Optimization of student personal course schedules with
Evolutionary Algorithms

Panagiotis Adamidis, Georgios Kinigopoulos
Dept. of Informatics,
Alexander Technological Educational Institute of Thessaloniki,
Thessaloniki GR-57400, Greece

Abstract
In this paper we present the optimization part of a course registration system through the web (EvoWebReg). The system consists of three parts. The first one is a web application which allows the students to submit their course preferences to the system’s database, through Internet. The second part is an administrative tool which controls the whole system and allows its smooth operation. The third part is an evolutionary algorithm which is responsible for the optimization of the student course schedules. The schedules are optimized according to the student’s submitted preferences, and the constraints imposed by the department. The results of the experimental tests of the evolutionary algorithm prove that our initial objectives to provide an open generic and effective tool, which can satisfactorily implement the course registration procedure, were achieved.

KEYWORDS
Optimization, Evolutionary Algorithms, Web course registration, educational timetabling.

1. INTRODUCTION

According to Carter and Laporte (1998) the course scheduling problem can be decomposed into the following sub-problems: course timetabling (assign courses or course sections to time periods satisfying some requirements and/or constraints), teacher assignment (assign teachers to courses maximizing a preference function), classroom assignment (events must be assigned to specific rooms to satisfy some criteria), and student scheduling (arises when courses are taught in multiple sections). The course timetabling systems are also distinguished as Master Timetables or Demand Driven, with their primary difference lying in the sequence in which the various sub-problems are being solved.

In this paper we consider the student scheduling problem with the Master Timetable approach, adapted to the requirements and characteristics of the Dept. of Informatics of ATEI Th. Although we discuss a specific case study, most methods are general enough and should be applicable to modeling and solving other instances of the problem. In our case study, a semester is the academic time unit. Course timetables and student sectioning must be done in the beginning of every semester.

A course can be either simple or mixed. A course is considered simple when it is taught only by lectures in big enough lecture rooms, and mixed when it is taught by lectures (theoretical part) and laboratory exercises (lab part). When a course is mixed, in order for a student to pass the course he/she must enrol in both parts of the course and succeed a passing mark in both parts. A simple course or the theoretical part of a mixed course is taught by lectures in rooms that are large enough to accommodate all enrolled students. It’s very rare to have two sections of a simple course. On the other hand, all laboratory rooms have limited capacity which is usually much smaller than the number of students registered in the theoretical part of the course. So the students of most mixed courses must be separated into lab groups (course sections for the lab part). The available number of lab groups for each mixed course is decided before the timetable creation, according to the number of students who are anticipated to enrol. Each lab group is assigned the following data: lab room, day of the week and time, teachers (1 or 2). The course timetable which is constructed and announced at the beginning of each semester contains the above data.
Every student prepares his own personal schedule selecting not only courses, but lab groups for the mixed courses as well, according to the already announced timetable. Every student’s schedule must satisfy a number of rules and constraints set by the department and by the available facilities (room number and sizes), and also by the student’s own preferences, concerning days, time and staff members.

The program of study of the department includes a limited number of lab rooms with specific places. The places are equivalent. The lab groups are considered as collections of places. Each place must be assigned to an appropriate student.

2. PROBLEM DESCRIPTION – CASE STUDY

As already mentioned, in this paper we consider the student scheduling problem with the Master Timetable approach, adapted to the requirements and characteristics of the Dept. of Informatics of ATEITh.

In our case study, a semester is the academic time unit. Course timetables and student sectioning must be done in the beginning of every semester. Studies last for 8 semesters.

A course can be either simple or mixed. A course is considered simple when it is taught only by lectures in big enough lecture rooms, and mixed when it is taught by lectures (theoretical part) and laboratory exercises (lab part). When a course is mixed, in order for a student to pass the course he/she must enrol in both parts of the course and succeed a passing mark in both parts. A simple course or the theoretical part of a mixed course is taught by lectures in rooms that are large enough to accommodate all enrolled students. It’s very rare to have two sections of a simple course. On the other hand, all laboratory rooms have limited capacity which is usually much smaller than the number of students registered in the theoretical part of the course. So the students of most mixed courses must be separated into lab groups (course sections for the lab part). The available number of lab groups for each mixed course is decided before the timetable creation, according to the number of students who are anticipated to enrol, judging from the previous semester. Each lab group is assigned the following data: lab room, day of the week and time, teachers (1 or 2). Although not necessary, usually lab groups for the same course are assigned on different days and times.

The course timetable which is constructed and announced at the beginning of each semester contains the above data.

Every student prepares his own personal schedule selecting not only courses, but lab groups for the mixed courses as well, according to the already announced timetable. Every student’s schedule must satisfy a number of rules and constraints set by the department and by the available facilities (room number and sizes), and also by the student’s own preferences, concerning days, time and staff members.

The program of study of the department includes a limited number of lab rooms with specific places. The places are equivalent. The lab groups are considered as collections of places. Each place must be assigned to an appropriate student.

3. COURSE SECTIONING: HISTORICAL PERSPECTIVE

The first attempts to solve the student scheduling problem were trying to produce conflict-free schedules and balance course section sizes. The algorithms proposed by Abell (1965) and Winters (1971), often produced solutions with invalid schedules. Faulkner (1965) introduced the idea of preference instead of only choosing a course.

Macon and Walker (1966) used a “Monte Carlo” algorithm to examine the case of assigning students to a fixed course schedule. Some of the courses were divided into sections and section choices for assignment must be made. Their algorithm was trying to reduce conflicts and balance the load between sections. A more stochastic algorithm was used by Busam (1967). They compared a non-preference and a preference (ordering of sections within the courses) algorithm and they concluded that both algorithms meet the requirements of machine sectioning equally well.

Later the approaches of Colijn (1973) and Miyaji et al. (1981) produced better results. Colijn’s algorithm produced conflict-free schedules, while preserving the balance between sections and satisfying some requirements. The approach of Miyaji et al. was based on goal programming, and was tested successfully on realistic examples. Sabin and Winter (1986) proposed the use of two programs using “greedy” algorithms.
Laporte and Desroches (1986) treated the student scheduling problem as an optimization problem satisfying a number of criteria. Some of the criteria were treated as constraints, while others were considered as objectives. They introduced the use of hard and soft constraints and used a “branch and bound” algorithm divided into the three phases (constructing student schedules, balancing section enrollments, and respecting room capacities), but it did not give students the possibility of section preference.

Aubin and Ferlan (1989) used the “Master Timetable” approach. First a timetable is generated and then students are assigned to course sections. They used the integer programming model improving the objective by working successively on the timetabling and the grouping sub-problems until a solution that cannot be improved, is reached. This approach was also used by Hertz (1991) with tabu search techniques to handle sub-problems. Her innovation was to accept as valid, schedules in which courses shared common teachers or students. Tabu search was also used by Alvarez-Valdes et al. (2000) in a two step student registration system. Initially students select courses but not sections, according to their priority (depending on their record and seniority).

Carter (2001) describes a system based on a “demand-driven” approach where students first choose their courses, and the system tries to find the best timetable to maximize the number of satisfied requests. The problem is decomposed into sub-problems which are solved using a greedy heuristic to assign times to sections, and a Lagrangian relaxation algorithm to assign classrooms.

Burke et al. (2004) gave that for each course there was only one lecture per week. The problem was modeled as an edge-coloring problem in a bipartite graph with students represented as left nodes and teachers by right nodes. Course sections (collections of time periods and resources) are represented by edges between left and right nodes. The approach was taking into consideration the balance of sections, the student preferences and the policy of the institute.

Amintoosi and Haddadnia (2005) proposed a new fuzzy sectioning algorithm. They use fuzzy clustering, a fuzzy evaluator and a novel feature selection method. Students are classified by a Fuzzy c-Means classifier after best feature selection for sectioning. A fuzzy function evaluates the produced clusters based on section balancing and similarity of student schedules within each section.

The most integrated solution was proposed by Murray et al. (2007). They introduced a university wide real time sectioning system for Purdue University where optimization happens at real time. The timetable is created and most students are sectioned based on demand data in student schedule requests, except from the beginning students, since their requests are unknown. Timetable construction considers actual course demand from students who have submitted schedule requests plus the projected demand for students who are anticipated to enroll.

4. SCHEDULING SUB-SYSTEM AND EVOLUTIONARY ALGORITHM

The operation of the system starts with the public announcement of the master timetable on the website. The timetable includes the usual necessary information in order for the students to be able to construct their personal schedules.

After the system accepts the student preferences, an evolutionary algorithm undertakes the responsibility of course sectioning (scheduling students to different lab groups).

The evolutionary algorithm produces the final personal schedules for each student. The results apart from the feasibility of the solution will also comply with the departmental policy. The requirements of the system, that concern the scheduling algorithm, are the following:

1. A student is assigned only to one lab group of the same course.
2. A student is assigned only one place of a lab group.
3. A student is assigned only one lab group at the same day and time (timeslot).
4. A student declares a course only when he has passed the prerequisite courses.
5. Industrial placement is declared only when all prerequisite courses are passed.
6. Final year thesis can be assigned only after a specific semester.
7. The number of TU that is assigned to a student is limited, depending on their seniority.
8. In special cases the TU of assigned courses are ignored.
9. No overlap of theoretical part and lab part of a course is allowed for a student, without the consent of the student.
10. Lab preferences are mandatory for all students.
11. The system supports lab rooms with different capacities.
12. The length of lecture times must be an integer number.
13. The system must fill as much as possible the available lab places.
14. Students have priority over the courses taught on their typical semester (typical semester is the number of semesters that have elapsed since they first registered in the Department).
15. The system provides the students with the capability to give a preference order of the lab groups for each and every course that they want to enrol (basic preferences).
16. The system gives the capability of declaring redundant courses as well. These will be used in case that, basic preferences cannot be assigned.
17. The system gives bonus to students according to their performance over the previous semester (do not withdraw from lab groups, and have better marks) and over all passed semesters (higher mark average)

The requirements 7, 8, and 9 are checked by the system when the students declare their preferences.

We’ve chosen to use evolutionary algorithms (EAs) as a search and optimization technique that can effectively search the space of feasible solutions and find the best of them. Moreover the quality function of EAs can easily accommodate new optimization criteria that can emerge in the future.

The evolutionary algorithm (EA) operates and communicates with other parts of the system as follows:
- Step 1. The root island starts operating. It communicates with the database to acquire all necessary data. The data are saved as objects in the main memory of the node that hosts the root island.
- Step 2. The root island caters for the initialization of the other islands. It is responsible for a) the creation of the number of islands defined in the parameters of the EA, and b) the initialization of the inhabitants (individuals) of the new islands with the data acquired from the database.

Requirements 4, 5, 6, and 10 are considered hard constraints and they must be satisfied in order for the solution to be valid. The requirements 16-20 are considered soft and are taken into consideration by the quality function of the EA. Other requirements are implemented in the others parts of EvoWebReg system.

### 4.1 Chromosome representation and Evaluation

Since the lack of sufficient lab rooms and accordingly lab places is the major problem of the department, this problem is central in designing the appropriate representation. A lab place can be assigned to a student or remain empty. The data that distinguish each lab place are: course, lab group of the course, student to be assigned the lab place, and student preference for this lab group. The number of lab groups is limited and defined in the Master timetable. The final chromosome is depicted in Fig. 4. It consists of a series of the lab part of mixed courses.

![Fig. 4 Chromosome representation](image-url)
Special care has been given to initialization which is biased towards the first lab group preference of the students.

The quality of an individual equals to the sum of the qualities of every gene: 

$$indivFitness = \sum_{i=0}^{N} G_i$$

(1)

where \(N\) is the number of genes, and \(G_i\) the quality of gene \(i\).

We distinguish between two cases for each gene: a) the lab place that corresponds to a gene is assigned to a student and b) the lab place remains empty. In the first case the quality of the gene is:

$$G = penTyp + p \cdot prefNo + m \cdot (11 - \text{avg}) + d \cdot \left( \frac{\text{regTU}}{\text{passTU}} \right) + \text{penWD} + \text{penRed}$$

(2)

where \(p\), \(m\), and \(d\) are some weight coefficients, set by the administrator, depending on the policy of the department. For example, if the policy is to reward the students with good performance in the previous semester then the weight of \(d\) must increase.

\(penTyp\) is a penalty concerned with the typical semester of the student. According to the policy of the department the students have absolute priority over the courses of their typical semester. If places are left then priority is conveyed upon the students of semester “\(\text{typical}+1\)”, then if there are still available places priority is conveyed to students with typical semester after the \(7^{\text{th}}\), and then to the rest of the students independent of semester.

\(prefNo\) is the preference number of the student for this lab group, \(\text{avg}\) is the average mark of all the courses that the student has passed successfully, \(\text{regTU}\) is the number of teaching units (TU) registered on previous semester, \(\text{passTU}\) is the number of TU passed successfully on previous semester, \(\text{penWD}\) is the penalty for a student that withdraws from a lab group in the middle of the semester, and \(\text{penRed}\) is the penalty for the use of a redundant declaration.

In the second case (the lab place remains empty) the quality of the gene is: 

$$G = \text{emptyPlace Penalty}$$

4.2 Genetic operators, mechanisms and parameters

We need to design appropriate reproduction operators (recombination & mutation) in order to keep solutions valid. We’ve implemented four different recombination operators: synchronous (1), asynchronous (2), independent (3) and place-based (4). All recombination operators affect a block according to a probability (recombination per block rate). In the first three operators, a block corresponds to a lab group. In the forth, it corresponds to a lab place.

The operators are applied in two phases. In the first phase, these operators recombine characteristics of the two parents, blocking exchanges that violate the validity of the chromosome. In the second phase the empty places are filled with appropriate students, i.e. students that have declared these lab groups in their preferences. The operators differ in the first phase and mainly in the way that they deal with exchanges which will result in invalid chromosomes. When the operators “synchronous” and “place-based” find an invalid exchange, they keep the previous values. The “independent” leaves empty places and “asynchronous” is a combination of the other operators.

We’ve implemented two mutation operators which are based on the same general idea: replacement of a student in some lab group place with another student that has also shown his preference on this lab group. Of course the mutation operators must check and verify that this replacement does not violate the validity of the chromosome. The first operator makes the student replacement without checking anything else. The second mutation operator does not allow replacements that deteriorate the fitness of the chromosome.

Elitism is implemented as follows: the best parent can replace the worst of fspring and not only a randomly selected one. Moreover the new mechanism was extended to support elitism for more than one individuals from the parent population.

The termination criteria of the EA are: maximum number of generations, or best (or goal) solution found, or maximum execution time, or best individual does not improve for a number of generations.

The system also offers the possibility of immigration of individuals with either constant migration probability or an alternative that increases the selection intensity (Cantu-Paz 2001).
5. DATA DESCRIPTION, EXPERIMENTS AND RESULTS

The scheduling system was tested on an experimental database created according to the courses offered by the Dept. of Informatics. It includes 51 courses, 14 simple courses and 37 mixed courses. The lab part of the 37 courses is separated into 113 lab groups. Each lab group has at most 25 places. The total number of available lab places in all mixed courses is 2825. The master timetable of the department offers courses in five days, and every day has 12 time periods (from 08:00 to 20:00).

The total number of students is 800 who are distributed in 10 semesters. The number of declared lab group places is 2845 (20 places more than the available). Considering the student preferences, the fact that the demand for places in some lab groups was extremely high and in others extremely low, and that the students did not declared their interest for 281 lab places, the maximum number of lab places that could assigned to students was 2544. Also, in some mixed courses the available places were more than the required (31 lab places would anyway stay empty), and the EA could not assign more than 2796 lab places to students in any case. From the previous semester data, we found that the students had withdraw from 347 lab places in the previous semester.

A factor of success of the students in the previous semester (TU passed against registered TU) was also taken into consideration.

The preferences of the students were randomly created using student declarations from previous semesters in order to have a simulation of the real conditions.

5.1 Parameter values of the evolutionary algorithm

We run eight experiments. Some of the operators and parameters of the EA were the same for all experiments and some were different. The values were set after some preliminary short experiments. Table 4 shows the parameters and their values that were the same over all experiments and table 5 the operators and parameters that were different.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination rate</td>
<td>0.80</td>
</tr>
<tr>
<td>Recombination per block rate</td>
<td>0.02</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.20</td>
</tr>
<tr>
<td>Mutation per block rate</td>
<td>0.10</td>
</tr>
<tr>
<td>Elitism</td>
<td>1 individual</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination type</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Mutation type</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Population size</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>50</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Selection method</td>
<td>tour</td>
<td>tour</td>
<td>tour</td>
<td>tour</td>
<td>tour</td>
<td>tour</td>
<td>tour</td>
<td>RW</td>
</tr>
</tbody>
</table>

The parameter “Recombination per block rate” refers to the recombination probability of every block (it can be either a lab group or a lab place) of the chromosome. The parameter “Mutation per gene rate” refers to the mutation probability of each lab place.

Numbers of recombination and mutation types correspond to their definition in section 6.3.

Selection methods used are either tournament selection of size 2 (tour) and roulette wheel. In the preliminary short experiments, ranking selection was also tested. The results were similar, so in the final experiments, we’ve decided to use only one of them, i.e. tournament selection.

Each experiment run for two hours, or for 5000 generations, or to the point that the best individual did not improve for 1000 generations.

5.2 Results
Before presenting the results, we have to remind that the EA searches only the space of feasible solutions, keeping the chromosomes valid during the evolution.

Results are presented in table 6. The stopping criterion that was mainly applied by the algorithm was the time limit (2 hours). The limit generation limit (5000 generations) was never applied. The EA was only occasionally stuck in local optima for 30 generations.

Table 6. Results of the eight experiments

<table>
<thead>
<tr>
<th>Exp 1</th>
<th>Exp 2</th>
<th>Exp 3</th>
<th>Exp 4</th>
<th>Exp 5</th>
<th>Exp 6</th>
<th>Exp 7</th>
<th>Exp 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>7227</td>
<td>7214</td>
<td>7211</td>
<td>7242</td>
<td>7202</td>
<td>7206</td>
<td>7203</td>
</tr>
<tr>
<td>Generation number</td>
<td>780</td>
<td>759</td>
<td>720</td>
<td>322</td>
<td>1419</td>
<td>380</td>
<td>813</td>
</tr>
<tr>
<td>Average fitness</td>
<td><strong>1,073</strong></td>
<td>1,098</td>
<td>1,097</td>
<td>1,236</td>
<td><strong>1,043</strong></td>
<td>1,121</td>
<td>1,233</td>
</tr>
<tr>
<td>Required places</td>
<td>2796</td>
<td>2796</td>
<td>2796</td>
<td>2796</td>
<td>2796</td>
<td>2796</td>
<td>2796</td>
</tr>
<tr>
<td>Assigned places</td>
<td><strong>2718</strong></td>
<td>2716</td>
<td>2717</td>
<td>2702</td>
<td><strong>2721</strong></td>
<td>2714</td>
<td>2702</td>
</tr>
<tr>
<td>1st semester</td>
<td>600</td>
<td>599</td>
<td>591</td>
<td>598</td>
<td><strong>600</strong></td>
<td>600</td>
<td>558</td>
</tr>
<tr>
<td>Typical semester</td>
<td><strong>1049</strong></td>
<td><strong>1047</strong></td>
<td>1046</td>
<td>1011</td>
<td><strong>1047</strong></td>
<td>1033</td>
<td>1011</td>
</tr>
<tr>
<td>Typical+1 semester</td>
<td>913</td>
<td>914</td>
<td>920</td>
<td>907</td>
<td>918</td>
<td>906</td>
<td>937</td>
</tr>
<tr>
<td>Other semester</td>
<td>156</td>
<td>156</td>
<td>160</td>
<td>186</td>
<td>156</td>
<td>175</td>
<td>196</td>
</tr>
<tr>
<td>1st preference</td>
<td><strong>2238</strong></td>
<td>2228</td>
<td>2181</td>
<td>1371</td>
<td><strong>2332</strong></td>
<td>1984</td>
<td>941</td>
</tr>
<tr>
<td>2nd preference</td>
<td>350</td>
<td>352</td>
<td>414</td>
<td>592</td>
<td>291</td>
<td>463</td>
<td>472</td>
</tr>
<tr>
<td>3rd preference</td>
<td>72</td>
<td>70</td>
<td>75</td>
<td>318</td>
<td>52</td>
<td>156</td>
<td>407</td>
</tr>
<tr>
<td>4th preference</td>
<td>38</td>
<td>41</td>
<td>21</td>
<td><strong>222</strong></td>
<td>25</td>
<td>69</td>
<td><strong>390</strong></td>
</tr>
<tr>
<td>5th preference</td>
<td>11</td>
<td>11</td>
<td>13</td>
<td><strong>115</strong></td>
<td>10</td>
<td>27</td>
<td><strong>269</strong></td>
</tr>
<tr>
<td>6th preference</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>55</td>
<td>1</td>
<td>5</td>
<td><strong>142</strong></td>
</tr>
<tr>
<td>7th preference</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>13</td>
<td>3</td>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td>8th preference</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>16</td>
<td>7</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>Basic prefs</td>
<td>2469</td>
<td>2480</td>
<td>2480</td>
<td>2513</td>
<td>2470</td>
<td>2485</td>
<td>2518</td>
</tr>
<tr>
<td>Redundant prefs</td>
<td>249</td>
<td>236</td>
<td>237</td>
<td><strong>189</strong></td>
<td>251</td>
<td>229</td>
<td><strong>184</strong></td>
</tr>
<tr>
<td>Withdraw: false</td>
<td>2589</td>
<td>2581</td>
<td>2584</td>
<td>2550</td>
<td>2585</td>
<td>2567</td>
<td>2537</td>
</tr>
<tr>
<td>Withdraw: true</td>
<td><strong>129</strong></td>
<td>135</td>
<td><strong>133</strong></td>
<td>152</td>
<td>136</td>
<td>147</td>
<td>165</td>
</tr>
<tr>
<td>Average of registered TUs</td>
<td>29.90</td>
<td><strong>29.89</strong></td>
<td>29.91</td>
<td>29.90</td>
<td>29.92</td>
<td>29.90</td>
<td>29.91</td>
</tr>
</tbody>
</table>

Comparing the results of the first 4 experiments we observe that using the 4th recombination operator the EA becomes considerably slower than using the others. This is due to the fact that it is applied at the level of lab places, so it has to do more checks in order to verify that no student has been assigned two places of the same lab.

The best results were obtained using the 1st recombination operator. This operator is the only one that succeeded in assigning all available places to students of the first semester. Besides, it managed to satisfy the student preferences to a higher degree than the others.

Comparing the results of experiments 3 and 7, we observe that the 1st mutation type (random mutation) has better results that the 2nd type.

The results on the population size agree with what is reported in the literature: small populations have better performance when the time is limited.

6. CONCLUSION

In this paper we presented the scheduling algorithm of a complete web registration system which was implemented by an evolutionary algorithm.

The results show that the proposed system succeeded in fulfilling our expectations. The evolutionary algorithm managed to:

- allocate most of the places (percentage of filled places is higher than 97%),
- assign to the students of the first semester all corresponding places,
- maintain the “typical semester” rule, the students of each semester are assigned the lab places from courses of their semester that they required.
The preferences of the students to enroll to specific lab groups were satisfied in percent higher than 85%.

Basic requests were satisfied in percent higher than 90%.

There was a small bonus to students with better performance.

At the same time the EA ensures the feasibility of the solutions.

REFERENCES


