Journal of The Korea Society of Computer and Information
Vol. 22 No. 9, pp. 9-16, September 2017

www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.09.009

Constraint Programming Approach for a Course Timetabling Problem

Chun-Sik Kim*, Junha Hwang**

Abstract

The course timetabling problem is a problem assigning a set of subjects to the given classrooms and different timeslots, while satisfying various hard constraints and soft constraints. This problem is defined as a constraint satisfaction optimization problem and is known as an NP-complete problem. Various methods has been proposed such as integer programming, constraint programming and local search methods to solve a variety of course timetabling problems. In this paper, we propose an iterative improvement search method to solve the problem based on constraint programming. First, an initial solution satisfying all the hard constraints is obtained by constraint programming, and then the solution is repeatedly improved using constraint programming again by adding new constraints to improve the quality of the soft constraints. Through experimental results, we confirmed that the proposed method can find far better solutions in a shorter time than the manual method.

▸Keyword: Timetabling, Course Timetabling Problem, Constraint Programming

I. Introduction

각 학교의 학과에서는 매 학기마다 수업 시간표를 작성하기 위해 많은 시간을 투입하고 있다. 수업 시간표 작성 문제는 기본적으로 각 교과목의 수업 시간을 배정하는 문제이지만, 수업 시간과 수업을 진행할 강의실뿐만 아니라 각 교수 및 학생에 대한 제약조건까지 고려해야 하는 매우 복잡한 문제이다. 수업 시간표 작성 문제에서 제약조건들은 필수 제약조건(hard constraint)과 선호 제약조건(soft constraint)으로 나누어진다. 필수 제약조건은 반드시 준수해야 하는 제약조건이며 선호 제약조건은 반드시 준수할 필요는 없으나 가급적 준수하면 좋은 제약조건이다. 일반적으로 탐색 문제에서 선호 제약조건은 목적함수로 표현되기 때문에 수업 시간표 작성 문제는 제약 만족 최적화 문제로 정의된다[1]. 수업 시간표 작성 문제는 NP-complete 문제로 알려져 있기 때문에 교과목의 개수가 증가함에 따라 최적해의 도출이 매우 어려워진다[2].

1962년 수업 시간표 작성 문제가 처음 제기된 이래로[3] 지금까지 이 문제를 해결하기 위해 다양한 방법들이 제시되어 왔는데 크게 경영과학과 인공지능 분야로 나눌 수 있다. 경영과학 분야에서는 이 문제의 해결을 위해 정수 계획법(integer programming)을 적용하는 방안이 제시되어 왔다[4, 5]. 제약조건들과 목적함수를 선형적으로 표현할 수 있다면 정수 계획법의 효율적인 탐색 기법을 적용하여 비교적 빠른 시간 내에 최적해의 도출이 가능하다. 그러나 대상 문제에 따라서는 매우 복잡한 제약조건과 목적함수가 포함될 수 있기 때문에 모든 제약조건과 목적함수를 선형적으로 표현하는 것이 현실적으로 불가능할 수 있다. 따라서 인공지능 분야에서는 유전 알고리즘(genetic algorithm)[6], 타부 탐색(tabu search)[7], 개미 시스템(ant system)[8] 등 메타 휴리스틱 탐색 기법들을 활용하여 준최적해를 도출하기 위한 노력을 기울여 왔다[9].

한편 제약 만족 문제를 해결하기 위해 개발된 제약 프로그래밍(constraint programming)은 제약조건들만으로 구성된 문제를 표현하고 해결하는 데 매우 유용한 것으로 알려져 있다[10]. 반면에 목적함수를 동반하는 최적화 문제에 있어서는 다른 최적화 기법에 비해 성능이 떨어지는 경향이 있다. 이로 인해 제약 프로그래밍은 주로 필수 제약조건들만으로 구성된 수

• First Author: Chun-Sik Kim, Corresponding Author: Junha Hwang
*Chun-Sik Kim (miracle_on@hanmail.net), Dept. of Computer Engineering, Kumoh National Institute of Technology
**Junha Hwang (jhhwang@kumoh.ac.kr), Dept. of Computer Engineering, Kumoh National Institute of Technology
• Received: 2017. 07. 11, Revised: 2017. 07. 31, Accepted: 2017. 08. 18.
• This paper was supported by Kumoh National Institute of Technology.

업 시간표 문제에 적용되어 왔다[11, 12]. 또한 제약 프로그래밍은 메타 휴리스틱 기법을 활용한 문제 해결 시 필수 제약조건들을 모두 만족하는 초기해를 생성하기 위한 수단으로 많이 활용되어 왔다[13, 14, 15].

본 논문에서 대상으로 하는 수업 시간표 작성 문제 역시 필수 제약조건뿐만 아니라 선호 제약조건을 모두 포함하는 제약만족 최적화 문제의 일종이다. 기존 연구 [16]에서는 이미 제약 프로그래밍만을 활용하여 대상 문제를 해결하기 위한 반복적 개선 탐색 기법을 제시한 바 있다. 먼저 제약 프로그래밍을 사용하여 필수 제약조건들만을 고려한 초기해를 도출한다. 그리고 선호 제약조건 측면에서 더 좋은 해를 도출하기 위해 제약 프로그래밍을 계속해서 적용하되 탐색 도중에 현재해보다 좋지 않은 해가 도출될 것으로 판단되는 방향으로는 더 이상 탐색을 진행하지 않음으로써 탐색 시간을 절약하게 된다. 이 방법은 제약 프로그래밍을 활용하여 최적화 문제를 해결하는 기본적인 방법이라 할 수 있다. 본 연구에서는 기존 연구 [16]과 마찬가지로 제약 프로그래밍만을 활용하여 대상 문제를 해결하는 방안을 제시하고 있다. 그러나 본 연구의 대상 문제에는 [16]과 달리 새로운 필수 제약조건들이 추가되었을 뿐만 아니라 각 교수별로도 가급적 선호 시간에 배정되어야 한다는 새로운 선호 제약조건이 추가됨에 따라 보다 적극적으로 제약 프로그래밍을 활용하고 있다. 즉, 반복적 개선 탐색 시 현재해의 각 교수별 시간표를 고려한 제약조건을 추가함으로써 기존 연구 [16]과는 달리 각 교수별로 현재해보다 좋지 않은 방향으로의 이동을 방지하여 가급적 많은 구성원이 만족할 수 있는 시간표가 작성될 수 있도록 하였다.

본 연구에서는 제안한 기법의 구현을 위해 IBM ILOG CP를 활용한다. IBM ILOG CP는 현재 전 세계적으로 상업적 또는 학술적으로 가장 널리 사용되고 있는 제약 프로그래밍 개발 라이브러리로서 다양한 제약조건을 표현하기 위한 클래스 및 함수들을 제공하고 있다[17]. 본 논문에서는 대상 문제를 제약 프로그래밍을 활용하여 해결하기 위한 설계 결과를 제시하며, 아울러 구현 방법 기술 시 IBM ILOG CP의 함수명 등을 그대로 사용함으로써 복잡한 제약조건이 실제로 어떻게 구현될 수 있는지 참고할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 대상으로 하는 수업 시간표 작성 문제에 대해 설명하며, 3장에서는 제약 프로그래밍을 적용하기 위한 설계 결과 및 구현 방법에 대해 기술한다. 4장에서는 실험 결과를 제시하고 분석하며, 마지막으로 5장에서 결론 및 향후 과제에 대해 기술한다.

II. Problem Description

1. Course Timetabling Process

일반적으로 대학의 수업 시간표는 학과 단위로 작성된다. 본 연구에서는 경북지역 중규모 4년제 대학 특정 학과의 수업 시간표 작성 문제를 대상으로 하고 있다. 해당 학과의 수업 시간표가 작성되기까지의 과정은 [그림 1]과 같다. 먼저 교무처에서는 교양교직과정부에 교양 및 기초도구 교과목에 대한 시간표 작성을 요청하며, 이 시간표가 확정되면 각 학과에 전공 및 학과기초도구 교과목에 대한 시간표 작성을 요청한다. 해당 학과에서는 먼저 해당 학기에 개설해야 할 교과목과 학년 별 학생 수를 고려하여 교과목 별 분반수를 결정한다. 예를 들면, 하나의 교과목에 대해 1개 내지 3개의 분반이 개설될 수 있다. 그러고 나서 각 교과목/분반 별로 담당교수를 결정한 후 수업 시간표를 작성하게 되는데, 이때 해당 교과목/분반 정보뿐만 아니라 교양 및 기초도구 시간표와 수업 배정이 가능한 강의실 정보를 모두 고려하여 작성하게 된다.

본 연구에서 대상으로 하는 수업 시간표 데이터는 총 3가지로 2016년 1학기, 2016년 2학기, 2017년 1학기 수업 시간표이다. 매 학기마다 수업 배정이 가능한 강의실은 총 6개로 동일하며, 이 중 실습실이 4개, 일반 강의실이 2개이다. 그 외에 매 학기 별 교과목(Subject)/분반(Class) 개수와 담당교수 수는 [표 1]과 같다. 담당교수는 전임교수(Full-time)와 시간강사(Part-time)로 구분된다. 그리고 각 교과목/분반은 주당 수업 시간이 2교시로 구성되어 있는 경우도 있고 3교시 또는 4교시로 구성되어 있는 경우도 있다. 참고로 각 교시는 1시간 단위로 수업이 진행된다. [표 1]의 “Total Period”는 해당 학기의 총 교시 수를 의미한다.

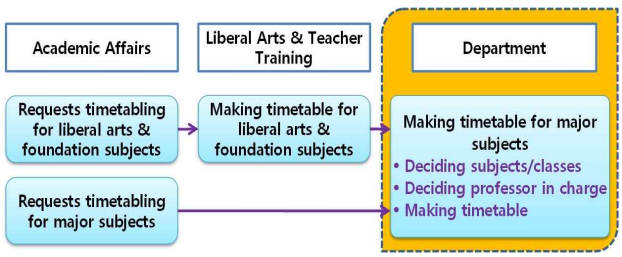

Fig. 1. General Timetabling Process

Table 1. Number of Subject and Professor per Semester

Semester	Subject	Class	Total Period	Professor	
				Full-time	Part-time
2016-1	22	56	160	13	4
2016-2	27	65	192	13	6
2017-1	23	53	155	14	5

2. Constraints

수업 시간표 작성을 위한 제약조건은 필수 제약조건과 선호 제약조건으로 구분된다.

필수 제약조건은 다음과 같이 총 12개로 구성되어 있다. C1~C5는 실제로 하나라도 위배하면 수업 진행이 어려운 상황이 발생할 수 있는 제약조건들이며, 나머지는 학과 또는 대학 차원에서 정책적으로 고려되어야 하는 제약조건들이다. 정책적

제약조건들은 부득이한 경우 선호 제약조건으로 변경될 수 있지만 실험 결과에 의하면 C1~C12의 모든 필수 제약조건들을 준수하는 해의 도출이 가능한 것으로 확인되었다.

- C1.강의실 중복 배제 : 한 강의실의 특정 시간에는 하나의 수업만 배정될 수 있다.
- C2.교수 별 불가능 시간 배제 : 담당교수 별로 수업이 불가능한 시간에는 해당 교수의 수업을 배정할 수 없다.
- C3.실습 교과목 실습실 배정 : 실습이 요구되는 교과목은 실습실에 배정해야 한다.
- C4.동일 학년 교과목/분반 간 비겹침 분반 존재 : 동일한 학년의 교과목들 사이에는 한 교과목의 각 분반에 대해 겹치지 않는 분반이 하나 이상 존재해야 한다. 이를 통해 특정 교과목의 분반을 선택함에 따라 수강이 불가능한 교과목이 발생되는 것을 방지한다.
- C5.교양 및 기초도구 시간 배제 : 기 작성된 교양 및 기초도구 시간표에 의거하여 미리 배정된 강의실의 해당 수업 시간에는 수업을 배정할 수 없다.
- C6.2시간 단위 그룹 고려 : 하나의 교과목은 2시간 단위로 수업이 형성되는데, 2시간 수업은 동일한 강의실에서 연강을 실시한다. 단, 2시간 그룹 외의 다른 그룹의 수업은 2시간 그룹과는 다른 요일에 배정하되 동일한 강의실에 배정한다. 예를 들어, 1주 3시간 교과목의 경우 2시간과 1시간 수업으로 나뉘는데 2시간 수업은 연강을 실시하며 1시간 수업은 2시간 수업과 다른 요일에 배정된다.
- C7.특정 교과목/분반들의 동일 시간 배정 : 특정 교과목의 경우 모든 분반을 동일한 시간에 배정해야 한다.
- C8.학과 회의시간 배제 : 학과 회의시간에는 전임교수의 수업을 배정할 수 없다.
- C9.주 3일 이상 배정 : 전임교수는 주 5일 중 3일 이상 수업을 배정해야 한다.
- C10.최대 연강 시간 이내 배정 : 담당교수 별로 최대 연강 시간이 지정되어 있으며, 이에 따라 담당교수 별 연강 시간은 최대 연강 시간을 초과할 수 없다.
- C11.교수 별 점심시간 확보 : 담당교수 별 시간표의 경우 각 요일 별로 점심 시간 확보를 위해 4교시와 5교시 중 하나는 수업을 배정할 수 없다. 참고로 1교시 수업은 9시부터 시작되며, 4교시와 5교시 수업은 각각 12시와 오후 1시에 시작된다.
- C12.학년 별 점심시간 확보 : 각 학년 별 시간표의 경우에도 점심 시간 확보를 위해 요일 별로 4교시와 5교시 중 하나는 수업을 배정할 수 없다.

선호 제약조건은 다음 C13, C14 제약조건과 같이 2개로 구성된다. 두 가지 제약조건 모두 각 교수 별 선호 시간과 관련된 것이다. 기본적으로 각 교수 별로 선호하는 수업 시간이 있고 선호하지 않는 수업 시간이 있다. 본 연구에서는 각 교수 별로 각 수업 시간에 대한 선호도를 0~5의 점수로 나타내었다. 5는 매우 선호, 4는 선호, 3은 보통, 2는 비선호, 1은 매우 비선호, 0은 수업 불가를 의미한다. 즉, 점수가 높을수록 선호도가 높음을 의미한다. C13은 담당교수 전체적으로 선호도가 높은 시간에 수업이 배정되어야 한다는 제약조건이다. 그러나 이 경우 전체적으로는 좋아질 수 있지만 특정 교수의 수업이 오히려 비선호 시간에 배정되는 상황이 발생할 수 있다. 따라서 C14와 같이 각 교수 별로도 가능하면 선호 시간에 수업이 배정되어야 한다는 제약조건을 추가하였다.

- C13.선호 시간 배정 최대화 : 담당교수 전체적으로 선호 시간에 더 많은 수업이 배정되어야 한다.
- C14.교수 별 선호 시간 최대화 : 각 교수 별로 선호 시간에 더 많은 수업이 배정되어야 한다.

III. Constraint Programming Approach

1. Constraint Programming

수업 시간표 작성 문제와 같이 다양하고 복잡한 제약조건들로 이루어진 문제를 제약 만족 문제(constraint satisfaction problem, CSP)라 하며, 이와 같은 문제를 해결하는 기법 중 하나가 제약 프로그래밍이다. 제약 프로그래밍은 대상 문제를 표현하는 방식에 있어서 다른 탐색 기법들에 비해 보다 자연스럽다는 장점이 있다[11]. 즉, 유전 알고리즘 등 많은 메타 휴리스틱 탐색 기법들의 경우 제약조건들을 하나의 목적함수로 변환하여 변형된 최적화 문제를 해결하고 있지만, 제약 프로그래밍에서는 제약조건 각각을 선언적인 방식으로 바로 표현하게 된다.

제약 프로그래밍에서 대상 문제는 다음과 같이 변수의 집합 $X = \{ x_1, ..., x_n \}$, 도메인, 즉 각 변수 xi가 가질 수 있는 값들의 집합 $D = \{ D_1, ..., D_n \}$, 그리고 제약조건의 집합 $C = \{ C_1, ..., C_t \}$로 표현된다. 제약조건들은 각각 각 변수의 범위 또는 변수들 간의 관계를 기술함으로써 각 변수가 취할 수 있는 값들을 제한하게 된다. 제약 프로그래밍의 목적은 각 변수의 값을 결정하되 모든 제약조건을 준수하는 값을 찾는 것이다. 제약 프로그래밍에서 해의 탐색 과정은 기본적으로 각 변수의 값을 차례로 하나씩 결정하는 트리 탐색 방식으로 동작한다. 이때 해를 보다 효과적으로 도출하기 위해 arc consistency와 같은 제약전파 기법을 사용하여 모든 변수들이 항상 모든 제약조건을 만족할 수 있도록 도메인을 재조정하게 된다.

2. CSP Model for the Timetabling Problem

대상 문제에서는 각 교과목/분반의 각 교시 별로 수업 시간과 강의실을 결정해야 한다. 수업 시간은 1주일 중의 수업 시간을 의미하는데, 요일과 수업 시간을 나누어 표현할 수도 있고 1주일 전체의 수업 시간으로 표현할 수도 있다. 따라서 모든 교과목/분반의 총 교시 수를 n이라 할 때 대상 문제를 표현하기 위해 사용한 변수와 도메인은 다음과 같다. 여기서 $D(R_i)$,

$D(D_i)$, $D(T_i)$, $D(DT_i)$는 해당 변수의 도메인을 의미한다.

- R = { R_1, R_2, ..., R_n } : 각 교과목/분반/교시의 강의실 변수
  - D(R_i) = { 1, 2, 3, 4, 5, 6 } : 각 강의실
- D = { D_1, D_2, ..., D_n } : 각 교과목/분반/교시의 요일 변수
  - $D(D_i)$ = { 1, 2, 3, 4, 5 } : 각 요일
- T = { T_1, T_2, ..., T_n } : 각 교과목/분반/교시의 하루 중 수업 시간 변수
  - $D(T_i)$ = { 1, 2, 3, 4, 5, 6, 7, 8, 9 } : 하루 중 수업 시간
- DT = { DT_1, DT_2, ..., DT_n } : 각 교과목/분반/교시의 1주일 중 수업 시간 변수
  - $D(DT_i)$ = { 1, 2, 3, ..., 44, 45 } : 1주일 중 수업 시간

1주일 중 수업 시간(DT)은 요일(D)과 하루 중 수업 시간(T)이 결정되면 [식 1]과 같이 자동으로 값이 결정될 수 있으며, 역으로 1주일 중 수업 시간이 결정되면 [식 2]와 [식 3]을 통해 요일과 하루 중 수업 시간 또한 자동으로 결정될 수 있다. 결국 시간 관련 제약조건을 표현하는 데 있어서 DT 변수 1개만을 사용해도 되고 DT 대신 D와 T를 사용해도 된다. 그런데 대부분의 제약조건들은 DT를 사용하는 것이 자연스럽지만 제약조건에 따라서는 D와 T를 사용하여 표현하는 것이 용이할 때도 있다. 따라서 제약조건에 따라 선별적으로 사용할 수 있도록 D, T, DT 변수를 모두 사용하고 있다. 물론 프로그램 구현 시 [식 1~3]이 이를 위한 제약조건으로 추가된다.

- $DT_i = 9 \times (D_i - 1) + T_i$ (1)
- D_i = { (DT_i − 1)을 9로 나눈 몫 } + 1 (2)
- T_i = { (DT_i − 1)을 9로 나눈 나머지 } + 1 (3)

[그림 2]는 교과목/분반/교시에 대한 변수의 예를 나타낸 것이다. 과목 A는 3교시로 구성되어 있으며 1개 분반만 존재한다. 과목 B는 2교시로 구성되어 있으며 2개 분반이 존재하며, 과목 C 역시 2교시로 구성되어 있으며 2개 분반이 존재한다. 교수 P1은 과목 A, 과목 B-1분반, 과목 B-2분반을 담당하며 교수 P2는 과목 C-1분만과 과목 C-2분을 담당한다. 최종적으로 제약조건을 모두 준수할 수 있도록 각 교과목/분반/교시 별로 강의실(R_i)과 수업 시간(D_i, T_i, DT_i)을 결정해야 한다.

	subject A			subject B class 1		subject B class 2		subject C class 1		subject C class 2	
R	3	3	3	1	1	1	1	5	5	5	5
D	2	2	4	1	1	3	3	4	4	3	3
T	5	6	2	3	4	5	6	7	8	3	4
DT	14	15	29	3	4	23	24	34	35	21	22
	← professor P_1 →							← profeessor P_2 →			

Fig. 2. An Example of Variable Expression

3. Implementation of Hard Constraints

제약조건 "C1.강의실 중복 배제"는 모든 서로 다른 교과목/분반의 각 교시 i와 j의 쌍에 대해 [($R_i \neq R_j$) ∨ ($DT_i \neq DT_j$)]라는 논리식에 의해 만족될 수 있다. 즉, 두 교시의 강의실이 서로 다르거나 수업 시간이 서로 다르면 된다는 것으로서 이는 두 수업 시간의 강의실이 같다면 수업 시간이 서로 달라야 된다는 의미와 동일하다. IBM ILOG CP에서는 ($\sim A \vee B$), 즉 ($A \rightarrow B$)의 논리식을 표현하기 위해 IloIfThen이라는 제약조건 클래스를 제공한다. 따라서 제약조건 C1은 [IloIfThen($R_i = R_j$, $DT_i \neq DT_j$)]과 같이 표현될 수 있다. 제약조건 C1과 같이 비교적 간단하게 표현이 가능한 제약조건들이 많이 있지만 몇몇 제약조건들의 경우 표현이 까다로운 경우도 있다. 여기서는 상대적으로 표현이 복잡한 제약조건인 C4, C6, C9, C10의 구현 방법에 대해 보다 자세히 설명한다.

- C4.동일 학년 교과목/분반 간 비겹침 분반 존재 : 동일한 학년의 교과목 A와 B가 있으며, A의 분반은 총 3개이고 B의 분반은 총 2개라고 가정하자. 교과목 A를 기준으로 1분반 수업 시간이 교과목 B의 1분반과 2분반 모두의 수업 시간과 겹치게 되면 교과목 A의 1분반 수업을 신청하는 경우 교과목 B의 신청이 불가능해진다. 따라서 교과목 A의 1분반에 대해 교과목 B의 분반들 중 수업 시간이 겹치지 않는 분반이 1개 이상 존재해야 한다. 교과목 A-1분반의 수업 시간과 교과목 B-1분반의 수업 시간이 겹치지 않는다는 것은 A-1분반의 모든 교시의 수업 시간들이 B-1분반의 모든 교시의 수업 시간들과 다르다는 것을 의미한다. A1을 A-1분반 변수들의 인덱스 집합이라 하고 B2를 B-2분반 변수들의 인덱스 집합이라 할 때, 이 제약조건은 [$\forall_{i \in A1, j \in B1}(DT_i \neq DT_j)$]와 같이 표현될 수 있으며 IBM ILOG CP에서는 IloAnd 제약조건을 사용하여 표현할 수 있다. A-1분반에 대해 B-2분반과 겹치지 않는다는 제약조건도 동일하게 표현할 수 있으며, 최종적으로는 B-1분반과 B-2분반 중 1개 이상에 대해 겹치지 않으면 된다. 이는 [$\sum_{k \in B}([\forall_{i \in A1, j \in Bk}(DT_i \neq DT_j)]) \geq 1$]와 같이 표현될 수 있다. 다시 말하면 A-1분반에 대해 B의 분반들 중 겹치지 않는 분반이 1개 이상이라는 것이다. 이 제약조건은 IBM ILOG CP의 IloSum 함수를 사용하여 구현할 수 있다. 이와 같은 제약조건은 각 분반을 중심으로 동일 학년의 다른 모든 교과목에 대해 추가된다.

- C6.2시간 단위 그룹 고려 : 교과목 A의 1분반 수업이 3교시로 구성되어 있으며 각각에 대한 변수의 인덱스가 i, j, k라고 하자. 1교시와 2교시는 연강을 실시해야 하고 3교시는 다른 요일에 수업이 배정되어야 한다. 이를 위해 [($T_i + 1 = T_j$) ∧ ($D_i = D_j$) ∧ ($D_i \neq D_k$)]라는 제약족건을 추가한다. 또한 강의실은 모두 동일해야 하므로 [($R_i = R_j$) ∧ ($R_i = R_k$)]라는 제약조건을 추가한다.

• C9.주 3일 이상 배정 : 어떤 한 전임교수의 수업들에 대한 변수 인덱스가 1, 2, 3, ..., k라고 가정하자. 주 5일 중 3일 이상 수업이 배정되어야 한다는 것은 요일을 의미하는 변수인 D_1, D_2, ..., D_k의 값들의 종류가 3개 이상이라는 의미이다. IBM ILOG CP에서 제공하는 IloCountDifferent 함수는 변수들의 집합에서 서로 다른 값의 개수와 관련된 제약을 추가하기 위해 사용될 수 있다. 즉, *DArray*가 해당 전임교수의 수업들에 대한 변수 D_i들을 포함하는 배열이라고 할 때, 본 제약조건은 [IloCountDifferent(*DArray*) ≥ 3]과 같이 표현될 수 있다.

• C10.최대 연강 시간 이내 배정 : 어떤 교수 P의 최대 연강 가능 시간이 4시간이라고 가정하자. 이 교수의 수업 시간이 4시간을 초과하는 연강을 배제하는 제약조건을 만드는 기본적인 아이디어는 교수 P의 수업 시간들 중 4시간을 초과하는 연강이 발생될 가능성이 있는 시간들의 조합에 대해 해당 시간들이 연강이 될 수 없도록 만드는 제약조건을 추가하는 것이다. 예를 들어, 교수 P의 담당 교과목이 총 3개로 각각 3시간, 3시간, 2시간으로 구성되어 있다면, 제약조건 C6에 의해 4시간을 초과하는 연강이 가능한 경우는 다음과 같이 총 3가지이다.

  ‣ (교과목1-2시간, 교과목2-2시간, 교과목3-2시간)
  ‣ (교과목1-2시간, 교과목2-1시간, 교과목3-2시간)
  ‣ (교과목1-1시간, 교과목2-2시간, 교과목3-2시간)

이 3가지 경우 각각에 대해 연강이 될 수 없다는 제약조건을 추가하면 된다. 여기서는 첫 번째 경우에 대한 제약조건 표현 방법을 설명한다. 먼저 (교과목1-2시간, 교과목2-2시간, 교과목3-2시간)에 포함된 각 교시에 대한 T_i 변수들을 포함하는 배열을 *TArray*라 하고, 해당 교시에 대한 D_i 변수들을 포함하는 배열을 *DArray*라 가정하자. 해당 수업 시간들이 연강이 되기 위한 기본 전제는 모든 수업 시간들이 동일한 요일에 배정되는 것이다. 이는 *DArray*에 포함된 모든 변수들이 한 가지 값만 가지게 됨을 의미하므로 [IloCountDifferent(*DArray*) = 1]와 같이 표현될 수 있다. 이때 해당 수업 시간들이 연강이 되지 않기 위해서는 *TArray*에 포함된 변수들의 최대값과 최소값의 차이가 4를 초과하면 된다. 따라서 최종적으로 이에 대한 제약조건은 [IloIfThen(IloCountDifferent(*DArray*) = 1, IloMax(*TArray*) - IloMin(*TArray*) > 4)]와 같이 표현된다. 여기서 IloMax는 해당 배열에 포함된 변수들의 값 중 최대값을 반환하며, IloMin은 최소값을 반환하는 함수이다.

[그림 3]은 제약조건에 대한 실제 소스 코드의 예로서 “C9. 주 3일 이상 배정” 제약조건에 대한 소스 코드이며, 왼쪽 숫자는 라인 번호를 의미한다. 각 수업 시간의 요일은 변수 D에 의해 표현된다. 앞서 설명한 바와 같이 본 제약조건은 어떤 전임교수의 D 변수, 즉, 요일에 대해 서로 다른 요일의 개수가 3개 이상이면 만족된다. 따라서 먼저 3~5라인에서 var 변수를 통해 해당 교수의 D 변수들을 가리키게 된다. 그리고 7~9라인과 같이 IBM ILOG CP에서 제공하는 IloCountDifferent 함수를 사용하여 var 변수 내에 서로 다른 값의 개수가 3개 이상이라는 제약조건을 추가하면 된다. 7~9라인은 사실상 “model.add(3 <= IloCountDifferent(var)”와 같이 하나의 문장으로 표현이 가능하다. 다만 많은 제약조건을 구현할 때 조건식을 표현하기 위해 IloExpr 클래스를 사용하고 있기 때문에 예시로서 이를 표현하였다.

```
 1: for (int i = 0; i < prof.size(); i++) {
 2:   if (prof.at(i).pf == 0) {          // full-time professor
 3:     IloIntVarArray var(env, prof.at(i).time_count, 1, 5);
 4:     for (int k = 0; k < prof.at(i).time_count; k++)
 5:       model.add(var[k] == D[k + prof.at(i).start_index]);
 6:
 7:     IloExpr expr(env);
 8:     expr = IloCountDifferent(var);
 9:     model.add(3 <= expr);        // add a constraint
10:   }
11: }
```

Fig. 3. Source Code for C9

4. Implementation of Soft Constraints

선호 제약조건을 목적함수로 다루기 위해서는 먼저 선호 제약조건인 “C13.선호 시간 배정 최대화”와 “C14.교수 별 선호 시간 최대화”를 점수화할 필요가 있다. 본 연구에서는 앞서 설명한 바와 같이 각 교수 별로 선호도에 따라 각 수업 시간을 0~5점 척도로 점수화하였다. [그림 4]는 교수 별 선호 테이블의 예이다.

	Mon.	Tue.	Wed.	Thu.	Fri.
1	0	1	1	1	0
2	1	2	2	2	0
3	1	2	0	5	0
4	3	3	0	3	0
5	3	3	3	3	0
6	5	5	5	5	0
7	5	5	5	5	0
8	4	4	4	4	0
9	2	2	2	2	0

⇨

	Mon.	Tue.	Wed.	Thu.	Fri.
1	0	10	10	10	0
2	10	19	19	19	0
3	10	19	0	49	0
4	29	29	0	29	0
5	29	29	29	29	0
6	49	49	49	49	0
7	49	49	49	49	0
8	39	39	39	39	0
9	19	19	19	19	0

Coversion Formula : $f(x_i) = \text{round}((x_i / \sum x_i) * 1000)$

(a) Score table of professor P　　(b) Normalized score table of professor P

Fig. 4. An Example of Preference Score Table

[그림 4](a)는 특정 교수 P의 수업 시간 별 점수를 나타낸 것이다. 그런데 각 교수 별로 각 점수의 분포를 다르게 책정하면 이에 따라 교수 간 선호 시간 배정에 있어 불공평한 상황이 발생할 수 있다. 그렇다고 해서 각 교수 별로 각 점수의 개수를 고정하면 점수표 작성이 불편해질 수 있다. 따라서 본 연구에서는 각 교수 별로 각 점수 별 개수에 구애받지 않고 0~5점 척도

의 점수표를 작성한 후 이를 정규화된 점수표로 변환하는 방법을 사용하였다. 구체적으로는 [그림 4]의 환산식과 같이 정규값에 1000을 곱한 후 소수점 이하 첫째 자리에서 반올림한 값을 사용하였다. [그림 4](b)는 [그림 4](a)의 점수표에 대한 정규화된 점수표를 나타낸 것이다. 단, 각 교수 별 점수표에 있어서 배정 불가를 나타내는 0의 개수 또한 교수 간 불평등의 원인이 될 수 있으므로 각 교수 별 0의 개수는 12개로 고정하였다. 하지만 시간 강사의 개인 사정 등과 같이 불가피한 사정이 있는 경우에는 해당 교수의 점수표 작성 시 0의 개수에 제한을 두지 않음으로써 자연스럽게 선호도가 높은 시간으로 수업이 배정될 수 있도록 하였다.

제약 만족 문제에서 선호 제약조건은 목적함수로 표현될 수 있다. “C13.선호 시간 배정 최대화”는 배정 결과 선호도 점수의 합을 최대화하는 것으로 정의되며, “C14.교수 별 선호 시간 최대화” 역시 배정 결과 각 교수 별로 선호도 점수의 합을 최대화하는 것으로 정의된다. 그러나 특정 교수에 대한 선호도의 합을 최대화하는 경우 다른 특정 교수의 선호도의 합이 오히려 낮아지는 상황이 발생할 수 있다. 이 문제는 일종의 다목적 최적화 문제로서 각 교수 별로 최적의 해를 구하는 것은 매우 어려운 일이며, 그에 앞서 최적해의 기준을 정의하는 것조차 어렵다. 본 연구에서 이와 같은 선호 제약조건을 고려하기 위해 제안한 전체적인 반복적 개선 탐색 알고리즘은 [그림 5]와 같다.

```
Algorithm Constraint Programming for Timetabling Problem
    Solver : CP solver.
    Solution : Current solution.
Begin
    Add all hard constraints to Solver
    Solution = Solve the problem with Solver
    While stopping condition is not met Do
        Set Solution as a Starting Point for Solver
        Add a constraint [Sum of preference > Sum of
            preference in Solution]
        For each professor Do
            Add a constraint [Sum of preference ≥ Sum
                of preference in Solution]
        End For
        Solution = Solve the problem with Solver
    End While
    return Solution
End Begin
```

Fig. 5. Iterative Improvement Search Algorithm for the Timetabling Problem

기본적으로 제약 프로그래밍을 기반으로 반복적 개선 탐색을 수행하되 C13 및 C14 측면에서 현재해보다 더 좋은 해가 도출될 수 있도록 하였다. 먼저 제약 프로그래밍을 통해 필수 제약조건들을 모두 고려한 초기해를 생성한다. 그러고 나서 다음 두 가지의 제약조건을 추가한 후 또 다시 제약 프로그래밍을 통해 더 좋은 해를 도출하는 과정을 반복 수행한다. 첫 번째 제약조건은 “모든 교수들의 선호도의 합이 현재해의 선호도의 합보다 크다”는 제약조건이며, 두 번째 제약조건은 각 교수 별로 “선호도의 합이 현재해의 선호도의 합과 같거나 크다”는 제약조건이다. 이 제약조건들을 통해 현재해의 각 교수 별 시간표의 질을 저해하지 않으면서 전체적으로 더 좋은 해의 도출을 유도할 것으로 기대된다. 물론 두 번째 제약조건을 추가하는 경우 첫 번째 선호 제약조건인 “C13.선호 시간 배정 최대화” 측면에서는 해의 개선에 불리할 것으로 예상된다. 따라서 문제 및 데이터의 특성에 따라서는 두 번째 제약조건의 추가 여부를 선택적으로 결정할 수도 있을 것이다. 이에 대해서는 실험 결과를 통해 보다 자세히 설명한다.

IV. Experimental Results

실험을 통해 다음 두 가지 요인과 그 관련하여 효과를 살펴보았다. 첫 번째는 수업 시간 관련 변수인 *D*와 *T* 및 *TD*의 효과이다. 즉, 제약조건을 표현하기 위해 D와 T만을 사용할 수도 있고 *D*와 *T*뿐만 아니라 *TD*를 함께 사용할 수도 있다. 두 번째는 반복적 개선 탐색 과정에서 C13만을 고려한 경우와 C13과 C14를 모두 고려하는 경우이다. 이에 대한 실험 결과는 [표 2]와 같다. “handwork”는 수작업 결과로서 매학기 2~3명의 인원이 약 2일에 걸쳐 시간표를 작성한 결과이다. “C13”은 C14에 대한 제약조건은 고려하지 않은 경우이며, “C13,C14”는 C14의 제약조건까지 고려한 경우이다. 그리고 “D,T”는 변수 *D*와 *T*만 사용한 경우이며, “D,T,TD”는 *D*와 *T* 외에 *TD*를 함께 사용한 경우이다. 각 실험의 수행 시간은 3시간이며 각 수치는 모든 교수들의 선호도의 합을 의미한다. 참고로 각 학기 별로 가장 좋은 결과는 빨강 및 이탤릭체로 표기하였다.

Table 2. Experimental Results

Semester	handwork	C13		C13,C14	
		D,T	D,T,TD	D,T	D,T,TD
2016-1	457	***686***	664	**639**	631
2016-2	668	***906***	***906***	796	**813**
2017-1	550	701	***707***	651	**662**

[표 2]를 통해 알 수 있는 가장 중요한 결과는 모든 경우에 있어서 수작업 결과보다 훨씬 좋다는 것이다. 이를 통해 제약 프로그래밍 기반의 반복적 개선 탐색이 수업 시간표 작성을 위해 효과적임을 확인할 수 있다.

둘째, 수업 시간 변수와 관련하여 요일과 수업 시간을 결합한 변수인 *TD*를 추가로 사용할 때 더 좋은 해의 도출이 가능한 경우도 있고 오히려 성능이 떨어지는 경우도 있으나, 그 차이는 크지 않은 편이다. 즉, 변수 *TD*의 사용으로 인한 성능의 차이는 미미하다고 볼 수 있다. 원래 변수 *TD*의 목적은 성능의 향상보다는 프로그래밍의 편의를 위한 것이다. 따라서 필요에 따

라 *TD*를 활용함으로써 성능에 큰 영향을 미치지 않으면서 보다 쉽게 제약조건을 표현할 수 있을 것으로 판단된다.

셋째, “C14.교수 별 선호 시간 최대화” 제약조건을 고려하지 않을 경우, 즉 전체적인 선호 시간 최대화 제약조건인 C13만 고려한 경우 보다 더 좋은 해의 도출이 가능함을 알 수 있다. 이는 예상한 바와 같은 결과로서 C14의 제약조건을 고려해야 하는 경우 모든 교수들에 대한 선호도의 합을 현재 이상으로 유지하면서 전체적으로 더 좋은 해를 찾아야 하기 때문에 그만큼 더 좋은 해의 탐색이 어려워지게 된다. 다시 말하면 제약조건 C14를 고려하는 경우 본 연구에서 제시한 반복적 개선 탐색은 일종의 언덕오르기 탐색 과정의 성격이 더 강해지며, 이에 따라 지역 최적해에서 머물게 되는 경향이 큰 것으로 해석된다.

그러나 C14의 제약조건을 고려하지 않을 경우 특정 교수에 대한 해의 질이 나빠지는 경우를 배제할 수 없다. [그림 6]은 2016년 1학기 데이터를 대상으로 제약조건 C14를 고려하지 않은 경우 초기 100초 동안의 일부 교수들에 대한 선호도의 합의 변화를 나타낸 것이다. 시간대 별 세로축의 값은 해당 교수의 수업 배정에 따른 선호도 점수의 합을 해당 교수의 총 수업 시간으로 나눈 값을 나타낸 것이다. 예상한 바와 같이 평균값(Average)은 지속적으로 향상되고 있지만 각 교수 별로는 이전해보다 좋지 않은 결과가 도출되는 경우를 자주 확인할 수 있다. 이로 인해 교수에 따라서는 최종적으로 초기해보다 좋지 않은 해가 도출될 수도 있을 것으로 우려된다. 그러나 다행히 본 연구에서 활용한 데이터들의 경우 3시간을 수행한 후에는 제약조건 C14를 고려하지 않더라도 거의 모든 교수들의 해가 초기해보다는 더 좋은 것으로 나타났다. 전반적으로 전체 선호 시간의 합이 초기값보다 크게 좋아지고 있기 때문에 각 교수별로 탐색 도중에 바로 이전해보다는 좋지 않은 해가 도출된다 하더라도 결국 최종적으로는 초기해보다 더 좋은 해의 도출이 가능한 것으로 판단된다. 따라서 데이터에 따라 제약조건 C14를 적용하지 않은 결과와 적용한 결과를 비교 검토하여 최종 시간표를 결정할 수 있을 것으로 사료된다.

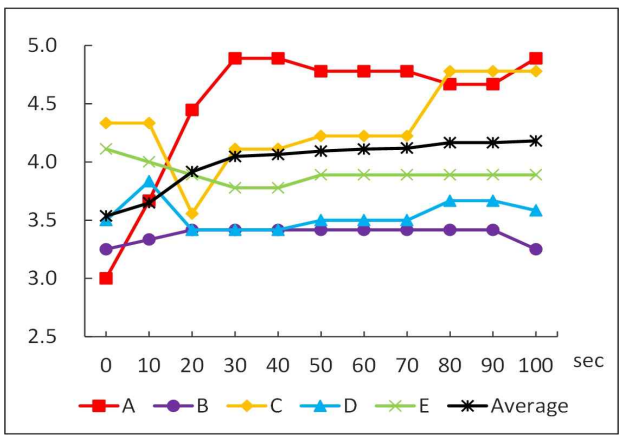

Fig. 6. Search Patterns for Each Professor with only C13

V. Conclusions

본 논문에서는 수업 시간표 작성을 위한 제약 프로그래밍의 적용 방안을 제시하였다. 대상 문제는 다양한 필수 제약조건과 선호 제약조건을 포함하고 있다. 본 논문에서는 제약 프로그래밍을 통해 필수 제약조건들을 표현하기 위한 방법을 제시하였다. 또한 선호 제약조건을 고려하기 위해 제약 프로그래밍을 활용하여 초기해를 생성한 후 반복적 개선 탐색 과정 내에서 또 다시 제약 프로그래밍을 활용함으로써 현재해보다 더 좋은 해의 탐색이 가능토록 하였다. 실험을 통해 보다 짧은 시간 내에 기존의 수작업 결과보다 더 좋은 해의 도출이 가능함을 확인하였다.

향후로 수업 시간표 작성을 위한 제약 프로그래밍의 성능 향상을 위해 다양한 연구가 가능할 것으로 판단된다. 일반적으로 제약 프로그래밍에서 변수와 변수의 값을 지정하는 순서가 탐색 성능에 영향을 미치는 것으로 알려져 있다. 따라서 이에 대한 추가 연구가 필요할 것으로 판단된다. 그리고 본 연구에서는 제약 프로그래밍만을 사용했지만 다른 최적화 기법을 사용하는 경우 또는 제약 프로그래밍과 다른 최적화 기법을 결합하는 경우에 대한 연구를 통해 수업 시간표 작성을 위한 성능 향상을 모색함과 동시에 각 기법의 장단점을 파악할 필요가 있다. 또한 필수 제약조건들을 모두 만족하는 해가 존재하지 않는 경우를 대비하여 필수 제약조건을 선호 제약조건으로 전환하여 문제를 해결하는 방법에 대한 추가 연구가 필요하다.

REFERENCES

[1] H. Babaei, J. Karimpour, and A. Hadidi, "A Survey of Approaches for University Course Timetabling Problem," Computers & Industrial Engineering, Vol. 86, pp.43-59, Aug. 2015.

[2] T. B. Cooper, and J. H. Kingston, "The Complexity of Timetable Construction Problems," Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT 1995), LNCS 1153, pp.281-295, 1996.

[3] C. C. Gotlieb, "The Construction of Class-teacher Timetables," Proceedings of IFIP Congress, pp.73-77, 1962.

[4] S Daskalaki, T. Birbas, and E. Housos, "An Integer Programming Formulation for a Case Study in University Timetabling," European Journal of Operational Research, Vol. 153, No. 1, pp.117-135, Feb. 2004.

[5] E. K. Burke, J. Mareček, A. J. Parkes, and H. Rudová, "A Branch-and-cut Procedure for the Udine Course

Timetabling Problem," Annals of Operations Research, Vol. 194, No. 1, pp.71-87, April 2012.
[6] S. Yang, and S. N. Jat, "Genetic Algorithms with Guided and Local Search Strategies for University Course timetabling," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 41, No. 1, pp.93-106, Jan. 2011.
[7] Z. Lü, and J. K. Hao, "Adaptive Tabu Search for Course Timetabling," European Journal of Operational Research, Vol. 200, No. 1, pp.235-244, Jan. 2010.
[8] C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl, "Solving the Post Enrolment Course Timetabling Problem by Ant Colony Optimization," Annals of Operations Research, Vol. 194, No. 1, pp.325-339. April 2012.
[9] R. Lewis, "A Survey of Metaheuristic-based Techniques for University Timetabling Problems," OR Spectrum, Vol. 30, No. 1, pp.167-190, Jan. 2008.
[10] R. Dechter, "Constraint Processing," Morgan Kaufmann, 2003.
[11] L. Zhang, and S. Lau, "Constructing University Timetable using Constraint Satisfaction Programming Approach," IEEE Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, No. 2, pp.55-60, IEEE. Nov. 2005.
[12] C. S. Kim, J. Hwang, G. Yoo, H. Lee, and J. Yoon, "Automatic Generation of Lecture Timetable Using Constraint Programming," Proceedings of the 2015 KIISE Symposium on Ubiquitous Computing and Web Information Technology, pp.137-140, 2015.
[13] T. A. Duong, and K. H. Lam, "Combining Constraint Programming and Simulated Annealing on University Exam Timetabling," Proceedings of RIVF Conference, pp.205-210, Feb. 2004
[14] L. T. Merlot, N. Boland, B. D. Hughes, and P. J. Stuckey, "A Hybrid Algorithm for the Examination Timetabling Problem," Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), LNCS 2740, pp.207-231, 2002.
[15] C. Valouxis, and E. Housos, "Constraint Programming Approach for School Timetabling," Computers & Operations Research, Vol. 30, No. 10, pp.1555-1572, Sep. 2003.
[16] C. S. Kim, and J. Hwang, "A Constraint Programming Model for Lecture Timetable Optimization," Proceedings of the Korean Society of Computer Information Conference, Vol. 25, No. 1, pp.13-14, 2017.
[17] "ILOG CPLEX Optimization Studio: CP Optimizer User's Manual," V12.6.2, IBM Corporation, 2014.

Authors

Chun-sik Kim received the B.S. degree in Computer Engineering from Kumoh National Institute of Technology, Korea, in 2015. He is currently a master student in Kumoh National Institute of Technology. He is interested in artificial intelligence, combinatorial optimization, constraint programming.

Junha Hwang received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Pusan National University, Korea, in 1995, 1997 and 2002, respectively. Dr. Hwang joined the faculty of the Department of Computer Engineering at Kumoh National Institute of Technology, Gumi, Korea, in 2002. He is currently a Professor in the Department of Computer Engineering, Kumoh National Institute of Technology. He is interested in artificial intelligence, combinatorial optimization, machine learning, and programming language.