Genetic algorithm implementation

# Creating the chromosome

Each individual chromosome is, in itself, a time table. The structure of the chromosome would be a 5×15 matrix, since it must represent a time table for a 5- day week, with 15 allotted time slots per day.

# Creating chromosomes

An individual is characterized by a set of parameters (variables) known as genes. Genes are joined into a string to form a chromosome (solution). We model our solutions as chromosomes.

In our example scenario, chromosomes will be different lecture sessions for students taking different modules. Consider that we have to coordinate student groups, modules, lecture halls, the days of the week and time. You can represent a lecture session as:

**<Module, Student Group, Day, Location, Lecture Time>**

You can encode the lecture session as a binary pattern to a chromosome.

You can give binary values for each value in each entity. You can change the encoding pattern as you wish. Given below is an example way you can encode the lecture session.

Get the list of modules and assign binary values.

Text

Description automatically generated

Get the list of students groups and give binary values.

Text

Description automatically generated

Similarly, you can come up with coding schemes for every entity in the lecture session.

Graphical user interface, text, application, chat or text message

Description automatically generated

# Creating an initial population

Different student groups take different classes within a week. Hence, you have to come up with different class combinations and create the initial population. You can decide upon the size of the population (number of classes).

Text

Description automatically generated

You have to encode these classes into chromosomes as mentioned before.

# Time slots

Each time slot is modelled as a structure in C containing the following members:

* A 1-bit variable which describes whether or not the slot is occupied (has an active class)
* A 3-bit variable describing the subject code of the active class.
* A 3-bit variable describing the faculty code (3MCT for example) of the active class.
* A 3-bit variable describing the teacher code who’s teaching the active class.
* A 2-bit variable describing the room code of the active class.

= 12-bit chromosome

Each chromosome would be a 5x15 array of a single time slot (as shown above in the 4 bullets). So from left to right we have the time slots and from up to down we have the days of the week!

The population would be an array of chromosomes. The chromosomes are initialized randomly. Each bit in every chromosome is populated randomly with a 0 or a 1.

# Fitness function

The fitness value of a chromosome is calculated using this formula:

Chart, box and whisker chart

Description automatically generated

The value of c is arrived at as follows:

* C is incremented every time multiple classes are scheduled in the same room in a given time slot
* C is incremented every time multiple classes are scheduled for the same faculty (study area), for example 3MCT has 2 classes at the same time slot.
* Every time the number of classes per week of a given subject is not equal to the predefined number, c is incremented by the absolute value of the difference between the two.

# Selection operator

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

It is like spinning a roulette wheel where each chromosome has a segment on the wheel proportional to its fitness. The roulette wheel is spun, and when the arrow comes to rest on one of the segments, the corresponding chromosome is selected. To select a chromosome for mating, a random number is generated in the interval (0,100), and the chromosome whose segment spans the random number is selected

Chart, table, pie chart

Description automatically generated

# Crossover operator

The point of crossover is a randomly generated number, and is distinct for each occurrence of crossover. Crossover pairs are generated randomly for the entire population. The crossover occurs separately for each row of the chromosome, rather than for the chromosome in a whole. A random point is selected among the 15 slots for each day in a given time table. Crossover would occur 5 times for a single chromosome.

# Mutation operator

1. Examine the first binary digit of a chromosome.
2. Toggle that bit with a probability of 0,05, or leave it unchanged, with a probability of 0,95.
3. Repeat the process with every bit of the chromosome.
4. Similarly, mutate all the chromosomes in the population.

The main disadvantage is that, due to the random nature of this mutation operator, it is equally likely to bring about variation that would increase or decrease the fitness of a given chromosome. To solve this issue, use the **guided mutation** operator.

1. Create a copy of the chromosome under consideration
2. Examine the first binary digit of the copy of the chromosome
3. Toggle that bit with a probability of 0,05.
4. Repeat the process with every bit of the chromosome copy.
5. Compare the fitness of the (mutated) copy with that of the (un-mutated) original.
6. If the (mutated) copy is fitter, replace the (un-mutated) original with the copy. Otherwise, discard the copy.
7. Similarly, perform guided mutation on all the chromosomes in the population.

When your genome is a list of binary numbers, there are several ways to apply mutation in a genetic algorithm. Here are some common methods:

* **Bit-flip mutation**: In this method, you randomly select one or more positions in the genome and flip the value of the bit at that position. For example, if the bit value is 0, it becomes 1, and vice versa. This operation can be applied with a fixed probability, such as 0.01, for each bit in the genome.
* **Inversion mutation**: In this method, you randomly select a range of positions in the genome and reverse the order of the bits in that range. For example, if the range is [3, 6], and the original genome was [1, 0, 1, 0, 0, 1, 1, 0], the mutated genome would be [1, 0, 1, 1, 1, 0, 0, 0].
* **Swap mutation**: In this method, you randomly select two positions in the genome and swap the values of the bits at those positions. For example, if the two positions are 2 and 5, and the original genome was [1, 0, 1, 0, 0, 1, 1, 0], the mutated genome would be [1, 0, 0, 0, 0, 1, 1, 0].
* **Random reset mutation**: In this method, you randomly select one or more positions in the genome and reset the value of the bit at that position to a new random value. For example, if the bit value is 0, it becomes 1 or 0 with equal probability. This operation can be applied with a fixed probability, such as 0.01, for each bit in the genome.