Journal of Software Engineering and Applications, **2011, 4, 482-486**
doi:10.4236/jsea.2011.48055 Published Online August 2011 (http://www.SciRP.org/journal/jsea)

On Some Basic Concepts of Genetic Algorithms as a Meta-Heuristic Method for Solving of Optimization Problems

Milena Bogdanović

Teacher Training Faculty, University in Niš, Vranje, Serbia.
Email: mb2001969@beotel.net, milenab@ucfak.ni.ac.rs

Received June 27th, 2011, revised July 25th, 2011, accepted August 1st, 2011.

ABSTRACT

The genetic algorithms represent a family of algorithms using some of genetic principles being present in nature, in order to solve particular computational problems. These natural principles are: inheritance, crossover, mutation, survival of the fittest, migrations and so on. The paper describes the most important aspects of a genetic algorithm as a stochastic method for solving various classes of optimization problems. It also describes the basic genetic operator selection, crossover and mutation, serving for a new generation of individuals to achieve an optimal or a good enough solution of an optimization problem being in question.

Keywords*: *Genetic Algorithm*, *Individuals*, *Genetic Operator*, *Selection*, *Crossover*, *Mutation

1. Introduction

The word ***heuristic*** is derived from the Greek word ***εύρίεσκω*** meaning I find. From here it can immediately that the heuristic algorithms incurred in fact from experimentation in order to obtain a satisfactory solution. An important feature of heuristic algorithms is that can approximately, but still good enough to solve the prob lems of exponential complexity. However, it should be stressed the problem heuristic algorithms may not lead to a satisfactory solution, and for some of problems, heuristic algorithms give relatively poor results.

Genetic algorithms are robust and adaptive methods which in addition to other fields of application can be used for solving combinatorial optimization problems. The central concept in the description of genetic algorithms is a population of individuals, which is usually between 10 and 200; each individual represents a possible solution in the search space for a problem (the space of all solutions).

These algorithms can be used for solving of various classes of problems because they are of general nature. By mode of action, genetic algorithms are among the methods of guided random search techniques of space solutions in looking for a global optimum. In the same group can be classified several other methods based on similar principles, such as:

- evolutionary strategies,
- simulated annealing,
- genetic programming.

Evolutionary strategies, developed in Germany in the sixties of the last century, have a lot of common features with a genetic algorithm. It is difficult to determine the border between the two approaches having in mind their various variants. Both methods work with a population of solutions over which are implemented defined operations which periodically repeat. The phases of this process having a model in the natural evolutionary flows are called *generations*.

Simulated annealing is a process having found a basis in the thermodynamic motion of matter to the energy-minimum in a gradual lowering of temperature as a parameter of the system. The method works with a single solution from which in the each iteration is required a "neighboring" solution. An old solution is always replaced with a new if one if it came to satisfying of criteria and it is possible a better solution to be replaced by a worse one is better if certain of stochastic are satisfied regulating the "temperature" of the system. A higher temperature gives a higher probability that a new, even a worse solution, to replace the old one. The process starts from a certain temperature, which allows a relatively

high probability of acceptance (more than 50%), this parameter to decrease exponentially until the motion becomes almost deterministic.

Genetic programming is an automated optimization process of developing computer programs, whose purpose is solving most complex problems in the field of computing, but also the problems that we encounter every day. The concept is based on ideas drawn from the general theory of genetic algorithms and other evolutionary methods. Simply put, the ultimate goal of genetic programming (as product) is a universal computer program that finds solutions to problems as input data.

The genetic programming can be accessed in several different ways and from different perspectives, such as from linear—most of them are universal access to a computer program as a formal tree in the context of graph theory. In fact, any computer program can be viewed as a tree or forest trees (in the broad sense), where the internal tree nodes have the role of the operator (or function of a number of variables), and leaves the role of operands. In this case, a set of operators (*function set*, *F*) and operands (*terminal set*, *T*) are predefined sets of the final.

In accordance with the general theory of evolutionary algorithms, we can say that the role of chromosomal genetic programming with non-linear structures play graphs and trees. It is the properties of trees as the strict mathematical objects, such as a simple recursive tour, sealed the developmental course of genetic programming in a pronounced direction.

Genetic algorithms simulate the natural evolutionary process. For the evolutionary process can be determined as follows:

- there is a population of individuals;
- some individuals are better (better adapted to the environment);
- better individuals have a higher probability of survival and reproduction;
- properties of individuals are written in the chromosomes using the genetic code;
- children inherit the properties (characteristics) of parents;
- Mutation can affect the individual.

The individuals represent potential solutions for genetic algorithm, while the environment is the objective function.

All data having indicates an individual, are written in a single chromosome. In the most general case, the chromosome can be any data structure who describes the characteristics of one individual. The chromosome represents a possible solution to a given problem for the genetic algorithm. It is necessary to define the genetic operators for each data structure. These genetic operators should be defined so that individuals do not create new solutions that are impossible, because in this way significantly reduce the performance of genetic algorithm.

No matter what kind of genetic algorithm works, the algorithm has the following parameters: population size, number of generations or iterations and the probability of mutation. For generations the genetic algorithm should also be mentioned, and the likelihood of hybridization. In eliminative genetic algorithm, instead of crossing probability, specifies the number of individuals for elimination.

2. Coding and Fitness Functions

As the most important aspects of a genetic algorithm, point out the encryption functions (coding) and adaptability (fitness), which is very important to be well adapted to the nature of a particular problem. It has been said that the usual binary encoding or of a higher cardinality alphabet. Preferably the connection between the genetic codes and solutions to the problem is an injective and onto mapping. Then it is possible that the application of genetic operators in a certain generation get called incorrect individual, or the individuals whose genetic code does not correspond to any solution. Overcoming this problem is possible in several ways. One possibility is to allocate any such individuals as the adaptation of the function is zero, so that operators are already using such selection to eliminate individuals. This approach proved suitable only when the ratio of the number of incorrect and correct individuals in the population is too large, which in practice often not the case. It is possible, however, the incorrect inclusion of individuals in the population, so that any unfair individuals assigned the value of penalty function. The goal is that individuals and unfair given a chance to participate in the crossing, but to be discriminated against in relation to the correct individual. Should take into account that the value of penalty function balance, because too small values can lead to the fact that some of the genetic algorithm codes incorrect decalrations of the solution, while, on the other hand, excessive punishment can cause loss of useful information from the incorrect individuals. There is another way to solve this problem—and it is unfair to individual repair to make them correct or incorrect that each individual is the correct replacement.

Calculating the function of adaptation is possible in several ways. Some of these methods are direct download, linear scaling, interval scaling, sigma clipping, etc. The simplest way of measuring the function of adaptation is a direct download, which means that the value function for the adaptation of a specimen is taken, its value of objective function. However, in practice this approach gives poor results.

The fitness for some individuals is calculated as a linear function of the objective function values and individuals in the case of linear scaling. For this, there are different approaches to choosing coefficients that accurately determine the linear function.

scaling functions in the unit interval adaptation looks at the interval [0, 1], with the best individual has the adaptability which is equal to 1, and the lowest adjusted 0.

If the formation of adaptation functions using sigma clipping, then the function of adaptation calculated from the formula (1)

$$f(x)=\begin{cases}0, x<\overline{x}-C\sigma\\ x-(\overline{x}-C\sigma), x\geq\overline{x}-C\sigma'\end{cases} \quad (1)$$

where $\overline{x}$ and σ are average value of objective function and standard deviation in the population.

3. Selection

The selection is directly related to the function of adaptation and the main mode of implementation of this genetic operator is the simple roulette selection. This method use a distribution where the probability of selection proportional to its adaptation to the individual. The individuals involved with the chances of roulette in accordance with them, pass or not pass the process of creating a new generation. The lack of a simple roulette selection is the possibility of premature convergence due to the gradual prevalence of highly adapted individuals in the population that do not correspond to the global optimum.

It can be used selections based on the ranking of genetic codes according to their adaptability to avoid this problem. The function of individual adaptation is equal to a range of pre-specified range of ranks, and depends only on the position of individuals in the population. It can be used in linear fashion, as well as other forms of ranking.

The tournament selection is another form of selection. In tournament selection on randomly generated subsets of the *N* units (*N* is the predetermined number), then in each subset, the principle of the tournament, selects the best individual that participates in the creation of a new generation. Usually the problem is the choice of *N* so as to reduce the adverse effects of stochastic, so that better and more diverse genetic material passed to the next generation. In cases where the size is perfect tournament is not an integer, it has proved successful fine-graded tournament selection (FGTS). A detailed description of these and other types of selection and its theoretical aspects can be found in [1]. The application of finely graduated tournament selection and comparison of the practice with other operators of selection are given in [2-4].

4. Crossover

Process of exchanging genetic material between individuals of parents, in order to form new offspring individuals is performed by the operator crossover. The most common type of operators are crossing one-position, two-position, multi-position, and uniform crossover, and can also be used for crossover mixing (shuffle), reduced surrogate crossing, crossing with the parent, intermediate recombination, and linear recombination.

The operator of crossing, which is implemented in a simple genetic algorithm, is the one-position crossover. In the one-position crossover is determined by the so-called position of the crossing. All genes from the predetermined positions, changing seats to make every parental pair created two offspring. In the two-position crossover were set at two positions and is the exchange of genetic material between the parents and two positions.

It should be noted that for each parental pair defines a binary array of length the same as the genetic parents as for the uniform crossover. This range is called a mask. The exchange of genes is performed only on those positions where the mask is 0, while the positions where there is one, the parents retain their genes.

The interference in crosses (or shuffle), choose only one crossing point, but before the parents exchange the genes they "stirred" (to deploy). The genes of offspring returning to the old place after the crossover in this way it's also eliminates the so-called positional preference (see [5]).

The crossover with the reduced surrogate is making the crossing of the sea, whenever possible, give the new offspring. Usually, this intersection is implemented by the location of the crossing limited to, those in which the values of different genes, about this technique more in [6].

There is also a technique of crossing with the parent. It simulates the propagation of insects (bees) in which one individual (parent) with the best feature of adaptation is involved in all the crosses with other individuals (drones). In some applications, this type of crossing operators achieve significantly better results than other techniques.

In intermediate recombination (crossing) is used in animals with the real values of genes and enables the production of new phenotypes around and between the values of parental phenotypes. The offspring is obtained according to the formula $O_1 = P_1 \times \alpha(P_1 - P_2)$, where α is a scaling factor that is chosen uniformly and randomly from an interval, usually [–0.25, 1.25], and P_1 and P_2 are parental individuals. Each variable of the seed is the result of combining the variables of the parents in a way that is given in the formula, by which the newly elected α for each parental pair, more about this operator in [7].

The linear recombination or crossover is similar to the

intermediary, except that it uses only one value for α in the intersection, [8].

The selection operators crossing must be adapted to the nature of problems being solved as in the case of determining the functions of adaptation and the other operators.

5. Mutation

The mutation operator is considered one of the most important and as such it can decisively influence the operation of genetic algorithm. If the genetic algorithm uses binary encoding and the population has no incorrect units, then it is usually implemented by a simple mutation operator that runs through the genes of the genetic code of the individual and for each of these checks is muted or not. The probability of gene mutations p_{mut} is a preset small size, usually taken from the interval [0.001, 0.01]. Simple mutation can sometimes be implemented through a binary number - *the mask*, which is randomly generated for each individual, and carries information about the position in which the genetic code is made between the genes.

To accelerating the realization of a simple mutation operator can use the binomial or normal distribution. The mutation using the binomial distribution using the fact that a random variable X_{ind} = number of mutated genes has a binary distribution of individual, $B(n, p_{mut})$ where n is the length of the genetic code, a p_{mut} level mutations. Let $F(K)$ its distribution function. With F^{-1} we find that the n_{mut} = number of genes to mutate in a given genetic code. The positions of genes that are mutated are chosen uniformly from $\{0,1,\cdots,n-1\}$. In the case of a long genetic code could result in errors when calculating the number of n_{mut}, so it is then convenient to use a mutation with a normal distribution. If $n \to \infty$, then the random variable X_{ind} can be approximated by a normal distribution $N(n * p_{mut}; n * p_{mut} * (1 - p_{mut}))$, provided that n is large enough and small enough p_{mut}. As with the previous case, we use the inverse function of the normal distribution functions in order to calculate the number of genes to mutate, and their positions determined in the same way. It is developed a variant of this operator that is applied to the entire population as the mutation using genetic codes of all specimens can be viewed as one entity, then.

If the mutation is not uniform, which normally happens when the genes of the genetic code are not equal, but some parts of the code necessary to mutate with a lower or higher probability, commonly used normal mutation (applied in accordance with normal distribution), exponential mutation (number of mutated genes in the code is exponentially decreasing) and so on.

When the genetic algorithm uses the entire coding or real numbers (floating point), it was necessary to develop other concepts of mutation, which was done. These are the replacement of genes randomly chosen number (random replacement), add or subtract a small value (creep), multiply the number is close to one (geometric creep) and so on. For both creep mutation operator required values are random and can have a uniform, exponential, Gaussian or binomial distribution (see [9,10]).

In some cases it is useful to genes, depending on the position in the genetic code, have different levels of mutation. In this regard it is especially important concept of frozen gene. Namely, if in a position of the genetic code in all or most of the population, the same gene, it is useful to the gene mutation has a higher level than the rest of the genetic code. This concept is used to restore lost diversity of genetic material, and these genes are called frozen.

6. Other Aspects of Genetic Algorithms

The successful application of genetic algorithm depends largely on the choice of policy replacement. Some of the most important politics of the generation: generational genetic algorithm, steady state genetic algorithm and the elitist strategy. Of course, it is possible to combine these principles.

When applied generational genetic algorithm, then apply to all individuals all the genetic operators, i.e. there are no privileged individuals are going into the next generation, or individuals who directly pass the selection process.

On the contrary, steady state genetic algorithm favors the best individuals in the population so as to them shall not apply operator selection, but they go directly to the next stage, while other individuals applied selection and they come to the remaining places.

The elitist strategy provides a direct passage into the next generation of one of the best individuals. For these individuals do not apply to operators of selection, crossing and mutation. By applying genetic operators on the population of individuals fill the remaining seats in the next generation.

This approach leaves room for another possible improvement of genetic algorithm, which is cached. As an elite individuals pass from generation to generation unchanged, it and its value remains unchanged. Therefore, it would be useful to value elitist individuals remain memorized, rather than constantly calculated, thus saving the time required for their computation. This process is called caching, and more details are found in [11] and [7].

Specifically, the values of the objective function of individuals are stored in so-called hash-row table, with the

use CRC codes that are associated with genetic code of individuals. If, during operation of the genetic algorithm, we get again the same genetic code, then the value of the objective function is taken from a hash-table, through the CRC code.

The technique, which is used to cache genetic algorithm, a well-known technique LRU—Least Recently Used strategy. There is a limit for this concept of genetic algorithm, and that is that the number of cached values of objective function is limited to N_{cache}, which depends on the implementation.

7. Conclusions

Mechanism of selection favoring above-average adjusted individuals and their above-average adjusted parts (genes), which receive a higher chance of their own play in the formation of a new generation. In this way, less-adapted individuals and genes get reduced chances of reproduction and gradually dying out.

Contribution to the diversity of genetic material gives the operator the crossing, which is recombination of genes of individuals. The exchange of genetic material between individuals, with the possibility that well-adjusted individuals generate better individuals as a result of the crossing structure is obtained, although non-deterministic. Mechanism of crossing operators and relatively less adapted individuals, with some well-adapted genes, get their chance to recombination of good genes produce well-adjusted individuals.

However, multiple applications of selection and crossing, there may be loss of genetic material, and some regions in the search space become unavailable. Mutation operator performs random change a particular gene, given the low probability P_{mut} which can restore the lost genetic material in the population. This is the basic mechanism for preventing premature convergence of genetic algorithm to a local extreme.

Genetic operators provide the offspring are similar but not identical with their parents, which allows the population to evolve a solution that was not present in the initial set of objects (individuals).

The evolutionary approach has enabled genetic algorithms to represent the true global optimization technique, which is able to identify solutions close to or identical with the global optimum. The genetic algorithms can be applied in traditional problem areas, such as tasks with a large number of local optima and discontinuities, how not to depend on local information.

REFERENCES

[1] V. Filipović, "A Proposal to Improve Operator Tournament Selection in Genetic Algorithms," Msa Thesis, Faculty of Mathematics, University of Belgrade, 1998, (in Serbian).

[2] V. Filipović, J. Kratica, D. Tošić and I. Ljubić, "Fine Grained Tournament Selection for the Simple Plant Location Problem," *Proceedings on the 5th Online World Con- ference on Soft Computing Methods in Industrial Appli- cation—WSC5*, 2000, pp. 152-158.

[3] V. Filipović, D. Tošić and J Kratica, "Experimental Results in Applying of Fine Grained Tournament Selection," *Proceedings of the 10th Congress of Yugoslav Mathematicians*, Belgrade, 21-24 January 2001, pp. 331-336.

[4] V. Filipović, "Fine-Grained Tournament Selection Operator in Genetic Algorithms," *Computing and Informatics*, Vol. 22, No. 2, 2003, pp. 143-161.

[5] F.J. Burkowski, "Shuffle Crossover and Mutual Information," *Proceedings of the 1999 Congres on Evolutionary Computation*, 1999, pp. 1574-1580.

[6] L. Booker, "Improving Search in Genetic Algorithms: In Algorithms and Simulated Annealing," Morgan Kaufmann Publichers, San Mateo, 1987, pp. 61-73.

[7] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization," *Evolutionary Computation*, Vol. 1, No. 1, 1993, pp. 25-49. doi:10.1162/evco.1993.1.1.25

[8] C. Höhn and C. R. Reeves, "Embedding Neighbour-Hoodsearch Operators in a Genetic Algorithm for Graph Bipartitioning," Ctac Technical Report, Coventry University, Reeves, 1995, pp. 33-34.

[9] J. Kratica, "Parallelization of Genetic Algorithms for Solving Some NP-complete Problems," Ph.D. Thesis, Faculty of Mathematics, Belgrade, 2000, (in Serbian).

[10] D. Tošić, N. Mladenović, J. Kratica and V. Filpović, "Genetic Algorithm," Institute of Mathematics SANU, Beograd, 2004, (in Serbian).

[11] J. Kratica, "Improvement of Simple Genetic Algorithm for Solving the Uncapacitated Warehouse Location Problem," In: R. Roy, T. Furuhachi and P. K. Chawdhry, Eds., *Advances in Soft Computing—Engineering Design and Manufactining*, Springer-Verlang London Limited, London, 1999, pp. 390-402.

