The introduction of Genetic Algorithm

1. Purpose

This article illustrates the genetic algorithm briefly,to tell the beginner who have no previous knowledge about the genetic algorithm,how dose it works.

2. Biological genetics and evolution

(1) Genome：

The collection of chromosomes within a cell holds all the information necessary to reproduce that organism.

(2) Crossover：

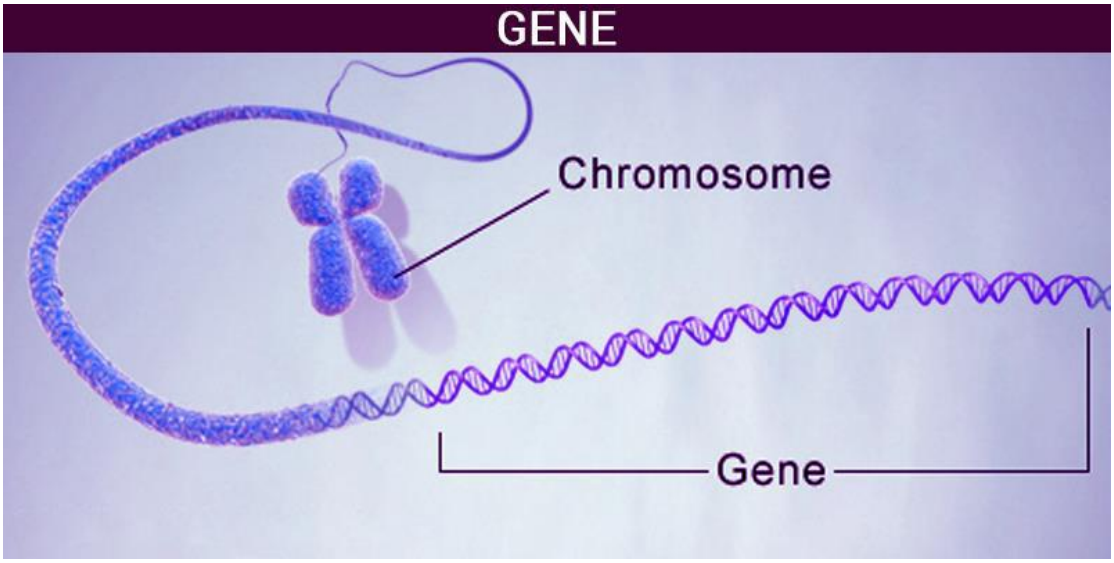
When two organisms mate and reproduce, their chromosomes are mixed together to create an entirely new set of chromosomes, which consists of genes from both parents. This process is called recombination or crossover. This could mean that the offspring inherits mainly the good genes, in which case it may go on to be even more successful than its parents (for example, it has better defense mechanisms against predators), or it may inherit mainly the bad genes, in which case it may not even be able to reproduce.

(3) Mutate：

when the genes are passed on to the offspring, there is a very small probability that an error may occur and the genes may be slightly changed. The evolution of creature comes from innumerable tiny mutations， The premise is that these mutations are beneficial to biological survival。

(4) Fitness：

The more fit the offspring are, the more likely they will go on to reproduce and pass on their own genes to their offspring. Each generation, therefore, will show a tendency to be better at survival and mating than the last.



3. Genetic algorithms used in computers

The working process of genetic algorithm in computer essentially simulates the biological evolution process.

(1) First, you figure out a way of encoding any potential solution to your problem as a "digital" chromosome.

(2) Then, you create a start population of random chromosomes (each one representing a different candidate solution) and evolve them over time by "breeding" the fittest individuals。

(3) In the meantime, a small amount of mutation should be added to certain positions on the chromosome.

(4) The genetic algorithm will converge upon a solution. Genetic algorithms do not guarantee a solution, nor do they guarantee to find the best solution, but if utilized the correct way, you will generally be able to code a genetic algorithm that will perform well.

(5) The best thing about genetic algorithms is Team LRN 99 that you do not need to know how to solve a problem; you only need to know how to encode it in a way the genetic algorithm mechanism can utilize.

4. The explanation of other nouns in genetic algorithm

(1)Crossover rate： The crossover rate is simply the probability that two chosen chromosomes will swap their bits to produce two new offspring.

(2)Mutation rate： The mutation rate is the probability that a bit within a chromosome will be changed

(3)TSP (Traveling Salesman Problem) ：

Given a collection of cities, the traveling salesman must determine the shortest route that will enable him to visit each city precisely once and then return back to his starting point.

5. The implement of genetic algorithm

Each chromosome is encoded in some way to represent a solution to the problem at hand. It may be a very poor solution, or it may be a perfect solution, but every single chromosome represents a possible solution (more on the encoding in a moment). Once an initial population is created ,then you do this：

Loop until a solution is found:

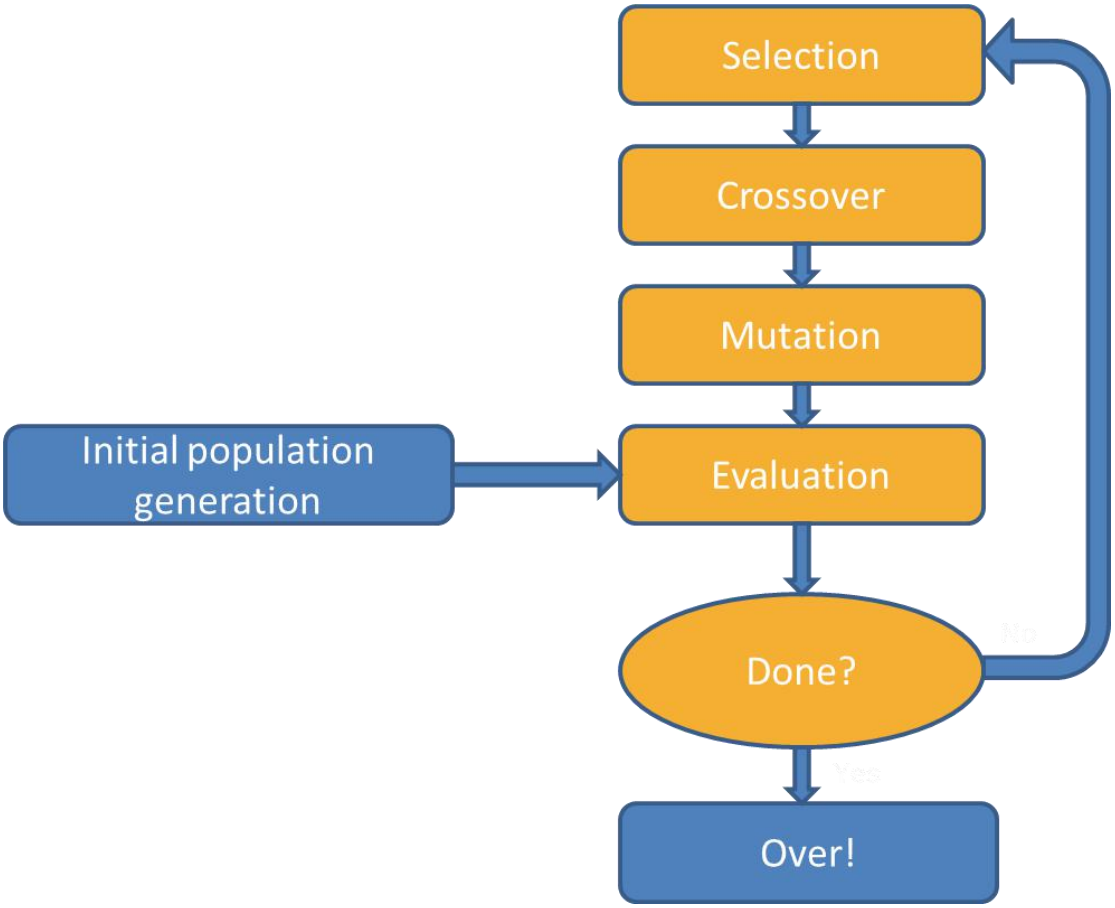
1. Test each chromosome to see how good it is at solving the problem and assign a fitness score accordingly.
2. Select two members from the current population. The probability of being selected is proportional to the chromosome's fitness—the higher the fitness, the better the probability of being selected. A common method for this is called Roulette wheel selection.
3. Dependent on the Crossover Rate, crossover the bits from each chosen chromosome at a randomly chosen point

4. Step through the chosen chromosome's bits and change dependent on the Mutation Rate.

5. Repeat steps 2, 3, and 4 until a new population has been created.

End loop

Each loop through the algorithm is called a generation (steps 1 through 5). We call the entire loop an epoch.



6. Common Operators used in Genetic Algorithms

(1) The Genetic algorithms of Chromosome binary coding：

Selection Operators

Elite Selection:

Elitism is a way of guaranteeing that the fittest members of a population are retained for the next generation.

Roulette Wheel Selection:

Roulette wheel selection is a method of choosing members from the population of chromosomes in a way that is proportional to their fitness—for example, the fitter the chromosome, the more probability it has of being selected. It does not guarantee that the fittest member goes through to the next generation, merely that it has a very good probability of doing so.

Crossover Operators

SinglePoint Crossover:

It simply cuts the genome at some random point and then switches the ends between parents.

DoublePoint Crossover:

Instead of cutting the genome at just one point, two-point crossover (you guessed it) cuts the genome at two random points and then swaps the block of genes between those two points.

MultiPoint Crossover:

to move down the length of the parents, and for each position in the chromosome, randomly swap the genes based on your crossover rate。

Mutation Operators

MultiPoint Mutation:

Along the chromosome length，depending on the crossover rate，start mutate operation

Fitness Scaling Operators

Rank Fitness Scaling:

Rank scaling can be a great way to prevent too quick convergence, particularly at the start of a run when it's common to see a very small percentage of individuals outperforming all the rest. The individuals in the population are simply ranked according to fitness, and then a new fitness score is assigned based on their rank.

(2) **The Genetic algorithms of chromosome decimal coding, used to solve TSP problems**:

Selection Operators

Roulette Wheel Selection:

Roulette wheel selection is a method of choosing members from the population of chromosomes in a way that is proportional to their fitness—for example, the fitter the chromosome, the more probability it has of being selected. It does not guarantee that the fittest member goes through to the next generation, merely that it has a very good probability of doing so.

Fitness Proportionate Selection:

Selection techniques of this type choose offspring using methods which give individuals a better chance of being selected the better their fitness score. Another way of describing it is that each individual has an expected number of times it will be chosen to reproduce. This expected value equates to the individual's fitness divided by the average fitness of the entire population.

Elite Selection:

Elitism is a way of guaranteeing that the fittest members of a population are retained for the next generation.

Steady State Selection:

Steady state selection works a little like elitism, except that instead of choosing a small amount of the best individuals to go through to the new generation, steady state selection retains all but a few of the worst performers from the current population. The remainder are then selected using mutation and crossover in the usual way. Steady state selection can prove useful when tackling some problems, but most of the time it's inadvisable to use it.

Stochastic Universal Sampling Selection:

Stochastic Universal Sampling (SUS for short) is an attempt to minimize the problems of using fitness proportionate selection on small populations. Basically, instead of having one wheel which is spun several times to obtain the new population, SUS uses n evenly spaced hands, The amount of pointers is equal to the amount of offspring required.

Tournament Selection:

To use tournament selection, n individuals are selected at random from the population, and then the fittest of these genomes is chosen to add to the new population. This process is repeated as many times as is required to create a new population of genomes. Any individuals selected are not removed from the population and therefore can be chosen any number of times.

Crossover Operators

Partially-Mapped Crossover Operator (PMX):

first choose two random crossover point, Then you look at the two center sections and make a note of the mapping between parents,.Now, iterate through each parent's genes and swap the genes wherever a gene is found that matches one of those listed. Step by step it goes

Order-Based Crossover (OBX):

To perform order-based crossover, several cities are chosen at random from parent and then the order of those cities is imposed on the respective cities in the child.

Position-Based Crossover (PBX):

This is similar to Order-Based Crossover, but instead of imposing the order of the cities, this

operator imposes the position.

Mutation Operators

Displacement Mutation(DM):

Select two random points, grab the chunk of chromosome between them, and then reinsert at a random position displaced from the original.

Exchange Mutation (EM):

Select two genes on chromosome to exchange

Insertion Mutation (IM):

This is a very effective mutation and is almost the same as the DM operator, except here only one gene is selected to be displaced and inserted back into the chromosome.

Scramble Mutation (SM):

Choose two random points and "scramble" the cities located between them.

Fitness Scaling Operators

Rank Fitness Scaling:

Rank scaling can be a great way to prevent too quick convergence, particularly at the start of a run when it's common to see a very small percentage of individuals outperforming all the rest. The individuals in the population are simply ranked according to fitness, and then a new fitness score is assigned based on their rank.

Sigma Fitness Scaling:

If you use raw fitness scores as a basis for selection, the population may converge too quickly, and if they are scaled as in rank selection, the population may converge too slowly. Sigma scaling is an attempt to keep the selection pressure constant over many generations. At the beginning of the genetic algorithm, when fitness scores can vary wildly, the fitter individuals will be allocated less expected offspring. Toward the end of the algorithm, when the fitness scores are becoming similar, the fitter individuals will be allocated more expected offspring.

Scaling Bolzmann Fitness:

using sigma scaling can keep the selection pressure constant over a run of your genetic algorithm, but sometimes you may want the selection pressure to vary. A common scenario is one in which you require the selection pressure to be low at the beginning so that diversity is retained, but as the genetic algorithm converges closer toward a solution, you want mainly the fitter individuals to produce offspring.

7. Further reading

About Genetic Algorithm:

https://zh.wikipedia.org/wiki/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95