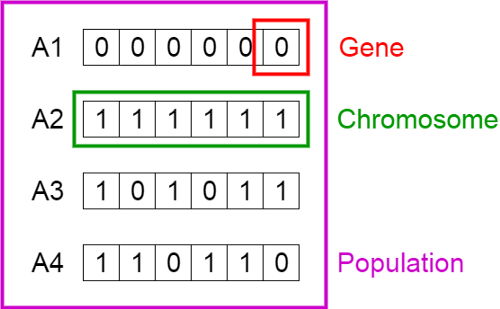
the 5 main phases of a genetic algorithm are,

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution).

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



**Fitness Function**

The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a **fitness score** to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

**Selection**

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.

**Crossover**

**Crossover** is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

**Mutation**

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.

For the baseline problem, one path in a solution looks something like 0110010100010111110000101110001011. It has 34 genes of 0 or 1 because there are 34 intervals in total. 0 represents a bus waiting still during an interval and does not carry any students; 1 represents a bus taking a trip either from A to B or from B to A. Please note that a gene of 1 does not necessarily mean that the bus carries students. The bus might go to B empty in order to fulfill the huge demand at the beginning of the next interval. One solution has a couple of buses, so a solution will be several path chromosomes concatenated together.

After a solution chromosome is generated, a fitness score can be calculated according to a fitness function. A fitness function has two components, a total cost, and a penalty cost. For the first component, although the price of each bus needs to be negotiated with the shuttle bus company, the prices generally depend on the duration starting from the beginning to the end of bus operation. **Please note that as long as a driver starts his day of work and hasn’t clocked out, all intervals marked 0 still cost money even though the bus stayed still in one place.** Therefore, I used the following mathematical formula:

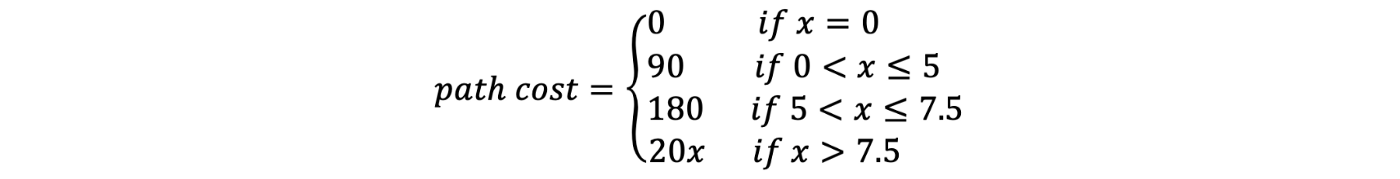
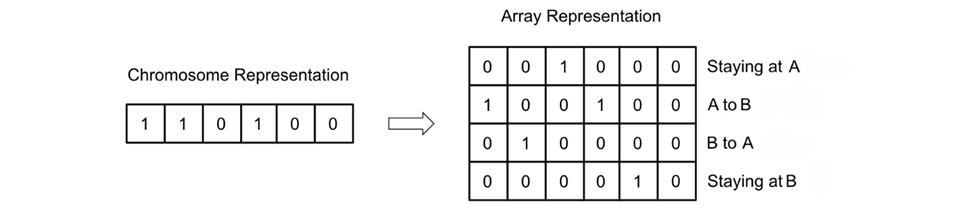


Image by Author

where x is the operation duration of a bus (unit: hours). This formula is counterintuitive in many ways, but we are just using it as a case. For example, the path chromosome 0110010100010111110000101110001011 operates 33 intervals, which is 16.5 hours, so the cost for this bus is 20 \* 16.5 = 330. Then, the total cost is the sum of the cost of each bus. The second component of the fitness score is the penalty for violating constraints. Depending on which version of the base function you are considering, you may have one penalty where you only enforce the demand constraint, or two constraints, or all three penalties of the demand constraint, the rush hour constraint, and the max working hour constraint. A different amount of small penalty is added to the fitness score every time a student does not get on the bus, or a driver finishes a trip in one interval during rush hours, or a driver works longer than 4 hours consecutively.

Specifically, here is how constraint violations are detected. First, for the demand constraint, the chromosome paths need to be encoded to an array of size 4 \* 34 which specifies what each 0 and 1 means.



In this example, the first 1 on the left corresponds to the first column on the right, in which the one and only 1 represents that the bus goes from A to B. Similarly, the rest means the same. What helps us calculate the total bus capacity is the second and third row of the array. We can then add up all the A to B buses and B to A buses and then times the bus capacity to compare with the demand. Second, for the rush hour constraints, we need to check each chromosome for consecutive 1’s from 7:30 to 8:30 and from 17:30 to 18:30. 10, 01, and 00 are valid during rush hours, but not 11. Third, for the max working hour constraint, we need to keep track of the number of 1’s in a row. In cases of rush hour, a 0 during the rush hour could also mean that a bus is running, so we need to take care of this special case as well. If the working duration is greater than 4 hours, it causes a penalty.