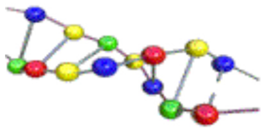
Introduction to
Genetic Algorithms



IV. Genetic Algorithm

Basic Description

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Example
As you already know from the chapter about search space, problem solving can be often expressed as looking for extreme of a function. This is exactly what the problem shown here is. Some function is given and GA tries to find minimum of the function.
You can try to run genetic algorithm at the following applet by pressing button Start. Graph represents some search space and vertical lines represent solutions (points in search space). The red line is the best solution, green lines are the other ones.
Button Start starts the algorithm, Step performs one step (i.e. forming one new generation), Stop

stops the algorithm and Reset resets the population.

Here is applet, but your browser does not support Java. If you want to see applets, please check browser requirements.

Outline of the Basic Genetic Algorithm

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

Some Comments

As you can see, the outline of Basic GA is very general. There are many things that can be implemented differently in various problems.

First question is how to create chromosomes, what type of encoding choose. With this is connected crossover and mutation, the two basic operators of GA. Encoding, crossover and mutation are introduced in next chapter.

Next questions is how to select parents for crossover. This can be done in many ways, but the main idea is to select the better parents (in hope that the better parents will produce better offspring). Also you may think, that making new population only by new offspring can cause lost of the best chromosome from the last population. This is true, so so called **elitism** is often used. This means, that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

Some of the concerning questions will be discussed later.

Maybe you are wandering, *why* genetic algorithms do work. It can be partially explained by **Schema Theorem** (Holland), however, this theorem has been criticised in recent time. If you want to know more, check other resources.

Previous Next