Electrical and Computer Engineering Department

ENCS3340 Artificial Intelligence, First Semester, 2022-2023

Project 1 - Department Courses Distribution

Prepared by:

Name: Lana Batnij ID:1200308

Name: Yasmeen kamal ID:1201146

Name: Anas Hawamda ID:1202121

Instructor: Dr.Aziz Qaroush

Data :15/1/2022

section: 2

Table of content:

1)The genetic Algorithm In general:

A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature." It is used to solve optimization problems in machine learning. It is one of the important algorithms as it helps solve complex problems that would take a long time to solve [1].

The genetic algorithm Consists of:

- **Population:** Population is the subset of all possible or probable solutions, which can solve the given problem.
- **Chromosomes:** A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.
- **Gene:** A chromosome is divided into a different gene, or it is an element of the chromosome.
- **Fitness Function:** The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.
- **Genetic Operators:** In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.
- **Selection:** After calculating the fitness of every existent in the population, a selection process is used to determine which of the individualities in the population will get to reproduce and produce the seed that will form the coming generation.

How Genetic Algorithm Work?

The genetic algorithm works on the evolutionary generational cycle to generate high-quality solutions. These algorithms use different operations that either enhance or replace the population to give an improved fit solution [1]. It basically involves five phases to solve the complex optimization problems, which are given as below:

- Initialization
- Fitness Assignment
- Selection
- Reproduction
- Termination

Initialization: The process of a genetic algorithm starts by generating the set of individuals, which is called population. Here each individual is the solution for the given problem. An individual contains or is characterized by a set of parameters called Genes. Genes are combined into a string and generate chromosomes, which is the solution to the problem. One of the most popular techniques for initialization is the use of random binary strings [1].

Fitness Assignment: Fitness function is used to determine how fit an individual is? It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function. The fitness function provides a fitness score to each individual. This score further determines the probability of being selected for reproduction. The high the fitness score, the more chances of getting selected for reproduction [1].

Selection: The selection phase involves the selection of individuals for the reproduction of offspring. All the selected individuals are then arranged in a pair of two to increase reproduction. Then these individuals transfer their genes to the next generation [1].

Reproduction: After the selection process, the creation of a child occurs in the reproduction step. In this step, the genetic algorithm uses two variation operators that are applied to the parent population. The two operators involved in the reproduction phase are given below [1]:

-Crossover: The crossover plays a most significant role in the reproduction phase of the genetic algorithm. In this process, a crossover point is selected at random within the genes. Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing the offspring [1].

-Mutation: The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population. It can be done by flipping some bits in the chromosomes.
Mutation helps in solving the issue of premature convergence and enhances diversification [1].

Termination: After the reproduction phase, a stopping criterion is applied as a base for termination. The algorithm terminates after the threshold fitness solution is reached. It will identify the final solution as the best solution in the population [1].

2) The problem Formalization:

We assumed that the Chromosome represent the solution (means the final table of courses with no conflict), in the other hand the first gen stands for the Course, while its timing, the days of teaching, and the sections of this course, all of these symbolize the length of that gen, but we also assume that we may have another gen in this chromosome which is the teacher. The teachers name and the courses that they teach describe the length of the second gen.

Basically, we start by creating two files, one for the courses data itself, that contain its name, its code, the year level, and the number of section that course all of them split by semicolon. The other one includes the teacher's information like the teacher's name, and the courses that they can teach and all of this split by colon. We use python in programming since much easier, then we built our fitness function which basically operates as: start with random state, then the computer starts to compare if there are a conflict in courses days or timing increase the cost by twenty, if there are a conflict in instructor of course increase it by fifty, because we assume that have a higher priority [the code of fitness is in appendix A].

After we generate this application for the first time, it gives us the best table for the courses, and that's since we try it on small amount of data, but the we try it on bigger set of data and it worked as we want it.

Furthermore, and as AI, the initial state was a random state to start with, while the final state was the courses table with the least number of conflicts. The operators represent as the fitness function and the cross over process.

3) Appendix A:

-the fitness function code:

```
def Fitness(Courses):
    fit = 0
    for code in Courses:
        for sections in Courses[code]:
            Days = Courses[code][sections][0]
            Instructor = Courses[code][sections][1]
            time = Courses[code][sections][2]
            lastSection = list(Courses[code].items())
            while int(lastSection[-1][0]) is not sections:
                Information = sections + 1


                if Days == Courses[code][Information][0] and time == Courses[code][Information][2]:
                    fit += 20
                if Days == Courses[code][Information][0] and time == Courses[code][Information][2] \
                        and Instructor == Courses[code][Information][1]:
                    fit += 50
                sections = sections + 1


    for code in Courses:
        for sections in Courses[code]:
            Days = Courses[code][sections][0]
            Instructor = Courses[code][sections][1]
            time = Courses[code][sections][2]
            level = int(Courses[code][sections][3])
            nextCode = False
            for code2 in Courses:
                while code2 != code and not nextCode:
                    continue
                if not nextCode:
                    nextCode = True
                    continue
                for sections2 in Courses[code2]:
```

```
                Days2 = Courses[code2][sections2][0]
                Instructor2 = Courses[code2][sections2][1]
                time2 = Courses[code2][sections2][2]
                level2 = int(Courses[code2][sections2][3])
                if Days == Courses[code][Information][0] and time == time2 \
                    and Instructor == Instructor2 and level == level2:
                  print(fit)
                  fit += 80
                elif Days == Days2 and time == time2 and level == level2:
                  print(fit)
                  fit += 50
return fit
```

4)Reference:

https://www.javatpoint.com/genetic-algorithm-in-machine-learning [Accessed on 9:15pm at 12/1/2020]