Faculty of Engineering and Technology

Electrical & Computer Engineering Department

ENCS3340

Artificial Intelligence

Genetic Algorithm Project

Prepared by:

Al-Ayham Maree 1191408

Sara Ammar 1191052

Supervised by:

Dr. Aziz Qaroush

Section:

1

15-1-2023

Introduction:

One of the hardest and most well-known search challenges is establishing schedules for academic institution course scheduling. Due of its enormous state space, it cannot be solved using conventional search techniques. There are several approaches for finding answers, including tests, and the studies have demonstrated that employing local search techniques to tackle this problem yields logical solutions. We will discuss the Genetic Algorithm approach to solving this issue.

Problem Specification:

The problem of scheduling time in the university can be defined as a problem full of restrictions related to several matters, such as the student and the doctor, the time and place of the lecture, here we will discuss this problem in terms of the appropriate times for students that give them more flexibility in registering their courses. The input provided to this problem is:

- A list of all courses and labs offered by the department of Electrical and Computer Engineering at this semester, each course has many sections, and each section is assigned to a specific doctor.
- A list of time slots,
  where classes with two lectures per week and a duration of one and a quarter hour for each class. It can be distributed as follows:

  - Times: (8:30 - 9:45), (10:00 - 11:25), (11:25 - 12:40), (12:50 - 14:05), (14:15 - 15: 30), (15:45 - 16:55).
  - Days: S/M, S/W, M/W, T/R.

  And classes with one lecture per week (laboratories), and its duration is three hours, and it can be distributed as follows:

  - Times: (8:30 - 11:25), (11:25 - 14:05), (14:15 - 16:55).
  - Days: S, M, T, W, R.

Problem Formalization:

a. Chromosome Design:

Using the genetic algorithm technique, a special chromosome is built for the current problem so that each chromosome represents a solution to the problem, each gene in this chromosome represents a Time slot ID of a specific time slot where the index of this gene represents the section ID with the course ID to which the time slot in this gene belongs.

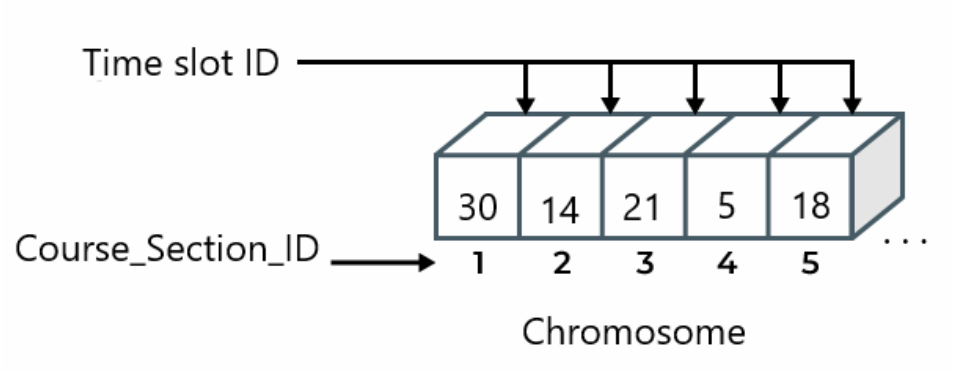

Figure 1: Chromosome Design

According to this representation, the length of the chromosome is given by the following formula:

$$\mathbf{Chromosome\ Length} = \sum_{i=0}^{n} \mathrm{Sci}$$

Where, n is the number of courses and S is the number of sections for course ci.

b. Population

An initial population of size 500 is generated. Each partition is randomly allocated to one time slot suitable for the class type. Through the genetic algorithm, the population grows to a maximum size of 1000. When the population size exceeds 1000, a number of chromosomes are removed according to a criterion discussed later, in order to maintain the size limit to 1000 chromosomes.

c. Parents Selection

Two chromosomes are selected in each iteration to generate two new children (off-springs). The best two chromosomes are selected from among the population, this method gave the best experimental results.

d. Crossover Operators

Crossover is applied with a high rate equal to 0.6 in order to continuously generate new off-springs. Two types of crossovers were used with equal probabilities (0.5 for both):

- Multi Point Crossover: where two points are chosen randomly, and the segment between these two points is swapped between the two parents.

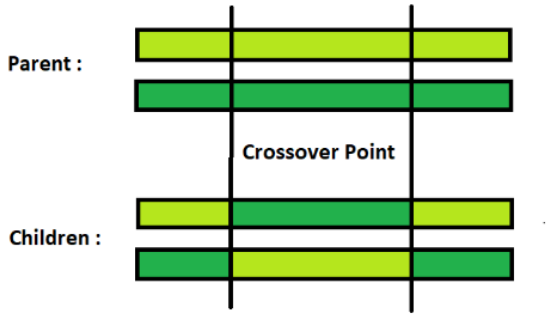

Figure 2: Multipoint Crossover

Uniform Crossover: where randomly some genes from different locations are swapped between the two parents.

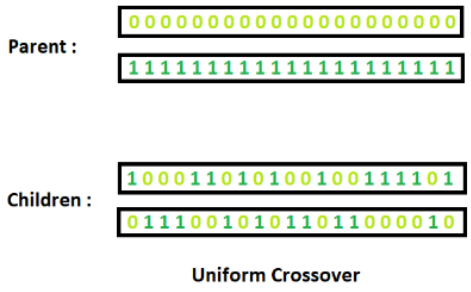

Figure 3: Uniform Crossover

These two types of crossovers are used rather than the classical one-point cross over, in order to explore more solutions. Experimentally, they give better results in our problem.

e. Mutation Operators

Mutation is used with a lower rate equal to 0.3, in order to avoid convergence to local optima and get new children by Replace one randomly selected gene in the parent, by another randomly generated gene.

f. Removing Criteria

As said earlier, there is a need to get rid of some chromosomes in order to avoid infinitely growing population. Here the worst chromosomes are removing from the population to prevent the population size to exceed the maximum. The purpose is to improve the total fitness for the population.

g. Fitness function

Fitness function evaluates how close a given solution is to the optimum solution of the desired problem. Each chromosome is given a score out of maximum fitness 100%. 60% of this fitness is given to the hard constraints and 40% to the soft constraints (distributed subjectively between them). Soft fitness is not computed till all hard constraints are satisfied. Otherwise, their fitness value has no meaning.

Hard fitness

1. There are no conflicts in assigning courses for the same instructor. The mathematical measurement of this constraint is defined by the equation:

$$fitness = \frac{\text{number of conflicted sections}}{\text{total number of sections}} * 60\%$$

2. No four or more consecutive lectures are allowed for the same instructor.

$$fitness = \frac{\text{maximum number of consecutive hours for instructor}}{\text{number of instructorsn}} * 40\%$$

Soft fitness

1. Reducing the number of Lab lectures at 14:00PM.

$$fitness = (1 - \frac{\sum_{i=0}^{n} \frac{Xi}{5}}{n}) * 10\%$$

Let n be the number of sections and Xi the number of days start at 14:00 PM for instructor i.

2. Reducing the number of sections in the same course that have a same time slot (time, day)

$$fitness = (1 - \frac{\sum_{i=0}^{n} Xi}{n}) * 20\%$$

Let n be the number of sections and Xi the number of sections in the same course that have a same time slot *for course i.*

3. Reducing the number of sections in the same course that have a same time different day

$$fitness = (1 - \frac{\sum_{i=0}^{n} Xi}{n}) * 5\%$$

Let n be the number of sections and Xi the number of sections in the same course that have a same time different day *for course i.*

4. Reducing the number of sections in the course (has 3 sections) that cross in one day

$$fitness = (1 - \frac{\sum_{i=0}^{n} Xi}{n}) * 20\%$$

Let n be the number of sections and Xi the number of sections in that cross in one day *for course i.*

5. Reducing the number of sections in the course (has 4 sections) that cross in one or two days

$$fitness = (1 - \frac{\sum_{i=0}^{n} F(Xi)}{n}) * 20\%$$

F(xi) = 1 if 2 days, 2 if one day

Let n be the number of sections and Xi the number of sections in that cross in one or two day *for course i.*

6. Reducing the number of sections in the course (has 5 or more sections) that cross in four or less days

$$fitness = (1 - \frac{\sum_{i=0}^{n} F(Xi)}{n}) * 25\%$$

Let n be the number of sections and Xi the number of sections in that cross in four or less days *for course i.*

Conclusion:

In conclusion, this project is an excellent example of how to use local search methods to solve problems with large state spaces. Practically we understood the fundamentals of problem solving with genetic algorithms. We observed the impact of fitness function selection on the quality of solutions produced. The good ones usually get close to the desired solution in a reasonable amount of time, whereas the bad ones usually converge to unacceptable solutions. And we discovered that in genetics, randomness and experiments to change some values and percentages plays an important role in preventing solutions from convergent to a local optimum.