OBJECTIVES:

Evolutionary algorithms are typically used to provide good approximate solutions to problems that cannot be solved easily using other techniques. Many optimization problems fall into this category. It may be too computationally-intensive to find an exact solution but sometimes a near-optimal solution is sufficient. In these situations evolutionary techniques can be effective. Due to their random nature, evolutionary algorithms are never guaranteed to find an optimal solution for any problem, but they will often find a good solution if one exists.

BACKGROUND STUDY

- Should have prior knowledge on any programming language to implement the algorithm.
- Need knowledge of artificial neural network.
- Need knowledge on computation complexity.

RECOMMENDED READING

- http://www.cs.cmu.edu/~02317/slides/lec_9.pdf
- BOOK-Russell and Norvig.
- Genetic Algorithms- David E Goldberg.

Genetic Algorithm

A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems. (GA)s are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified to form a new population. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Vocabulary

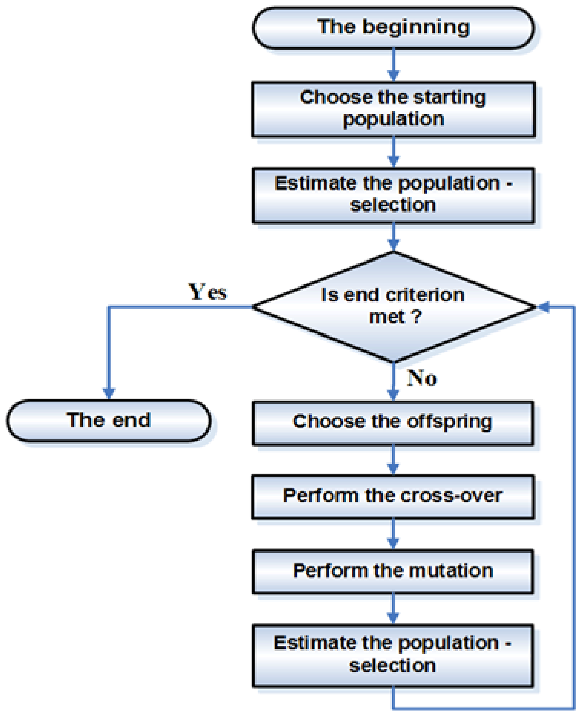
• Individual - Any possible solution

• Population - Group of all individuals

• Fitness – Target function that we are optimizing (each individual has a fitness)

• Trait - Possible aspect (features) of an individual

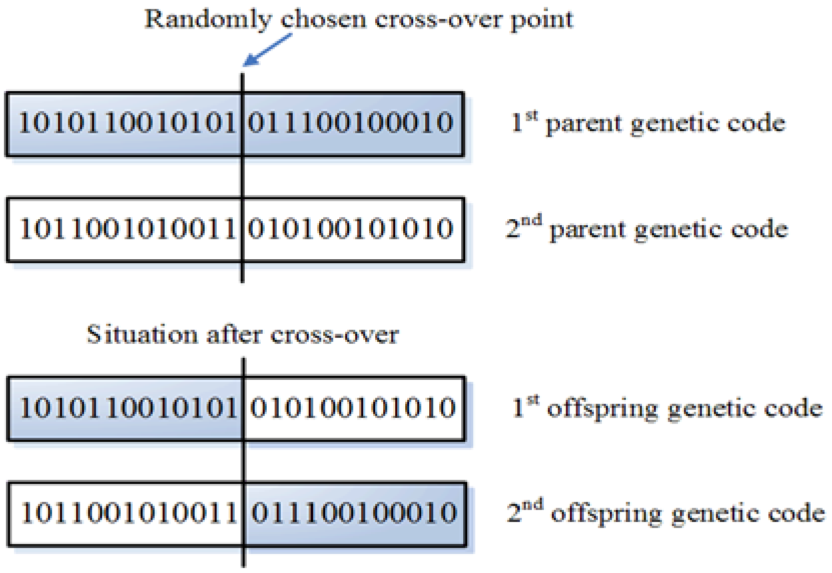
• Genome - Collection of all chromosomes (traits) for an individual.



Genetic algorithm construction blocks

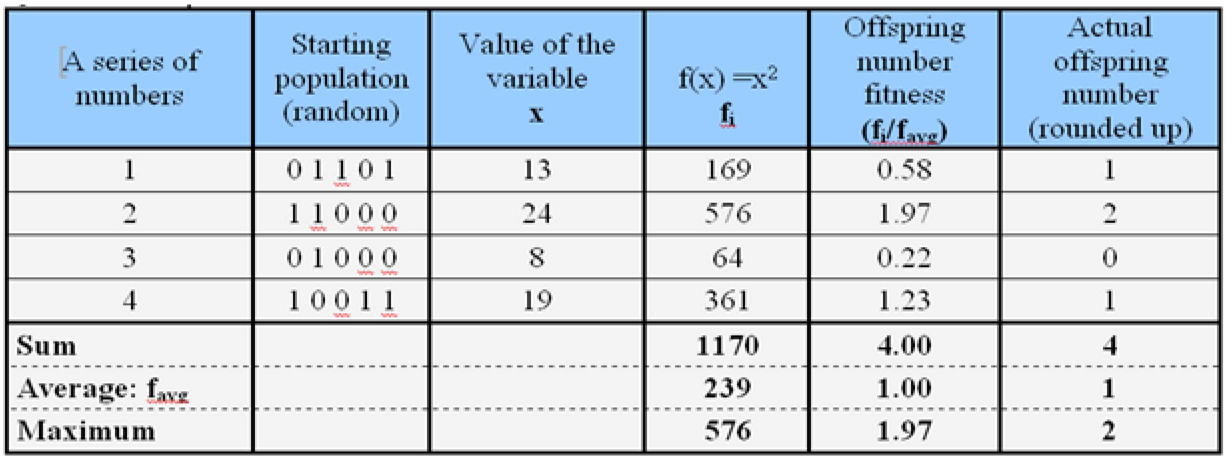
Demo Example:

For a better idea of what this section is about, let us have a look at a simple genetic algorithm and use it in an everyday application: searching for a maximum of the function , $f(x) = x^2$ where x can take values between 0 and 31.



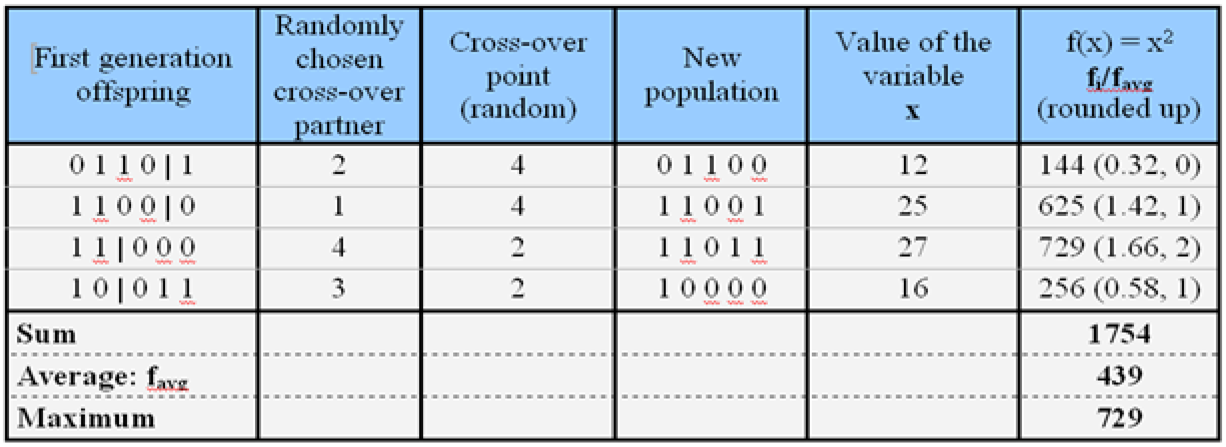
Genetic code of the parents and the offspring before and after the cross-over

It can easily be seen that a good subject, which has many copies, has an advantage in comparison with the weak one, because his genetic material can cross-over with more subjects than the genetic material of the weak ones. What follows is also the mutation for which we set the probability of 0.001 in this experiment. This execution of the genetic algorithm was simple, but it still shows the essential element of its operation, which is that the result of the optimisation is improving from generation to generation.



A series of numbers	Starting population (random)	Value of the variable x	$f(x) = x^2$ f_i	Offspring number fitness (f_i/f_{avg})	Actual offspring number (rounded up)
1	0 1 1 0 1	13	169	0.58	1
2	1 1 0 0 0	24	576	1.97	2
3	0 1 0 0 0	8	64	0.22	0
4	1 0 0 1 1	19	361	1.23	1
Sum			**1170**	**4.00**	**4**
Average: f_{avg}			**239**	**1.00**	**1**
Maximum			**576**	**1.97**	**2**

Table.1 The computation of a randomly chosen population characteristics



First generation offspring	Randomly chosen cross-over partner	Cross-over point (random)	New population	Value of the variable x	$f(x) = x^2$ f_i/f_{avg} (rounded up)
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144 (0.32, 0)
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625 (1.42, 1)
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729 (1.66, 2)
1 0 | 0 1 1	3	2	1 0 0 0 0	16	256 (0.58, 1)
Sum					**1754**
Average: f_{avg}					**439**
Maximum					**729**

Table.2 The presentation of the cross-over and the first generation offspring characteristics computation

Table1 shows an example of the characteristics (quality) computation of a randomly chosen starting population (first generation or the starting parents). On the basis of the computed characteristics of the individual subjects, a selection is made and only the best subjects are allowed to continue with the cross-over (multiplication). Table.2 shows the cross-over of this first generation and the "birth" of the offspring and their characteristics (quality) computation with the criterion function $f(x) = x^2$. When we cross-over the parents (first column), we get a new population with better characteristics. We assumed the value 0.001 for the mutation. Since there are four subjects in each generation, each of the subjects being five bits (binary places) long, the probability that one of them would mutate is 4*5*0.001=0.02. In the first step, none of the bits has mutated. We can keep computing in the same way until we reach the value x = 31 in just a few iterations. It takes many more modifications and improvements for successful operation at solving a very complex problem, however, they are beyond this section.

LABROTORY EXERCISES

1. Implement the above example to maximize $f(x)=x^2$
   Input: User desired size of population, individuals of population, and maximize function.
   Output: Number of iterations, finally surviving population set and maximum value of the given function.