Genetic algorithms

History

Idea of evolutionary computing was introduced in the 1960s by I. **Rechenberg** in his work "***Evolution strategies***" (*Evolutionsstrategie* in original). His idea was then developed by other researchers. **Genetic Algorithms** (GAs) were invented by John **Holland** and developed by him and his students and colleagues. This lead to Holland's book "*Adaption in Natural and Artificial Systems*" published in 1975.

In 1992 John **Koza** has used genetic algorithm to evolve programs to perform certain tasks. He called his method "**genetic programming**" (GP). LISP programs were used, because programs in this language can expressed in the form of a "parse tree", which is the object the GA works on.

Introduction

Genetic algorithm is one of the best and at the same time one of the simplest evolutionary algorithms. Genetic algorithm by itself contains all the basic evolutionary algorithm construction blocks. Therefore, the study of genetic algorithm is also a basis for the study of other evolutionary algorithms.

Genetic algorithm is used as an **optimisation algorithm**. This means that it is one of the so-called optimisation methods for searching optimums (global maximums or minima).

There are several **definitions** of the algorithm; the most common and **simple** one is that *this is a sequence of precisely defined steps which eventually lead to a solution to the given problem*. Historically, the algorithm was invented very early for the realization of the simplest works, as well as complicated procedures, such as constructing the pyramids in the old Egyptian times. Algorithms follow us everywhere we go, whether at brushing our teeth, cooking lunch, or washing a car, etc. However, algorithms are in most cases said to be mathematical processes which describe the course of a working task step by step.

Algorithms consist of a series of construction blocks or simple commands. It could be said that the complication of the algorithms lies in their simplicity. Since they consist of simple commands (steps), they are difficult to be designed. Namely, man, as an intelligent creature who is capable of performing unusually complex operations, is not used to work with the simple mind maps that are required by algorithms. The main problem of us, the designers of algorithms, and consequently of algorithms themselves, is that the details that we take for granted can be easily overlooked.

Optimisation

Today's science is marked by two typical problems. The first one is automatic system designing and the other one is the behaviour prediction of large coincidental systems such as the stock exchange, the weather, the society, etc. Here, we will limit ourselves to the first problem - automatic system designing, where the so-called optimisation methods were put into effect in the last decade. Optimisation roughly

means that we are looking for the best one among various possible solutions. Almost always we have to make a compromise between the good and the bad qualities of the solution we get. In very few cases, we run into complex problems which only have one solution. Optimisation methods, sometimes also called searching schemes, are searching for maximums and minima (in one word, optimums). There are two kinds of optimums: local and global optimums. The algorithms should of course find global optimums, but the majority of the so-called standard optimisation methods have problems if the local optimums are present as well. The Figure 3.1 shows a two-dimensional example of an optimisation. Some optimisation methods (gradient methods) function in a way that they compute the function inclination in a given point, [make a step in that direction], compute the inclination again, and make a step, etc.; when the computed inclination is small enough, the method stops. The inclination is positive if we are looking for a maximum, and negative if we are looking for a minimum. A method like this will go astray to a local minimum (optimum) very soon for the function in Figure 3.1, if we choose T_1 as the starting point, or it will reach the global minimum (optimum) if we start in the point T_2. The example can show only two or three dimensions; if there are more, the function cannot be evident and, thus, it is much more difficult to estimate whether the optimum is local or global. The only thing we can do is to examine the space from multiple points at the same time, which of course increases the difficulty level of the method computation at an exponent degree with regard to the number of variables. Genetic algorithms of this kind do not have these problems and this is the basic reason for their popularity among the engineers in the fields of technics and mechatronics since 1992.

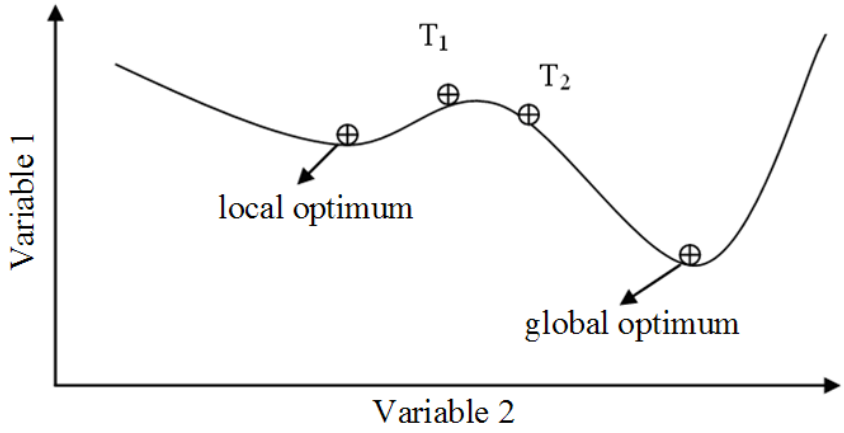


Figure 3.1: Illustration of the optimisation method

Presentation of genetic algorithm

Searching for useful methods which would be robust enough to stay away from local optimums and at the same time simple enough so that they could examine spaces with many variables and find a global optimum or even several optimums (in cases with several equivalent solutions), man started to follow nature's example. If nature was capable of constructing a complicated creature such as man, why could we not learn from it? These aspirations to imitate nature caused that the methods of evolutionary algorithms were developed, and one of the most important and most successful among them is the

source: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_01.html

genetic algorithm, which uses the 'mating' (cross-over or combination) of the solutions in order to seek a new generation of solutions.

Genetic algorithm was in fact invented by nature. Charles Darwin named it 'evolution'. Genetic algorithm, which was first presented and named by J.H. Holland in the early seventies of the previous century, imitates the natural algorithm of the evolution entirely.

Individuals or their genetic material were presented with a series of ones and zeros (binary system), and the genetic algorithms performed operations on these series, not on the solutions themselves.

Holland's programme selected only the best samples, just like it happens in nature. Each had its 'quality value' variable (also fitness function), which was used to estimate the sample quality with regard to the rest of the population (generation). The higher the estimated quality value, the higher was the probability of survival. This leads to the **reproduction operation** which represents the subject copy from the current generation to the next one, at which good subjects have more copies than bad ones.

The next operation is the **cross-over**. This is combining the genetic material of randomly selected pairs of subjects.

Another operation in this programme was **the mutation,** which is randomly changing the genetic material to individual subjects (not to all). This operation also resembles that which happens in nature, where the mutation means regenerating the lost genetic material. When a specific part of the genetic material is lost in any way, the subject generates it again, but it does it randomly. This operation turns out to be very important, because without it, the genetic algorithm could get caught in a snare (local optimum) and it could not find its way out. If this happens, the mutations take care of continuing the evolution (creating the genetic material heterogeneity).

source: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_01.html

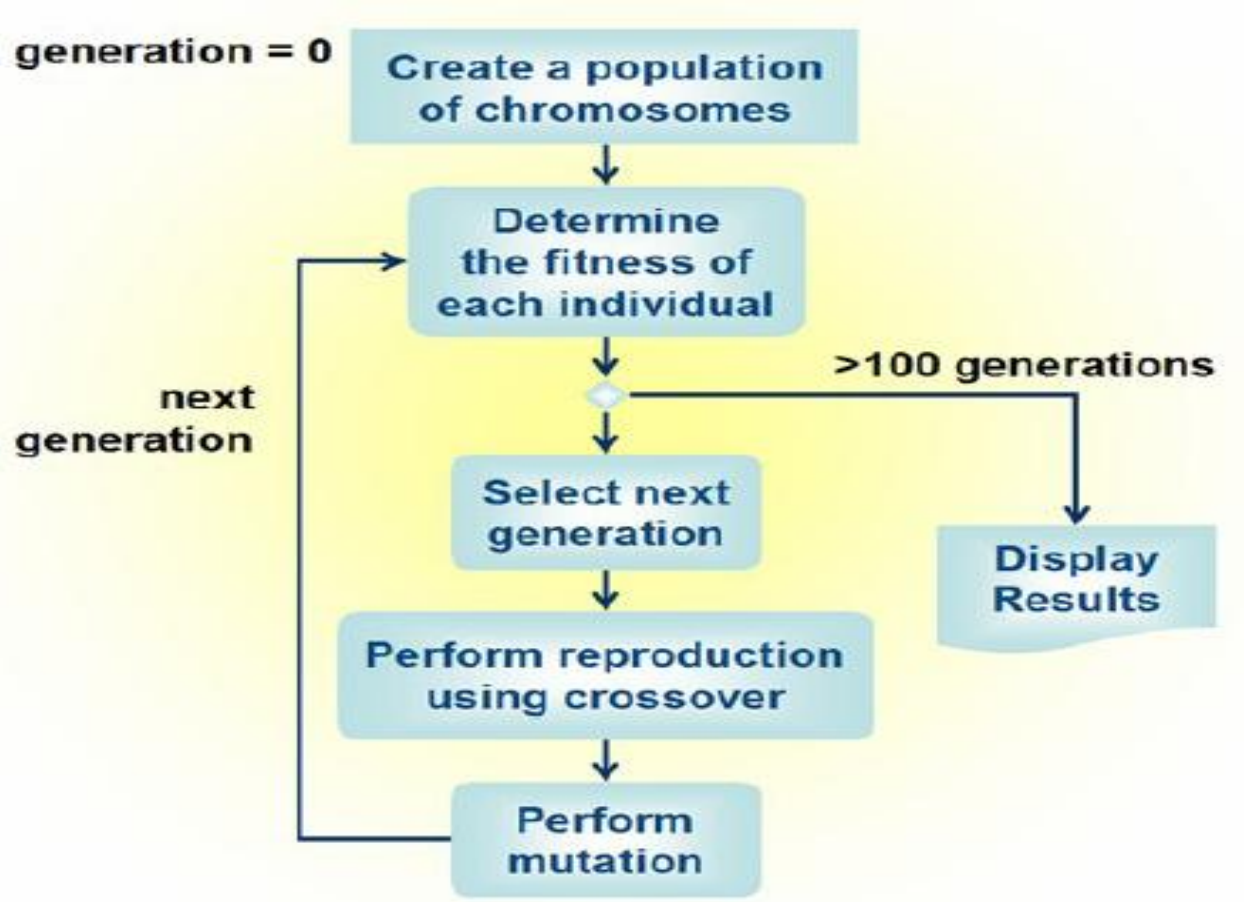
Keypoints:

- Genetic algorithms are inspired by Darwin's theory of natural evolution.
- In the natural world, organisms that are poorly suited for an environment die off, while those well-suited, prosper.
- Genetic algorithms search the space of individuals for good candidates.
- The chance of an individual's being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.

Outline of the Basic Genetic Algorithm

- [**Start**] Generate random population of *n* chromosomes (suitable solutions for the problem).
- [**Fitness**] Evaluate the fitness *f(x)* of each chromosome *x* in the population.
- Repeat until terminating condition is satisfied
  - [**Selection**] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
  - [**Crossover**] Crossover the parents to form new offsprings (children). If no crossover was performed, offspring is the exact copy of parents.
  - [**Mutation**] Mutate new offspring at selected position(s) in chromosome).
  - [**Accepting**] Generate new population by placing new offsprings.
- Return the best solution in current population

source: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_01.html



Issues involved

- How to create chromosomes and what type of encoding to choose?
- How to perform Crossover and Mutation, the two basic operators of GA?
- How to select parents for crossover?

Termination of Loop

- Reaching some (known/hoped for) fitness.
- Reaching some maximum allowed number of generations.
- Reaching some minimum level of diversity.
- Reaching some specified number of generations without fitness improvement.

source: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_01.html

Advantages and Disadvantages of GA

- Applicable when little knowledge is encoded in the system.
- Effective way of finding a reasonable solution to a complex problem quickly.
- NP-complete problems can be solved in efficient way.
- Parallelism and easy implementation is an advantage.
- However, they give very poor performance on some problems as might be expected from knowledge-poor approaches.

source: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_01.html