양의 인위 변수를 필요로 하고, 제약조건의 논리합(disjunction) 관계 및 <, >, ≠로 이루어진 제약조건들을 표현하고, 처리하기가 까다로운 반면에, 본 논문의 제약 중심 스케줄링 시스템은 이들을 단순히 도메인에서 제거하는 방식으로 처리함으로써 훨씬 효과적으로 할 수 있는 것이다. 뿐만아니라 본 시스템은 복잡한 모델화 작업을 거치는 LP 방법에 비해 간단하게 구성함으로써 시스템에 대한 수정과 변경이 상대적으로 쉽다는 장점을 가지고 있다.

그러나 이 시스템은 프롤로그 언어 자체에 내장되어 있는 탐색 기법이 그대로 사용되어서 방대한 양의 탐색 공간을 필요로 하고, 해를 구하기 위해서는 잦은 백트랙킹이 발생되어야만 하였다. 이러한 불합리한점을 개선하기 위하여 두 번째 시스템에는 본 논문에서 구현한 도메인 여과에 의한 휴리스틱 탐색 기법을 적용하였다. 또한 이 시스템은 휴리스틱 탐색 기법을 적용하기 위하여 필요한 메인 메모리를 자유롭게 사용할 수 있고, 계산 속도가 빠른 HP apollo Series 700 모델의 워크스테이션상에서 유닉스 버젼의 논리 언어인 CLP(R)을 이용하여 구현되었다. CLP(R)은 기존의 논리 언어의 단점인 제약 만족(constraint satisfaction)과 최적화(optimization) 문제를 통합한 제약 논리 프로그래밍(Constraint Logic Programming) 언어로서 표현력이 뛰어나고, 기존의 논리 언어의 단점인 수학적 연산 관계의 처리를 보다 쉽게 할 수 있다.

이와 같이 개발된 시스템의 성능을 측정하기 위하여 본 논문에서는 시스템을 실제적인 문제로 실험하였다. 시스템에 대한 성능 측정은 기본적으로 첫번째 시스템과 두번째 시스템의 계산 속도상의 차이를 기준으로 하고 있다. 물론 성능 측정이 같은 조건하에서 이루어지도록 하기 위해 첫번째 시스템을 CLP(R)의 환경에 맞게 옮겨서 두번째 시스템과 비교하도록 하였다. 시스템을 단순히 CLP(R)로 옮겨서 실행할 경우에는 제약논리 프로그래밍 언어로서의 CLP(R)이 아닌 유닉스 버젼의 프롤로그 언어상에서 수행되는 것으로

간주할 수 있다. 시스템이 다루고 있는 문제는 11명의 교수들과, 총 108학점에 해당하는 교수의 과목, 4개 학년, 그리고 4개의 강의실로 구성되어 있고, 그 실험 결과는 아래의 (표 4-1)과 (표 4-2)를 통하여 제시하였다.

표 1. 휴리스틱 적용 결과

탐색 기법 휴리스틱의 사용여부	기존의 프롤로그에 내장된 탐색 기법	구현된 도메인 여과 기법
	sec	*sec*
순서적 할당	10800.3425	1925.6840
재배열 1	648.7170	379.6830
재배열 2	102.4670	29.7500

표 2. 탐색공간의 크기에 따른 구현된 도메인 여과 기법의 성능 평가

탐색 기법 탐색공간의 크기	기존의 프롤로그에 내장된 탐색 기법	구현된 도메인 여과 기법
	sec	*sec*
중복 할당 불가 (상대적인 탐색 공간대)	102.4670	29.7500
부분적 중복 할당 허용 (상대적인 탐색 공간 중)	20.2833	23.7167
중복 할당 허용 (상대적인 탐색 공간 소)	11.4167	13.4500

위의 (표 4-1)은 시스템이 해를 구하는데 있어서 2가지의 휴리스틱을 적용한 결과를 보여 주고 있고, (표 4-2)는 과목들을 선택 과목과 필수 과목으로 분리해서 처리해서 과목에 대한 중복 할당을 허용했을 때와 허용하지 않았을때에 달라지는 탐색 공간의 크기에 대해서 구현된 도메인 여과 기법의 수행 결과를 보여 주고 있다. 또한, 전체적으로 (표 4-1)과 (표 4-2)에는 자체적인 도메인 여과 기법과 일반적인 프롤로그의 탐색 기법을 비교하고 있다. (표 4-1)의 결과에서 보면, 시스템에 휴리스틱을 적용한 결과 훨씬 시간이 단축됨을 알 수 있다. 즉, 시스템의 수행 능력은 단순히 순서적으로 스케줄링하는 것보다, 휴리스틱을 적용하여 다른 교수들보다 상대적으로 강의 시간수가 많거나 제약조건이 많은 교수에 대해 우선적으로 스케줄링할 때 훨씬 향상된다. 스케줄링할 교수에 대한 재배열 방법은 재배열 1과 재배열 2로 구분해서 할 수 있다. 재배열 1은 설강 과목수가 보다 많은 교수를 우선적으로 스케줄링하는 방법이고, 재배열 2는 보다 많고 복잡한 제약조건을 갖고 있는 교수를 우선적으로 스케줄링하는 방법이다. 결과적으로, 시스템은 재배열 1을 적용했을 때보다 재배열 2를 적용했을 때 더 빠르게 수행하였다.

또한, 휴리스틱 탐색 기법을 적용시켰을 때 일반적인 프롤로그의 탐색 기법으로 해를 구했을 때보다 수행 시간상의 효율을 가져올 수 있었다. 그러나, 이미 언급한대로 (표 4-2)의 경우와 같이 스케줄링 동안에 백트랙킹이 거의 발생되지 않을 때에는 휴리스틱 탐색 기법의 수행 능력은 기존의 프롤로그의 탐색 기법에 비해서 뛰어나다고 할 수 없다. 즉 일부 과목을 선택 과목으로 간주해서 같은 시간에 동시에 서로 다른 두 과목을 중복해서 할당하는 것을 허용할 경우에는, 상대적으로 문제가 간단해져서 시스템이 백트랙킹을 적게 일으키면서 해를 구할 수 있게 된다. 이 경우 휴리스틱 탐색 기법은 다소 비효율적이라 할 수 있다. 물론 모든 과목을 선택 과목으로 간주해서 모든 과목에 대해 중복 할당을 허용할 경우에는 구현된 휴리스틱 탐색 기법의 수행 능력은 더 비효율적일 것이다. 그러나, 휴리스틱 탐색 기법은 (표 4-2)의 첫번째 경우와 같이 문제가 복잡하여 도메인에 비해서 탐색 횟수가 많아지고 백트랙킹이 자주 발생하게 되는 경우일수록 좋은 효과를 발휘할 수 있다.

V. 결론 및 향후 연구

본 연구에서는 여러 제약조건들이 서로 복잡하게 연관되어 있는 코스 스케줄링 문제를 해결하는 시스템을 인공 지능적 측면에서 구현하였다. 시스템은 스케줄링을 수행하는데 있어서 제약 중심의 방법을 사용하기 때문에 기존의 LP나 전문가 시스템등의 다른 방법처럼 제약조건을 표현하기가 어렵고, 제약조건 처리에 대해서 스케줄링 수행시에 매번 처리해야한다는 어려움을 갖지 않도록 하였다. 즉 도메인으로부터 제약조건에 위배되는 부분을 미리 제거하고 제약조건에 일치되는 부분에 대해서만 할당을 시도함으로써 문제에 대한 모델링이 쉽고, 스케줄링을 효과적으로 수행하도록 하였다.

또한 시스템의 성능을 향상시키기 위해 시스템이 스케줄링을 수행하기 전에 탐색 공간을 미리 검사하여 백트랙킹의 발생을 가능한 한 줄이도록 하는 탐색 기법을 구현하였으며, 유용한 몇 가지의 휴리스틱을 적용하여 해를 구하는 시간을 단축하도록 하였다. 이러한 기법들을 사용한 결과 본 연구에서는 시스템이 이러한 기법을 사용하지 않았을때보다 수행 시간에 있어서 수 배의 성능 향상을 가져 올 수 있었다.

전체적으로는 시스템이 스케줄링 과정을 메타프로그램에 의해 제어함으로써 스케줄링에 필요한 각각의 처리 루틴들이 모듈 단위로 분리 수행되어 시스템에 대한 수정및 유지가 용이하게 하였다. 뿐만아니라 현재 구현중인 동적 재스케줄링 매체의 경우에 있어서도 재스케줄링시 메타프로그램이 스케줄링의 수행 순서를 조정함으로써 처음부터 다시 스케줄링하지 않고, 변화에 영향을 받는 부분에 대해서만 부분적으로 재스케줄링할 수 있도록 하였다.

현재 본 연구실에서는 제안된 시스템의 동적 재스케줄링 매체를 구현함과 동시에 동적 재스케줄링의 수행시에 사용자와의 대화식 스케줄링(interactive scheduling)이 가능하도록 GUI(Graphic User Interface)를 디자인하고 있다. GUI의 중요성은 갈수록 높아지고 있고, 스케줄링 시스템의 경우 사용자로부터의 요구를 시스템이 직접 받아들여 실시간에 가깝게 처리하는 능력은 동적 재스케줄링에 있어서 중요하다 할 수 있는 것이다. 따라서 본 연구에서는 사용자가 화면을 통해 스케줄링 결과를 편집하고, 재스케줄링을 직접 수행시킬 수 있도록 시스템의 GUI 부분을 연구할 것이다.

참고문헌

1) Bitner, J. R. and Reingold, E. M., Backtrack Programming Techniques, *Communications of the ACM, Vol. 18*, pp.651~655, 1975.

2) Bratko, I., PROLOG Programming for Artificial Intelligence 2nd edition, *Addison-Wesley*, 1990.

3) Dantzig, G., Linear Programming and Extensions, *Princeton University Press*, 1963.

4) Dhar, V. and Ranganathan, N., Integer Programming vs. Expert Systems : An Experimental Comparison, *Communications of the ACM March*, Vol. 33, Number 3, pp.323~333, 1990.

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

6) Fox, M.S. and Smith, S.F., ISIS-A Knowledge-Based System for Factory Scheduling, *Expert Systems*, Vol.1, No. 1, 1984.

7) French, S., Sequencing and Scheduling : An Introduction to the Mathematics of the Job-Shop, *Ellis Horwood Limited*, 1982.

8) Geun-Sik Jo, Constraint Logic Programming, Tutorial, *Proceeding of '93 Korea/Japan Joint Conference*

on Expert Systems, 1993.

9) Haralick, R. M. and Elliot, G. L., Increasing Tree Search Efficiency for Constraint Satisfaction Problems, *Artificial Intelligence*, Vol. 14, pp.263~313, 1980.

10) Nadel, B. A., Constraint Satisfaction Algorithm, *Computational Intelligence 5, pp.188~224*, 1989.

11) Ow, P.S., Smith, S.F. and Thirez, A.., Reactive Plan Revision, *CAIA*-88, pp.77~82, 1988.

12) Prosser, P., A Reactive Scheduling Agent, *International Joint Conference on Artificial Intelligence August*, pp.1004~1009, 1989.

13) Rick Evertsz, The Development of Syllabus-An Interactive, Constraint-Based Scheduler for Schools and Colleges, *Innovative Applications of Artificial Intelligence* 3, pp.38~51, 1991.

14) Sterling, L. and Shapiro, E., The Art of Prolog Advanced Programming Techniques, *The MIT Press Cambridge*, 1986.

15) Van Hentenryck, P. and Dincbas, M.., Domains in Logic Programming, *In Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 759~765, Menlo Park, Calif : AAAI, 1986.

16) Yoshikawa, M., Kaneko, K. Nomura, Y. and Watanabe, M., A Constraint-Based Approach to High-School Timetabling Problems : A Case Study, *AAAI* '94, Vol.2, pp.1111~1116, 1994.