International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184 Volume 4, Number 3(June2015), pp.83-95

http://www.meacse.org/ijcar

University Course Timetable using Constraint Satisfaction and Optimization

Tarek El-Sakka[*, **]

[*]IT Department, Community College, University of Sharjah, Sharjah, UAE

[**]Central Laboratory for Agricultural Expert Systems (CLAES), Cairo, Egypt

Abstract

Timetable problem is a well-known multidimensional, constraint assignment problem that focuses in the assignment of courses to faculty members in classrooms within limited time slots. Hence, it is a challenging time-consuming problem facing universities and it belongs to the NP-hard class of problems. In particular, universities regularly need an optimal solution for the course timetable problem. However, a manual solution to this problem by considering all constraints usually requires a long time and hard work to offer proper, optimized solution. Specifically, timetabling problem scan be modeled as Constraint satisfaction problems (CSPs), which are combinatorial in nature. Particularly, a variety of approaches are used to investigate CSPs, such as constraint logic programming (CLP) that combines the declaratively of logic programming with the efficiency of methods from operations research and artificial intelligence. This paper introduces a model that applies CLP to the timetabling as a CSP on the use of an Optimization Programming Language (OPL). The proposed model is tested against real data obtained from the Community College (CC), at University of Sharjah (UoS) that has a so sophisticated timetable with much interlaced, limited resources and limited time slots.

Keywords: Timetable, University Timetable, Course Scheduling, Constraint satisfaction problems, Constraint Programming, Optimization Programming Language, CP Optimizer, IBM ILOG

Introduction

Indeed, timetabling is a multidimensional assignment problem, which needs to be solved regularly at educational institutions. It is the assignment of courses to faculty members and the assignment of these courses to classroom and time slots [1] in a way that makes optimal use of the available resources [2]. Such a timetable must satisfy certain constraints such as no single teacher teaches more than one class at the same time, no single room is allocated for more than one class at the same time, and so on. Further, it may try to achieve certain objectives such as maximum utilization of classrooms, assigning teachers to his or her preferred courses, etc. Practically, a typical university timetabling problem may comprise thousands of courses, thousands of students, hundreds of instructors, hundreds of classrooms and other resources.

Moreover, timetabling problem has been classified as NP-hard optimization problem (i.e., no polynomial time algorithm is known to solve the problem) [3], meaning that if all combinations were to be examined, the time to solution for reasonable problems would rise dramatically. Therefore, in order to find optimal solutions to such problem, it is necessary to consider all possible solutions to choose the best one that satisfy a wide range of constraints, preferences, and participants and it must be solved in reasonable time.

Although, the university timetabling (UTT) is a major, regular and complex administrative activity in most academic institutions [3], only a few organizations possess reliable automated timetable solvers, and fewer still possess solvers that require no manual intervention. However, most institutions employ the knowledge and experience of expert personnel with regard to the production of good timetable that satisfy all given requirements. Categorically, UTT problems can be classified into two main categories: course timetabling (CTT) problems and examination timetabling (ETT) problems, each with its own sets of constraints and requirements. The focus of this paper is on the CTT.Therefore, lots of research has been invested in order to

provide automated support for solving a real-world timetabling problem. Contributions come from the fields of operations research (e.g., graph coloring, network flow techniques) and artificial intelligence (e.g., simulated annealing, tabu search, genetic algorithms and constraint satisfaction [4]). Barták et al. [5] surveyed the main definitions and techniques of constraint satisfaction, planning and scheduling from the Artificial Intelligence point of view.

This paper refers to methods from constraint satisfaction and how they were presented and developed using Constraint Programming (CP)/Constraint Logic Programming (CLP). Concerning CLP (an emerging Artificial Intelligence field), it is used for solving Constraint satisfaction problems (CSPs) and combinatorial optimizations [6]. Particularly, CLP combines the declaratively of logic programming with the efficiency of methods from operations research and artificial intelligence [4]. Its major advantage is its ability to give a precise declarative description of the problem in terms of constraints [7]. Although, in many cases, the definition of the problem's constraints is ambiguous, and it generally requires many refinements and interactions with the user, CLP exploits the constraint satisfaction structure of the problem.

Definitely, CSPs are combinatorial in nature [8]. For many categories of CSPs, an efficient algorithm is unlikely to exist (these problems are NP-complete). Thus, an algorithm that guarantees to find a solution that satisfies all constraints is enumerative and hence has an exponential time requirement in the worst case. In practice, it may be sufficient to find a solution at a reasonable computational expense, which satisfies most of the constraints. If all or as many constraints as possible are satisfied, the solution is exact; otherwise, it is approximate.

In reality, CTT problems have been seemed more suitable to be solved with constructive methods implemented in general-purpose CLP languages. Consequently, applying classical methods from constraint satisfaction requires modeling the timetabling problem as a CSP, i.e., a set of variables each associated with a domain of values it can take on, and a set of constraints among the variables [9]. In contrast, the timetabling problem is modeled as a constraint optimization problem (COP) and addressed with CLP languages.

Considering the timetable problem at CC, it is a very complex timetabling problem with much, interlaced resources as the college has five branches distributed over five faraway cities. Each branch/campus has two shifts for students to study in CC, therefore the time slots for timetabling is so limited. In addition, students are enrolled in many specialized programs with different curricula that need many courses to be set in the CTT. Most lecturers have to teach courses at different branches. This paper introduces a model using the OPL that solved by the CP solver. This model is developed with the IBM ILOG CPLEX 12.6 Studio. The model is tested against real data obtained from the CC.

The next section of this paper provides a review of some relevant work on solving the timetable problem. Section 3 provides the description of the timetabling problem at CC. Section 4 presents the proposed model to solve the CC timetabling problem. Section 5 presents the implementation of the CTT model. Section 6 presents a discussion of the results and findings. Finally, section 7 concludes the paper with a summary of the work was done and possible directions for future work.

RELATED WORK

Regarding timetabling problems, many works have been done to accomplish a good solution to their using different operation research (OR) and artificial intelligence (AI) approaches since fiftieth. A large number of problems in AI and other areas of computer science can be viewed as special cases of the CSP [10]. In addition, many problems in OR fall within the general framework of CSP [8]. The development of effective solution techniques for CSPs is an important research problem. Timetabling problems are considered as combinatorial problems in OR, which can be modeled as CSPs. Researchers in AI usually assume constraint satisfaction approaches as their preferred methods when undertaking such problems [8].A variety of approaches can be used to process the CSPs. For instance, LP techniques can be applied to find an exact solution. On the other hand, there are various approaches provide an approximate solution, including local search methods and neural networks as a special purpose technique used for solving CSPs.

There are many research works provide the usage of constraint-based approaches. In these methods, the events of a timetabling problem are modeled by a set of variables to which values have to be assigned subject to a number of constraints[11]. When the propagation of the assigned values leads to an infeasible solution, a

backtracking process enables the reassignment of value(s) until a solution is found that satisfies all of the constraints.

To begin with, Abdennadher and Marte [12]showed how to model the timetabling problem as a partial constraint satisfaction problem and gave a concise finite domain solver implemented with Constraint Handling Rules that allows for making soft constraints an active part of the problem solving process. They improved efficiency by reusing parts of the previous year's timetable. Zhang and Lau [13] developed a CSP model for a university timetabling problem. They investigated a sample case study problem and implemented an approach for constraint satisfaction programming using ILOG Scheduler and ILOG Solver. They used various goals in ILOG to investigate the performance of the CSP approach.

Further, Ojugo et al. [14]build constraint satisfaction models to search the space for the timetable scheduling state and satisfies all constraints and criteria that guarantee the reasoning process through an explicit structure that conveys data about the problem. They aimed to find a complete assignment that satisfies certain constraints to yield a valid schedule. In addition, they provided a study that surveyed NP-complete task of academic timetabling at the University of Benin, Nigeria and they adopted a rule-based expert system to yield an initial solution for the models. They showed that their models yielded a valid schedule for the University of Benin in Nigeria considering student preference, medium constraints of high priority.

In addition, Gu´eret et al. [15]showed how CP is well suited for solving timetabling problems and they presented different approaches used for solving timetabling problems of an institute in a university by using the CLP programming language CHIP. While, Legierski[16]dealt with the effectiveness of CP for timetabling problem. He used the CP language Mozart-Oz to express complex constraints. He also created complicated, custom-tailored distribution strategy for solving a real-world timetabling problem.Likewise, Zervoudakis and Stamatopoulos[17]presented a model for the CTT using theILog Solver, a constraint programming object-oriented C++ library. They showed how their model might be extended to cover the needs of a specific CTT.

Goltz et al. [18]developed methods, techniques and concepts for a combined interactive and automatic timetabling system of university courses using used CLP. They developed a timetabling system for the Charité Medical Faculty at the Humboldt University, Berlin. They showed how their system was flexible enough to take into account special user requirement to allow constraints to be modified easily. In addition, they provided an interactive user intervention facility to allowing user alter a timetable if no hard constraints are violated.Similarly, Fen, Deris and Hashim[19]investigated the constraint-based reasoning algorithm to solve a complex university course timetabling problems by searching for the best preference value base on the student capacity for each lecture. They have tested their work using a real world data for two higher education institutions.

In a like manner, Shue, Lin and Tsai [20]developed a university timetabling decision support system that considers both hard and soft constraints of the problem. They stated that their system has satisfied all hard constraints and partially met soft constraints. They modeled the problem as a Constraint Satisfactory Problem and adapted lexicographic optimization approach to implementing the solution procedure where each soft constraint is treated as an objective with a priority.Moreover, Gavanelli[21]modeled a university timetabling problem as a COP and addressed his model with the CLP language, ECLiPSe. His paper showed how the data was represented in ECLiPSe and what the phases to build a timetable were.

Additionally, Rudová and Murray [22]described and applied the development of an automated timetabling system for Purdue University by using an extension of constraint logic programming that allows for the weighted partial satisfaction of soft constraints. They implemented a soft constraint solver for the proposed solution approach that allows constraint propagation for hard constraints together with preference propagation for soft constraints. They presented and discussed the model and search methods applied to the solution of the large lecture room component.

In addition, Yeung et al. [23]reported their experience to automate the timetable scheduling process in their college by applying constraint logic programming. They built a system that consists of a core for scheduling as well as an X-window user interface. Their system was implemented in CHIP, which was able to produce a timetable for the whole faculty in just a few seconds. They showed that applying constraint logic programming is a practically viable means for timetable scheduling in a university.

At the same time, there are various approaches applied to the CTT problem including Linear Programming (LP). LP is widely used for the solution of timetabling problems, like Bakir and Cihan[24], Ribi and Konjicija[25], Wattanamano, Thongsanit and Hongsuwan[1] and Czibula et al. [26]. Other approaches have been recently applied to solve the CTT problem including heuristics and meta-heuristic, graph coloring, network flows, genetic algorithms and other OR and/or AI methods [27].

Problem Description

In fact, CTT is a tedious task for university administration that aims to arrange periods, modules, rooms and lecturers to all courses in an academic term. Such a CTT must satisfy the required constraints and certain objectives of optimally using the university resources, which is limited and valuable. A typical CTT may comprise thousands of courses, thousands of students, hundreds of instructors, hundreds of classrooms and other resources. CTT should place these courses that share resources, such as lecturers or classrooms, in a weekly calendar.

Indeed, in our real case study, CC has six different departments. Each department has different active programs with different, specialized curricula. All CC programs are sharing some courses called general education (GE) courses that are taught by different departments. Those GE courses are categorized as university requirements and college requirements categories. In particular, each course is the main module in CTT problem that defined as a class. Each class shares the available resources in the CC, e.g. teachers, rooms, or time slots. In addition, courses are composed of theory and/or practical sessions. To clarify, theory sessions are usually set as three credit hours lectures per week and they are held in classrooms. On the other hand, practical sessions are usually set as four credit hours per week. They are composed of two hours lecture in a classroom and two hours practical in a lab.

CC has five different branches geographically distributed over five faraway cities. Furthermore, each branch has two shifts. The first shit in the morning (from 8am to 14pm). While the other shift in the afternoon (from 15pm to 21pm) – for students who cannot study in the morning (e.g. employees). In addition, most lecturers of departments may teach classes at different branches at same working day. Hence, CC has a very complex timetabling problem because resources are interlaced, e.g. time slots, rooms and lecturers.Similarly, to the higher education institutes, CC regularly tries to build its optimal timetable every academic semester. CC departments should define some courses from their specialized programs to be set in a new CTT as classes. This CTT should allow each student enrolled in a CC program to set his/her timetable. At the same time, several constraints have to be considered during development of the CTT. Those constraints are set to use the available resources optimally at CC, e.g. no two lectures can take place at the same room at the same time, and no lecturer can teach more than one class at the same time. In addition, no lecturer can teach classes in more than one branch at the same day or the capacity of a room must not be less than the number of students attending a lecture. Furthermore, classes should be distributed on weekdays as one of the two days patterns. The first pattern is a three days in a week, Sunday-Tuesday-Thursday. The other pattern is two days per week, Sunday-Tuesday or Monday-Wednesday. There are many other constraints to be satisfied in the CTT at the CC.

Thus, we propose a model for CTT problem at the CC that is modeled with the IBM ILOG CP Optimizer, which provides a scheduling language supported by a robust and efficient automatic search [28]. This CP Optimizer uses the OPL as a scheduling language. IBM ILOG CP Optimizer is a component of the ILOG OPL-CPLEX Development System, which is a rapid development system for optimization models. It streamlines modeling with high-level data types and an algebraic language specifically designed for building optimization models [29].

In detail, the ILOG OPL-CPLEX Development System consists of the different components. The first is the OPL for developing optimization models. The second is an Integrated Development Environment (IDE) to execute and test optimization models. The third is a command line tool (oplrun) to execute models from the command line. The fourth is an application Programming Interfaces (APIs) to embed models into standalone applications. The last is the CPLEX engine that can solve the OPL model as CP model or MP model.

Therefore, the OPL is a modeling language that allows to state CP, LP, Mixed Integer Programming (MIP), and CSPs, as well as combinations of all of them [30]. In addition to the constraints of the model, a search heuristic is stated that is then executed by the CP Solver, which offers CP algorithms. OPL Studio is built on top of CP Solver allowing solving CSPs modeled in OPL. The CSP model is addressed with the CP language. Consequently, we define our model as a CP model that solved by the CPLEX engine. Practically, a CP model uses only discrete decision variables for which you must define a domain. Therefore, the product of all domain variables makes up the search space. Hence, the CP models are further characterized by specific constraints, expressions, and search.

Actually, there are many approaches to satisfaction problems. These algorithms seek solutions of a CSP or try to prove that no solution exists. Those approaches vary from consistency techniques to systematic search or combination of tree search algorithms and consistency techniques. The consistency techniques try to remove values that cannot be part of a solution. The systematic search methods explore systematically the whole search space. While the combination of both applies the consistency techniques, which prune some parts of the search tree and hence improve the efficiency of the tree search algorithm [31].

Similarly, the CP optimizer uses constructive search to build a solution by fixing decision variables to values. In addition, the CP optimizer uses other heuristics to improve search. In that case, there are three types of search available in CP Optimizer: “restart”, “depth-first” and “multipoint”. Particularly, we use the depth-first search type with the CP optimizer to get an optimal solution to the CTT problem. The depth-first search is a tree search algorithm such that each fixing, or instantiation, of a decision variable, can be thought of as a branch in a search tree [8]. Actually, the CP optimizer works on the subtree of one branch until it has found a solution or has proven that there is no solution.

Concerning CSP, it is defined (in [10, 8, 32, 33]) by a finite set of variables, each of which has a finite domain of values, and a set of constraints. Each constraint is defined over some subset of the original set of variables and restricts the values these variables can simultaneously take. We develop our model with the most common definition and syntax of CSP. A CSP is defined as three sets: X, D, and C, where:

- $X = \{x_1, \ldots, x_n\}$ is the set of variables called *domain variables*;
- $D = \{D_1, \ldots, D_n\}$ is the set of *domains*(the possible values for the corresponding variable);
- $C = \{C_1, \ldots, C_c\}$ is the set of constraints (relations defined on a subset of all variables).

Proposed Model of CTT

We have built our model for the CTT with the OPL model using the CSP definitions to be solved by the CP solver. Moreover, the OPL model uses an algebraic modeling system, which consists of several parts including the definition of the *model data*, the declaration of the *decision variables*, and the statement of *constraints* and *objective*. Initially, the *model data* in our CCT model are listed in Table 1.

TABLE 1. THE DOMAIN VARIABLES FOR THE MODEL DATA

Variable	Meaning
NumberOfDaysPerPeriod	Number of working days per week
DayDuration	Number of time slots per day
BreakUnitTime	The break duration for class
ClassRequirementSet	The educational program:{"*class*", "*discipline*", "*courseType*", *hours, repetition*}
TeacherDisciplineSet	Pair Set of preferred disciplines for a teacher
Room	Set of rooms in which courses are held
DedicatedRoomSet	Pair Set of dedicated rooms for a discipline. The other rooms support all disciplines.
MorningClass	Set of morning classes in a campus
EveningClass	Set of evening classes in a campus
PossibleDays	Set of possible day names
MorningStartHour	Starting time for morning classes

As an illustration, an example fora model data is shown below.

ClassRequirementSet = {
<"Law-80","Introduction to law", 3, 2>,
<"Law-80","Sources of Obligation", 3, 2>,
<"Law-80","Islamic", 3, 2>,

<"Law-80","Arabic", 3, 2>,
<"Law-80","Principles of Management", 3, 2>,
<"Law-80","IT", 4, 2>, . . . }

Secondly, our model set the values of the*domain variables* from the model data. These*domain variables* are listed in Table 2.

TABLE 2. THE DOMAIN VARIABLES

Class	Set of classes extracted from the educational program
Teacher	Set of teachers extracted from TeacherDisciplineSet
Discipline	Set of disciplines extracted from the educational program
MaxTime	Number of time slots in working days
Time	A time counter
PossibleTeacherDiscipline[x]	Array of teacher *x* teaches preferred disciplines
PossibleTeacherIds[d]	Array of teachers' Ids who teach a discipline *d*
PossibleRoom[d, x]	Array of discipline *d* held in a room *x*
PossibleRoomIds[d]	Array of room Ids that hold a discipline *d*
Section	For a given class, it is one course occurrence. {Class, discipline, Duration, repetition, id, SectionId}

Specifically, an example for a domain variable is shown below.

discipline: *{"Introduction to law", "Sources of Obligation", "Islamic", "Principles of Management", "IT" "Penal Law 1", "Rules of Obligation", "Comm. law and companies", "Administrative law", "Arabic", "CS", "Prog 1", "CG", "BCO", "ESP for IT 1", "IWD", "Net 1", "AWD", "Video", "Introduction to Psychology", "Principals of Statistics"*}

Thirdly, our model contains a set of constraintsthatthe CP solver verifies(to solve the CTT problem) over the set of variables. These constraints are illustrated below:

C_1: A teacher is required once at any time point
C_2: A room in a campus is required once at any time
C_3: A class follows one course at a time, so for a specific class a student can only attend one section at a time
C_4: A teacher can teach the discipline
C_5: A teacher is always the same, for given class and discipline
C_6: A course is taught in the same room for given class and discipline
C_7: A room in a campus can support the discipline
C_8: A course ends after it starts
C_9: A course is taught at same time - every two days if the course repetition is two times per week
C_{10}: A course is taught at same time - every three days if the course repetition is three times per week
C_{11}: A course starts and end the same half day
C_{12}: Avoid having the same discipline taught twice a day for the same class
C_{13}: A morning class ends in the morning(from 8 am to 14 pm)
C_{14}: An evening class ends in the evening(from 15 pm to 21 pm)
C_{15}:A morning discipline ends in the morning
C_{16}: An evening discipline ends in the evening
C_{17}: Insert break between specified disciplines
C_{18}: Instances of sections are chronological for a class
C_{19}: Help proving optimality by minimizing the maximum used time slots (makespan)
C_{20}: Students must have a pause time between classes according to lecture's length.
C_{21}:Some specified disciplines have break between them.
C_{22}: Classes may have a break every day between lectures for activities or non-faculty lectures.

Certainly, the CTT problem requires a number of soft and hard constraints to be satisfied [34]. Hence, our OPL model uses constraints that are classified as soft and hard constraints. In the first side, soft constraints are considered while generating the required timetable for the CTT problem [35]. For example the soft constraints in our model such as: avoid scheduling classes in the last timeslot of the day or in the early morning; avoid scheduling more than two classes in a row for a student; and avoid scheduling one class in a day for a student, etc. On the other hand, hard constraints such as: the students can only be scheduled to one event in a time slot; event rooms meet all required features and their capacity is respected; no more than one event is allowed per room and per timeslot, etc.

Implementation

The main task of the OPL CP solver is to find an assignment of each value for each decision variable such that the assignment satisfies all the soft and hard constraints in order to solve our model. For instance, the constraintthat ensures that each course has to be taught in the same room for every session of a course, is represented in our OPL model as follow:

$\forall_{i,j \in InstanceSection} room[i] = room[j]$
where i.id < j.id&&i.SectionId == j.SectionId

```
// ensure course is taught in the same room for every section
 forall(i, j in InstanceSection
      : i.id < j.id &&i.SectionId == j.SectionId) room[i] == room[j];
```

After all, we set the CP optimizer to use heuristics algorithm, Depth-first, as the search algorithm. Particularly, in the Depth-first algorithm, the decision variable is assigned a value from its domain. This assignment is then checked against the current partial solution; if any of the constraints between this variable and the past variables is violated, the assignment is abandoned and another value for the current variable is chosen. If all values for the current variable have been tried, the algorithm backtracks to the previous variable and assigns it a new value. If a complete solution is found, i.e. a value has been assigned to every decision variable, the algorithm continues to find new solutions and terminates if no more solution is found. If there are no solutions, the algorithm terminates when all possibilities have been considered.

Precisely, the decision variable is an unknown in an optimization problem and has a domain, which is a compact representation of the set of all possible values for the variable [36]. The task is to find an assignment of a value for each decision variable such that the assignments satisfy all the constraints and optimize a specific objective function. Therefore, the feasible solution to a CSP is satisfiable if there is an assignment of a value to every decision variable [8]. We want to find an optimal and/or a good solution for our CTT problem. There are six decision variables used in our OPL to be verified by all constraints and objective functions. The decision variables are listed in the Table 3.

TABLE 3. THE DECISION VARIABLES WITH MEANING

Start[InstanceSection]	Array of starting points for courses from *InstanceSet* in a *Time*
End[InstanceSection]	Array of ending points for courses from *InstanceSet* in a *Time*
classTeacher[Class,Discipline]	Array of teachers working once per time point to teach a class'sdiscipline
teacher	Set of teachers
room	Set of rooms in which courses are held
makespan	ending date of last course

Finally, the objective function used in our OPL model is the*minimize*function, which minimizes the value of decision variable *makespan* – the maximum time slots used in the timetable. The function *minimize* is satisfied by the following two constraints:

$makespan = max_{r \in InstanceSection} \; End[r]$
$makespan >= max_{c \in Class} \sum_{r \in InstanceSection} \; r.Duration$
Where r.Class == c

Particularly, we use real data from the CC to implement our OPL model. The selected data comes for different classes related to different academic programs in the CC. Each class data defines the educational program for a set of students at a different level. Hence, the resulting timetable could allow students to register in their timetable without conflict as well as follow the constraints of the CC timetable.

Alternatively, the totalnumber of teachers is forty-two teachers from different departments. Given that, those teachers have to teach at different five distant campuses at the same time. In addition, teachers should teach during the whole day in two scheduling shifts. The first shift is in the morning (from 8:00 am to 14:00 pm) while the other shift is in the afternoon/evening (from 15:00 pm to 21:00 pm). Furthermore, the students from all different programs share the same GE courses at each branch and in each time shift. Those GE courses are the university and college requirement courses that all academic programs have to share. The GE courses are taught by different departments. Additionally, the limitation of the available rooms to hold sessions have to be satisfied in the results of our model. Consequently, the resulting CTT satisfies all

constraints and minimizes the usage of limited resources available at the CC. So, it is considered as an optimal timetable for both students and teachers.

Actually, we build the model data for different nine classes. Each class has five different courses. Each course has a number of sessions (repetitions) per week that could be two or three, according to the setting of the educational program for each class. These sessions could be theoretical only or theoretical and practical. Indeed, theoretical sessions are held in classrooms while practical sessions in labs.

In addition, we have implemented (using the OPL language) in our CTT model the calculation and accumulation of the teaching load for teachers. Consequently, the resulting timetable contains the total teaching load assigned for each teacher from the different department. Finally, we define the session time as some time blocks. Each time block is a fixed thirty minutes. Then, we have defined the break time unit for each time slot as applied in the UoS timetable.

Results

Actually, running the proposed CTT OPL model,using the IBM ILOG OPL-CPLEX Development Studio Systemis reasonably simple.This OPL Studio provides the solution for our CTT OPLmodel by finding the assignments of all decision variables that satisfiesall constraints and the optimization instruction with the objective function *minimize*. An example for the solution of a decision variable, is shown below for the *classTeacher* array – the assignment of teachers to sections of each class. The *classTeacher* array contains rows for the sections' instances of a class.Whilethe value for each element in that instances are the teachers ID.

classTeacher = [[0 0 5 7 9 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 1 2 2 4 0 0 0 0 0 0 0 0 0]
[0 0 5 0 0 0 0 0 0 0 9 10 14 12 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 10 12 13 11 0 0]
[0 0 0 7 9 0 0 0 0 5 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 1 2 2 0 0 0 0 0 0 0 0 6 0]
[0 0 4 0 0 0 0 0 0 0 9 12 11 13 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 13 13 10 0 0 5]]

The solution for a feasible timetable forthe OPL CTT model, is the assignment of the two-dimensional array, *timetable*. The array values are collected from the decision variables in our OPL model. Specifically, we built the *timetable* array using the type *course*, which is a tuple (record) used to represent section instances for CTT. The *course* tuple contains the following data elements: *teacher*, *discipline*, *campus*, *room*, *id*, *repetition*, *startingPoint* and *endingPoint*. The two dimensions of the array are *time* and *class*. The *time* is ranged from zero to *MaxTime*-1 while the class is the set of classes for the educational program at CC.

Consequently, when running of our model, the resulting timetable with full contents will display the solution for the CTT problem at the CC. We have formatted the resulting timetable to be compatible with the current timetables used in the CC by all faculties, students or staff.The result shows the timetable for nine different classes. Each class has five courses. Each course has specific information such as*Days*, *Time*, *Campus*, *Room*, and *Instructor*as shown in Fig. 1. Therefore, the resulting timetable shows the teaching load assigned for each teacher. In addition, the figure shows the breaks between each course session.

// solution with objective 94

Class	Course	Days	Time	Campus	Room	Instructor
Law-80	Introduction to law	Sunday-Tuesday	08:30 - 09:45	SHW	Common Room	Law Lecturer 1
Law-80	Islamic	Sunday-Tuesday	11:00 - 12:15	SHW	CR A2	BS Lecturer 2
Law-80	IT	Monday-Wednesday	08:00 - 09:40	SHW	Lab A3	IT Lecturer 1
Law-80	Sources of Obligation	Monday-Wednesday	10:00 - 11:15	SHW	Common Room	Law Lecturer 1
Law-80	Principles of Management	Monday-Wednesday	11:30 - 12:45	SHW	CR A2	AFS Lecturer 1
Law-81	Arabic	Sunday-Tuesday	09:00 - 10:15	SHW	CR B1	BS Lecturer 3
Law-81	Administrative law	Sunday-Tuesday	10:30 - 11:45	SHW	Common Room	Law Lecturer 3
Law-81	Comm. law and companies	Sunday-Tuesday	12:00 - 13:15	SHW	Common Room	Law Lecturer 3
Law-81	Rules of Obligation	Monday-Wednesday	08:30 - 09:45	SHW	Common Room	Law Lecturer 2
Law-81	Penal Law 1	Monday-Wednesday	11:30 - 12:45	SHW	Common Room	Law Lecturer 2
IT-80	CG	Sunday-Tuesday	09:30 - 11:10	SHW	Lab B1	IT Lecturer 3
IT-80	BCO	Sunday-Tuesday	12:00 - 13:40	SHW	Lab A1	IT Lecturer 4
IT-80	Prog 1	Monday-Wednesday	08:00 - 09:40	SHW	Lab A2	IT Lecturer 4
IT-80	CS	Monday-Wednesday	10:30 - 12:10	SHW	Lab A2	IT Lecturer 1
IT-80	Islamic	Monday-Wednesday	12:30 - 13:45	SHW	CR A1	BS Lecturer 3

IT-82	AWD	Sunday-Tuesday	08:30 - 10:10	SHW	Lab A3	IT Lecturer 2
IT-82	Net 1	Sunday-Tuesday	11:30 - 13:10	SHW	Workshop 2	IT Lecturer 1
IT-82	ESP for IT 1	Monday-Wednesday	08:00 - 09:15	SHW	CR A1	BS Lecturer 1
IT-82	IWD	Monday-Wednesday	09:30 - 11:10	SHW	Lab B2	IT Lecturer 2
IT-82	Video	Monday-Wednesday	11:30 - 13:10	SHW	Lab B1	IT Lecturer 3
IT-83	ESP for IT 2	Sunday-Tuesday	08:00 - 09:15	SHW	CR A1	BS Lecturer 1
IT-83	CM	Sunday-Tuesday	09:30 - 11:10	SHW	Workshop 1	IT Lecturer 1
IT-83	SAD	Sunday-Tuesday	12:30 - 13:45	SHW	Lab A2	IT Lecturer 2
IT-83	Self Development	Monday-Wednesday	09:30 - 10:45	SHW	CR B1	AFS Lecturer 1
IT-83	IR	Monday-Wednesday	11:00 - 12:40	SHW	Lab A3	IT Lecturer 4
Law-30	IT	Sunday-Tuesday	16:00 - 17:40	SHM	Lab A2	IT Lecturer 3
Law-30	Introduction to law	Sunday-Tuesday	19:00 - 20:15	SHM	Common Room	Law Lecturer 1
Law-30	Sources of Obligation	Monday-Wednesday	15:00 - 16:15	SHM	Common Room	Law Lecturer 1
Law-30	Arabic	Monday-Wednesday	16:30 - 17:45	SHM	CR A1	BS Lecturer 3
Law-30	Principles of Management	Monday-Wednesday	18:00 - 19:15	SHM	CR B2	AFS Lecturer 2
Law-31	Penal Law 1	Sunday-Tuesday	16:00 - 17:15	SHM	Common Room	Law Lecturer 2
Law-31	Rules of Obligation	Sunday-Tuesday	17:30 - 18:45	SHM	Common Room	Law Lecturer 2
Law-31	Introduction to Psychology	Sunday-Tuesday	19:30 - 20:45	SHM	CR A1	BS Lecturer 4
Law-31	Comm. law and companies	Monday-Wednesday	17:00 - 18:15	SHM	Common Room	Law Lecturer 3
Law-31	Administrative law	Monday-Wednesday	18:30 - 19:45	SHM	Common Room	Law Lecturer 3
IT-30	BCO	Sunday-Tuesday	15:00 - 16:40	SHM	Lab A1	IT Lecturer 4
IT-30	Prog 1	Sunday-Tuesday	17:00 - 18:40	SHM	Lab A3	IT Lecturer 2
IT-30	CG	Sunday-Tuesday	19:00 - 20:40	SHM	Lab B1	IT Lecturer 3
IT-30	Islamic	Monday-Wednesday	15:00 - 16:15	SHM	CR B2	BS Lecturer 3
IT-30	CS	Monday-Wednesday	17:30 - 19:10	SHM	Lab A1	IT Lecturer 1
IT-32	ESP for IT 1	Sunday-Tuesday	15:00 - 16:15	SHM	CR B2	BS Lecturer 1
IT-32	Principals of Statistics	Sunday-Tuesday	16:30 - 17:45	SHM	CR B1	BS Lecturer 3
IT-32	Net 1	Sunday-Tuesday	19:00 - 20:40	SHM	Workshop 1	IT Lecturer 2
IT-32	IWD	Monday-Wednesday	16:00 - 17:40	SHM	Lab B1	IT Lecturer 4
IT-32	AWD	Monday-Wednesday	18:00 - 19:40	SHM	Lab A3	IT Lecturer 3

Teacher Load

Law Lecturer 1: 12
Law Lecturer 2: 12
Law Lecturer 3: 12
BS Lecturer 1: 9
BS Lecturer 2: 3
BS Lecturer 3: 15
BS Lecturer 4: 3
AFS Lecturer 1: 6
AFS Lecturer 2: 3
IT Lecturer 1: 20
IT Lecturer 3: 20
IT Lecturer 4: 20
IT Lecturer 2: 19

Fig 1: **The Solution for the CTT problem at the CC**

Further, we convert the timetable produced in Fig 1 into graphical representation as shown in Fig. 2 and Fig. 3, in which each student group belongs to a specific class (educational program) could register their courses easily. Altogether, those figures show that there is no conflict for students, teachers, and rooms as the required solution for the CTT problem at the CC. In addition, these figures show that each group of students has a break time during the day for extra-curriculum activities.

	Sunday-Tuesday					Monday-Wednesday				
Section/Time	Law-80	Law-81	IT-80	IT-82	IT-83	Law-80	Law-81	IT-80	IT-82	IT-83
8:00					ESP for IT 2 -> 08:00 - 09:15 -> SHW -> CR A1 -> BS Lecturer 1	IT -> 08:00 - 09:40 -> SHW -> Lab A3 -> IT Lecturer 1		Prog 1 -> 08:00 - 09:40 -> SHW -> Lab A2 -> IT Lecturer 4	ESP for IT 1 -> 08:00 - 09:15 -> SHW -> CR A1 -> BS Lecturer 1	
8:30	Introduction to law -> 08:30 - 09:45 -> SHW -> Common Room -> Law Lecturer 1			AWD -> 08:30 - 10:10 -> SHW -> Lab A3 -> IT Lecturer 2			Rules of Obligation -> 08:30 - 09:45 -> SHW -> Common Room -> Law Lecturer 2			
9:00		Arabic -> 09:00 - 10:15 -> SHW -> CR B1 -> BS Lecturer 3								
9:30			CG -> 09:30 - 11:10 -> SHW -> Lab B1 ->		CM -> 09:30 - 11:10 -> SHW -> Workshop				IWD -> 09:30 - 11:10 -> SHW -> Lab	Self Development -> 09:30 -

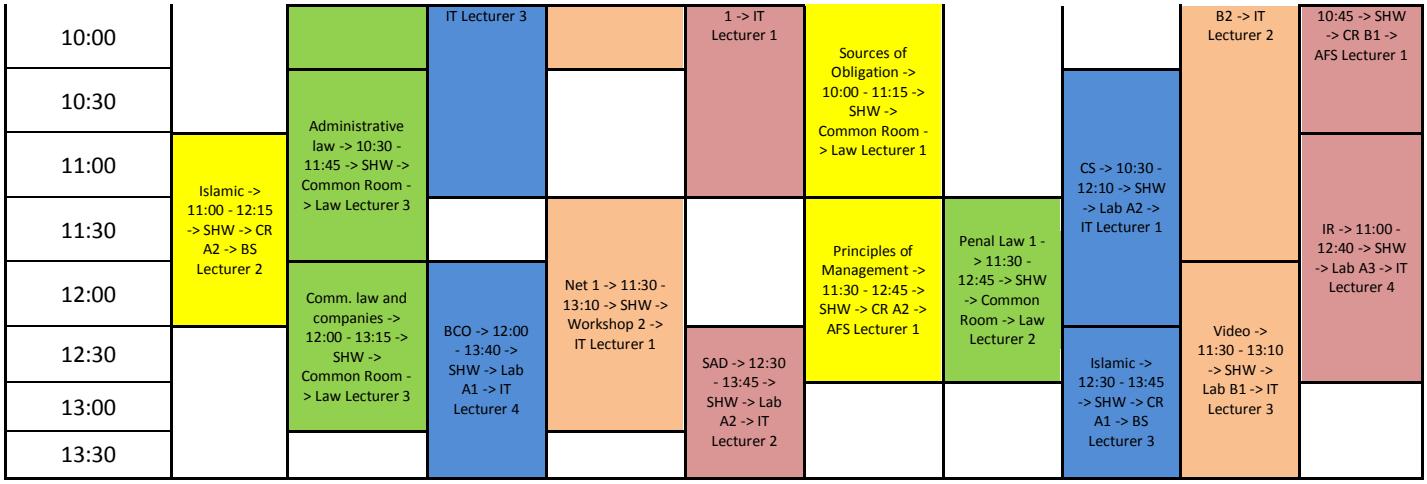

Fig 2: Graphic Representation for the Morning Shiftof CTT

	Sunday-Tuesday				Monday-Wednesday			
Section/Time	Law-30	Law-31	IT-30	IT-32	Law-30	Law-31	IT-30	IT-32
15:00			BCO -> 15:00 - 16:40 -> SHM -> Lab A1 -> IT Lecturer 4	ESP for IT 1 -> 15:00 - 16:15 -> SHM -> CR B2 -> BS Lecturer 1	Sources of Obligation -> 15:00 - 16:15 -> SHM -> Common Room -> Law Lecturer 1		Islamic -> 15:00 - 16:15 -> SHM -> CR B2 -> BS Lecturer 3	
15:30								
16:00	IT -> 16:00 - 17:40 -> SHM -> Lab A2 -> IT Lecturer 3	Penal Law 1 -> 16:00 - 17:15 -> SHM -> Common Room -> Law Lecturer 2		Principals of Statistics -> 16:30 - 17:45 -> SHM -> CR B1 -> BS Lecturer 3	Arabic -> 16:30 - 17:45 -> SHM -> CR A1 -> BS Lecturer 3	Comm. law and companies -> 17:00 - 18:15 -> SHM -> Common Room -> Law Lecturer 3		IWD -> 16:00 - 17:40 -> SHM -> Lab B1 -> IT Lecturer 4
16:30								
17:00		Rules of Obligation -> 17:30 - 18:45 -> SHM -> Common Room -> Law Lecturer 2	Prog 1 -> 17:00 - 18:40 -> SHM -> Lab A3 -> IT Lecturer 2				CS -> 17:30 - 19:10 -> SHM -> Lab A1 -> IT Lecturer 1	
17:30								
18:00					Principles of Management -> 18:00 - 19:15 -> SHM -> CR B2 -> AFS Lecturer 2	Administrative law -> 18:30 - 19:45 -> SHM -> Common Room -> Law Lecturer 3		AWD -> 18:00 - 19:40 -> SHM -> Lab A3 -> IT Lecturer 3
18:30								
19:00	Introduction to law -> 19:00 - 20:15 -> SHM -> Common Room -> Law Lecturer 1		CG -> 19:00 - 20:40 -> SHM -> Lab B1 -> IT Lecturer 3	Net 1 -> 19:00 - 20:40 -> SHM -> Workshop 1 -> IT Lecturer 2				
19:30		Introduction to Psychology -> 19:30 - 20:45 -> SHM -> CR A1 -> BS Lecturer 4						
20:00								
20:30								

Fig 3: Graphic Representation for the Evening Shift of CTT

The IBM Development Studio provides the solution for our OPL model in few seconds. The IBM Development Studio has been run on i7 CPU at 2.10 GHz with 6.00 GB RAM and 64-bit Operating system. Conversely, currently the manual solution of the CC timetabling problem requires several workdays.

Conclusion

We reported a model for the university timetable that based on finite domain technique for CSP. Our system solves the problem efficiently. The development time for the program is much shorter than time spending on the manual approach. Our experience shows that applying constraint logic programming with the optimization language is a practically viable means for timetable scheduling in a university.

The results from running our model of the CTT problem at the CC conclude and prove that all the required constraints are successfully verified. We found that the resulting timetable could be used easily by students to register their courses with conflict-free. Moreover, an important issue is appeared in the result timetable, our model balance the teaching load for teachers that the college faculties For further work, we are able to enrich this model with concepts and data of the other colleges at the UoS to build the university CTT. The further

development of timetabling model includes timetabling for any college in the UoS, dedicating courses for specific times, adding breaks for specific courses, dedicating rooms to be free in a specific times, dedicating teachers to be free in a specific times, improvements of problem modeling, and using the APIs of the IBM IDE solver to build a standalone system. The study of adding or combining different modeling techniques and heuristics techniques will be the main parts of the further research.

Finally, the applying of a comparable model for the ETT with the required methods, concepts and data from the CC will be also the scope of further research.

References

[1] R. Wattanamano, K. Thongsanit and P. Hongsuwan, "The Development of Mathematical Model for a University Course Timetabling Problem," Silpakorn University Science and Technology Journal, vol. 5, no. 2, pp. 46-52, 2011.

[2] K. Kumar, Sikander, R. Sharma and Kaushal, "Genetic Algorithm Approach to Automate University Timetable," International Journal of Technical Research(IJTR), vol. 1, no. 1, 2012.

[3] A. O. Adewumi, B. A. Sawyerr and M. M. Ali, "A heuristic solution to the university timetabling problem," Engineering Computations: International Journal for Computer-Aided Engineering and Software, vol. 26, no. 8, pp. 972-984, 2009.

[4] S. Abdennadher and M. Marte, "University Course Timetabling using Constraint Handling Rules," Journal of Applied Artificial Intelligence, vol. 14, no. 4, pp. 311-325, 2000.

[5] R. Barták, M. A. Salido and F. Rossi, "Constraint satisfaction techniques in planning and scheduling," Intelligent Manufacturing, vol. 21, no. 1, pp. 5-15, 2010.

[6] F. Rossi, P. van Beek and T. Walsh, Handbook of Constraint Programming, 1 ed., Elsevier Science, 2006, p. 978.

[7] A. Schaerf, "A Survey of Automated Timetabling," Artificial Intelligence Review, vol. 13, no. 2, pp. 87-127, 1999.

[8] S. C. Brailsford, C. N. Potts and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," European Journal of Operational Research, vol. 119, pp. 557-581, 1999.

[9] S. Abdennadher and M. Marte, "University Course Timetabling using Constraint Handling Rules," Journal of Applied Artificial Intelligence, vol. 14, no. 4, pp. 311-325, 2000.

[10] V. Kumar, "Algorithms for Constraint-Satisfaction Problems: A Survey," AI Magazine, vol. 13, no. 1, pp. 32-44, 1992.

[11] S. Petrovic and E. Burke, "University timetabling," in Handbook of scheduling: algorithms, models, and performance analysis, Chapman and Hall/CRC, 2004, pp. 1-23.

[12] S. Abdennadher and M. Marte, "University timetabling using constraint handling rules," JFPLC, pp. 39-50, 1998.

[13] L. Zhang and S. K. Lau, "onstructing university timetable using constraint satisfaction programming approach," Computational Intelligence for Modelling, Control and Automation, vol. 2, pp. 55-60, 2005.

[14] A. Ojugo, I. Iyawa., F. Aghware., M. Yerokun and E. Ugboh, "Comparative Study of the Timetable Constraint Satisfaction Problem," in 5th International Conference on Cir-cuits, Systems, Control, Signals (CSCS '14), Salerno, Italy, 2014.

[15] C. Gu´eret, N. Jussien, P. Boiz and C. Prins, "Building university timetables using constraint logic programming," in Practice and Theory of Automated Timetabling, Springer-Verlag LNCS 1153, 1996, p. 130–145.

[16] W. Legierski, "Constraint-based reasoning for timetabling," in Artificial Intelligence Method, Gliwice, Poland, 2002.

[17] K. Zervoudakis and P. Stamatopoulos, "A generic object-oriented constraint-based model for university course timetabling," Lecture Notes in Computer Science, vol. 2079, pp. 28-47, 05 Sep 2001.

[18] H.-J. Goltz, G. Küchler and D. Matzke, "Constraint-based timetabling for universities," in INAP'98, 11th International Conference on Applications of Prolog, 1998.

[19] H. S. Fen, S. Deris and S. Hashim, "Investigating Constraint-Based Reasoning for University Timetabling Problem," in Ho Sheau Fen, Safaai-Deris, Siti Zaiton-Mohd Hashim, "Investigating Constraint-Based Reasoning for University Timetabling Problem", Proceedings of the International Multi Conference of Engineers and Computer Scientists, 2009.

[20] L.-Y. Shue, P.-C. Lin and C.-Y. Tsai, "Constraint Programming Approach for a University Timetabling Decision Support System with Hard and Soft Constraints," in Opportunities and Challenges for Next-Generation Applied Intelligence Studies in Computational Intelligence, vol. 214, 2009, pp. 93-98.

[21] M. Gavanelli, "University Timetabling in ECLiPSe," Association for Logic Programming, vol. 19, no. 3, 2006.

[22] H. Rudová and K. Murray, "University course timetabling with soft constraints," in Practice and theory of automated timetabling IV, Gent, Belgium, 2003.

[23] C.-m. Yeung, S.-M. Leung and H.-F. Leung, "Applying constraint satisfaction technique in university timetable scheduling," Evolutionary Computing, vol. 3, pp. 683-695, 1995.

[24] M. A. Bakir and A. Cihan, "A 0-1 integer programming approach to a university timetabling problem," Hacettepe Journal of Mathematics and Statistics, vol. 37, no. 1, p. 41 – 55, 2008.

[25] S. Ribi and S. Konjicija, "A Two Phase Integer Linear Programming Approach to Solving the School Timetable Problem," in 32nd International Conference on Information Technology Interfaces, Cavtat, Croatia, June 21-24, 2010.

[26] O. Czibula, H. Gu, A. Russell and Y. Zinder, "A Multi-Stage IP-Based Heuristic for Class Timetabling and Trainer Rostering," in 10th International Conference of the Practice and Theory of Automated Timetabling, York, United Kingdom, 2014.

[27] K. Zervoudakis and P. Stamatopoulos, "A Generic Object-Oriented Constraint-Based Model for University Course Timetabling," in Practice and Theory of Automated Timetabling III, vol. 2079, Springer Berlin Heidelberg, 2001, pp. 28-47.

[28] P. Laborie, "IBM ILOG CP Optimizer for detailed scheduling illustrated on three problem," in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer Berlin Heidelberg, 2009.

[29] P. Van Hentenryck, "A Preview of OPL," Department of Computing Science and Engineering, UCL, Belgium, 2000.

[30] M. SCHULZ and A. EISENBLÄTTER, Solving Frequency Assignment Problems with Constraint Programming, Technische Universität Berlin, Institut für Mathematik, 2003.

[31] K. Vermirovsky, Algorithms for constraint satisfaction problems, Diss. Master Thesis of Purdue University, 2003.

[32] Z. Liu, Algorithms for Constraint Satisfaction Problems (CSPs), M.S. thesis, Department of Computer Science, University of Waterloo, Waterloo, Canada, 1998.

[33] M. Abril, M. A. Salido, F. Barber, L. Ingolotti, P. Tormos and A. Lova, "Distributed Constraint Satisfaction Problems to Model Railway Scheduling Problems," in the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006), Ambleside, 2006.

[34] B. McCollum and N. Ireland, "University Timetabling: Bridging the Gap between Research and Practice," in 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT), 2006.

[35] S. M. S. Shatnawi, . F. Albalooshi and K. Rababa'h, "Generating Timetable and Students schedule based on data mining techniques," International Journal of Engineering Research and Applications (IJERA), vol. 2, no. 4, pp. 1638-1644, 2012.

[36] P. Van Hentenryck, L. Michel, L. Perron and J. C. Régin, "Constraint programming in OPL," in Principles and Practice of Declarative Programming, Berlin, Heidelberg, 1999.