Proceedings of the 1st WSEAS / IASME Int. Conf. on EDUCATIONAL TECHNOLOGIES, Tenerife, Canary Islands, Spain, December 16-18, 2005 (pp57-63)

A Software Tool for Lectures Timetable

DAMIR KALPIC, TOMISLAV RAJNOVIC, VEDRAN MORNAR
Department of Applied Computing
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb
CROATIA
http://www.zpm.fer.hr/kalpic/

Abstract: - A software tool has been developed as result of a graduation thesis. This tool has substantially helped in producing timetables for lectures at the postgraduate study, where the students enjoy high freedom of choice among the numerous offered courses. The aim is to produce a timetable of lectures such that respects all the limitations the lecturers have due to their other obligations, to avoid collisions from the standpoint of lecturers and to minimise the collisions from the standpoint of students. Genetic algorithm has been applied and the performances were very satisfactory.

Key-Words: - Education, Timetable, Genetic algorithm, Collision, Elective courses, Graduation thesis

1 Introduction

A complex timetable can be defined as a set of events scheduled in time when they should happen. The problem is omnipresent; from educational or medical institutions to public transportation.

An important aspect in organisation of education is producing the timetable of educational activities. If the list of courses that the students enrol is mostly predefined, the main problem is to assign the premises and lecturers to granules of time in order to keep the timetable feasible. The solution to that problem can be based on the pallet loading problem [2]. Such solution has been in full use at our institution for 15 years. However, the applicability is limited to undergraduate study, characterised by a multitude of about 4000 students with mostly deterministic enrolment.

Postgraduate study appears as a simpler problem because it involves about ten times less students. However, postgraduate students have no compulsory courses and they rather freely chose from a list of over hundred courses. Due to their smaller number, assignment of premises is of less concern and it can be solved manually. The problem of availability of lecturers is basically the same as in undergraduate study but somewhat more restricted, because the postgraduate study is superimposed on the already running undergraduate study. The main difficulty arises from attempts to avoid collisions from the students' standpoint. The number of students who cannot attend some lecture because they have simultaneously another one, should be minimised. It is a difficult combinatorial problem. The solution was attempted by genetic algorithm and it has resulted with very satisfactory and practically applicable results.

2 Problem Formulation

The university course timetabling has been already investigated and described. Doctoral theses have also been defended on highly respected universities, like [4] or [6]. The ambition of this paper is principally to describe a useful practical solution and to point out that from a graduation thesis [1] such a useful outcome may result. To describe the problem mathematically and for better understanding, some formulations deriving from [3] will be cited.

Construction of the timetable of lectures for students who can freely choose the courses is an optimisation problem under requirements describable by a triple (P, D, Z). $P= \{p_0, p_1, \ldots, p_N\}$ is a finite set of N courses that satisfy the condition that enough students are enrolled (in concrete case >5), so that the lectures will take place. $D= \{d_0, d_1, \ldots, d_N\}$ is a finite set of domains for the corresponding courses in P. Let $T= \{t_1, t_2, \ldots, t_M\}$ be the finite set of possible time granules when the lectures can take place. Accordingly, a feasible domain for each course can be a subset of all the available times, $d_i \subseteq T$. $Z = \{z_1, z_2, \ldots, z_N\}$ is a set of requirements to be respected. Now the problem of timetable can be described as the problem of optimum assignment (p_i ,t_j) for each course $p_i \in P$, in order that all the requirements remain satisfied. The search space is large and corresponds to M^N.

2.1 Constraints

In the considered problem two types of constraints are considered: the prohibited times for lecturers and the desirable times for lecturers. The assignment of lecture rooms was neglected as easily solved for a relatively small number of students. Prohibited times for lecturers come mostly from their obligation in the undergraduate study. External lecturers have obligations at their regular work.

2.2 Fitness Function

Collision avoidance is expressed as pairs of courses which can be lectured simultaneously without collision from the students' standpoint. Violation of this requirement will not mean that the solution is infeasible but it will be evaluated as less favourable.

3 Problem Solution

3.1 The Genetic Algorithm

For a problem with nonexistent algorithmic solution and with possibility to gradually progress towards the solution, genetic algorithm can be a good choice. A deeper insight can be obtained from literature, like [7] or [8].

The algorithm can be briefly described by the following pseudo code:

```
t = 0
Generate an initial population of potential solutions P(0);
Repeat {
    t = t + 1;
    Select a subset of better solutions P'(t) from P(t-1);
    Crossover the members from P'(t) and store the offsprings into P(t);
    Mutate the members of P(t);
    Evaluate the solution P(t);
} Until the prescribed conditions are met;
```

The solutions are represented by chromosomes and a proper choice of this representation is of most importance for the algorithm efficiency.

A chromosome represents a solution from the domain of feasible solutions. It is of constant length. Genetic operators must also be defined taking care not to produce infeasible solutions.

Most often, a binary chromosome representation is used. For timetables, the binary representation is not appropriate. The problem of timetable is reduced to assigning of courses $p_i \in P$ to a corresponding granule of time t_i when the lecture will take place.

A chromosome representation is possible as an array where the array index is the course and the array member value is the time granule code. The contents in position # 0 reveals the time granule when the course p_0 should take place, in position # 1 is stored the code of the time granule for the course # 2, etc.

3.2 Genetic Operators

3.2.1. Crossover

New solutions are produced by crossover in the simplest way as taking a half from one parent and the other half from the other randomly selected parent. If both parents represented feasible solutions, than their offspring is also feasible because it contains for each course only those time granules that respect the prohibited and the desirable time allocations.

3.2.2 Mutation

In order to avoid local optima or loss of a possibility to find a solution, mutation is used. For the considered problem a random mutation of chromosomes will be used and a given number of genes will mutate. Likewise as at crossover, infeasible solutions are not allowed as result of mutation.

3.2.3 Selection

The simplest selection, to sort the solutions according to its fitness is too simple and unsatisfactory. The 3-way tournament selection has been chosen. Three solutions are randomly selected and evaluated. The best two are crossed and their offspring replaces the worst of these three solutions. The advantage is that there is no abrupt division among generations and the process is continuous. Only new solutions have to be evaluated, there is no need for cumulative evaluations or sorting. All this improves the algorithm speed. The quality of elitism, i.e. persevering of the best solution, is also maintained.

3.2.4 Evaluation

Evaluation of the fitness function is most important for selection and perseverance of each solution. It

assures that the average quality of solutions in the population grows through generations. For the considered problem, the number of collisions is inversely proportional to fitness of solution. In every occasion when two courses are held simultaneously and there are students who enrolled in both, a collision is present. The number of students being affected by a collision gives the fitness function.
There is also a possible collision from the standpoint of lecturers. Their initial declared times when they are already occupied with other activities are taken into account while producing feasible solutions. However, collisions may be created in the process since some of them teach multiple courses. Collisions from the standpoint of students are unwanted but collision from the standpoint of a lecturer makes the solution unacceptable. Therefore, a weight of 100 is put to lecturer's collisions. If the fitness function reaches the value of zero, the solution is found.

Another reason to stop the algorithm is exceeding of the number of iterations. There have been situations where collisions could not be reduced to zero. One can assume that there had not existed a solution without collisions. In such cases the number of iterations caused the algorithm to stop. Practical experience has shown that for the considered problem, 200000 iterations were more than sufficient.

4 Software Implementation

The whole software implementation is the result of [1]. The programming language is C# and the Microsoft Visual Studio .NET 2003 framework was used. The input data were received from the Student Administration System [9] and stored in a local MS Access database. The necessary data consist of five relations:

- A table with students' code and name,
- A table with lecturers' code, name and position,
- A table with courses' code and name,
- A table connecting lecturers' codes with courses' codes,
- A table connecting students' codes to enrolled courses' codes.

4.1 Algorithm Parameters

Any genetic algorithm requires some parameters like the population size, number of iterations and probability of mutation. Choice of these parameter values may represent an optimisation problem by itself.

5 Example of Use

A few screens were captured in order to briefly illustrate the program features.
The screen in Fig. 1 contains desirable times for certain courses. This feature is not much popularised and the lecturers are asked to submit the data about their prohibited times instead. This is shown in Fig. 2. The Fig. 3 contains default values for the optimisation parameters, which can be changed. Fig. 4 is a part of the resulting timetable with 10 unresolved collisions. Manual editing is allowed and in that case possible changes that do not worsen the solution are highlighted.

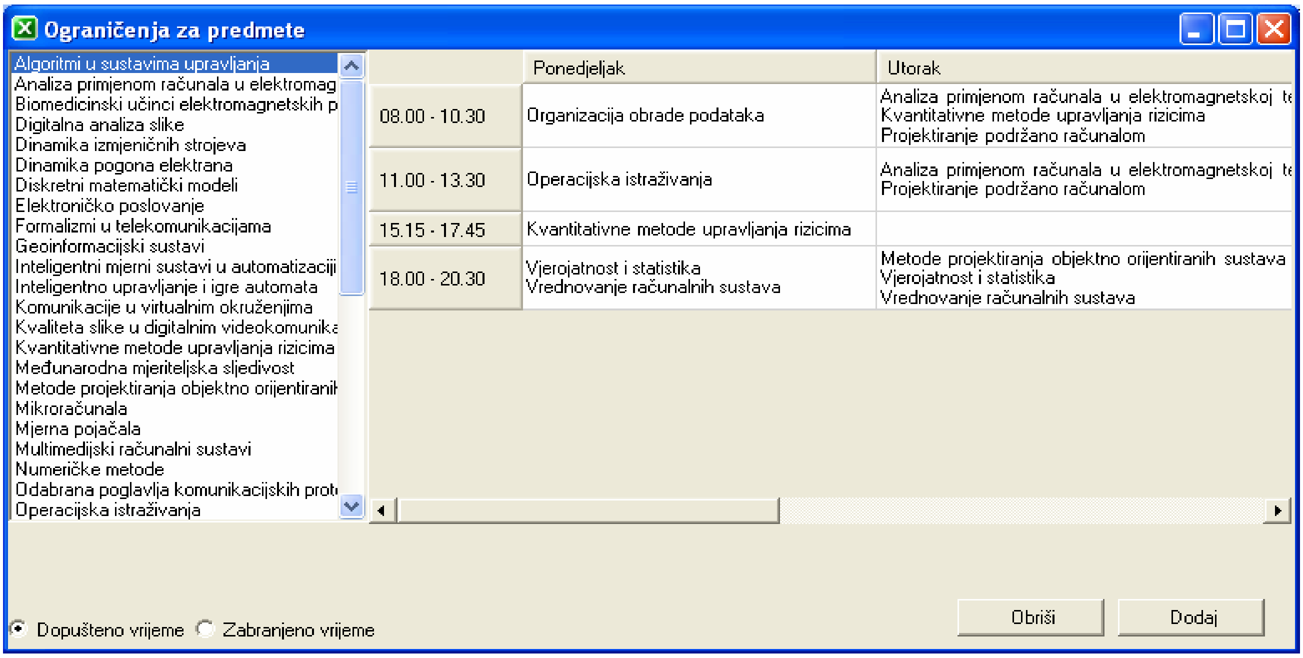

Fig. 1. Desirable times for courses

Some explanations:

In the left hand side of the screen is the list of all the involved courses. In the right hand side, a part of the desirable timetable is currently visible, for Monday (Ponedjeljak) and for Tuesday (Utorak).

- “Dopusteno vrijeme” means “Allowed (or desirable) time”.
- “Zabranjeno vrijeme” is “Forbidden time”.
- “Obrisi” means “Delete”.
- “Dodaj” means “Add”.

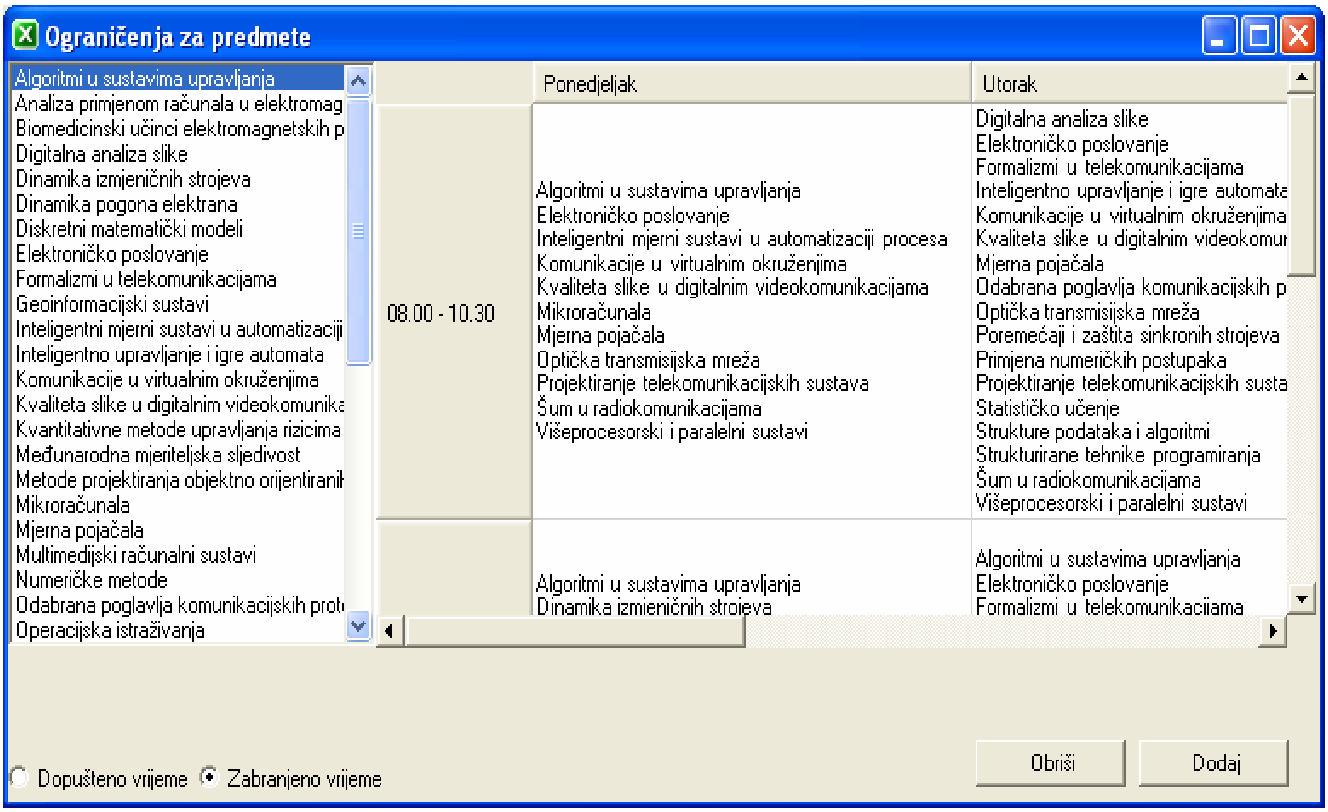

Fig. 2 Forbidden times for courses

The screen is the same as for Desirable times, but the number of data is much more abundant because it reflects the already existent fixed obligations of the lecturers, due to their other activities, primarily lecturing on the undergraduate study.

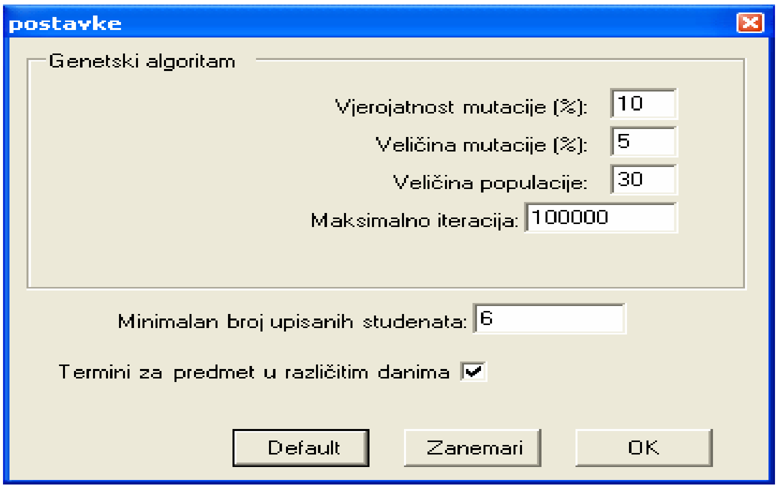

Fig. 3 Default parameter values

For the considered problem empirically determined default parameters are:

- Probability of mutation 10%
- Maximum percentage of mutated genes 5%
- Population size 30
- Maximum number of iterations 100000
- Minimum number of enrolled students to schedule the course (Faculty rule is 6)
- Two terms of lectures cannot be within the same day.

These values can be changed, but the above defaults can be restored.

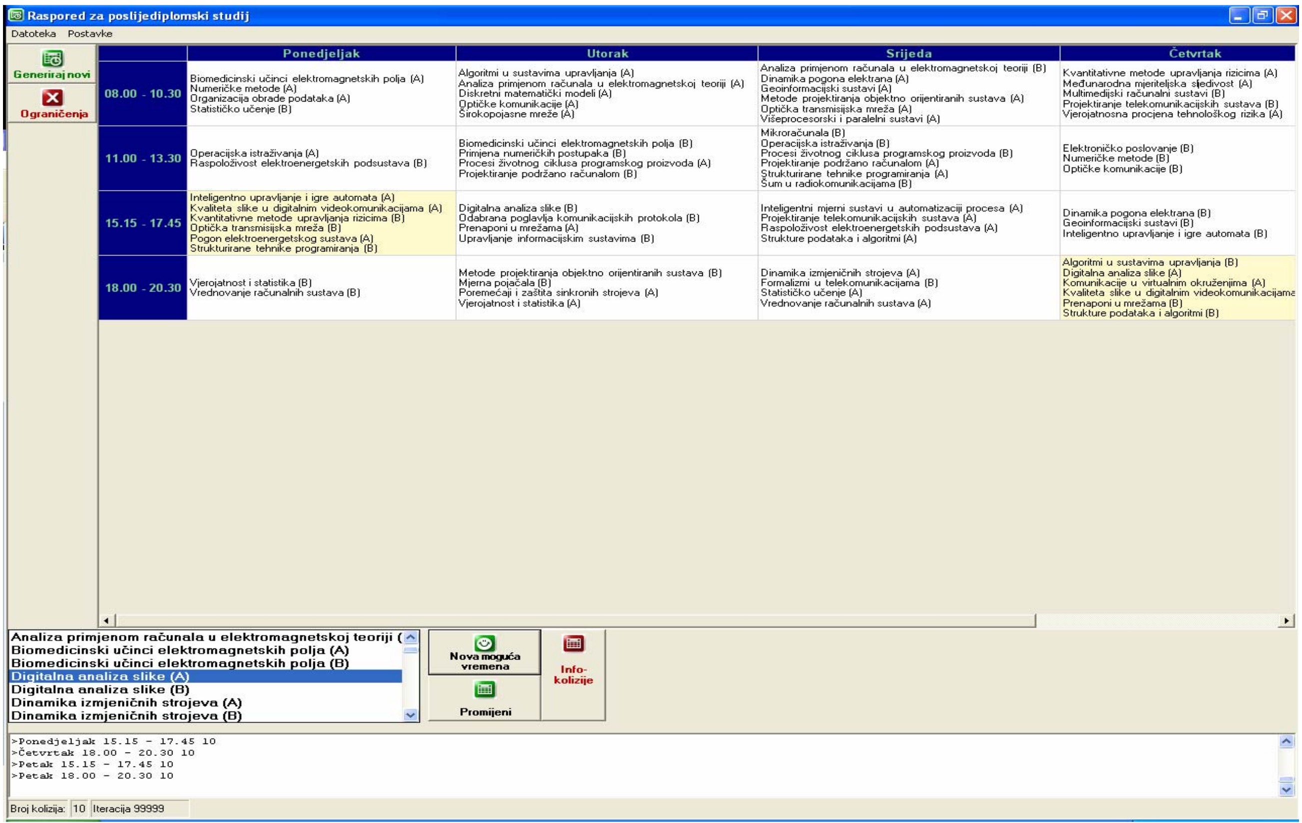

Fig. 4 Resulting timetable with highlighted alternative lecture times for the course “Digitalna analiza slike”

Slightly highlighted fields mark the positions in time where the selected course “Digitalna analiza slike” can be manually relocated without introducing new collisions.

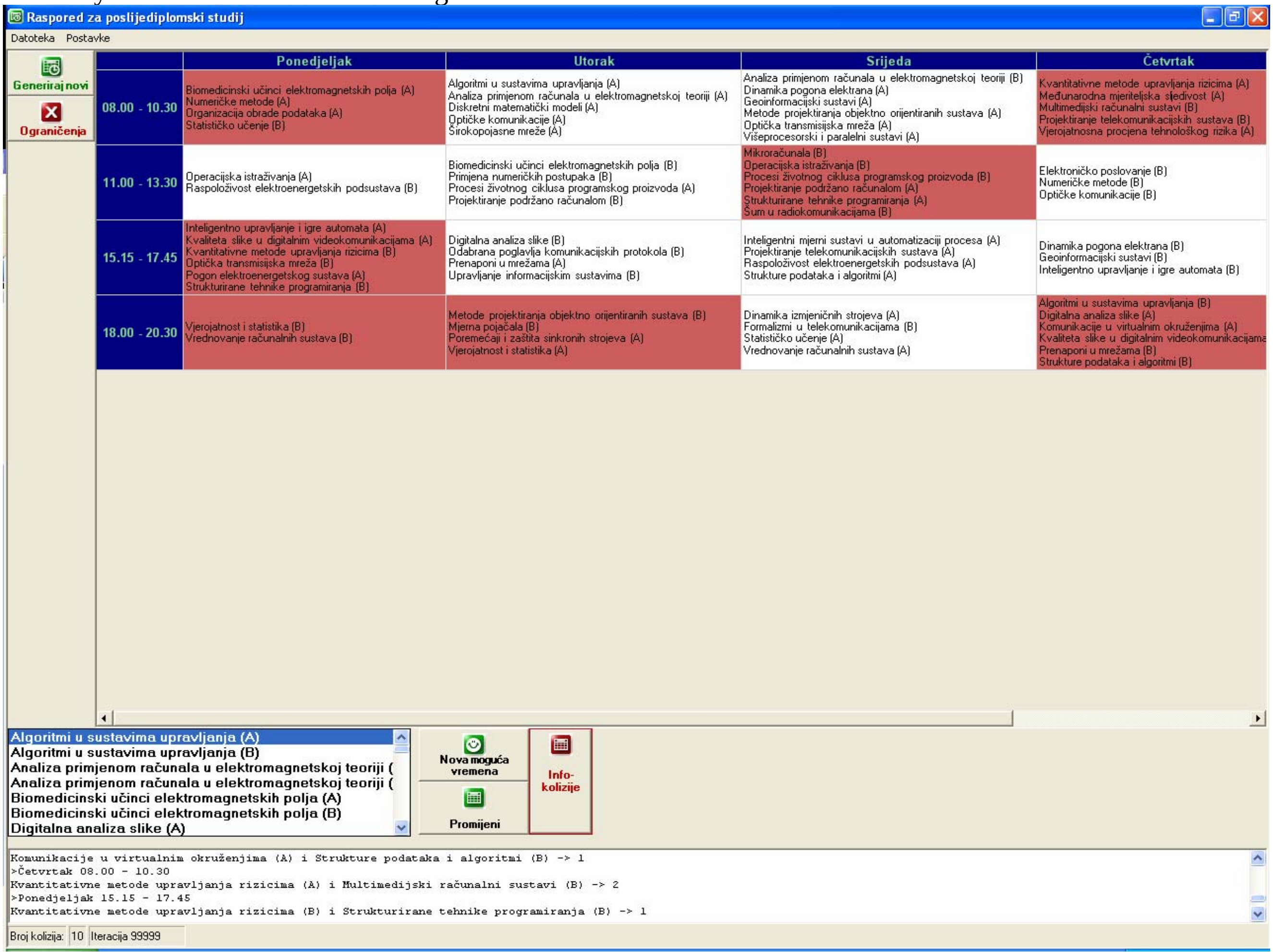

Fig. 5 Highlighted fields mark collisions which (10 of them) are listed at the bottom of the screen

Duration of the optimisation was 30 seconds on a PC with Pentium 4 CPU 3.00 GHz, 1 GB of RAM.

6 Conclusion

This paper mostly resulted from a graduation thesis guided by the first and defended by the second author. It has shown substantial practical value. In the autumn and the spring semester of the academic year 2004/05 at the Faculty of Electrical Engineering and Computing timetables were produced for the postgraduate study, resulting with minimum dissatisfaction from both sides; lecturers and the students. All the lecturers’ prohibited times and in some cases even desirable times were respected. There were only a few collisions left, even if up to 6 courses were taught simultaneously. The only reason for criticism is that there is currently no built in feature to start a new optimisation in vicinity of the existent one. This possibility should be built in some future version because producing of such a timetable involving about 50 lecturers is in practice an interactive task. Even if the lecturers declare their prohibited times by e-mail in advance, after the timetable is produced and displayed on the Faculty intranet, some of them are unsatisfied because they forgot to mention some of their prohibited time granules. In order to please them, their constraints are added and a new timetable is produced. This new timetable is often very different from the previous one, what causes dissatisfaction among some of the others who had liked the previous version. The experience was that about 8 iterations were necessary to obtain an acceptable solution. Without such a tool it would be a very difficult task.

References:

[1] T. Rajnovic: *Programsko pomagalo za izradu vremenskih rasporeda uz minimalnu koliziju (Software Tool for Construction of Timetables with Minimum Collision)*, Faculty of Electrical Engineering and Computing, University of Zagreb, Graduation thesis mentored by D. Kalpic, Zagreb, September 2004

[2] V. Mornar: A Heuristic Approach to a Class of the Pallet-Loading Problems, *ITA* vol.10, No. 1-4, 1991, pp.137-146

[3] Alkan, E. Özcan, Memetic Algorithms for Timetabling, *Proc. of 2003 IEEE Congress on Evolutionary Computation*, pp. 1796-1802, December 2003

[4] M. Marte, *Models and Algorithms for School Timetabling – A Constraint-Programming Approach*, Doctoral Thesis, Ludwig-Maximilians-Universität, München, 2002.

[5] K. S. Aggour, A. Moitra, Advances in Schedule Optimization With Genetic Algorithms, *General Electric Global Research*, 2003

[6] M. Bartschi Wall: A Genetic Algorithm for Resource-Constrained Scheduling, Doctoral Thesis, Massachusetts Institute of Technology, 1996.

[7] D. Whitley, A Genetic Algorithm Tutorial, Tech. Rep. CS-93-103, Department of Computer Science, Colorado State University, Fort Collins, CO 8052, March 1993.

[8] T. Back, Optimization by Means of Genetic Algorithms, 36th International Scientific Colloquium, Technical University of Ilmenau, 1991, pp16-169

[9] D. Kalpic, M. Baranovic, V. Mornar, S. Krajcar: Development of an Integral University Management System, International Conference on System Engineering, Communications and Information Technologies, ICSECIT 2001 Proceedings, Universidad de Magallanes, Punta Arenas, Chile, 2001.