UTSE: Construction of Optimum Timetables for University Courses - A CLP Based Approach

Harikleia Frangouli Vassilis Harmandas Panagiotis Stamatopoulos

University of Athens, Department of Informatics
Panepistimiopolis, TYPA Buildings
157 71 Athens, Greece
E-mail : {bxpro,vassilis,takis}@di.uoa.ariadne-t.gr

Abstract

The construction of timetables for universities or schools is an extremely complex problem, whose manual solution requires much effort. The set of all possible solutions, that is the search space of the problem, is very large, at least in the real-world examples. An acceptable solution is one that satisfies all the problem constraints. The problem becomes even more difficult if someone wants to generate an optimum timetable according to some heuristic criteria. Various attempts have been made so far on the automatic solving of the timetabling problem by a computer. In this paper, a method is proposed for the construction of optimum timetables for university courses. A specific system is presented which has been used for the timetabling procedure of the Department of Informatics of the University of Athens. The software platform of the implementation is an instance of the Constraint Logic Programming class of languages, the ECL^{i}PS^e system. ECL^{i}PS^e is proved to be an appropriate vehicle for managing the complexity of the timetabling problem.

1. Introduction

Many real-life problems lead naturally to combinatorial search which is a very computationally intensive task. Unfortunately, no general method exists for solving problems of this kind efficiently. The automatic construction of timetables [dWe85] falls under this general class of problems.

Much work has been done so far in the area, even since the early sixties [Got62, Law69, BMc79, CdW89]. The reason for this is that at least once a year, schools and universities have to solve an instance of the timetabling problem whose manual solution requires a lot of manpower. It would be desirable to have a program schedule courses and/or exams instead of a person. What is more is that in many cases, an

optimum timetable is required according to some heuristic criteria. This problem is much more difficult than the construction of a simple working timetable.

In this paper, the construction of optimum timetables for university courses is tackled using the Constraint Logic Programming (CLP) [vHe89, FH+92] framework. The system which is presented is called UTSE (University Timetable Scheduling in ECL^{i}PS^e) and has been used on the real case of the Department of Informatics of the University of Athens (DoI/UoA). The ECL^{i}PS^e language [ECL93], an instance of the CLP class of languages which supports constraint satisfaction techniques over finite domains, is used for the modelling and implementation of the system.

In order to get an idea about the complexity of the timetabling problem, consider a semester at a university where 30 different subjects have to be scheduled, each subject splitting into 4 teaching periods per week. Let us also assume that there are 5 working days in a week with 10 possible teaching periods in every day and that there are 3 rooms for hosting lectures. In this case, which is quite typical, the search space to be examined for the construction of a weekly timetable is as big as $(5 \cdot 10 \cdot 3)^{4 \cdot 30} \cong 1.35 \cdot 10^{261}$. The problem is to find a point, i.e. a specific timetable, in this huge search space that satisfies all the imposed constraints. A more difficult problem is to find an optimum timetable.

Various approaches have been followed so far for the solution of the timetabling problem. Graph colouring algorithms have been used [Meh81], where each vertex in the graph represents a triplet (subject, teaching staff member, room). Two vertices are connected if they cannot be scheduled simultaneously, that is if they have a subject, a teaching staff member or a room in common. The problem is reduced to find a colouring of the graph with p or fewer colours, if one exists. Mathematical programming algorithms have been used as well [Tri80, FRo85], where the central idea is to solve an optimization problem whose purpose is to reduce constraint violations. Both graph colouring algorithms and mathematical programming are approaches from the Operations Research (OR) area. Other methods include the use of database management systems [Joh93], genetic algorithms [CDM90] and general purpose constraint solving tools [YK+94]. Quite recently, logic programming approaches have been considered [KWh92] and sometimes a CLP framework is exploited [ABa94, BDP94].

What is achieved by a CLP approach is an ***a priori*** pruning of the search space through a constraint propagation mechanism. At each step, the constraints are exploited as much as possible, and then a choice is made which triggers more propagation. Choices are made non-deterministically. In this way, the resolution mechanism of a logic programming language is combined with constraint satisfaction techniques resulting in a very powerful and elegant scheme of computation.

The purpose of this paper is to exhibit the appropriateness of CLP for mastering the combinatorial timetabling problems faced by schools and universities. The main issue discussed is the composition of optimum timetables according to some widely accepted criteria. The quantification of these criteria is empirically done, resulting to an objective function that has to be optimized. The course scheduling of the DoI/UoA is studied, but as it will be seen, CLP is powerful enough to accommodate the needs of other specific timetabling problems as well.

In the following, the timetabling problem is introduced and CLP as well as ECL^{i}PS^e are briefly described. Then, the specific problem faced by the DoI/UoA is presented. The way this problem is modelled and implemented in ECL^{i}PS^e and how the resulting UTSE system is behaving are given next. Finally, the adopted approach is discussed and possible further work is highlighted.

2. The Timetabling Problem

As it is stated in [dWe85], the timetabling process consists of two distinct phases:

- The curricula, i.e. lists of subjects, are defined for each class and various resources (manpower and/or equipment) are assigned to them.
- A detailed timetable is composed which is compatible with the previously defined requirements and with some additional constraints as well.

The second phase is the one which is usually referred to as the "timetabling problem". Due to the complexity of this problem, it is worthwhile trying to solve it by a computer instead of tackling it manually.

Actually, the previous problem is a course scheduling one. Another problem often found in the literature is the exam scheduling. Both problems are quite similar, though they present a few differences:

- There is usually only one exam but several lectures for a subject in one semester.
- Conflicts are not allowed in exam scheduling, but some may exist in course scheduling.
- In general, exam scheduling is less complex than course scheduling.

In the following, we will concentrate on course scheduling, considering that this problem might benefit a lot from a possible automation.

The constraints that a solution to the timetabling problem has to respect may be either general or specific. A general constraint is one that does not refer to a specific subject, room or member of the teaching staff. For example, the following are general constraints that the solution of every timetabling problem has to satisfy:

- No member of the teaching staff can give two lectures simultaneously.
- No room can host two lectures simultaneously.

- No student can attend two lectures simultaneously (though in some cases, this constraint has to be applied for subjects of a specific type only, while for other subjects conflicts in their lectures may arise, in order to get at least one solution).
- The room capacities have to be respected.

For a given timetabling problem, there is also a set of specific constraints that have to be satisfied as well. Some examples are:

- Prof. Kathigitis can teach only on Wednesdays and Fridays.
- Room A is not available on Mondays after 17:00.
- The subject "Calculus I" has to be scheduled on Tuesdays and Thursdays from 9:00 to 11:00.

A distinction that is usually made is whether the timetabling problem is faced by a school or by a university. The differences lie only in the specific constraints that have to be satisfied. For example, a university student may attend lectures in the same day that have gaps between them, though this possibility should be minimized, while in a school, gaps are not allowed. As far as complexity is concerned, both problems are of similar difficulty. This paper concentrates on university course scheduling, however it is a matter of constraints definition to use the proposed method for schools as well.

The timetabling problem is formulated either as a feasibility problem or as an optimality problem. In the first case, just a single feasible solution is required, while in the second case, the optimum solution according to an objective (or cost) function has to be found. Both formulations are very similar, but for the optimality problem a good definition of the cost function has to be given, so as to reflect subjective opinions about the features characterising a timetable as a good one. Some factors that may affect the quality of a timetable are the existence of gaps between the various lectures in the same day, the reasonable distance in time of the lectures for the same subject, the exploitation of the rooms, etc. There is no widely accepted cost function taking into account these factors as well as other ones, but it should be mentioned that a real-world timetable construction system has to consider a flexible enough cost function that may satisfy the needs of every potential user.

Sometimes, what is required to be computed is a new timetable, because of a change in the constraints that have to be satisfied. For example, the availability of a member of the teaching staff or of a room might change, or even a new subject might be introduced. In this case, it is not wise to reschedule the whole timetable from scratch, since such a solution might affect the personal schedules of a lot of people unnecessarily. A new timetable is required with the additional property of having the minimum distance, and thus the less side-effects, from the existing timetable.

A bad result that may be produced from the processing of a timetabling problem is the proof of the unfeasibility of the problem. This means that no solution exists that satisfies all the given constraints. In this case, what may be done is to consider that some constraints are not compulsory and try to relax them, in order to find a working timetable. Indeed, this is the case in the real world, since many of the constraints, mainly specific ones, state just a preference and not an obligation. For example, a member of the teaching staff might have requested not to teach before 11:00, but if a change to his/her personal schedule would result to a desirable timetable, this should be done. Another approach is instead of doing constraint relaxation *a posteriori*, to assign weights to the constraints and try to minimize the total weight of the violated constraints.

3. CLP and ECL^{i}PS^e

CLP is a novel programming framework that enhances logic programming with constraint satisfaction techniques. While preserving the declarative nature of logic programming, it exhibits surprising efficiency for solving a certain class of combinatorial problems. While the straightforward solution of such problems might be done through a "generate and test" approach, CLP introduces a new method of computation, the "constrain and generate" one. In cases of large scale combinatorial problems, "generate and test" cannot lead to results within finite time.

CLP is often referred to as CLP(X) where X stands for the set over which the problem constraints are stated. There exist languages where X is the set of real numbers, the set of rational numbers, a set of boolean values, etc. A very interesting case is the one where a problem is described via constraints which involve variables that range over finite domains. Then, the corresponding instance of CLP(X) is called CLP(FD). The first representative of the CLP(FD) scheme of computation is the CHIP language [Dv+88]. CHIP was developed at the European Computer-Industry Research Centre (ECRC). The constraint facilities of CHIP have now been integrated with ECRC's SEPIA and Megalog systems, resulting into the ECL^{i}PS^e language.

ECL^{i}PS^e is a Prolog system whose aim is to serve as a platform for integrating various logic programming extensions. Many interesting extensions of ECL^{i}PS^e are based on a special data type, which is called metaterm. Various constraint related libraries are supported, which have been built on the metaterm facility, namely the finite domains library, the generalised propagation library (Propia), the constraint handling rules library, etc. The first is the one which brings to ECL^{i}PS^e the constraints functionality of CHIP for finite domains.

The finite domains library of ECL^{i}PS^e implements constraints that involve integer as well as atomic data. The concept of domain variables is used which range over finite domains. Constraints are either arithmetic or symbolic. An arithmetic constraint is an equality, disequality or inequality relation between two linear terms, where a linear term is composed of domain variables and integers in a way respecting its linearity. There are various symbolic constraints that operate upon both integer and atomic data. Enumeration as well as optimization predicates are also provided by ECL^{i}PS^e.

The procedure for solving a constraint satisfaction problem using the finite domains library of ECL^{i}PS^e is to define a set of domain variables, together with their domains, that model the problem concepts. Then, the constraints that define the relations among these variables have to be stated. Actually, these constraints define the subset of the search space that contains the solutions of the problem. The last step is to trigger an enumeration procedure, which co-operates with the internal constraint propagation mechanism and with the backtracking mechanism of the Prolog engine, leading finally to the required solutions, if they exist. In case of an optimality problem, ECL^{i}PS^e provides a branch-and-bound technique that computes the optimum solution of the enumeration procedure, with respect to a given cost function.

4. UTSE - A Timetable Construction System

An attempt to solve the timetabling problem mentioned previously has been made with the UTSE system. A number of assumptions and constraints have been taken into account, so that the system accommodates the requirements of the DoI/UoA.

More specifically, three main assumptions have been considered:

- Each subject is split into two or three lectures depending on its duration (except from those with only one teaching period per week). Each of these lectures must be given in different days.
- Each subject can be either tutorial or practical. Tutorial subjects are being taught in a classroom, whereas practical ones are being taught in a laboratory. Both are assumed to have a fixed student capacity.
- Each subject belongs to a specific teaching year and falls under one out of four separate categories, namely compulsory, basic, optional and free choice. Moreover a subject that is not compulsory belongs to one out of three areas.

Apart from those assumptions, four main constraints have been taken into consideration:

- No two lectures can be scheduled the same day and period of time in the same room.

- No two subjects belonging to the same teaching year should overlap if they are compulsory or fall under the same subject area.
- No two lectures should be given by the same teaching staff member at the same time. Moreover no teaching staff member should be assigned a lecture at a period of time, at which he/she has declared to be unavailable.
- During the semester each subject is being taught in the same room (classroom or laboratory), which must be large enough to fit the number of students attending it. A room may be occupied for non-tutorial reasons for some hours during the week, therefore no lecture should be scheduled in that room at that period of time.

The constraints concerning the availability of a teaching staff member or room, can be considered as second order ones, meaning that they could be relaxed if this would help finding a solution to the timetabling problem. This is by no means the case for the other constraints (first order constraints) which can never be altered or relaxed. It must be noted however that UTSE makes no distinction between first and second order constraints and tries to find a solution that satisfies them all.

The most common case is that if a solution to the timetabling problem, which of course satisfies all the constraints, can be found, then a great number of solutions can also be found. UTSE tries to find the best of these solutions, assigning a cost to each of them and ultimately presenting the one with the minimum cost. The evaluation of each solution's cost is made according to how well this solution satisfies some predefined criteria: the more a solution satisfies those criteria the less its cost is. This technique can yield a far better solution than the first one that would be returned, if the cost evaluation process was not used.

Three criteria have been set in the present system, which of course can be easily expanded and improved. Each one of the following criteria adds a cost to the solution summing up to the total cost :

- When a subject is scheduled in two or more different days, it is preferred that these days have a distance of two or three days between them. A distance of four days is not so good and a distance of one day is even worse.
- In order to achieve the best utilisation of each room, it is desirable that each subject is scheduled in a room with capacity proportional to the number of students attending this subject.
- It is desirable that all subjects belonging to the same teaching year, if scheduled in the same day, should be taught in periods of time that have as few and small gaps between them as possible. This means that the ideal assignment of these lectures would be in consecutive teaching periods.

It should be mentioned here that these criteria have not been defined as constraints for reasons of adding the flexibility to the system to accept as good enough

a solution, which satisfies them to some point rather than trying to find the solution that completely satisfies them (a process which can turn out to be extremely time consuming or very well lead to no solution).

A weight factor is assigned to each one of these criteria determining how much they will influence the final solution: the greater the weight factor is the bigger gravity the corresponding criterion will have on the solution's cost. The user can therefore interfere to the cost evaluation process by altering these weight factors. If, for instance, the weight factor of the third criterion is set to zero, then this criterion will not be taken into account in the evaluation of the solution's cost.

4.1 Representation

All information regarding the details of subjects, teaching staff members, rooms, etc., is encoded in a particular way. More specifically :

- Each day of the week is represented by a fact with predicate day/2 containing a day code and the name of the corresponding weekday. In a similar way teaching periods in a day are represented as facts with predicate teaching_period/2 containing a code and the corresponding time period in a day. For instance, day(1,'Monday') means that 'Monday' is identified by code 1, and teaching_period(4,12:00-13:00) means that the teaching period between 12:00 and 13:00 is identified by code 4.
- All data dealing with a particular subject is encoded in a fact with predicate subject/9. Such data is the subject code, the teaching year, the category and the subject area it falls under, the room type (all classrooms are identified by code 0, while laboratories by their full names), the teaching periods it occupies per week, the name of the teaching staff member responsible for the subject, the number of students attending it, and its full name. For instance, subject(1,3,basic,2, 'Lab 1',5,'Halatsis',80,'Artificial Intelligence') represents the subject called 'Artificial Intelligence', with code 1, belongs to the 3rd teaching year, is basic of the 2nd subject area, is held in laboratory 'Lab 1' for 5 teaching periods per week, is taught by Prof. 'Halatsis' and attended by 80 students.
- All the information regarding teaching staff members is stored in facts with predicate teaching_staff_member/3. Such a fact contains the code and full name of the teaching staff member, as well as a list representing the periods of time when he/she is not available for teaching any subject. The list elements are compound terms each of which identifies a period of time by stating the codes of the day of week and the time period in this day that a teaching staff member is unavailable. For instance, Prof. 'Lector', whose code is 12, and is not available on Mondays, Wednesdays and Thursdays from 9:00 to 12:00, is represented by

teaching_staff_member(12,‘Lector’,[no(1,1),no(1,2),
no(1,3),no(3,1),no(3,2),no(3,3),no(4,1),no(4,2),no(4,3)]).

- Classrooms are encoded as facts with predicate classroom/4 containing the code, the name, the room size, and finally a list of compound terms representing the periods of time that the classroom is occupied for non-tutorial reasons. In the same way a fact with predicate lab/4 contains similar information that deals with laboratories. For instance, if classroom ‘Classroom A’, with code 1, can hold up to 60 students and is occupied every Friday afternoon from 17:00 to 19:00, the corresponding fact will be classroom(1,‘Classroom A’,60, [no(5,9),no(5,10)]).

A solution to the timetabling problem is represented by three finite domain variable lists, using the ECL^{i}PS^e terminology. The first list contains day codes, the second teaching period codes and the third room codes. Each list element corresponds to a specific teaching period of a subject. Therefore the scheduling of every teaching period is identified by the respective elements of the lists of days, teaching periods and rooms.

4.2 Implementation

The UTSE system is quite flexible regarding the input of data. It provides the user with a window environment interface which is designed so that it can be easily handled by a non-expert. The user can select from a given data store the subjects, teaching staff members, classrooms and laboratories that will be included in the timetable or create his/her own data by modifying (i.e. adding, removing or changing) some of these elements. He/she can also set the constraints concerning the availability of teaching staff members and rooms (i.e. define the days and periods of time in these days in which they are not available). Finally, there is the possibility to define the labeling and optimization type that will be used in the process of searching for a solution, as well as the weight factors for the three optimization criteria.

As soon as the data input stage has been completed, the main process starts, which includes the following phases:

Phase One : Primal checking

- The system checks if there is a subject which is attended by more students than the maximum capacity classroom can fit. In that case it splits the subject into two or more identical subjects.
- The system checks whether there is a teaching staff member whose total teaching hours exceed his/her total hours of availability (as they have been defined at the data input stage) and if so, the execution of the program is terminated.

Phase Two : Finite domain variables definition and cost function evaluation

- The lists of finite domain variables are defined. As mentioned in the previous section, there are three lists representing the days, periods of time in each day and rooms in which the lectures will be held.
- The cost function which will evaluate the cost of each solution is constructed. At first a cost is evaluated according to each one of the optimization criteria mentioned previously. The addition of these partial costs results in the cost function. When this phase is completed, every subject is assigned a cost which the system will try to reduce at the labeling stage.

Phase Three : Set of constraints

- The constraints on the teaching staff members are set. This means that the finite domain variables corresponding to the subjects that a teaching staff member is assigned, are constrained not to take as values the periods of time that he/she is not available.
- The constraints on the rooms are set. Each subject must be held in the same room during the week. This room must be large enough to hold the students attending the subject. Moreover, a subject should not be assigned to a room at the periods of time that the room is not available. All the tutorial subjects are arbitrarily assigned to one of the available classrooms (as long as its constraints are satisfied). Each practical subject is assigned to the predefined for that subject laboratory.
- The constraints on the subjects are set. More specifically, only one subject can be taught in a room at a certain period of time, a teaching staff member can teach only one subject at a certain period of time, and subjects of the same teaching year and area must not be scheduled simultaneously.
- The subjects are split into lectures spread in different days in the week.

Phase Four : Values assignment

- The finite domain variables of each one of the three lists are given a value. There are three ways of labeling, determined by the user choice made at the data input stage. The labeling type is concerned with the order in which the three lists of domain variables will be assigned values. The three types of order currently provided are: days-teaching periods-rooms, teaching periods-days-rooms and rooms-days-teaching periods. The selection of these types of order has been made heuristically, as they have been proved more efficient in yielding a solution from all the possible combinations of the three domain variable lists.

  The labeling process is dominated by an optimization procedure which takes into account the cost function. This procedure, which is carried out by the

min_max/5 built-in of ECL^{i}PS^e, goes as follows: as soon as a solution has been found, the system is forced to find another solution with less cost than that of the previous one. This process carries on until no better solution (i.e. with less cost) can be found or the cost of the last solution found is less than a minimum threshold defined by the user. In that case the search terminates and the solution is returned.

Once again there are two types of optimization: the first one ensures that there will be no subject with "bad" cost (i.e. a subject that does not satisfy the three optimization criteria at some minimum point) and the second one ensures that the majority of subjects have a "good" cost (i.e. they satisfy the three criteria at a satisfactory point) but also permits some subjects with "bad" cost to exist. In general, the first optimization type yields a balanced solution, while the second one yields a non-balanced solution. In practice the optimization procedure in the first case tries to minimize the maximum subject cost, while in the second case tries to minimize the overall cost that results from the addition of every subject cost.

When all four phases have been successfully completed the solution is displayed on the screen or printed on a printer device. In case a solution cannot be found either because the data are too constrained or because the system cannot estimate the best solution within a reasonable period of time, the user can terminate the search, relax the constraints or define a different labeling and/or optimization strategy, and start the process from the beginning.

4.3 UTSE at Work

For a timetable scheduling system to be complete, a flexible user interface should be provided, so that the specific requirements of the problem can be stated (subjects, teaching staff members, rooms available, etc.). UTSE provides such an interface.

The UTSE interface has been developed using the KEGI (Kernel for ECRC Graphical Interface) and PCE (Programmable Computing Environment) extensions of ECL^{i}PS^e. It is intended to be ported to the Tcl/Tk extension of ECL^{i}PS^e as well. The interface allows the user to define the system parameters as preferred. Not only does it make it possible for the user to enter the particular subjects, etc. that are to be scheduled and the constraints regarding teaching staff members' and rooms' availability, it also enables the dynamic definition of the labeling procedure and the optimization parameters (weight factors of the cost function as described in the

previous section, and optimality bounds) that will be used when a solution is searched for.

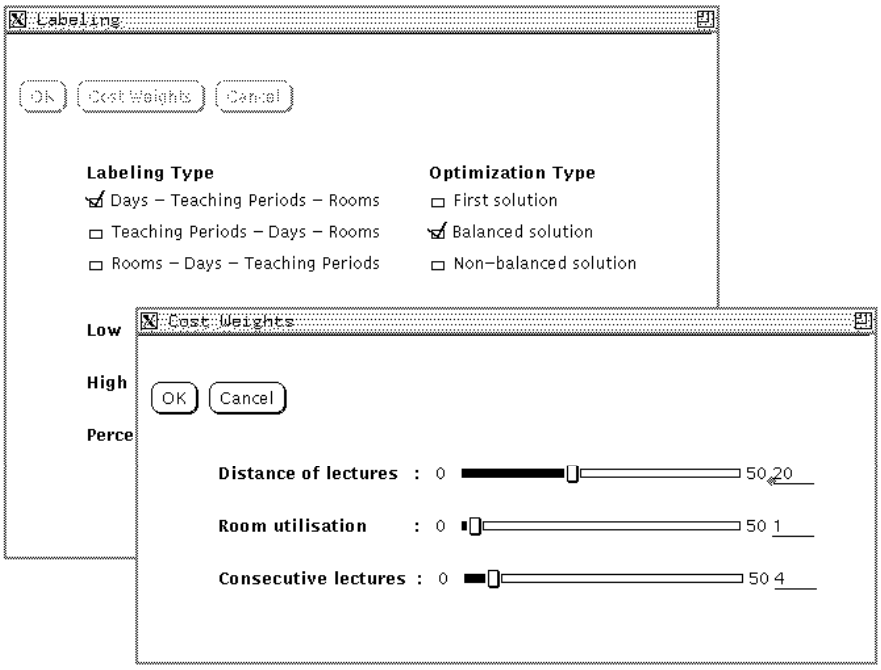

Figure 1 : Definition of control parameters

Another important feature of UTSE is the possibility provided to the user to experiment with different types of labeling and optimization, in such a way that the solution returned by the system is the one that is closest to his/her particular requirements. For instance, best room utilisation in a day (i.e. most rooms are equally used for teaching purposes) is achieved when the first or second labeling types (as referred to in the previous section) are selected. On the other hand, the third labeling type would proceed by exploiting each room to the full extent of its availability before trying to schedule lectures in other rooms. As far as the optimization type is concerned, if relatively small weight factors are assigned to the first and second optimization criteria, then the best solution returned will be mainly satisfying the third criterion, possibly leaving the other two partially unsatisfied. Furthermore, to reach a solution that only considers the third criterion while totally ignoring the other two, one could simply assign zero weight factors to the first and the second. Figure 1 shows a snapshot of the UTSE interface regarding the selection of the labeling and

optimization types as well as the definition of the weight factors. Table 1 shows a typical timetable for one day, as it is generated by UTSE.

Time Period	Classroom A	Classroom B	Classroom C
09:00 - 10:00	Calculus I (1st year-compulsory)	Electronics I (2nd year-compulsory)	Operating Systems (3rd year-compulsory)
10:00 - 11:00	Calculus I (1st year-compulsory)	Electronics I (2nd year-compulsory)	Operating Systems (3rd year-compulsory)
11:00 - 12:00	Algorithm Design (4th year-basic 2)	Discrete Mathematics (2nd year-compulsory)	Data Structures (1st year-compulsory)
12:00 - 13:00	Algorithm Design (4th year-basic 2)	Discrete Mathematics (2nd year-compulsory)	Data Structures (1st year-compulsory)
13:00 - 14:00	Computer Algebra (4th year-optional 1)	Calculus III (2nd year-compulsory)	Signal Processing (3rd year-basic 3)
14:00 - 15:00	Computer Algebra (4th year-optional 1)	Calculus III (2nd year-compulsory)	Signal Processing (3rd year-basic 3)
15:00 - 16:00	Differential Equations (4th year-optional 1)	Calculus III (2nd year-compulsory)	Linear Algebra (3rd year-basic 1)
16:00 - 17:00	Differential Equations (4th year-optional 1)	Parallel Computers (4th year-optional 2)	Linear Algebra (3rd year-basic 1)
17:00 - 18:00		Parallel Computers (4th year-optional 2)	Mathematical Logic (3rd year-optional 1)
18:00 - 19:00			Mathematical Logic (3rd year-optional 1)

Table 1 : Winter semester - Monday's schedule

With regard to the performance of the system at run-time, UTSE has been tested on a Sun SPARCclassic X and has yielded most satisfactory results, not only at respecting the user preferences that had been entered but also at reaching an optimum solution to the specific timetabling problem. In particular, the system has been used with a number of 37 subjects, with a total of 135 teaching periods in a week, 31 teaching staff members, 3 classrooms and 3 laboratories to hold lectures, which are the actual data from the DoI/UoA winter semester schedule. The constraints regarding teaching staff members' and rooms' availability have been defined in such a way that the corresponding real-life constraints are represented.

Whenever a solution exists, UTSE can return the first solution within reasonable time. If some optimization criteria have been specified, the time period required before reaching the final solution is highly dependent on the optimization type and bounds selected. Of course the same applies to the selection of a labeling type: the time performance and the efficiency of each order of labeling vary considerably. In general, one can expect that a first solution should be reached within 10 minutes. Experimentation with different input has lead to the conclusion that if a solution is to be found that satisfies the optimality requested, then the search process should be successfully terminated within 35-40 minutes. If more time elapses and no satisfactory solution has been found yet, the system is probably trapped in a search process which is unlikely to terminate within finite time. Such a case is rather common in problems where optimality proof (meaning proof that the solution found so far is indeed the optimal one) is difficult. Therefore the user is prompted to terminate the search procedure (through the system interface) and alter some of the initial preferences (labeling and/or optimization type).

5. Discussion

The CLP approach adopted for the development of UTSE is a programming framework that fits perfectly with the timetabling problem faced by the DoI/UoA, as well as with any other timetabling problem. The system is quite flexible both from the implementation and the user points of view. The flexibility in the implementation, which is actually based on the declarative style of programming supported by CLP, offers the possibility to easily add more constraints, which might be very different in nature from the existing ones. Moreover, other optimization criteria may be programmed and this can be done surprisingly fast. On the other side, the user is supplied with a graphical user interface through which many kinds of input data may be given. All the information that describes the data for the specific problem is user defined (or may be taken from an existing data store). The user has also the opportunity to give control information, such as the weights of the optimization criteria and the desirable labeling strategy.

Commenting on the other approaches that have been tried in the past to tackle the timetabling problem, the following have to be mentioned. The first attempts to create timetables automatically have been made with the exploitation of OR methods [Tri80, Meh81, FRo85]. These methods are applied to mathematical formulations of

the problem and lead to very good results, as far as efficiency is concerned. However, their major problem is that they do not facilitate the maintenance of the resulting code, since they lead to implementations of special purpose complicated algorithms in imperative languages. Another approach that has been recently used for the timetabling problem is based on genetic algorithms [CDM90]. It is a promising method which models the principles of evolution and has been used in the past for various kinds of combinatorial problems. As far as its applicability to the timetabling problem is concerned, quite good results have been produced. However, it is doubtful whether a timetabling program based on genetic algorithms is easily adaptable to many specific cases. Logic programming is an approach that overcomes the previously mentioned problems. It has been used as a platform to support the automatic construction of timetables [KWh92], but an even better programming framework is the CLP one. The application of CLP to timetabling is quite new. In [BDP94], the exam timetabling problem is considered and a solution is presented using CHIP. The results are satisfactory, however course scheduling is a more difficult problem and it seems that CLP is fruitful in this case as well. In [ABa94], this latter problem is described and the way it is solved using an extension of CHIP is presented. However, what has not been tackled so far and is covered by the UTSE system presented in this paper is the construction of optimum course timetables. The optimality refers to the generation of the best timetable, concerning some criteria of quality. In some other systems, the term "optimum" is used to denote a solution with a minimum number of constraint violations. In UTSE, all constraints are respected, both the internal ones and those stated by the user.

The UTSE system has been used to compose the timetables of the winter and spring semesters of the DoI/UoA. The performance is very good and the quality of the proposed solutions is of a very high degree. The timetables constructed by UTSE are similar or even better to the ones composed manually by the human timetabler so far and, what is more, in much less time.

As it happens with every real-world system, there is much space for improvements in UTSE. For instance, if the assumptions made for the development of UTSE were to be modified, then the system should be altered to satisfy any new requirements. Subjects could be split into weekly lectures in a particular way, or the constraints imposed concerning conflicts of lectures might be altered. Since ECL^{i}PS^e allows for such preferences and constraints to be stated explicitly, without affecting other parts of the system, such as labeling and optimization procedures, UTSE could be expanded with relatively little effort.

Furthermore, the optimization criteria that are being used in the evaluation of the cost function could also be modified. For example, another factor that might be

taken into consideration when evaluating the optimality of a solution, is concerned with the minimum movement of lectures that fall under the same teaching year and/or subject area from one room to another, so that students do not have to waste time going from one lecture to another.

Finally, a more complex adaptation that could be considered would be the construction of a timetable that satisfies some constraints and optimization criteria, but does not start from scratch. If some timetable already exists, the system should try to satisfy any new requirements by effecting as few changes to the existing timetable as possible.

To conclude this discussion, there are some points that have to be mentioned as well. The first is that the term "timetabling problem" refers to a general class of problems and cannot be defined precisely in a way applicable to every special case. The development of a universal timetable construction system would be indeed desirable, but is seems rather difficult that this might be achieved. The focus of attention should be to implement easily portable systems and this is feasible if the underlying technology facilitates it. The authors' opinion is that in this direction CLP is indispensable.

Another issue is whether there is a need for constraint relaxation in timetabling problems. This is completely dependent on the specific case under consideration. If the problem that has to be solved is much constrained, the likelihood of unfeasibility, and thus the need for constraint relaxation, is high. However, as it has been found for the DoI/UoA, it is not very probable that a feasible timetable cannot be constructed. The most common case is the existence of many solutions, the majority of which are rather bad. Consequently, the problem is to find the best solution, or at least a very good one, even if this is very well hidden inside the huge search space.

A final remark is that a current trend is to develop interactive systems for timetable construction [CdW89]. The reason for this is that sometimes humans do not like to be scheduled by machines. This is true mainly in cases where computers are not widely accepted. An advantage of the interactive systems is that the expert might assist the program to produce even better results. For example, for the construction of optimum timetables, it is not very easy to quantify the quality of solutions and, sometimes, a human timetabler might judge more reasonably whether one timetable is better than another.

6. Conclusions

In this paper, the well-known timetabling problem is discussed and a specific system, called UTSE, that constructs optimum timetables for university courses is presented.

UTSE has been implemented in ECL^{i}PS^e, a language that combines the logic programming paradigm with constraint satisfaction techniques. The timetabling problem, being a constraint satisfaction one, is mapped naturally to the constraint facilities provided by ECL^{i}PS^e. Moreover, the modelling of the problem under consideration profits a lot from the declarative style of programming which is supported by ECL^{i}PS^e.

The UTSE system that has been presented is a real-world one and is currently in use by the Department of Informatics of the University of Athens. It is a very flexible system, since it allows the user not only to define the basic problem data (subjects, teaching staff members, rooms, constraints, etc.) but also to control various run-time parameters. Such parameters include the weights of the optimization criteria, the order of variable enumeration, etc. All input data are given through a graphical user interface, thus providing easy access to the system even by the non-expert.

The performance of UTSE is quite satisfactory, considering that it is a program that has to run once or twice a year for the construction of the timetable of an educational organization. Some improvements which are currently under development will certainly enhance the system's functionality.

References

[ABa94] F. Azevedo, P. Barahona: *Timetabling in Constraint Logic Programming.* Proceedings of the World Congress on Expert Systems '94, 1994.

[BMc79] S. Bloomfield, M. McSharry. *Preferential Course Scheduling.* Interfaces, 9:24-31, 1979.

[BDP94] P. Boizumault, Y. Delon, L. Peridy. *Planning Exams Using CLP*. Proceedings of the 2nd International Conference on the Practical Applications of Prolog, pages 79-93, London, 1994.

[CdW89] N. Chahal, D. de Werra. *An Interactive System for Constructing Timetables on a PC*. European Journal of Operational Research, 40:32-37, 1989.

[CDM90] A. Colorni, M. Dorigo, V. Maniezzo. *Genetic Algorithms: A New Approach to the Time-Table Problem.* Lecture Notes in Computer Science - NATO ASI Series, Vol. F82, Combinatorial Optimization, pages 235-239, Springer Verlag, 1990.

[dWe85] D. de Werra. *An Introduction to Timetabling*. European Journal of Operational Research, 19:151-162, 1985.

[Dv+88] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier. *The Constraint Logic Programming Language CHIP*. Proceedings of the International Conference on Fifth Generation Computer Systems, pages 693-702, Tokyo, 1988.

[ECL93] *ECL^{i}PS^e User Manual*, ECRC, 1993.

[FRo85] J. Ferland, S. Roy. *Timetabling Problem for University as Assignment of Activities to Resources.* Computers and Operations Research, 12(2):207-218, 1985.

[FH+92] T. Fruhwirth, A. Herold, V. Kuchenhoff, T. Le Provost, P. Lim, E. Monfroy, M. Wallace. *Constraint Logic Programming - An Informal Introduction*. Lecture Notes in Computer Science, Vol. 636, Logic Programming in Action, pages 3-35, 1992.

[Got62] C. Gotlieb. *The Construction of Class-Teacher Timetables*. Proceedings of the IFIP Congress, pages 73-77, Amsterdam, 1962.

[Joh93] D. Johnson. *A Database Approach to Course Timetabling*. Journal of the Operational Research Society, 44(5):425-433, 1993.

[KWh92] L. Kang, G. White. *A Logic Approach to the Resolution of Constraints in Timetabling*. European Journal of Operational Research, 61:306-317, 1992.

[Law69] N. Lawrie. *An Integer Programming Model of a School Timetabling Problem*. Computer Journal, 12:307-316, 1969.

[Meh81] N. Mehta. *The Application of a Graph Coloring Method to an Examination Scheduling Problem*. Interfaces, 11:57-64, 1981.

[Tri80] A. Tripathy. *A Lagrangian Relaxation Approach to Course Timetabling.* Journal of the Operational Research Society, 31:599-603, 1980.

[vHe89] P. van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.

[YK+94] M. Yoshikawa, K. Kaneko, Y. Nomura, M. Watanabe. *A Constraint-Based Approach to High-School TimeTabling Problems: A Case Study*. Proceedings of the Twelfth National Conference on Artificial Intelligence AAAI-94, Seattle, 1994