Journal of The Korea Society of Computer and Information
Vol. 24 No. 4, pp. 27-34, April 2019

www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.04.027

A Constraint Programming-based Automated Course Timetabling System

Junha Hwang*

Abstract

The course timetabling problem is a kind of very complex combinatorial optimization problems, which is known as an NP-complete problem. Sometimes a given course timetabling problem can be accompanied by many constraints. At this time, even if only one constraint is violated, it can be an infeasible timetable. Therefore, it is very difficult to make an automated course timetabling system for a complex real-world course timetabling problem. This paper introduces an automated course timetabling system using constraint programming. The target problem has 26 constraints in total, and they are expressed as 24 constraints and an objective function in constraint programming. Currently, we are making a timetable through this system and applying the result to the actual class. Members' satisfaction is also much higher than manual results. We expect this paper can be a guide for making an automated course timetabling system.

▸Keyword: Timetabling, Course Timetabling Problem, Constraint Programming

I. Introduction

수업 시간표 작성 문제는 각 교과목의 수업 시간과 강의실을 결정하는 문제로서 교수, 학년, 강의실 등과 관련된 다양한 제약조건을 포함할 수 있다. 수업 시간표 작성 문제에서 제약조건은 필수 제약조건(hard constraint)와 선호 제약조건(soft constraint)로 나누어진다. 필수 제약조건은 반드시 준수해야 하는 제약조건이며 선호 제약조건은 가능한 준수하면 더 좋은 제약조건이다. 일반적으로 선호 제약조건은 목적함수로 표현되며 필수 제약조건은 제약조건으로 표현된다. 따라서 수업 시간표 작성 문제는 목적함수와 제약조건들로 이루어진 제약 만족 최적화 문제로 정의될 수 있다[1].

지금까지 수업 시간표 작성 문제를 풀기 위해 유전 알고리즘, 타부 탐색, 시뮬레이티드 어닐링, 제약 프로그래밍 등을 활용한 다양한 최적화 기법들이 제시되어 왔다[2]. 특히 2002년과 2007년에는 수업 시간표 작성 관련 국제 경진대회인 ITC-2002와 ITC-2007이 개최되었으며[3, 4], 많은 연구들이 해당 대회의 데이터를 토대로 이루어져 왔다[5, 6, 7]. ITC-2002 데이터는 3개의 필수 제약조건과 3개의 선호 제약조건을 포함하며, ITC-2007 데이터는 4개의 필수 제약조건과 4개의 선호 제약조건을 포함한다. 그러나 실제 교육 현장의 수업 시간표 작성 문제에서 요구하는 제약조건은 학교 또는 학과마다 천차만별이며, 경우에 따라 매우 복잡한 제약조건을 포함할 수 있어 자동화된 시스템의 구현을 어렵게 만든다. 예를 들어 본 논문의 대상 문제만 하더라도 총 26개의 제약조건을 포함하고 있다. 따라서 엄격히 얘기하면 지금까지의 많은 연구들은 가상의 데이터를 기반으로 기법들 간의 성능을 비교하기 위한 연구였다고 할 수 있다. 이로 인해 실세계 문제에 대한 적용 사례가 드문 편이다.

실세계 데이터를 대상으로 한 경우라 하더라도 실제 교육 현장에서의 활용 여부에 대한 언급이 전무하며, 나아가 기존 수작업 결과 대비 구성원들의 만족도에 대한 비교 또한 전무한 상황이다. 본 연구와 같이 실세계 데이터를 대상으로 한 연구로는 [8], [9]가 있다. [8]은 9개의 필수 제약조건과 10개의 선호 제약조건을 포함하는 수업 시간표 작성 문제를 대상으로 유전 알고리즘을 적용하였으며, [9]는 7개의 필수 제약조건과 2개의 선호 제약조건을 포함하는 문제를 대상으로 정수계획법을 적용하였다. 그러나 [8]에서는 수작업 결과 대비 정량적인 차이점

• First Author: Junha Hwang, Corresponding Author: Junha Hwang
*Junha Hwang (jhhwang@kumoh.ac.kr), Dept. of Computer Engineering, Kumoh National Institute of Technology
• Received: 2019. 03. 20, Revised: 2019. 04. 12, Accepted: 2019. 04. 17.
• This research was supported by Kumoh National Institute of Technology(2018-104-139).

만 기술하였을 뿐이며 실제 적용 여부 및 구성원들의 만족도에 대한 언급은 없었다. [9]에서는 교수 만족도 향상에 대한 가능성을 언급하고 있으나 이에 대한 분석 결과 및 실제 적용 여부에 대해서는 언급하지 않았다.

기존 연구 [10]에서는 본 논문과 동일한 수업 시간표 작성 문제를 대상으로 하였다. 해당 연구에서는 총 14개의 제약조건을 해결하기 위한 방안을 제시하였으나 실제 현장에 적용하기에는 부족한 점이 있었다. 본 연구는 기존 연구 [10]을 토대로 하고 있다. 먼저 제약 프로그래밍(constraint programming)을 사용하여 필수 제약조건을 만족하는 해를 도출하며, 또 다시 제약 프로그래밍을 반복적으로 적용함으로써 선호 제약조건 측면에서 지속적으로 더 나은 해를 탐색하게 된다.

본 연구에서는 실제 적용이 가능한 수업 시간표를 작성하기 위해 총 26개의 제약조건을 제시하며, 제약 만족 최적화 문제의 틀 내에서 각 제약조건에 대한 구현 방안을 제시한다. 구현 결과는 2018년 1학기부터 2019년 1학기까지 3회에 걸쳐 적용되었다. 그 결과, 기존의 수작업 결과에 비해 구성원들의 만족도가 더 높은 수업 시간표를 도출할 수 있었다. 본 연구의 대상 문제는 분반 개념 등 기존의 수업 시간표 작성 문제에 비해 보다 복잡한 요소들을 포함하고 있다. 따라서 본 논문에서 제시한 방법은 다양한 수업 시간표 작성 문제를 해결하기 위한 지침이 될 수 있을 것으로 기대된다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 대상 문제에 대해 설명하며, 3장에서는 각 제약조건을 반영하여 수업 시간표를 작성하기 위한 방안에 대해 기술한다. 4장에서는 구현 결과 및 실험 결과를 제시하고 분석하며, 5장에서 결론 및 향후 과제에 대해 기술한다.

II. Problem Description

1. Course Timetabling Process

본 연구에서는 경북지역 4년제 대학 특정 학과의 수업 시간표 작성 문제를 대상으로 하고 있다. 학과의 수업 시간표를 작성하기 전에 교양교직과정부에서 교양 및 기초도구 교과목의 수업 시간표를 작성하며, 이를 참고하여 학과의 수업 시간표를 작성하게 된다. 학과 수업 시간표 작성 시에는 먼저 해당 학기에 개설되는 교과목 별 분반 수를 결정하고 각 분반 별 담당교수를 결정한다. 대상 학과의 경우 한 학년의 재학생 수가 100명을 초과한다. 따라서 하나의 교과목에 대해 1개 내지 4개의 수업이 개설될 수 있는데, 이를 각각 “분반”이라 한다. 개설 교과목/분반 및 담당교수가 결정되면 이 데이터를 기반으로 본 연구에서 대상으로 하는 수업 시간표를 작성하게 된다.

본 연구에서 대상으로 한 2018년 1학기, 2018년 2학기, 2019년 1학기의 교과목(Subject)/분반(Class) 개수와 총 수업 시간수(Total Hour) 및 교수 수는 [표 1]과 같다. 교수는 전임교수(Full-time)와 시간강사(Part-time)로 구분된다. 교과목에 따라 주당 수업시간은 1시간~4시간으로 구성된다. 매 학기 수업 배정이 가능한 강의실은 총 7개로 동일하며, 실습실 4개, 일반 강의실 2개, 대학원 강의실 1개를 포함한다.

Table 1. Number of Subject and Professor per Semester

Semester	Subject	Class	Total Hour	Professor	
				Full-time	Part-time
2018-1	27	71	196	15	5
2018-2	30	73	206	15	5
2019-1	27	69	191	14	5

2. Constraints

필수 제약조건은 다음과 같이 총 24개로 구성되어 있다.

- HC1.하나의 강의실에는 특정 시간에 하나의 분반만 배정될 수 있다.
- HC2.교양 또는 기초도구 수업이 이미 배정되어 있는 강의실의 해당 수업 시간에는 다른 수업을 배정할 수 없다.
- HC3.담당교수 별로 배정이 불가능 시간에는 해당 교수의 수업을 배정할 수 없다.
- HC4.분반 별로 요구하는 강의실 종류에 따라 실습실, 이론 강의실, 대학원실 중 하나로 배정한다.
- HC5.필요 시 특정 분반의 수업 시간을 고정할 수 있다. 이 경우 지정된 시간으로 배정한다.
- HC6.한 학생이 해당 학년의 모든 교과목을 신청할 수 있도록 보장되어야 한다.
- HC7.2시간 이상의 수업들은 2시간 단위로 나누어 배정하고 각 단위 별로 다른 요일에 배정해야 한다. 즉, 3시간 수업은 2시간과 1시간 단위로 배정하며, 4시간 수업은 2시간 단위로 배정한다.
- HC8.한 분반의 모든 수업 시간은 동일한 강의실로 배정한다.
- HC9.야간의 특정 시간에 고정되어 있는 수업을 제외한 모든 수업은 주간 시간에 배정한다.
- HC10.특정 교과목의 분반들은 모두 같은 시간에 배정한다.
- HC11.특정 교과목의 분반들은 모두 같은 강의실로 배정한다.
- HC12.전임교수의 분반은 학과 회의시간에 배정할 수 없다.
- HC13.전임교수는 1주일 중 3일 이상 배정해야 한다.
- HC14.교수 별로 최대 연강 시간을 지정할 수 있으며, 해당 시간 이내로 연강이 가능하다.
- HC15.대학원 수업은 금요일에 배정하지 않는다.
- HC16.교수 별로 점심시간 확보를 위해 매일 4교시 또는 5교시에는 해당 교수의 수업을 배정하지 않는다.
- HC17.학년 별로 점심시간 확보를 위해 매일 4교시 또는 5교시에는 해당 학년의 수업을 배정하지 않는다.
- HC18.교수 별로 1주일 중 최대 수업 일수를 지정할 수 있으며, 해당 일수 이내로 수업을 배정한다.

- HC19.교수 별로 하루 최대 수업 시간을 지정할 수 있으며, 매일 해당 시간 이내로 수업을 배정한다.
- HC20.특정 분반들 사이에는 선후행 관계를 준수해야 한다. 예를 들어, "교과목 A의 수업 시간은 교과목 B의 수업 시간보다 앞서야 한다"는 제약을 추가할 수 있다.
- HC21.한 교수가 특정 교과목의 여러 분반을 담당하는 경우 동일 단위 순으로 요일을 배정한다. 예를 들어, 한 교수가 수업 시간이 3시간인 교과목 2개 분반을 담당한다면 모두 2시간, 1시간 순으로 배정하거나 1시간, 2시간 순으로 배정한다.
- HC22.한 교수가 특정 교과목의 여러 분반을 담당하는 경우 동일한 한 단위가 다른 동일 단위보다 항상 앞서도록 배정한다. 예를 들어, 한 교수가 수업 시간이 3시간인 교과목 3개 분반을 담당한다면 모든 분반의 2시간 단위 수업 시간들을 1시간 단위 수업 시간들보다 앞서 배정하거나 1시간 단위 수업 시간들을 2시간 단위 수업 시간들보다 앞서 배정한다.
- HC23.특정 교과목의 분반 번호는 1, 2, 3과 같이 일련번호로 표시한다. 한 교수가 특정 교과목의 여러 분반을 담당하는 경우 각 단위 별로 일련번호가 빠른 분반의 수업 시간을 다른 분반의 수업 시간보다 앞서 배정한다.
- HC24.선택 교과목은 비선호 수업 시간인 1교시에 배정하지 않는다.

선호 제약조건은 다음과 같이 총 2개로 구성되어 있다.

- SC1.교수 별 선호 시간 배정을 최대화한다.
- SC2.한 교수가 특정 교과목의 여러 분반을 담당하는 경우 동일 단위 별로 같은 요일에 배정하는 경우를 최대화한다. 예를 들어, 한 교수가 수업 시간이 3시간인 교과목 2개 분반을 담당한다면 2시간 단위끼리 같은 요일에 배정되고 1시간 단위끼리 같은 요일에 배정되는 것을 선호한다.

이상과 같이 총 26개의 제약조건으로 구성되어 있지만, 필수 제약조건과 선호 제약조건의 구분은 모호할 수 있다. 엄격히 따질 경우 공간 또는 시간적인 물리적 제약에 의해 필수적으로 만족해야 하는 제약조건에는 HC1과 HC2만 포함되며, 현실적 여건을 고려할 때 HC3~HC6 또한 필수 제약조건에 포함될 수 있다. 이외에 HC7~HC24 역시 학교의 정책, 학과의 여건, 교수 또는 학생의 사정에 따라 준수하지 않으면 곤란하므로 필수 제약조건으로 포함시켰다. 다만 HC19~HC24는 준수하지 않아도 무방하지만 가급적 준수하는 것이 좋다. 이런 측면에서 보면 HC19~HC24는 오히려 선호 제약조건에 가깝지만 실험 결과에 의하면 해당 제약조건들을 모두 만족하는 해의 도출이 가능하므로 필수 제약조건에 포함시켰다. 이와 유사한 제약조건이 SC2이다. SC2 또한 가능하다면 필수 제약조건으로 다루고자 하였으나 이를 만족하는 해의 도출이 불가능하였다. 따라서 선호 제약조건으로 포함시켜 가급적 준수하는 경우를 최대화하도록 유도하였다. 마찬가지로 데이터에 따라 HC19~H24 또한 언제든지 선호 제약조건으로의 전환이 가능하다. 결과적으로 진정한 선호 제약조건은 SC1이 유일하다.

III. Constraint Programming Approach

1. Why Constraint Programming?

본 연구에서는 수업 시간표 작성 문제를 해결하기 위해 제약 프로그래밍을 사용한다. 제약 프로그래밍에서 주어진 문제는 변수, 각 변수의 도메인, 제약조건들로 표현된다. 도메인은 해당 변수가 가질 수 있는 값들을 의미하며 제약조건은 변수 자체 또는 변수들 간의 관계를 의미한다. 제약 프로그래밍의 목표는 제약조건을 모두 만족하는 각 변수의 값을 찾는 것이다. 이를 위해 각 변수의 값을 유효한 값들 중 하나로 배정해 나가는데, 이때 제약조건의 위배 여부를 보다 빠르게 알아내기 위해 forward checking, AC-3 등 다양한 제약 전파 알고리즘들이 동원될 수 있다[1].

본 논문의 대상 문제와 같이 목적함수를 포함하는 제약 만족 문제를 제약 만족 최적화 문제라 한다. 제약 프로그래밍에서는 기본적으로 branch and bound 방식을 활용하여 제약 만족 최적화 문제의 최적해를 도출한다[1]. 제약 프로그래밍에서는 제약조건을 만족하는 해를 찾기 위해 깊이 우선 탐색을 시도한다. 최적해를 찾기 위해서는 유효한 해를 도출한 후에도 깊이 우선 탐색을 계속 진행하게 되는데, branch and bound에 의해 현재 해보다 목적함수 값이 좋지 않은 방향으로의 분기는 중단하게 된다. 따라서 이론적으로는 탐색 초기에 더 좋은 해를 도출할수록 더 빨리 분기를 중단할 수 있게 되어 보다 빨리 최적해의 도출이 가능하게 된다.

본 연구에서 제약 프로그래밍을 동원한 주된 이유는 제약조건을 쉽게 표현할 수 있다는 것이다. 제약 프로그래밍에서는 제약조건을 어떻게 해결하느냐 보다는 제약조건이 무엇인지를 선언적으로 표현하는 데 보다 초점을 맞추게 된다. 예를 들어, 제약조건 HC1의 경우 서로 다른 특정 분반의 수업 시간 인덱스를 i, j라 할 때 [if Time(i) = Time(j) then Room(i) != Room(j)]와 같이 표현하면 된다. 여기서 Time(i)는 i에 대한 수업 시간 변수, Room(i)는 i에 대한 강의실 변수를 의미한다. 이와 같이 본 논문의 대상 문제와 같이 다양하고 복잡한 제약조건을 포함하는 문제의 경우 제약 프로그래밍을 활용하면 수정 및 확장이 매우 용이할 것으로 판단하였다. 많은 제약 프로그래밍 라이브러리들은 복잡한 제약조건을 쉽게 표현할 수 있는 방법을 제공하는 등 프로그램을 보다 쉽게 구현할 수 있도록 유용한 도구들을 제공하고 있다.

수업 시간표 작성을 위해 제약 프로그래밍 외에도 크게 정수 계획법과 지역 탐색 기반 기법들을 고려해 볼 수 있다. 정수 계획법은 최적해의 도출을 보장하지만 모든 제약조건들을 선형식으로 나타낼 수 있어야만 적용이 가능하다는 단점이 있다[9]. 반면에 제약

프로그래밍은 제약조건을 별도의 변환없이 그대로 표현할 수 있기 때문에 보다 쉽게 적용이 가능하다. 지역 탐색 기반 기법들은 유전 알고리즘, 타부 탐색, 시뮬레이티드 어닐링, 개미 시스템 등 다양한 휴리스틱 최적화 기법들을 포함한다[2]. 이 방법들은 제약조건들을 모두 선호 제약조건, 즉, 하나의 목적함수로 다루게 된다. 따라서 목적함수 측면에서 어느 정도 좋은 해의 도출이 가능할 수 있지만, 특정 제약조건 하나라도 만족되지 않는 경우 실행 불가능한 시간표가 될 수도 있다. 반면에 제약 프로그래밍은 해가 도출되기만 한다면 필수 제약조건의 준수를 보장한다. 또한 교육 현장의 수업 시간표 작성 문제는 제약조건이 언제든 추가, 삭제, 수정될 수 있으며, 프로그래밍 전문가가 아닌 경우에도 비교적 쉽게 다룰 수 있어야 한다. 이와 같은 이유로 제약 프로그래밍이 수업 시간표 작성 문제의 해결을 위해 가장 적합한 것으로 판단하였다.

2. CP Modeling for the Timetabling Problem

대상 문제를 위한 변수의 표현 방법 등 제약 프로그래밍 모델링은 기본적으로 기존 연구 [10]과 동일하다. [그림 1]과 같이 각 교수가 담당하는 각 교과목/분반 별 수업 시간이 하나의 변수로 표현되는데, 강의실 지정을 위한 변수 *R*과 수업 교시 지정을 위한 변수 *DT*가 있다. 변수 *R*은 0부터 6까지 총 7개의 배정 가능한 강의실을 도메인으로 가진다. 변수 *DT*는 0부터 69까지 1주일 중 배정 가능한 수업 교시를 도메인으로 가지는데, 하루 기준으로 주간 9교시 및 야간 5교시를 포함한다. *D*는 요일을 의미하는 변수이고 *T*는 하루 중 수업 교시를 의미하는 변수이다. 변수 *D*는 0부터 4까지 각 요일을 의미하는 값을 도메인으로 가지며, 변수 *T*는 0부터 13까지 하루 중 수업 교시를 도메인으로 갖는다. *D*와 *T*의 값은 *DT*의 값이 결정되면 자동으로 결정되며, 역으로 *DT*의 값은 *D*와 *T*의 값이 결정되면 자동으로 결정된다. 사실상 *DT*만 사용해도 모든 제약조건의 표현이 가능하며, 거꾸로 *DT* 대신 *D*와 *T*만 사용해도 모든 제약조건의 표현이 가능하다. 다만 둘 다 사용할 경우 제약조건에 따라 적절한 변수를 사용함으로써 보다 자연스럽게 표현이 가능하다. 실험에 의하면 둘 다 사용할 때와 둘 중 하나만 사용할 때의 성능 차이는 없는 것으로 확인되었다. *DT*와 *D*, *T* 변수 사이의 값의 자동 변환 관계는 식 (1)~(3)과 같은 부가적인 제약조건을 추가함으로써 해결된다.

	subject A			subject B class 1		subject B class 2		subject C class 1		subject C class 2	
R	2	2	2	0	0	0	0	4	4	4	4
DT	18	19	43	2	3	32	33	48	49	30	31
D	1	1	3	0	0	2	2	3	3	2	2
T	4	5	1	2	3	4	5	6	7	2	3
	professor P_1							professor P_2			

Fig. 1. An Example of Variable Expression

$$DT_i = 14 \times D_i + T_i \quad (1)$$

$$D_i = DT_i\text{를 14로 나눈 몫} \quad (2)$$

$$T_i = DT_i\text{를 14로 나눈 나머지} \quad (3)$$

3. Implementation of Hard Constraints

필수 제약조건들 중 비교적 복잡한 제약조건으로는 HC6, HC7, HC13, HC14, HC20, HC21이 있다. 이 중에서 HC7, HC13, HC14에 대한 구현 방안은 기존 연구 [10]과 동일하므로 본 논문에서는 HC6, HC20, HC21에 대한 구현 방법에 대해 자세히 설명한다.

① HC6.어떤 학년의 학생을 기준으로 볼 때 해당 학년의 모든 교과목을 신청할 수 있어야 한다. 예를 들어, 교과목 A의 분반 A-1이 교과목 B의 모든 분반과 겹친다면 분반 A-1을 신청하는 경우 교과목 B를 신청하지 못하게 된다. 분반이 1개인 교과목은 다른 교과목의 모든 분반들과 겹쳐서는 안 된다. 따라서 분반이 1개인 교과목은 이후 설명에서 제외한다.

[그림 2]와 같이 분반 수가 각각 2개, 3개, 3개, 4개인 교과목들이 있다고 가정하자. 분반이 2개인 교과목은 한 분반을 A그룹으로, 다른 한 분반을 B그룹으로 지정한다. 분반이 3개인 교과목은 각 분반을 A, B, C그룹으로 지정한다. 분반이 4개인 교과목은 2개의 분반을 A그룹으로, 다른 2개의 분반을 B그룹으로 지정한다. 그리고 [그림 2]의 점선과 같이 A그룹 분반들끼리, B그룹 분반들끼리 그리고 C그룹 분반들끼리 동시에 수강신청이 가능하도록 만든다. 단, C그룹 분반의 경우 분반이 2개 또는 4개인 교과목의 모든 분반을 동시에 신청할 수 있도록 함으로써 C그룹 분반을 신청한 학생들이 다른 교과목의 A그룹 또는 B그룹으로 골고루 배분될 수 있도록 한다.

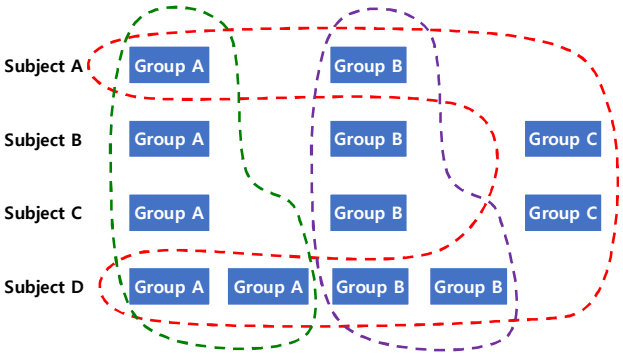

Fig. 2. Class Groups to Ensure Registering Simultaneously

이를 위해 동일한 그룹의 분반들끼리는 수업 시간이 서로 다르다는 제약조건을 추가하면 된다. 예를 들어, 수업 시간이 2시간인 분반 A-1과 분반 B-1의 변수 인덱스가 a_1, a_2이고, B-1의 변수 인덱스가 b_1, b_2라 하자. 분반 A-1과 분반 B-1의 수업 시간이 다르다는 제약조건은 [($DT(a_1)$!= $DT(b_1)$) and ($DT(a_1)$!= $DT(b_2)$) and ($DT(a_2)$!= $DT(b_1)$) and ($DT(a_2)$!= $DT(b_2)$)]와 같이 쉽게 표현이 가능하다.

② HC20.특정 분반들 사이에는 선후행 관계를 준수해야 한다. 대상 문제에서는 교과목 A가 있고 A를 기반으로 한 실습 과목 B가 있을 경우 B는 A 수업이 완료된 후에 시작될 것을 요

구하고 있다. 이를 위해서는 교과목 A의 모든 분반의 수업 시간이 교과목 B의 모든 분반의 수업 시간보다 앞서야 한다. 교과목 간 선후행 관계는 입력 데이터에 기록된다. 예를 들어, 교과목 A와 B의 수업 시간이 각각 2시간이고 분반 A-1의 변수 인덱스가 a_1, a_2, 분반 B-1의 변수 인덱스가 b_1, b_2라 가정하자. HC20은 [$DT(a_1) < DT(b_1)$ and $DT(a_1) < DT(b_2)$ and $DT(a_2) < DT(b_1)$ and $DT(a_2) < DT(b_2)$]와 같은 제약조건으로 표현될 수 있으며, 교과목 A와 교과목 B의 모든 분반에 대해 이상과 같은 제약조건을 추가하면 된다.

③ HC21.한 교수가 수업 시간이 3시간인 교과목 A의 3개 분반을 담당한다고 가정하자. 이 경우 3개 분반 모두 [그림 3](a)와 같이 2시간, 1시간 순으로 배정하거나 거꾸로 3개 분반 모두 1시간, 2시간 순으로 배정한다. 즉, [그림 3](b)와 같이 A-1과 A-2분반은 2시간, 1시간 순으로 배정하고 A-3분반은 1시간, 2시간 순으로 배정하지 않는다는 것이다. 물론 본 제약조건만으로는 [그림 3](a)와는 달리 A-1의 2시간과 A-2의 2시간의 순서가 서로 뒤바뀌거나 A-1의 1시간 단위 수업이 A-3의 2시간보다 앞서는 등 기대와는 다르게 나타날 수 있다. 이를 위해 제약조건 HC22와 HC23을 동시에 만족토록 하여 [그림 3](a)와 같은 시간표를 도출하게 된다.

A-1의 변수 시작 인덱스가 a_{11}이고 A-2의 변수 시작 인덱스가 a_{21}이며 A-3의 변수 시작 인덱스가 a_{31}이라고 가정하자. 참고로 수업 시간이 3시간인 분반인 경우 처음 2개의 변수는 2시간 단위를 의미하며 항상 함께 이동하게 된다. 따라서 각 분반의 1시간 단위의 인덱스는 각각 a_{11} + 2, a_{21} + 2, a_{31} + 2가 된다. 그러므로 HC21은 [{ ($D(a_{11}) < D(a_{11} + 2)$) and ($D(a_{21}) < D(a_{21} + 2)$) and ($D(a_{31}) < D(a_{31} + 2)$) } or { ($D(a_{11}) > D(a_{11} + 2)$) and ($D(a_{21}) > D(a_{21} + 2)$) and ($D(a_{31}) > D(a_{31} + 2)$) }]와 같이 표현될 수 있다.

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

(a) satisfied by HC21

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

(b) unsatisfied by HC21

Fig. 3. An Example of a Professor's Schedule

4. Implementation of Soft Constraints

목적함수로 다루어지는 선호 제약조건은 SC1과 SC2이다. 먼저 SC1에 대한 점수 obj_1을 계산하고 SC2에 대한 점수 obj_2를 계산한 후 식 (4)와 같이 하나의 목적함수로 표현한다. P는 교수의 집합을 의미하며, α 값은 obj_1 대비 obj_2의 상대적인 중요도를 의미한다. 즉, α 값이 커지면 SC2를 더 많이 만족하는 해를 찾기 위해 노력하게 된다.

$$\text{Maximize } z = \left(\sum_{j \in P} c_i \times obj_{1i}\right) + \alpha \times obj_2 \qquad (4)$$

SC1에 대한 점수인 obj_{1i}는 i번째 교수의 선호 시간 배정에 대한 점수를 의미한다. c_i는 1 이하의 양수로서 i번째 교수의 선호 시간 배정에 대한 가중치를 나타낸다. 이 값이 크면 클수록 해당 교수에 대해서는 상대적으로 더 선호하는 시간으로 배정하도록 유도하게 된다. 교수 별로 각 수업 시간에 대한 선호도는 [그림 4]와 같이 0~5점 척도의 선호 테이블로 나타낸다. 5점은 매우 선호, 4점은 선호, 3점은 보통, 2점은 비선호, 1점은 매우 비선호, 0점은 불가를 의미한다. 그런데 0점이 너무 많을 경우 교수 간 시간표의 질에 있어서 차이가 날 수 있을 뿐만 아니라 심지어는 유효한 시간표 자체의 도출이 불가능할 수 있다. 따라서 교수 당 0점의 개수는 12개 이내로 제한하였다. 단, 학과 및 교내 회의 등으로 인한 부득이한 경우에 한해 0점을 추가할 수 있도록 하였다. 물론 시간강사의 경우 특별한 제한이 없다. 각 교수 별 선호도 점수는 배정된 수업 시간에 대한 선호 테이블 점수를 합산하면 되고, 최종적으로 obj_1의 값은 모든 교수의 선호도 점수를 합산하면 된다.

	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

Fig. 4. An Example of Preference Score Table

SC2에 대한 점수인 obj_2는 각 교수 단위로 한 교과목의 여러 분반을 담당하는 경우 동일 수업 시간 단위 별로 같은 요일에 배정되는 정도를 점수화한 것이다. 다시 말하면, 수업 시간이 3시간인 교과목의 분반들이 있다면 2시간 단위끼리 같은 요일에 배정하고 1시간 단위끼리도 같은 요일에 배정하는 경우 점수가 높아지게 된다. 이에 대한 구현은 다음과 같다. 예를 들어, 한 교수가 수업 시간이 3시간인 교과목 A의 분반 A-1과 A-2를 담당한다고 가정하자. A-1과 A-2의 2시간 단위 변수 인덱스가 각각 i_1과 i_2이고, 1시간 단위 변수 인덱스가 각각 j_1과 j_2라면 obj_2의 점수는 [2 × ($D(i_1)$ == $D(i_2)$) + ($D(j_1)$ == $D(j_2)$)]와 같이 쉽게 표현된다. A-1과 A-2의 동일 수업 단위가 같은 요일에 배정되면 ($D(i_1)$ == $D(i_2)$)와 ($D(j_1)$ == $D(j_2)$)의 값이 각각 1이 된다. 인덱스 i_1과 i_2에 대해 2를 곱한 것은 해당 수업 단위가 2시간이기 때문이다. 이와 같은 방식으로 모든 교수에 대한 점수를 합산하면 된다. 본 구현 방식은 필수 제약조

건의 구현 방식과 매우 유사하다. 본 제약조건을 필수 제약조건으로 다루고자 한다면 각 쌍에 대한 제약에 해당하는 부분을 제약조건으로 추가하기만 하면 된다. 앞의 예의 경우 ($D(i_1)$ == $D(i_2)$))와 ($D(j_1)$ == $D(j_2)$))를 제약조건으로 추가하면 된다. 이와 같은 방식으로 다른 제약조건들의 경우에도 필요에 따라 필수 제약조건 또는 선호 제약조건으로 쉽게 전환할 수 있다.

제약 프로그래밍을 활용한 전반적인 해의 도출 과정은 [그림 5]와 같다. 먼저 대상 문제에 대한 변수와 각 변수의 도메인을 설정한다. 그러고 나서 필수 제약조건들을 추가하고 목적함수를 설정한 후 지정된 최대 수행 시간 동안 계속해서 더 나은 해를 도출하게 된다. 물론 최대 수행 시간이 경과하기 전에 최적해를 찾게 된다면 프로그램은 종료하게 된다.

- *CP* = Create a constraint programming solver
- Set variables and domains for *CP*
- Add all hard constraints to *CP*
- Set objective_function = Maximize(obj1 + α × obj2) to *CP*
- Set max_time = *t* // *t* : maximum execution time
- While current_time < max_time
  - Get next solution with *CP*
  - Store the solution

Fig. 5. Automated Timetabling Process

IV. Implementation Results

본 시스템은 Intel Core i7-4790 CPU 3.6GHz, 8GB RAM 및 Windows 7 64비트 운영체제 상에서 수행되었으며, 제약 프로그래밍 개발 라이브러리인 IBM ILOG CPLEX 12.7.1 및 DOcplex를 활용하여 Python 언어로 구현하였다[11].

[그림 6]은 2018-1, 2018-2, 2019-1 각각의 데이터에 대한 시간대 별 도출해의 목적함수 값을 그래프로 나타낸 것이다. 2018-1과 2019-1은 24시간 동안 수행하였으며, 2018-2는 초기해가 매우 늦게 도출된 관계로 30시간 동안 수행하였다.

2018-1과 2019-1은 모든 필수 제약조건을 만족하는 해의 도출이 가능하였다. 다만 2019-1의 경우 전임교수 수가 1명 적기 때문에 목적함수 값 또한 2018-1보다 작은 값이 도출되고 있다. 2018-1은 약 1.3시간 후에 목적함수 값이 998인 초기해가 도출되었으며 최종적으로 1099인 해가 도출되었다. 2019-1은 약 3시간 후에 목적함수 값이 922인 초기해가 도출되었고 최종해의 목적함수 값은 1072이다. 해의 도출 패턴을 보면 특정 시간대에 집중적으로 많은 해들이 도출된다는 것을 알 수 있다. 이는 한 번 해가 도출되면 그와 비슷한 형태의 해들이 여러 개 도출될 수 있기 때문인 것으로 해석된다. 그러나 탐색 후반으로 갈수록 지금까지 도출된 가장 좋은 해보다 좋은 해가 드물기 때문에 새로운 해의 도출이 어려워진다.

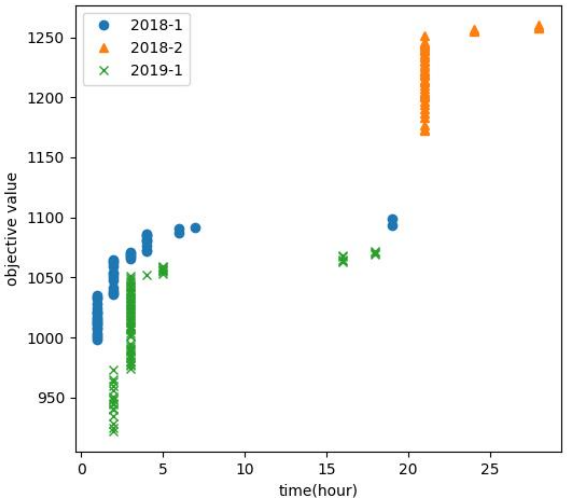

Fig. 6. Search Patterns for Each Data

2018-2의 경우 모든 필수 제약조건을 만족하는 해의 도출이 불가능하였다. 이에 따라 HC23과 HC24를 선호 제약조건으로 간주하고 SC2와 같이 목적함수로 추가하였다. 이로 인해 다른 데이터들보다 훨씬 큰 목적함수 값을 가진 해들이 도출되고 있다. 2018-2의 경우 약 21.3시간 후에 목적함수 값이 1172인 초기해가 도출되었으며, 최종해의 목적함수 값은 1260이다. 2018-2는 다른 데이터에 비해 분반수도 많고 전체 수업시간 또한 많은 편이다. 이로 인해 다른 데이터에 비해 모든 필수 제약조건을 만족하는 해의 도출이 매우 어려운 것으로 판단된다. 해의 도출 패턴은 다른 데이터와 마찬가지로 특정 시간대에 집중적으로 많은 해들이 도출되고 있음을 확인할 수 있다.

본 시스템을 통해 수업 시간표 작성이 완료되면 최종적으로 교수 별 시간표, 강의실 별 시간표, 학년 별 시간표가 엑셀 파일로 출력된다. [그림 7]과 [그림 8]은 2019-1의 결과의 예를 보인 것으로서 2019년 1학기 현재 해당 결과에 따라 수업을 진행하고 있다.

[그림 7]은 2019-1의 최종해에 대한 특정 교수의 시간표의 예이다. 각 수업 시간 별로 선호 점수, 교과목, 강의실이 기술되어 있으며, 교과목은 학년, 필수(R)/선택(O), 교과목명, 분반 순으로 나열되어 있다. 해당 교수가 요청한 수업일수는 4일이고 하루 최대 수업 시간은 4시간이며, 최대 연강 시간은 2시간으로 [그림 7]의 시간표는 이를 모두 만족하고 있다. 해당 교수는 또한 모든 교과목에 대해 강의실 “RA”로의 배정을 요청하였으며 이에 따라 배정되었다. 해당 교수는 교과목 3OA 3개 분반을 담당하고 있는데, 제약조건 H21에 따라 모든 분반이 각각 2시간, 1시간 순으로 배정되어 있고 제약조건 H22에 따라 모든 분반의 2시간 단위가 1시간 단위보다 앞서 배정되었으며, 제약조건 H23에 따라 1분반, 2분반, 3분반 순으로 배정되었다. 그리고 목요일 SOA1, SOA2, SOA3과 같이 목적함수 SC2에 따라 동일 과목의 같은 단위끼리는 가급적 같은 요일에 배정되고 있음을 알 수 있다. 물론 월요일 3OA1과 같이 다른 분반과 같은 요일에 배정되지 못하는 경우도 있다. 이는 해당 교수가 요청한

하루 4시간 이내 배정 제약에 의해 원칙적으로 같은 요일 배정이 불가능하기 때문이다. 가령 같은 요일에 배정이 가능하다 하더라도 다른 선호 제약조건을 고려하여 같은 요일에 배정되지 못할 수도 있다. 즉, SC2 측면에서의 목적함수 최대화는 단지 같은 요일 배정을 위해 최대한 노력함을 의미하는 것이다. 한편 목적함수 SC1 측면을 볼 때, 비록 목요일 8교시에 배정된 3OA3과 같이 선호 점수가 2점인 수업 시간에 배정되는 경우도 있지만 전반적으로 5점인 수업 시간에 가장 많이 배정되고 있으며 간혹 4점 또는 3점에 배정되고 있음을 확인할 수 있다.

Period / Day	1	2	3	4	5	6	7	8	9
Mon.	0	3 5 3OA1 RA		1	1	5	5 2RA1 RA	4	3
Tue.	0	5 5 3OA2 RA		1	1	5	5	4 3 3OA3 RA	
Wed.	3 5 2RA1 RA		0	0	1	5	5	4	3
Thur.	0	5 3OA1 RA	5 3OA2 RA	1	1	4	4	2 3OA3 RA	2
Fri.	0	0	0	0	0	0	0	0	0

Fig. 7. An Example of a Schedule for a Professor

[그림 8]은 2학년 시간표를 예로 보인 것으로서 각 수업 시간 별 교과목을 표기하였다. 총 8개 교과목의 24개 분반이 있으며, 구체적으로는 2RA 3개 분반, 2RB 3개 분반, 2RC 4개 분반, 2RD 3개 분반, 2RE 3개 분반, 2OA 3개 분반, 2OB 2개 분반, 2OC 3개 분반으로 구성되어 있다. 노란색 배경은 A그룹을 의미하고 초록색 배경은 B그룹, 보라색 배경은 C그룹을 의미한다. 제약조건 HC17에 의해 매일 점심 시간인 4교시 또는 5교시에는 수업이 배정되어 있지 않다. 그리고, 2RD는 2RE와 밀접한 관련이 있는 교과목으로 제약조건 HC20에 따라 2RD가 2RE보다 앞서고 있다. 제약조건 HC24에 의해 선택 교과목(O)들은 1교시에 배정되어 있지 않다. 단, 목요일 1교시에 배정된 2OA2는 담당교수의 요청에 의한 것이다. 제약조건 HC6에서는 모든 교과목의 신청을 보장토록 하고 있으며, 이를 위해 동일 그룹의 분반끼리는 중복을 피하도록 하였다. [그림 8]을 보면 모든 수업 교시에 있어서 동일한 그룹의 분반들이 존재하지 않음을 알 수 있다. 예를 들어, 금요일 3, 4교시의 경우 2RA, 2OC, 2RE 교과목 모두 서로 다른 분반, 즉, 서로 다른 그룹이 배정되어 있다. 수요일 8, 9교시의 경우 2OA 교과목이 3개 분반이므로 2OA2는 B그룹이고, 2RC 교과목이 4개 분반이므로 2RC2는 A그룹이 되어 서로 다른 그룹이 배정된 것이다.

모든 필수 제약조건을 만족하고 선호 제약조건 측면에서 최적화된 수업 시간표를 작성한다 하더라도 담당교수의 만족도가 높지 않다면 좋은 수업 시간표라 할 수 없다. 즉, 담당교수의 만족도는 본 시스템이 대상 문제에 대한 요구 사항을 얼마나 잘 파악하고 구현하였는지를 나타내는 가장 중요한 척도라 할 수 있다.

Period / Day	1	2	3	4	5	6	7	8	9
Mon.		2RD1			2OB1		2RA1	2OC1	
		2RC3			2RC-4		2RD2		
Tue.		2RC1			2OA1		2OB1	2RC2	2RC4
		2OB2			2OA3		2RC3		
Wed.	2RA1		2RA2	2RA3	2OA1	2OB2	2OC3	2OA2	
		2RD3			2OC2	2RB1		2RC2	
Thur.	2OA2	2RD1	2RC1		2RD2	2OC1			
		2RB2					2RB3		
Fri.	2RD3	2OA3	2RA3			2OC3		2RE3	
	2RA2		2OC2			2RE2			
			2RE1						

Fig. 8. An Example of a Schedule for the Second Grade

[표 2]는 전임 교수를 대상으로 한 설문조사 결과이다. 질문은 “2017년 시간표 대비 본인의 시간표에 대한 만족도”로서 매우 만족(5점), 만족(4점), 보통(3점), 불만족(2점), 매우 불만족(1점) 중 하나로 답하도록 하였다. 2017년 시간표는 수작업으로 작성한 것으로 3명의 교수 및 조교가 약 3일에 걸쳐 작성한 것이다. 즉, 본 질문은 수작업 대비 만족도를 의미한다. 참고로 2018년 1학기와 2018년 2학기에는 제약조건 HC23과 H24가 적용되기 전이었다. 따라서 2018-1과 2018-2는 HC23과 H24를 제외한 결과를 대상으로 한 것이며, 2019-1은 모든 제약조건을 반영한 결과를 대상으로 한 것이다. 점수를 보면 1학기 시간표에 대한 만족도가 2학기 시간표에 대한 만족도보다 월등히 높음을 알 수 있다. 이는 2학기 대상 교과목수, 분반수 및 전체 수업시간이 1학기보다 많아 문제 자체가 더 어렵기 때문인 것으로 분석된다. 이로 인해 초기해를 도출하기까지도 훨씬 많은 시간이 소요되었고 최종적으로 해의 질 또한 상대적으로 좋지 않은 것으로 판단된다. 그러나 2018-2의 3.8점 또한 수작업 결과 대비 만족도는 높은 편이라 할 수 있다. 한편 2019-1의 만족도가 2018-1보다 훨씬 높게 나타났다. 2019-1의 경우 2018-1에 비해 분반수도 2개가 더 적고 교수수도 1명이 적기 때문에 문제 자체가 다소 쉬울 것으로 예상된다. 뿐만 아니라 앞서 설명한 바와 같이 2019년 1학기에는 HC23과 H24를 포함한 모든 제약조건이 반영됨으로써 교수들의 만족도가 그만큼 향상된 것으로 분석된다.

Table 2. The Result of Professors’ Satisfaction

Semester	Number of Total Prof.	Number of Responders	Average Score
2018-1	15	11	4.2
2018-2	15	12	3.8
2019-1	14	12	4.8

V. Conclusions

본 논문에서는 제약 프로그래밍을 활용한 자동화된 수업 시간표 작성 시스템을 제시하였으며, 현재 학교 현장에 적용된 결과를 함께 제시하였다. 대상 문제를 해결하기 위해 24개의 필수 제약조건과 2개의 선호 제약조건을 반영하였다. 그 결과 수작업으로 작성한 결과보다 훨씬 만족도가 높은 수업 시간표를 작성할 수 있었다. 많은 제약조건을 동반하는 수업 시간표 문제를 해결하기 위해 제약 프로그래밍은 매우 유용한 도구가 될 수 있으며, 본 연구의 결과는 다양한 수업 시간표 작성 문제를 해결하기 위한 길잡이 역할을 할 수 있을 것으로 판단된다.

향후 개선 과제로는 다음과 같이 세 가지를 고려할 수 있다. 첫째, 최근 대학 교육 현장에서는 집중 이수제, 융합형 교육 등 다양한 형태의 수업을 요구하고 있으며, 이에 따라 수업 시간 또한 다양한 형태로 변화될 가능성이 있다. 따라서 다양한 형태의 수업을 수용할 수 있도록 보다 유연한 시스템으로 개선하기 위한 연구가 필요하다. 둘째, 2018-2 데이터의 경우 초기해 도출까지 과도한 시간이 소요되고 있으므로 탐색 성능 향상을 위한 추가 연구가 필요하다. 이를 위해 제약 조건의 추가 및 변경을 고려할 수 있으며, 제약 프로그래밍에 있어서 변수 지정 순서 및 변수값 지정 순서 등을 함께 고려해 볼 필요가 있다. 셋째, 실험 데이터에 따라서는 실행 가능해가 도출되기 위해서는 필수 제약조건 중 일부를 선호 제약조건으로 처리해야 한다. 그러나 아직까지는 선호 제약조건으로 처리할 필수 제약조건을 자동으로 선정하는 것은 불가능하다. 따라서 이에 대한 자동화 방안에 대해서도 검토해 볼 필요가 있다.

REFERENCES

[1] E. Tsang, "*Foundations of Constraint Satisfaction*," Academic Press, London, 1993.

[2] 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.

[3] B. Paechter, "International Timetabling Competition," *http://sferics.idsia.ch/Files/ttcomp2002/*, 2001.

[4] L. di Gaspero, A. Schaerf, and B. McCollum, "The Second International Timetabling Competition: Curriculum-based Course Timetabling (Track 3)," Technical Report, Queen's University, Aug. 2007.

[5] S. Abdullah, and H. Turabieh, "On the Use of Multi Neighbourhood Structures within a Tabu-based Memetic Approach to University Timetabling Problems," Information Sciences, Vol. 191, pp.146-168, May 2012.

[6] R. Bellio, S. Ceschia, L. di Gaspero, A. Schaerf, and T. Urli, "Feature-based Tuning of Simulated Annealing Applied to the Curriculum-based Course Timetabling Problem," Computers & Operations Research, Vol. 65, pp.83-92, Jan. 2016.

[7] A. Bettinelli, V. Cacchiani, R Roberti, and P. Toth. "An Overview of Curriculum-based Course Timetabling," TOP, Vol. 23, No. 2, pp.2-37, July 2015.

[8] H. H. Kim, and Y. N. Choi, "An University Timetabling System Using Genetic Algorithm," Journal of Science & Culture, Vol. 2, No. 1, pp.333-342, Feb. 2005.

[9] N. Ahn, "Mathematical Modeling Approach for Classroom Assignment Problem," Journal of the Korea Academia-Industrial cooperation Society, Vol. 18, No. 10, pp.580-587, 2017.

[10] C. S. Kim, and J. Hwang, "Constraint Programming Approach for a Course Timetabling Problem," Journal of The Korea Society of Computer and Information, Vol. 22, No. 9, pp.9-16, Sep. 2017.

[11] "IBM Decision Optimization CPLEX Modeling for Python", V2.9, IBM Corporation, 2019.

Authors

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.