Mathematical Software - Genetic Algorithms - Tutorial

Genetic Algorithms - Introduction

Genetic algorithms (GA), considered as one of numerical methods, serve for computer simulations of genetic populations. Genetic algorithms (GA) can be divided into two main groups of haploid and diploid type. Both genetic models deal with populations of sequences of bits. Each sequence represents parameter or set of parameters written in

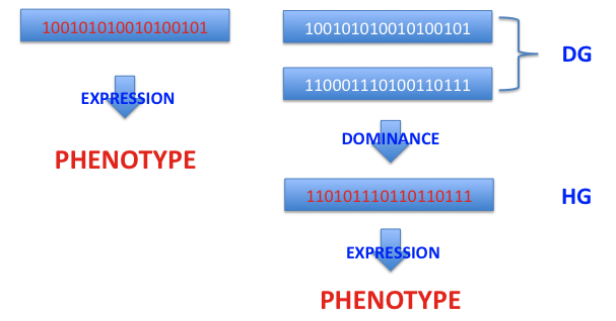
- a single binary string (constituting an artificial chromosome) in haploid GA
- a pair of strings (a pair of chromosomes) in diploid GA.

Information from that genotype is decoded to a numeric value (values) that is (are) called a phenotype. That decoding is often done according to commonly known representation of an integer number as a sequence of bits. Having this done the obtained numeric value (phenotype) is used to compute desired function according to a mathematical formula determining problem to solve. One-parametric problems are represented by scalar functions whereas more complicated multi-parametric ones are described by vector functions. In the first case, on the basis of a scalar representation of a system a corresponding **fitness function** is computed. In the second one, the **Pareto approach** is often applied (to learn more see The Pareto Condition.)
The fitness function is the essential measure of chromosome strength to occur in the next generation. The higher fitness of chromosome the greater chance to reproduce. Reproduction in population is thought as a randomized process based on the fitness function. Only these chromosomes that are

selected can take part in producing the next generation. In haploid models, a pair of chromosomes is a pair of parents that exchange their genetic information through the crossover process whereas, in diploid models, the process of gametogenesis uses the crossover mechanism to exchange genetic information between gametes. In spite of that small difference between haploid and diploid models in both cases it is demanded to find the crossover point or points (in models with multiply crossovers) on two parent chromosomes with assumed crossover probability. Apart from that random process genetic information can be modified through mutations. They also are random and occur with prescribed mutation probability. To enhance genetic variability of newly created population also genetic reconfigurations can be applied to chromosomes.

Genetic Algorithms - Haploidy vs. Diploidy

Haploid genotypes (HG) are built from single chromosomes whereas diploid ones (DG) from pairs of chromosomes. Making genetic material double results in enriching genetic expression process of dominance while a single chromosome is directly decoded to the phenotype.



Genetic Algorithms - Diploid Model - Dominance

Feature of dominance in a simple diallelic genotypes can be set as

	0	1
0	0	1
1	1	1

Diploid AG can be enlarged to triallelic genotypes where the bit of 1 value occurs in two variants i. e. of the weaker 1^R and the stronger 1^D type. Then a dominance scheme is shown in the table below

	0	1^R	1^D
0	0	0	1
1^R	0	1	1
1^D	1	1	1

Genetic Algorithms - Fitness Function

A fitness function is a non-negative criterion of quality computed for each individual in population. The better fitness the higher value of the fitness function. This feature of a genetic genotype can be hired to find the optimum solution (numeric value) for some mathematical problems like searching for maximum or minimum of a function. To do so, it is necessary to define a problem to solve as a **f(x)** function being *the aim function*. Let us consider two cases:

- finding minimum of a function f(x). That problem is equivalent to the following definition of the fitness function F(x):

$$F(x) = f_{max} - f(x)$$

where f_{max} constant ensures non-negative value of the F(x) function.

- finding maximum of a function f(x). Now the fitness function F(x) is defined as:

$$F(x) = C_{min} + f(x)$$

where C_{min} constant ensures non-negative value of F(x) function e.g. $C_{min} = |f_{min}|$.

Genetic Algorithms - Constraints - Punishment Function

Punishment function H is a penalty added/subtracted to/from *the aim function*

f(x) when any of constraints is violated. Consider two cases:

- finding minimum of a function f(x) with constraints given by $h_i(x)$ functions

where i = 1, 2, ..., n

$$f_{new}(x) = f(x) + r \sum H[h_i(x)]$$

where r is penalty coefficient and H function can be set as e. g. $h_i^2(x)$.

- finding maximum of a function f(x) with constraints given by $h_i(x)$ functions

where i = 1, 2, ..., n

$$f_{new}(x) = f(x) - r \sum H[h_i(x)]$$

where r is a penalty coefficient and H function can be set as e. g. $h_i^2(x)$.

Genetic Algorithms - Fitness Function - Scaling

Scaling of the fitness function allows to modify a survival ability of a chromosome in population. It is often done for two reasons. The first one is to strengthen reproduction of weaker genotypes in first generations. In that way population slower tends to the optimum and longer constitute more flexible genetic population. The second reason is to make greater differences between very similar genotypes in last generations when the average fitness of whole population is very close to the sought solution. This improves the resolution of the method. The following types of the fitness function scaling are used:

- linear

$$F_{sc} = aF + b$$

where a and b coefficients are found under two conditions:

  - conserving the average fitness i. e. $F_{sc,\ ave} = F_{ave}$

❍fitness maximum after scaling is set as $F_{sc,\,max} = C\,F_{ave}$ where a C multiplier comes from the range [1.2, 2.0]

Genetic Algorithms - Fitness Function - Sharing Function

A **sharing function** is a function of a distance between chromosomes in a population. This distance between chromosomes can be computed on the basis of their phenotypes or genotypes. When phenotypes are a measure of chromosomes similarity the distance function is given by the formula

$$d_{ij} = |x_i - x_j|$$

where x_i and x_j are phenotypes of *i*-th and *j*-th chromosome, respectively.

Otherwise, when the computed distance reflects similarity between two genotypes **the Hamming measure** is used. It is defined as the number of different bits on corresponding sites on both chromosomes of equal length divided by the length of the first or the second genotype. The **sharing function** S(d) can be set as a linear or a power function of a distance d_{ij} between chromosomes i. e.

$$S(d_{ij}) = 1 - d_{ij}^{p}$$

where the power p is a parameter in a genetic algorithm. The **sharing function** is computed in order to modify the fitness function F_i of *i*-th chromosome in the way:

$$F_{i,\,sharing} = F_i / \sum S(d_{ij})$$

where the sum is over j.
Applying a sharing function to the genetic algorithm leads to splitting a population into a set of sub-populations. Thus the sharing function approach

can be done to simulate species creation in genetic populations. For that reason, in mathematical optimization problems, it is used in searching for optimum of multi-modal functions.

Genetic Algorithms - Fitness Function - Selection

Having all population built with fitness factors computed for every chromosome in population the group of parents of next generation can be chosen. A selection mechanism is based on the one main rule *the better fitness the greater reproduction (more children)*. It can be done by one of the following methods:

- **roulette:** each genotype has computed its fitness factor:

$$F_i/F_{sum}.$$

The whole roulette is divided into sectors of areas proportional to fitness factors. Computer simulation allows to choose one genotype each roulette turn. The roulette is turned up to complete a new population.

- **modified roulette:** The main part of algorithm is the same as in the roulette method. The difference occurs when a chromosome is selected to replication. Then its fitness factor is diminished by an assumed value (e.g. 0.5).

- **deterministic:** each genotype has computed its fitness factor (F_i/F_{sum}).

Then number of assigned copies to the *i*-th genotype is computed as the integer part of a number of expected copies

$$n_i = n*(F_i/F_{sum})$$

where F_i is a fitness of *i*-th genotype and n denotes the population size. The

rest of free places in a new population is given to those genotypes which have greater floating parts of n_i.

- **random based on rests - type I:** first step is the same as for deterministic

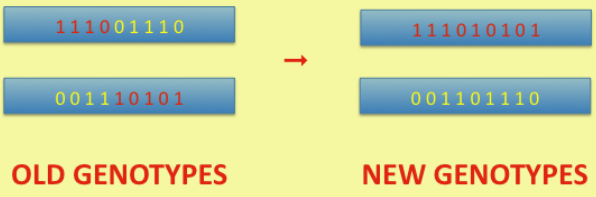
selection. The rest of free places in a new population is taken from the roulette method. The roulette is built from the floating parts of fitness factors.

●**random based on rests - type II:** first step is the same as above. This time floating parts are treated as success probabilities in the Bernoullie's method.

●**Wetzel's method:** at the beginning pairs of genotypes are chosen by the roulette method. From each pair that genotype which have greater fitness factor is chosen to a new population.
In each case, the selection is done up to complete a new population.

Genetic Algorithms - Reproduction - Crossover

Crossover is a process applied to a pair of chromosomes. As the first step the **crossover point** is randomly chosen with assumed probability. That point defines a position of dividing bit on both chromosomes. Bits taken before the crossover point from the first chromosome are merged with bits from the second one lying on and after that point. In that way a new genotype is created. The second genotype from the new pair is formed equivalently. In the example shown below cross point is 5.



Genetic Algorithms - Reproduction - Mutations

Mutations are applied to each new chromosome with prescribed probability. If mutation occurs a bit value is changed to its opposite charge i. e. $1 \rightarrow 0$ and $0 \rightarrow 1$ in diallelic models whereas in triallelic ones this change is like this one: $1^R \rightarrow 1^D$ or $1^R \rightarrow 0$.

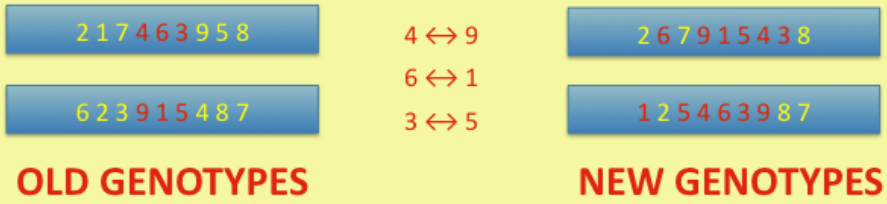
Genetic Algorithms - Reproduction - Reconfigurations

Reconfigurations are applied to chromosomes having more than one gene. In

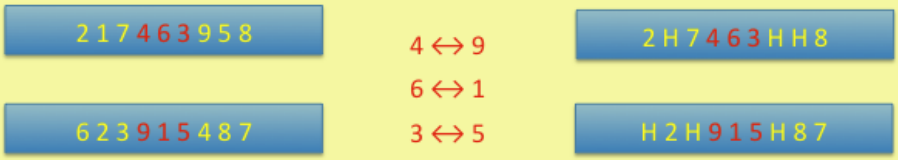
such cases allele location (locus) on a chromosome is not constant. It can vary depending on reconfigurations that occurred. The list below presents main types of reconfigurations

●**inversion**: during inversion chromosome is cut in two randomly chosen points. The part between points is reversed and merged with both ending parts. A little modification to this type of reconfiguration can be applied. The inversion (linear) is done with probability 0.75 otherwise with probability 0.125 right-end-inversion or left-end-inversion is done.

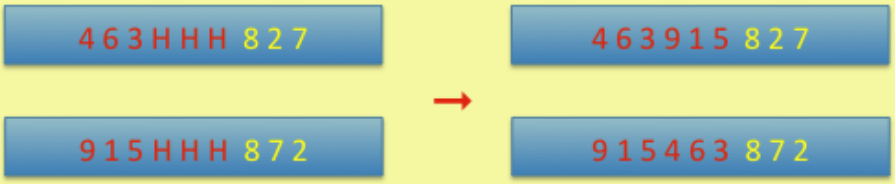
●**partially matched crossover**: two chromosomes exchange genetic information. The matching section is indicated by two points randomly found. Then alleles from both chromosomes from matching section create a map that is used to construct new chromosomes through transpositions. The numbers below present alleles located in ordered way on a chromosome. So far alleles were set on a chromosome from the first to the last e. g. 1 to 9. Now alleles can have permuted ordering.



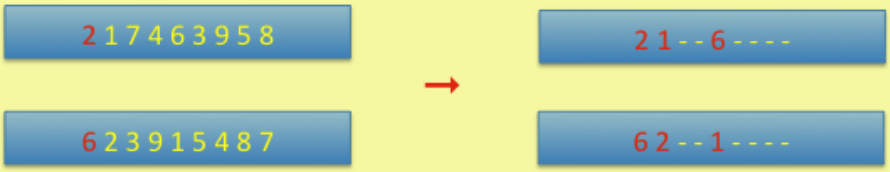
●**order crossover**: the matching section is indicated by two points randomly found as previously. Then alleles from matching section create a transposition map. The alleles from the section are substituted by holes (H).



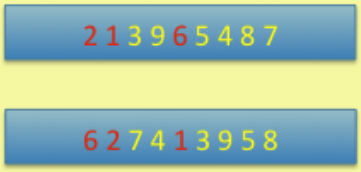
Then the part before matching section is added to the part after it and holes are shifted to the matching section and replaced by alleles from the matching section of the complementary chromosome.



●**cycle crossover**: at the beginning the first alleles from both chromosomes are set as starting values, in turn, the corresponding allele from the complementary chromosome is set on the current chromosome. It lasts as long as the starting allele is reached.



The free sites are taken from the complementary chromosome.



Genetic Algorithms - Pareto Approach

The Pareto condition says that a vector x is lower (partially) than a vector y when the following condition is fulfilled

$$(\forall i)(x_i \leq y_i) \wedge (\exists i)(x_i < y_i)$$

where x_i and y_i are *i*-th components of vectors x and y, respectively. These vectors may represent fitness vectors **F** belonging to different chromosomes in a population. Then problem of finding better solutions on the basis of higher value of scalar fitness function is replaced by searching for a group of chromosomes fulfilling the above-written condition. The Pareto condition, due to the way it is formulated, can be naturally applied to find a function minimum. When opposite problem is put forward a fitness vector should be multiplied by

-1. Then above-written condition can be used straightforward.
That part of chromosomes that satisfies the Pareto condition has the same power to reproduce which results in the same rang value in selection routine. Having selected the first group of *Pareto-positive* chromosomes the next one can be chosen in the same manner within a sub-population of *Pareto-rejected* chromosomes. The group of genotypes has assigned successive rang value, obviously. The lower rang the greater number of children associated with. That procedure continues up to finding rang values for all chromosomes in population.
To obtain better results one can associate the multi-parameter Pareto-based problems with a technique of species forming (e.g. using the sharing function approach).

Genetic Algorithms - OptFinder

Package for genetic search - OptFinder.
To test this software partially, go to Online Optfinder Page.

Genetic Algorithms Machine Learning

Genetic algorithms (GA) play also an important role in learning systems called the *Genetic Based Machine Learning Systems*. One type of these systems is called the **Classifier System**.
Classifier systems are systems that learn some rules to act in given circumstances. They are built from the following sub-systems

- a rule and message system
- a credit algorithm
- a genetic search system (GA)

To learn more go to machine learning page.

Genetic Algorithms - References

1.^ David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc. 1989

Last update: April 2, 2019

Privacy PolicyTerms of Use