Genetic Algorithm for University Timetable Planning in FTI

Dedy Suryadi and Rizaldy Pilipus
Department of Industrial Engineering
Parahyangan Catholic University, Bandung 40141, Indonesia

Abstract

Timetable planning is an activity that must be done at the beginning of each semester in Industrial Technology Faculty (FTI). Since it is done manually, it takes a huge effort and time. This research proposes a Genetic Algorithm for timetable planning. The chromosome is encoded using value encoding. Sub-chromosome and mod-chromosome are defined to accommodate the crossover and mutation procedures. The objective is to minimize the overall penalty, which is given whenever a constraint is violated. The timetable generated by the algorithm has a significantly higher fitness and it requires a considerably less time than the manual planning.

Keywords
Genetic Algorithm, timetable planning, scheduling

1. Introduction

As an educational institution, one of the most important activities of a university is offering courses for its students. In each semester, it involves a number of lecturers and usually even more number of students. Thus, the courses offered in each semester must be scheduled such that a feasible and satisfactory schedule is obtained. However, sometimes it is not a simple matter because university timetable planning is a NP-complete problem [1]. In this problem, generally there are several constraints that have to be considered as well. For example, the number of classrooms is usually limited.

In this research, the planning is done for the timetable of courses offered at Industrial Technology Faculty (Fakultas Teknologi Industri, FTI), Parahyangan Catholic University. Up to now, the timetable planning is still done manually by the staffs of Faculty Administration Office and the Vice Dean of Academic and Students Affairs. Such manual planning consumes quite a long time, generally about a working week. It certainly disrupts the other tasks of both Administration Office and the Vice Dean.

The manual planning requires a long time because it is basically done by trial-and-error method. It takes lots of trials even just to generate a feasible timetable. Any cause for infeasibility is often left unrealized until the timetable is almost fully generated, such that it requires the planners to repeat the timetable planning process. The example of a common cause of infeasibility is scheduling two (or more) different courses for the same class of students. In FTI, the students of each year are grouped into four classes, i.e. A, B, C, and D. The grouping is permanent, so a student who is initially grouped as Class D, for example, will always be in Class D for all courses until his/her graduation. The freedom to switch classes is only obtained when a student takes courses of a higher semester or re-takes courses that he/she had failed previously.

According to the Vice Dean, a good timetable should be comfortable for both lecturers and students. For lecturers, the comfort is defined by the matching between a lecturer’s preference teaching time and the real teaching time he/she is scheduled. For information, at the end of a semester, each lecturer who teaches FTI is given a form to fill in his/her preference teaching time on the next semester. For students, the comfort is defined by a schedule that does not contain too much empty hours between a course and the next one.

The timetable planning problem may be solved using analytical methods, such as integer programming or dynamic programming. However, analytical approaches are usually unable to handle the problem when it gets larger or more complicated, in which it is said to face a “combinatorial explosion” (Zhang, 2006). Therefore, this research uses a Genetic Algorithm approach to solve the university timetable planning in FTI specifically.
Genetic Algorithm is a technique to search a solution for a problem, which is based on the theory of natural selection and genetics [2]. The timetabling problem in different academic institutions has been previously solved by applying metaheuristics such as Genetic Algorithm [3][4][5], Particle Swarm Optimization [6], and Multi-Agent System [7].

2. Timetable Planning Inputs and Constraints
There are several inputs for the timetable planning. The first input is the preference teaching time for each lecturer who teaches in FTI. This preference becomes stricter for the lecturers who are not permanently based in FTI. The reason is that those lectures may also teach in other faculty in the same university, or even in other university or institution. In FTI, they are usually labeled as Extraordinary Lecturers. For FTI-based lecturers, the preference is exactly a preference, i.e. they are assumed to be available and willing to teach outside his preferred time. This preference may be considered as a soft constraint, because it should be fulfilled if possible, but it is not a must.

The second input is the course(s) taught by each lecturer. Related to each course are the year and the class of students who attends the course, as well as the credits of the course. The hours allocated for a course is as many as its credit. For example, 3 hours are allocated for a 3-credit course. Table 1 shows an example of the second input related to a particular lecturer and the courses that he teaches.

<table>
<thead>
<tr>
<th>Lecturer’s Name : Ali Sadiyoko</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>IIE304</td>
</tr>
<tr>
<td>IIE310</td>
</tr>
</tbody>
</table>

As it can be seen on Table 1, the Information System course is taught by a team of lecturers, as indicated with the note “(Team Teaching)” in the column of Year & Class of Students.

Based on the observation and interview with the Vice Dean, there are several constraints of the timetable planning in FTI. Those constraints are as follows:
1. A lecturer must not teach more than one course at any particular time.
2. A class of students must not be scheduled for more than one course at any particular time.
3. A classroom must not be scheduled for more than one course at any particular time.
4. All courses must be scheduled within the FTI’s academic week, i.e. Monday to Friday. Each day starts from 7AM and ends at 17PM.
5. The limited number of classrooms at any particular time.
6. The courses taught by a team of lecturers and laboratory practices are initially scheduled and their initial schedules are considered fixed.

3. Genetic Algorithm for Timetable Planning in FTI
Genetic Algorithm works on solutions. Therefore, the solution encoding into a chromosome is required initially. Later, the chromosome must be able to be decoded into a solution. This section discusses the encoding, decoding, as well as the crossover, mutation, and selection processes in Genetic Algorithm.

3.1 Gene Encoding
There are several encoding techniques that may be used in Genetic Algorithm. Those techniques are binary encoding, permutation encoding, value encoding, and tree encoding [8]. This research uses value encoding to encode the chromosome.

The smallest part of a chromosome is a gene. In other words, a chromosome consists of a number of genes. In this research, each gene is designed to contain six digits. An example of a gene is shown in Figure 1.
The first digit of a gene explains the preference of a lecturer at a particular hour. The first digit is filled with 1 if a lecturer states his/her preference to teach at that hour, otherwise it is 0. Another possibility is to fill the first digit with 2, which means a course has been fixed at a particular time. The digit 2 is used for the courses which are taught by a team of lecturers and for laboratory practices. Those genes start with the digit 2 are not processed by the Genetic Algorithm, because they have been fixed at certain positions before the Genetic Algorithm begins. In other words, those courses are scheduled before others and their schedules must not be changed.

The second digit carries the information of assignment to a lecturer at a particular hour. If he/she is assigned, the digit’s value is 1. Otherwise, it is 0. Once the second digit contains 0, the remaining digits must be 0. It is determined that way because if a lecturer is not assigned to a particular hour, it logically means there are no courses and no students to teach at that hour.

The third digit contains the information of the course. This digit contains integer which starts from 1 up to the number of courses taught by a lecturer. Suppose it contains 2, it refers to the second course taught by a lecturer. Suppose a lecturer teaches Elementary Physics and Calculus, for that lecturer 1 represents Elementary Physics and 2 represents Calculus. The numbering is given during the input process.

The fourth digit shows the year of students. For Optional Courses, this digit is filled with N, because those courses are not only designed for a specific year of students. The fifth digit shows the class of students. The last digit shows the credits of the course mentioned in the third digit.

Using the example in Figure 1, the gene represents a condition in which a lecturer prefers to teach at a particular hour and he/she is indeed assigned at that hour. The course he/she teaches is the third course on his/her list. The students take the course is from the Year 2007, Class A. The course has 3 credits.

**3.2 Chromosome, Sub-Chromosome, and Mod-Chromosome Encoding**

A complete chromosome should wholly represent a solution. Therefore, in this research, a chromosome represents a whole timetable of FTI, involving all lecturers, classes, and courses altogether. An example of a part of a chromosome is shown in Figure 2. The part shown is only for Monday and for four lecturers only.

<table>
<thead>
<tr>
<th>Lecturer</th>
<th>Monday</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAD</td>
<td>7-8</td>
</tr>
<tr>
<td></td>
<td>000000</td>
</tr>
<tr>
<td>DS</td>
<td>000000</td>
</tr>
<tr>
<td>CT</td>
<td>000000</td>
</tr>
<tr>
<td>CBN</td>
<td>000000</td>
</tr>
</tbody>
</table>

**Figure 2: Example of a chromosome**

The genes with gray shade show each lecturer’s teaching time preferences. A gene that contains “000000” indicates a lecturer does not prefer to teach at that time and he/she is not assigned to that time either. On the other hand, a gene contains “100000” indicates a lecturer prefers to teach at that time but he/she is not assigned to that time.
The timetable for each particular lecturer is called a sub-chromosome. If a chromosome may be thought as a table as shown in Figure 2, a sub-chromosome may be thought as a single row of the table which is related to a single lecturer only. Furthermore, a sub-chromosome is modified into a mod-chromosome. First, genes whose first digit is 0 or second digit is 0 are deleted. Subsequently, any duplication of a gene is deleted as well. An example of the modification of a sub-chromosome to become a mod-chromosome is shown in Figure 3. The first two genes are deleted, as they contain 0’s only. The third one is kept, while the next two are deleted. The sixth gene contains “100000” is deleted, while the seventh one is kept. Finally, the last three genes are deleted.

Sub-chromosome:  
Mod-chromosome:  

Figure 3: Example of a sub-chromosome and its mod-chromosome

3.3 Crossover

The crossover used in this research is inspired by the position-based crossover. However, a modification must be made to the method in order to avoid producing illegal chromosomes. Illegal chromosomes are the ones unable to be decoded into a solution. In other word, in this research, illegal chromosomes are the ones unable to be translated into a timetable.

The modification is that the crossover is not done between two chromosomes, but between two mod-chromosomes. Since mod-chromosomes come from sub-chromosomes, the crossover keeps the correct courses taught by each lecturer. If it is applied to the whole chromosomes, the courses taught by a lecturer might be mixed up with other lecturer’s. The same length is guaranteed as well when crossover is done between two mod-chromosomes.

The crossover starts by picking a certain chromosome based on the determined crossover probability and another one arbitrarily. From the first chromosome, each sub-chromosome is converted into a mod-chromosome. For each gene in the mod-chromosome, its original position in the sub-chromosome is recorded. For example, a gene whose position is the third in a sub-chromosome becomes the first gene in a mod-chromosome, but its original position as the third gene in sub-chromosome is recorded.

An example of a mod-chromosome is shown in Figure 4. The small number above each gene indicates its original position in the sub-chromosome. For example, the third gene in this mod-chromosome, 1127D2, is originally positioned at the 14th in the sub-chromosome. This first mod-chromosome is called Parent 1. It acts as the base of the timetable pattern.

Figure 4: Example of Parent 1 for crossover

In the second chromosome, all sub-chromosomes are also modified into mod-chromosomes. Parent 2 is the mod-chromosome that comes from the sub-chromosome related to the same lecturer as that of Parent 1. An example of mod-chromosome as Parent 2 is shown in Figure 5.

Figure 5: Example of Parent 2 for crossover

After two parents are obtained, genes from Parent 1 are selected randomly. These genes go straightly into the offspring’s mod-chromosome. The remaining genes in the offspring’s mod-chromosome are completed by the same order of the genes from Parent 2. The crossover procedure to create an offspring is illustrated in Figure 6.
The offspring’s mod-chromosome is converted back into a sub-chromosome by expanding each gene. For example, the first gene, 1137B3, is expanded into three genes in sub-chromosome. The location is from the third until the fifth gene. The second gene, 1127B2, is expanded into two genes in sub-chromosome and located at the ninth until tenth gene. The remaining empty genes are filled with the way they are in the Parent 1’s sub-chromosome.

This crossover procedure may result in an infeasible offspring, i.e. a gene in sub-chromosome contains two courses to be scheduled at that particular time. However, such offspring is maintained because infeasible offspring might lead to a better solution in the next generations after going through other crossovers and mutations. In complex cases, relatively good feasible solutions might be difficult to obtain if the search is only applied to the feasible area [2].

### 3.4 Mutation

The mutation is applied to all chromosomes in a population and the offsprings of the generation, based on the mutation probability. Like the crossover, mutation is also operated on a mod-chromosome. A gene is selected randomly and it is inserted into another position, which is also selected randomly. The insertion itself is done on a sub-chromosome. The mutation procedure, which creates a feasible mutation result, is shown in Figure 7.

The procedure may generate infeasibility as well, however it is again maintained with the expectation of obtaining better chromosomes in later generations. An example of an infeasible mutation result is shown in Figure 8.

### 3.5 Selection

After the population goes through crossover and mutation, the selection is done in order to select the best chromosomes that will go to the next generation. The method used is deterministic selection and the selection is applied to an enlarged sampling space, i.e. chromosomes from the current population (let’s say there are $\mu$ chromosomes) and the generated offsprings (let’s say there are $\lambda$ offsprings). Using deterministic selection, the best $\mu$ chromosomes are chosen among $(\mu + \lambda)$ chromosomes in order to become the new population in the next generation.

In order to measure the quality of a chromosome, it is needed a fitness function. The higher fitness a chromosome has, the better quality it has as well. In this research, the fitness function consists of penalties. Penalties are used because there might be infeasible chromosomes involved.

Very large penalties must be given for the violations of hard constraints, i.e. the ones that result in an infeasible timetable. Smaller penalties are given for the violations of soft constraints, i.e. the ones that actually would not result in an infeasible timetable. Therefore, the objective is to minimize the overall sum of all penalties.

The violation types and amount of penalties are shown in Table 2.
In fact, the fifth violation may cause the timetable to be infeasible. However, in reality, sometimes the non-FTI-based lecturers may be negotiated to change his/her preferred time. Therefore, the penalty value for this violation is not as large as the first four violations.

### 3.6 The General Algorithm

The general algorithm for university timetable planning is as follows.

1. Enter the inputs, i.e. lecturers and the respective courses taught by each lecturer; as well as each lecturer’s preferred teaching time.
2. Determine the GA parameters, i.e. crossover probability, mutation probability, population size, and maximum number of generations.
3. Set the initial best fitness, Fitbest, equal to a very large number.

<table>
<thead>
<tr>
<th>No</th>
<th>Violation</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A gene contains two courses to be scheduled</td>
<td>1,000,000</td>
</tr>
<tr>
<td>2</td>
<td>A year of student is scheduled for more than a course at a particular time</td>
<td>1,000,000</td>
</tr>
<tr>
<td>3</td>
<td>The number of classrooms required at any particular time exceeds the available classrooms</td>
<td>1,000,000</td>
</tr>
<tr>
<td>4</td>
<td>A course is scheduled along two different days</td>
<td>1,000,000</td>
</tr>
<tr>
<td>5</td>
<td>A non-FTI-based lecturer is scheduled at his/her non-preferred time</td>
<td>10,000</td>
</tr>
<tr>
<td>6</td>
<td>A FTI-based lecturer is scheduled at his/her non-preferred time</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>An empty hour between two courses scheduled for a year of student at a particular day</td>
<td>10</td>
</tr>
</tbody>
</table>
4. Create the initial population, which contains chromosomes as many as the determined population size. For each chromosome in the initial population, the assignment of courses taught by each lecturer (which refers to each sub-chromosome) is done by randomly picking a gene which is related to the preferred time of that lecturer. If a course has been assigned to that gene, the random picking is repeated. If after three picks the genes are always occupied, then a gene among the genes related to the not-preferred time is picked randomly.

5. Start the first generation.

6. Do the crossover.

7. Do the mutation.

8. Compute the fitness of the enlarged population (original chromosomes and the offsprings) and do the selection.

9. Update the best fitness, Fitbest, if there is any chromosome of the current population has better (smaller) fitness than the previously recorded Fitbest. Keep the chromosome that has the best fitness.

10. If the number of generation has reached its maximum, then go to Step 11. Otherwise, add the number of generation by 1 (one) and go back to Step 6.

11. The algorithm has been finished. The best solution found is the one whose fitness is recorded as Fitbest.

4. Results and Discussions
Since there are two parameters in Genetic Algorithm, i.e. crossover and mutation probabilities, which affect the searching procedure, the algorithm implementation is started by experimenting several values of those parameters. The general rule-of-thumb is that the crossover probability should be relatively high, but the mutation probability should be quite low. Therefore, the crossover probabilities experimented are 50%, 80%, and 95%, while the mutation probabilities experimented are 5%, 30%, and 50%.

Based on the experimentation upon a hypothetical case, which has significantly smaller size than the real one, it is obtained that crossover probability, mutation probability, and their interactions all significantly affect the fitness function value. Moreover, it is also found that the best fitness results from the combination of the highest crossover probability, 95%, and the lowest mutation probability, 5%. The experimentation results over those parameters are shown in Figure 9.

![Figure 9: Experimentation result for crossover probabilities, mutation probabilities, and their interactions](image)

The algorithm is implemented for the real case using the following parameters:
- crossover probability 95%
- mutation probability 5%
- population size: 30
- maximum number of generations: 2000

The case involves 53 lecturers, 13 classrooms. There are students from four different years and there are six classes for each year’s student. The implementation result for 10 replications is shown in Table 3.
Table 3: Fitness and Computation Time for 10 Replications

<table>
<thead>
<tr>
<th>Replication</th>
<th>Fitness</th>
<th>Computation Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>82740</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>73540</td>
<td>59</td>
</tr>
<tr>
<td>3</td>
<td>103290</td>
<td>58</td>
</tr>
<tr>
<td>4</td>
<td>42930</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>93010</td>
<td>58</td>
</tr>
<tr>
<td>6</td>
<td>72710</td>
<td>58</td>
</tr>
<tr>
<td>7</td>
<td>33220</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>33400</td>
<td>59</td>
</tr>
<tr>
<td>9</td>
<td>52860</td>
<td>58</td>
</tr>
<tr>
<td>10</td>
<td>102630</td>
<td>58</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>69033</strong></td>
<td><strong>58.6</strong></td>
</tr>
</tbody>
</table>

The best fitness is obtained at the seventh replication. If the existing timetable for the semester in which this research was done is measured by the utilized fitness, it has much worse fitness, i.e., 4,144,080. It means the existing timetable must have violated any of the hard constraints. The existing timetable contained some courses taught by the same lecturer to be scheduled at the same time. It was later solved because the lecturer finally opted to close one of the courses, which was fortunately just an optional course. If such option is not available, usually the Vice Dean and administration staff of FTI must repeat the timetable planning from the very beginning.

From the viewpoint of time, the suggested algorithm also provides a huge benefit for the timetable planner. The computation time is roughly an hour, which is very shorter compared to the existing time spent for planning the timetable, i.e., at least 3 workdays.

From the 10 replications, it can be seen as well that the standard deviation is high, i.e., 27068.07552. It might suggest that the algorithm has not reached near enough to convergence after 2000 generations. Therefore, it can also be further investigated whether the addition of either population or the number of maximum generations may help the algorithm to converge better, and reach better fitness as well. A local optimization method may also be added into the algorithm in order to find better solution [9].

5. Conclusions
The Genetic Algorithm for university timetable planning has been developed. The result is satisfactory when it is applied to the real case in FTI. Based on the determined criteria, the best timetable generated by the algorithm has significantly better fitness than the one generated manually. Moreover, the algorithm also cuts the required time to create a timetable from more than 3 workdays to less than an hour.

Based on the experiments, the well-known rule-of-thumb for Genetic Algorithm parameters are confirmed. The higher crossover probability is found to be better than the lower ones. Also, the lower mutation probability is found to be better than the higher ones.

References


