University Automated Timetabling

Nuno Mamede and Pedro Soares
IST/INESC, Rua Alves Redol 9, Lisboa, Portugal

Automated school timetabling has been a goal for a long time, but the high combinations of this problem have maintained these systems in the academic research field, with very few successful commercial applications. To solve this problem several methodologies, such as scheduling, constraint programming, and genetic algorithms have been applied with relative success. This article describes an implementation of a timetable generator that uses constrained heuristic search, combining constraint programming with search techniques. This system was conceived to work in a university with seven thousand students.

Keywords: University timetabling, scheduling, constraint programming, real-life problems.

1. Introduction

School timetabling is a very complex task because of the interdependence of different sources of information: teachers, classrooms, classes, course curricula, etc. The time factor is also responsible for all the choices that loom when trying to assign or move lessons. The problem of assigning lessons to periods in such a way that no teacher or classroom is involved in more than one lesson at a time has been proved to be NP-complete. In face of this result it is indispensable to have good heuristics to help finding (at least) one of the "good" solutions.

Although being a hard problem, in every school school timetables are manually done by devoted handicrafters, who, during a lingering process, verify all the restrictions, producing at last the student, teacher, and classroom timetables. Sometimes, this manual process is unsatisfactory, mainly because the quantity of the information manipulated is huge, which implies infinite concentration and the use of many empirical rules. In a task with such characteristics the motivation and patience to come up with a better timetable are difficult to maintain.

This article describes a system for automatic school timetabling satisfying two goals: soundness (all incorrect solutions are eschewed) and quality. In what regards the first one, it is necessary to satisfy some rules such as: no teacher, student group,[1] or classroom is involved in more than one lesson at a time; a sequence of more than k hours of consecutive lessons without a break is not allowed; between noon and 2:00 PM a lunch break is mandatory; etc. The second goal (quality), is related with the absence of idle times between lessons for student groups and teachers; uniform occupation of the classrooms in the campus; similar schedules for student groups with the same courses; satisfaction of teacher preferences; etc.

Since in our university, timetables are done twice a year, it was decided to focus on the quality rather than on the speed of the process.

2. Problem definition

Timetabling in an educational setting covers a wide range of scheduling problems: the scheduling of students, teachers and other resources, in both time and space. This kind of problem is defined by a set of activities, a set of resources and a set of relations between them called constraints.

An activity is an operation that is characterised by being time-based (start time, end time, duration), resource-based (which and how many

[1] The expression student group will be used to refer to the concept of a group of students that share exactly the same lessons every week of a term (3 to 6 months).

resources are needed) and, possibly, having special restrictions (pre-conditions for execution).

A resource is an entity necessary for the execution of an activity. The most usual resource characteristic is its capacity, which restricts the number of activities the resource can process at a given time.

A constraint restricts the values that can be assigned to a variable. Constraints are applied to activities and resources to express their interdependencies. Constraints can be hard (have to be satisfied) or soft (will be satisfied, if possible).

Constraints found in scheduling problems include (Le Pape 94):

- temporal constraints: define the temporal relations between activities: precedence, simultaneity, etc. (Allen 83);
- capacity constraints: this class of constraints is related to the resources constraints (restrict the number of activities that can be processed at a given time).

In this perspective, it is possible to view a lesson as an activity, and classrooms, teachers, and student groups as resources. Each lesson needs one instructor, one classroom, and is associated with one (or more) student groups. The "impossibility of assigning the same resource to two or more lessons at the same time" and "two lectures (theoretical lessons) of the same course and student group have to be assigned to different days" are examples of constraints.

The goal of school timetabling is to assign to each lesson a start time, a teacher and a classroom (the student groups are known before the whole process starts), satisfying all the hard constraints and trying to optimise certain criteria. These criteria (soft constraints) define the pedagogic preferences and the preferences of the teachers, which should be taken into account to generate good timetables.

One important characteristic of this problem is that all the activities can be known before starting the scheduling process: examining the curricula of the courses and the number of student groups it is possible to compute which lessons have to be considered.

3. Techniques and solutions

The main techniques used in the scheduling problems have been developed in two scientific research fields: Operational Research and Artificial Intelligence.

Operational Research has provided frameworks based on approximate algorithms and it suffers from certainproblems such as lack of flexibility and relative by low quality of the result. Artificial Intelligence (Zweben and Fox 1994) has contributed to the study of this problem proposing new paradigms, much more flexible and reaching. Some of these new approaches include constraint programming and genetic algorithms.

Genetic algorithms are supported in the Darwin's species evolution theory and follow a repair approach (Johnston and Minton 1994) (Zweben et. al. 1994): at the beginning they generate a population of legal (or valid) timetables, that keep evolving (increasing its quality) during the computation.

Constraint programming (Kumar 1992) (Mackworth 1987), used in the system being described, is based on a symbolic representation of the following concepts: activity, resource and constraint, and in the use of constraints reduction of the possible combinations as timetables are being built. As an example, if two teachers are alternative in what regards lessons L1 and L2 at a given time, then if one of them is associated with lesson L1, the other teacher is automatically attached to the other lesson (L2). Using this technique the problem complexity is reduced as it is being solved.

Constraint programming

Constraint programming (Sadeh and Fox 1991) (Le Pape 1994) (Zweben and Fox 1994) (Fox, Sadeh and Baykan 1989) (Fox and Sadeh 1990) (Yoshikawa et. al. 1994) can be successfully used in problems of constraint satisfaction, where scheduling problems can be included.

One constraint satisfaction problem is defined by a set V of n variables $\{v_1, v_2, \ldots, v_n\}$, with each variable v_i associated with a domain D_i of possible values; and by a set C of m constraints $\{c_1, c_2, \ldots, c_m\}$, between those variables (the arity of each constraint is given by the number

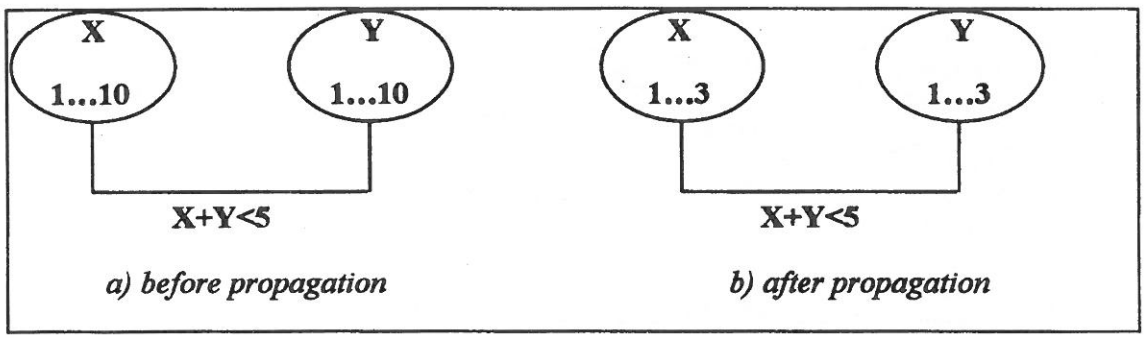

Fig. 1. Example of a constraint between two variables.

of variables it restrains). One solution consists of a tuple of n assignments (one for each variable) satisfying the set C (Kumar 1992).

A constraint satisfaction problem can be seen as a graph in which each node is a variable (with the corresponding domain), and each arc represents one binary restriction between the connected variables. It is possible to convert constraint satisfaction problems with n-ary constraints to another equivalent constraint satisfaction problem with binary constraints (Rossi, Petrie and Dhar 1989).

Consider the following example: two variables, X and Y, having the same domain, the interval between 1 and 10, and a constraint between them "$X + Y < 5$" (see Figure 1a). The constraint reduces the domains of both variables: the values that do not guarantee the satisfaction of the constraint are removed. As a consequence, the domains of these variables are automatically updated (see Figure 1b).

This procedure is used to ensure validity (Yoshikawa et. al. 1994), or in other words, in each moment of the computation, the domains of the variables only contain the values that can still be assigned to them.

As was previously said, this kind of problems also focus on the quality of the solutions, which means that preferences have to be modelled. In the timetable problemes, there is usually a great number of solutions, and the difficult task is to find out (in "limited" time) one of high quality. The adequate methodology has to conciliate constraint programming (legality) with heuristic search (quality).

4. School timetable modelling

As mentioned before, lessons are considered as activities, and teachers and classrooms as resources. Lessons have a temporal domain (possible start times), a pre-defined duration and a set of required resources. These requirements may be mandatory or alternative, which means that a lesson may need a specific teacher or one teacher from a set (of teachers). The same can be said about classrooms.

In what regards student groups, there are only mandatory requirements. Since lessons are generated at the beginning of this process, it is possible to know which student groups have to share a given lesson (usually a lecture).

Every resource maintains an occupation table that is updated as resources are assigned to lessons (an example in Figure 2). This table is used to detect bottlenecks where it is not possible to assign resources to the remaining lessons.

Other lessons relate constraints are temporal relations, such as: (i) same day; (ii) different days; (iii) same start time; (iv) different start times; (v) during; (vi) not overlapped; (vii) at the same time. Other constraints implemented are: (i) same teacher; (ii) different teacher; (iii) same classroom; (iv) different classroom.

These constraints are used to define a given problem. At the beginning when all activities have been generated and all the resources have been defined, the constraints are used to relate activities and to relate activities and resources.

The preferences, or in other words, the choice criteria enabling comparison of states, are used to guide the system towards high quality solutions. Examples of preferences have already

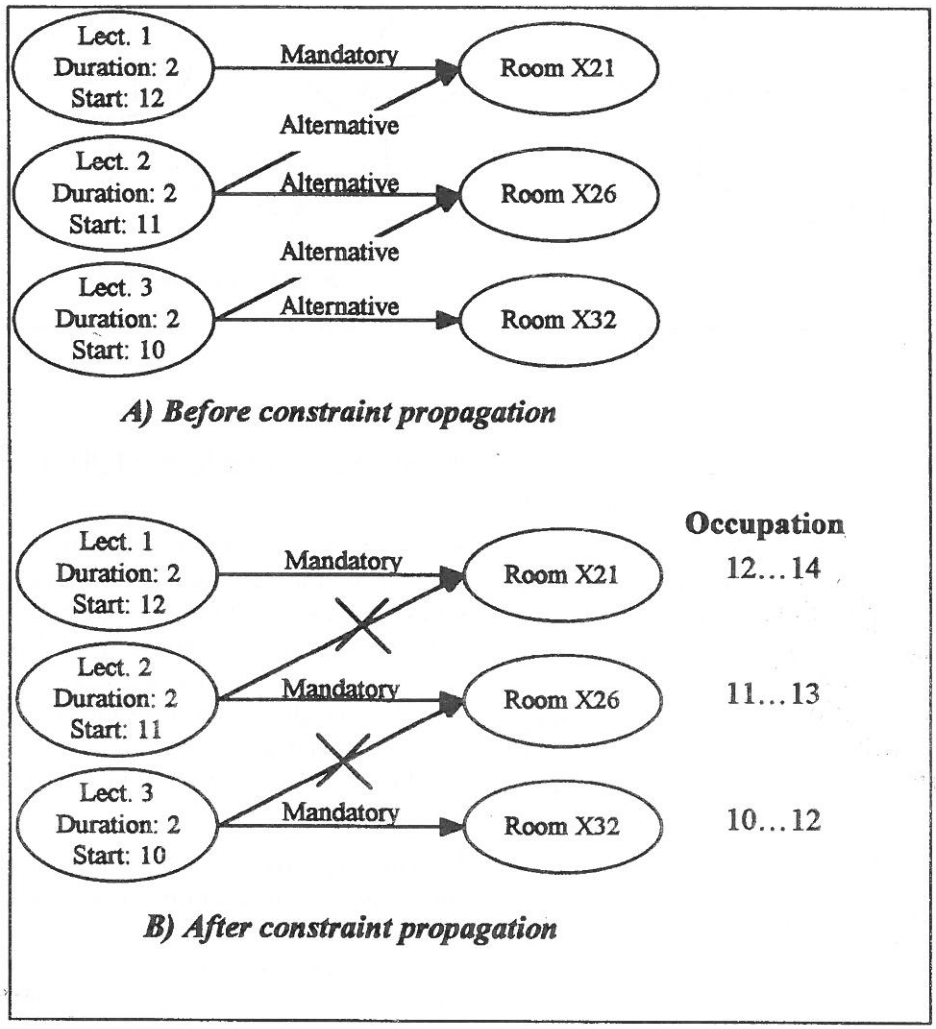

Fig. 2. Example of constraint propagation.

been described. In this system there exists a pre-defined set of preferences, and the user has to adjust their relative importance when solving a given problem.

Finally, the system is composed of several independent schedulers, one for each degree. Each scheduler is responsible for assigning the resources and a start time for all lessons of the corresponding degree.

5. Implementation

This system is designed to work in an automatic mode. First, the system collects all relevant data from an ORACLE database and then processes the information read, creating the schedules: student group schedules and resource schedules. A WWW interface was developed to allow the visualisation of all the produced schedules.

The development was made in the C++ programming language, with some specific libraries supplied by ILOG.

Main Algorithm

The program follows this sequence of steps:

1) generation of all lessons;
2) feasibility test;
3) scheduling.

In the first step, all lessons (for all degrees) are created. This process is oriented by the degree's curricular information and the number of student groups defined.

The feasibility test is necessary to find out if the schedules can be created with the data read. The

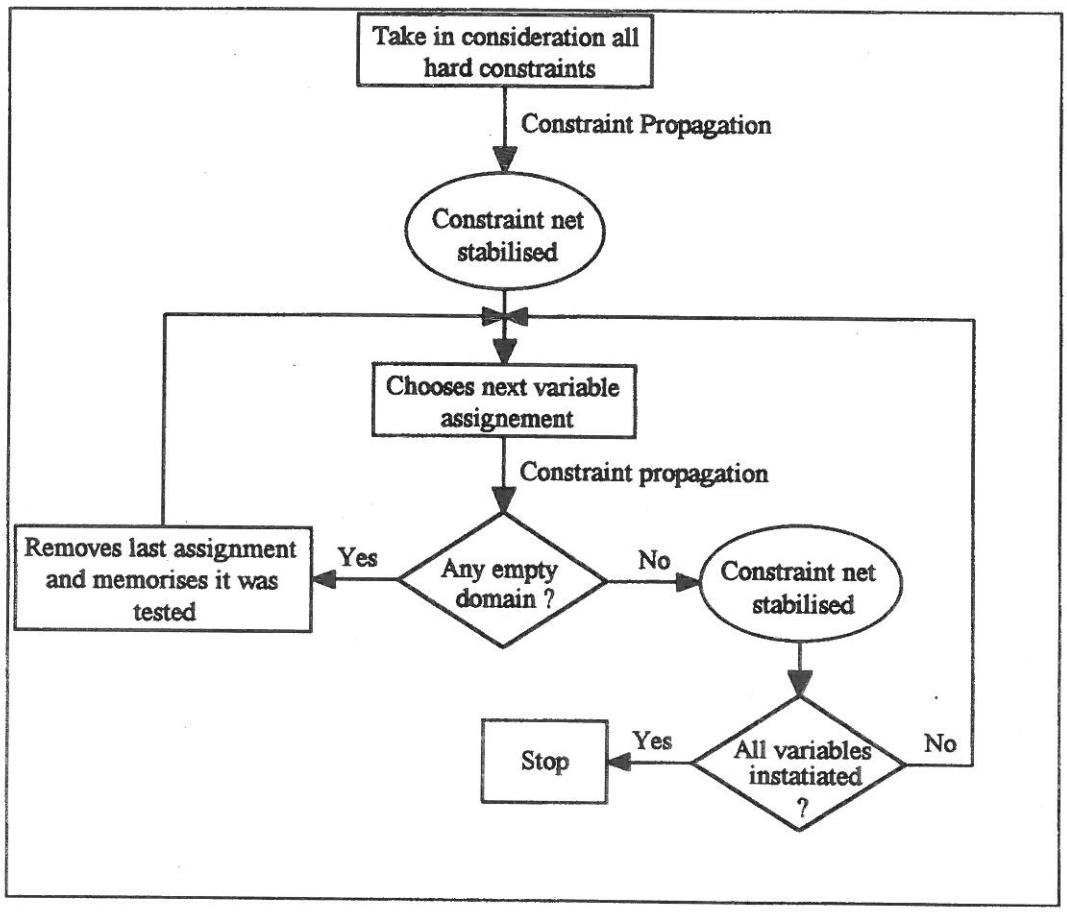

Fig. 3. Scheduling algorithm.

idea is to compare the sum of the duration of all lessons (demand) and the maximum available time of all rooms (supply).

If the demand is higher than the supply, the problem cannot be solved and must be redefined (reduce the number of student groups, make other rooms available, etc.).

If the test is surpassed, we can move to the next phase — scheduling (see Figure 3). In the scheduling step, we begin by representing all hard constraints. These constraints will be helpful to reduce the problem complexity, i.e., reducing the domains of the existing variables (start-times, possible-teachers, possible-rooms, etc.), as we continue with the computation.

When the constraint net is stabilised, we need to "make a move", i.e., choose what binding is to be made first, for example, to bind the start-time of one lesson. After this binding, some constraint will propagate and the net will stabilise again. This process finishes when all variables become bound.

If the domain of any variable becomes empty, it means that with the current set of bindings the problem cannot be solved, so, it is necessary to undo the last binding (chronological backtracking) and try another binding.

Because backtracking should always be avoided, it is necessary to implement heuristic methods of choosing the next binding to do. This heuristic method is responsible to guide the search and must consider two, sometimes incompatible, points of view: quality (good solutions) and efficiency (few backtrackings). Predicting backtracking situations is very hard. Usually they happen in resource bottlenecks, i.e., many lessons requiring the same resources. If we have a bad set of bindings we endanger a solution. This is the reason why it is so important to use look-ahead techniques to avoid bottlenecks.

One way to deal with this problem is to use Con-

straint Heuristic Search and the Micro-Boss algorithm (Sadeh 94). The goal of this algorithm is to use demand profiles to represent how a lesson requires a resource along the time. First, for any lesson requiring resources, one demand profile is calculated. Second, for any timetable resource, all demand profiles (regarding that resource) are aggregated and the most contented peak of all aggregated profiles is chosen because it represents the most critical bottleneck at that moment. Third, we identify the demand profile with the greater contribution to that contention peak, and that lesson is selected to make the binding.

In this way, the focus is always placed on the most constrained resource and the bindings that could provoke bottlenecks are done earlier.

Classroom-pools

When there is the intention of binding a classroom to a certain lesson there are usually many alternatives. Since the enumeration of alternatives is very costly, we have created classroom-pools to reduce the number of alternatives. This abstraction is used to represent groups of classrooms that are very similar in capacity.

The step "choose a classroom" is replaced by the steps "choose a classroom-pool" and "choose one classroom from the pool".

By using classroom-pools we don't loose much constraint propagation. For example: if the cardinality of a given classroom-pool and the number of lessons requiring a certain pool in a certain time is known, it can exclude certain lessons for that time.

Meals

Following this approach, the natural way to represent meals is to create an activity (a "fake" lesson, since all activities are lessons) that has to be included daily in all student groups' timetable. The set of lessons is extended to include meals, taking advantage of the constraint propagation that avoids activities (lessons) to be double-booked.

6. Conclusion

This paper presents and describes a real system that automatically creates university timetables. The scheduling part of the problem is solved using AI Techniques, namely Constraint Programming. These techniques allow a natural representation of the timetable problem and a consequent adaptation to several variations of the problem. The representation of preferences (soft constraints) strongly related to quality, can also be tuned to any particular problem.

Although this system was conceived to work in Instituto Superior Técnico (Lisboa), a strong effort was made to achieve a generic timetabling platform.

Future Work

This system can be complemented with the reparative approach, namely genetic algorithms. After the constructive phase, an optimising process could be initiated, using reparative methods. The repairs would start with a good and sound "seed" and eventually converge to a better solution.

Another possible improvement is to allow a manual control mode: instead of a fully automatic mode, we could allow the user to express his/her consent about each step of the timetabling process. When this "manual" mode becomes available, it will be possible to switch between the automatic mode (performs n allocations without the user interference) and the manual mode (each assignment needs the user's consent to be considered). An explanation module is being developed to assist the user.

Idle times between lessons in a student group timetable are easily recognisable after the end of the computation, but during the scheduling process they are hard (or even impossible) to recognise. The main culprit of this situation is the search method used, since we would have to know in advance when, in a given position of the timetable there would be an idle time. This problem is a consequence of the heuristic that handles idle times work with partial completed timetables. The reparative methods are better

at solving this problem since they manipulate completed timetables. We are trying to reduce this problem by using genetic algorithms.

Acknowledgement

This work was partially supported by Grant PRAXIS XXI/BM/6861/95 of Junta Nacional de Investigaçao Científica (JNICT).

References

ALLEN, J. F. (1983), "Maintaining Knowledge about Temporal Intervals", Communications of the ACM 26 (11).

FOX, M. S., SADEH, N. AND BAYKAN, C. (1989), "Constrained Heuristic Search". Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann Pub. Inc., San Francisco, CA, pp. 309–316.

FOX, M. S. AND SADEH, N. (1990), "Why Is Scheduling Difficult? A CSP Perspective". Proceedings of the 9th European Conference on Artificial Intelligence, John Wiley and Sons, NY, pp. 754–767.

JOHNSTON, M. D. AND MINTON, S. (1994), "Analysing a Heuristic Strategy for Constraint-Satisfaction and Scheduling". Intelligent Scheduling, ed. by M Zweben and M.S. Fox, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 257–289.

KUMAR, V. (1992), "Algorithms for Constraint-Satisfaction Problems: A Survey". AI Magazine, Vol. 13, Number 1.

LE PAPE, C. (1994), "Implementation of Resource Constraints in ILOG Schedule: A Library for the Development of Constraint-Based Scheduling Systems". Intelligent Systems Engineering, Vol. 3, pp. 355–66.

MACKWORTH, A. K. (1987), "Constraint Satisfaction". Encyclopaedia of Artificial Intelligence, Volume 1, S. C. Shapiro (ed.), John Wiley and Sons, New York, pp. 205–211.

ROSSI, F., PETRIE, C. AND DHAR, V. (1989), "On the Equivalence of Constraint-Satisfaction Problems", Technical Report ACT-AI–222–89, MCC Corp., Austin, Texas.

SADEH, N. AND FOX, M. S. (1991), "Variable and Value Ordering Heuristics for Hard Constraint Satisfaction Problems: An Application to Job Shop Scheduling", Technical Report CMU-RI, TR-91–23, The Robotics Institute, Carnegie Mellon University.

SADEH, N. (1994), "Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler", in Intelligent Scheduling, ed. by M. Zweben and M. S. Fox, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 99–135.

YOSHIKAWA, M., KANEKO, K., NOMURA, Y. AND WATANABE, M. (1994), "A Constraint-Based Approach to High-School Timetabling Problems: A Case Study", Proceedings of the 12 th National Conference on Artificial Intelligence, Vol 2, AAAI Press, Seattle, pp. 111–116.

ZWEBEN, M. AND FOX, M. S. (1994), "Intelligent Scheduling", Morgan Kaufmann Publishers Inc., San Francisco, CA.

ZWEBEN, M., DAUN, B., DAVIS, E. AND DEALE, M. (1994), "Scheduling and Rescheduling with Iterative Repair", Intelligent Scheduling, ed. by M Zweben and M.S. Fox, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 241–255.

Received: July, 1996
Accepted: October, 1996

Contact address:
Nuno Mamede, Pedro Soares
IST/INESC
Rua Alves Redol 9
1000 Lisboa
Portugal
e-mail: Nuno.Mamede@inesc.pt
e-mail: Pedro.Soares@inesc.pt

NUNO JOAO NEVES MAMEDE was born in 1958, received the BSc. (1981) and MSc. (1985) degrees in Electrical and Computer Engineering, and the PhD. (1992) in Artificial Intelligence, all by the Instituto Superior Tecnico (School of Engineering, Technical University of Lisbon). In October 1992 he was appointed as Assistant professor at the Instituto Superior Tecnico, and in 1983 as a researcher of Instituto de Engenharia de Sistemas e Computadores (INESC), where he has lead or participated in several research projects. His research interest is in artificial intelligence and natural language processing. He is one of the Organiser-chairs and Program-chairs of the Seventh Portuguese Conference on Artificial Intelligence, and a member of the editorial board of the "Advanced Manufacturing Forum" magazine (Trans Tech Publications).

PEDRO SOARES is a MSc. student at the Technical University of Lisbon where he received his BSc. degree. For the last three years he has been studying the timetabling domain. His research interests also include scheduling and intelligent agents.