**Solution To Timetable Scheduling using Genetic Algorithms**

Hammad Ikhlaq (16L-4281), Shamoon Shahid (16L-4093), Muhammad Zain Anjum (16L-6307)

*National University of Computer & Emerging Sciences*

**Abstract**

University Course Timetabling Problem (UCTP) is a well-known Constraint Satisfaction Problem (CSP) problem that has exponential number of solutions based on course conflicts, teacher’s empty times and other parameters. This is a NP-Hard problem. Scheduling is a major debate on planning which can be used in trains scheduling, classroom scheduling, traffic even in schools and universities. The Scheduling leads to organizing tasks and removing tasks interference which is important. The goal of solving UCTP is setting times for courses and teachers in weekdays in order to reach minimum courses conflicts. It is also ideal for teachers to have joint days for teaching in the least weekdays. Of course, subject to the restrictions of classes and teachers program this scheduling is very difficult. Generally,

Evolutionary Algorithms (EA) are efficient tools to solve this problem. The final timetabling must be optimum which means that there is no conflicts if possible and best scheduling generate for teachers. In this paper we solve this problem based on genetic algorithm and implement this algorithm with java based toolbox on given dataset. The implementation results show that genetic algorithm is efficient tools that can reach to the near of global optimum.

**Keywords:** Course Scheduling, Constraint Satisfaction Problem, Genetic Algorithm, Optimization, University Course Timetabling Problem**.**

# Introduction

The issue of tabulation of courses, in principle, includes allocating lessons per week holding periods and room class. If conditions such as failure to allocate one class at a time to several lessons, or lack of time in courses of a teacher are added to this issue, it becomes clear that the problem of designing scheduling, can be a constraint satisfaction problem (CSP). Due to the increasing number of students, new fields, lack of classrooms, conference rooms, laboratories and increasing number of courses offered to the students, we face many limitations, there for designing a time tables for this problem base on teachers, students and courses parameters is very difficult .Different methods have been used to solve these types of issues, some of which include: Graph coloring algorithm, Use of heuristic functions or experimental functions, Population-oriented or developmental methods. In these methods, the evolutionary ideas and population improvement are used and a variety of methods in common include application of this schedule include: Genetic Algorithm. The algorithm used in this paper is genetic algorithm, which is one of the strongest and most widely used algorithms in search and optimization problems. One reason for the popularity of the genetic algorithm is that it does not require a mathematical model of high-level and advanced one. This algorithm was first introduced by John Holland based on the Darwin’s evolutionary theory. Genetic algorithms follow the law of development. To start, these algorithms need a random primary generation which is a set of problem answers. Then the answers are better than before until reaching a global optimum level. In this process chromosomes those which have higher fitness transfer to the next generation with higher probability. In this study each Chromosome is a university course timetabling and solve this problem based on Genetic Algorithm in terms of features and constraints of the problem .the rest of paper has organized as follows. In section 2 we introduce general issues of genetic algorithm. In section 3 we define the problem of university course timetable. Section 4 is devoted to description of the proposed algorithm. Section 5 concerns numerical experiments and their results were studied. Finally, section 6 summarizes the paper.

# Introduction to Genetic Algorithm

The experience of recent decades shows that genetic algorithm is one of the strongest methods inspired from the nature of genetics and natural selection phenomenon, which is one of the best forms of numerical optimization problems in science and engineering. This algorithm, with a heuristic random search, find the most appropriate answers from the coded information (the chromosomes). The Components of GA include: Chromosomes and genes, genetic population, fitness function and genetic operators. Table 1 compares natural genetic systems and GA.

In the genetic algorithm, genetic population (society) there is the set of chromosomes, the genetic operators are crossover and mutation, and the quantitative parameters are population size, the mutation rate and so on. Other components are explained in Table 1.

|  |
| --- |
| **Genetic algorithm** |
| **Individual:** Possible responses of the issue that have been encrypted like a string of numbers. |
| **Fitness function:** Is to evaluate the quality of a chromosome that has a mathematical formula. |

***Table 1 – Genetic algorithm sementics***

|  |
| --- |
| **Crossover:** is a process of taking more than one parent solutions and producing child solutions from them. |
| **Mutation:** one bite of chromosome is selected randomly and changes. |
| **Reproduction**: Repeating the algorithm to achieve optimal solution or reach the termination condition. |

**2.1 Description of the Genetic Algorithm Flowchart**

Genetic algorithm has flowing stages:

1. Creating random population (The initial population) and evaluation.
2. Parental choice and their combinations to make children.
3. Selecting members of the population to apply a mutation and mutation population creation.
4. The main population integrate the children of the original population and creating new mutants.
5. If the termination condition is not achieved, we repeat stage 2.

The general flowchart of genetic algorithm and pseudo code of typical evolutionary algorithm is shown in figure 1

and figure 2 respectively

.

Population

Parents

Offspring

Initialization

Termination

Replacement

Selection

Crossover

Mutation

***Figure 1- The flowchart of Genetic Algorithm***

**Algorithm:**

*Evolutionary algorithm*

**BEGIN**

INITIALISE

population

EVALUATION each individual

**REPEAT UNTIL**

(

termination condition

)

1

.

SELECT parents

2

.

RECOMBINE pairs

of parents

3

.

MUTATE the resulting offspring

4

.

EVALUATE new

individuals

5

.

SELECT individuals for the next

generation

**END REPEAT**

**END**

***Figure 2 - Pseudo code of evolutionary algorithms***

# Problem Definition

The university course timetabling problem we want to find and optimum course scheduling in which all conflict are satisfied, therefore this is and optimization problem. Because the genetic algorithm is and optimization process, we can use this algorithm for solving this problem to reach the global optimum albeit genetic algorithm almost doesn’t reach to the global optimum, but it can reach to the sub optimal points. In this problem the sub optimal point is a solution (course timetabling) that some conflicts are not satisfied. The input-output diagram of UCTP is shown in figure 3.

Courses Teacher’s empty times Courses conflict

1

.

Population size

2

.

Generation

3

.

Probability of crossover

4

.

Probability of mutation

5

.

Elitism

factor

Optimal course timetabling

***Figure 3 - Input – Output diagram of the Course scheduling model time.***

The parameters of this problem are as following:

**Inputs:** Course Names, Teachers timetable, Courses conflict.

**Output**: Optimal Course timetabling.

**Parameters:**

|  |  |  |
| --- | --- | --- |
|  | 1. | Population size |
|  | 2. | Generations |
|  | 3. | Probability of mutation |
|  | 4. | Probability of crossover |
|  | 5. | Number of teachers |
|  | 6. | Number of courses |

# Our proposed algorithm

In the previous section we introduce UCTP parameters. In the following we identify the components of our algorithm. First, a set of teacher’s name times, list of courses and courses conflict is given to the algorithm. Then initial population is randomly produced including a set of courses timetable. After using parent selection method, a set of chromosomes are selected and combine them to produce new generation. Finally, chromosome with maximum fitness is introduced as the best answer. In the next section we review the components of our proposed algorithm.

**4.1 Components of algorithm**

**Length of the chromosome:** The length of each chromosome is number of meeting time.

**Fitness:** Our fitness is considering the conflicts among the following:

1. Number of teachers present in same class.
2. Number of times that teachers have more than one class at the same time.
3. Number of times teacher teaches the same section.
4. Number of times that two courses are at the same time.

**Selection:** For parent selection we used tournament selection; this method works as follows: Choose some number of individuals (tournament size) randomly from the population and copy the best individual from this tournament group into the population and repeat n times. In this paper, we figure out “tournament size=3” is suitable value for this problem.

**Crossover:** We used uniform crossover in our algorithm; in this method, each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosomes. When there is a 1 in the crossover mask, the gene is copied from the first parent, and when there is a 0 in the mask the gene is copied from the second parent. A new crossover mask is randomly generated for each pair of parents. Therefore Offspring contain a mixture of genes from each parent. The crossover rate is 0.9

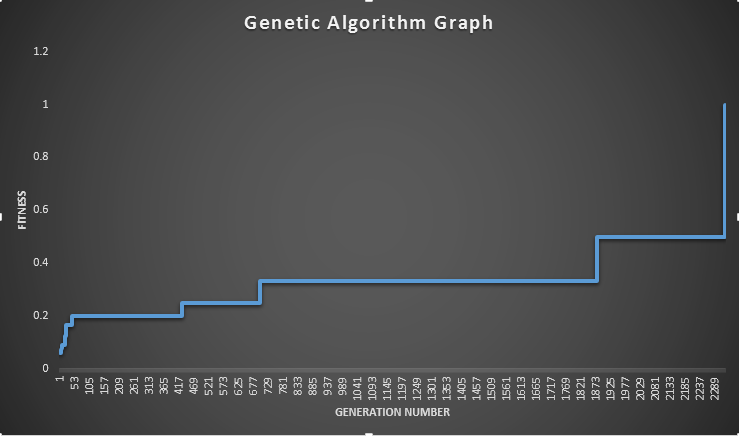
**Mutation:** For mutation, we used shuffle indexes mutation; in this method, Shuffle the attributes of the input individual and return the mutant. The P parameter is the probability of each attribute to be moved. Usually this mutation is applied on vector of indices. In our implementation, we set “P=0.1” and mutation probability is 0.1.

**Elitism:** We used elitism size=1. It means that we keep best individual in each iteration of generations.

**Termination condition:** In this method, after generation giving fitness while not equal to one, algorithm keeps on running.

# Simulation Results

For implementation of our propose algorithm we use Java programming language. We generate datasets based on given files and apply genetic algorithm functions with Java IDE. For creating output diagrams we use excel.

******We simulate our proposed algorithm on random generated schedules. Diagram of fitness changes in various generations is shown in figure below. This figure shows that fitness increase, and near to the optimal solution (max diagram). Therefore the population converge to the global optimum and best solution is near to the global optimum (fitness=1).

The parameters value in this experiment are:

Number of generations = 2289

Population size =9

Number of courses = 170

Number of teachers = 83

# Conclusion

In this paper we study University Course Timetabling Problem based on genetic algorithm. This problem has been widely studied in literatures but there is no optimal solution for this problem yet that reach to global optimum. Results show that solving this type of problems with evolutionary algorithms is very efficient. Also average evaluation function on different random datasets in various generations shows that this algorithm can improve its performance during iterations of each generation until reach to the suboptimal solution. For future works we can improve this algorithm with another parameters of genetic algorithm such as different crossover algorithm or use several crossover methods.