ISSN 1512-1232

SOLVE OF A PROBLEM OF OPTIMIZATION USING GENETIC ALGORITHMS

Natela Ananiashvili
Faculty of Engineering, Georgian Aviation University, 16 Ketevan Dedopali Avenue. Tbilisi, Georgia

Abstract
Heuristic algorithm of solution of a problem of optimization is offered. The algorithm is based on main principles of classic genetic algorithm. Genetic operators are considered. Selection is made with technique of inbreeding or outbreeding. Operators of crossover and mutation are offered. Algorithms of encoding and decoding of true numbers are selected. Algorithm is approved for test problems. The quantitative results prove efficiency of genetic operators and their combination.

Keywords: *genetic algorithm, real and binary encoding, operators of crossover and mutation.*

1. Introduction

The main complexity of solution of optimization problems is related to non-linearity of functions and scales of problems. Searching of optimal values with classical techniques often finishes without results. That is why genetic algorithms were developed. Genetic algorithm is one of the types of random search.

Majority of classic algorithms of optimization [1] and their modifications [2-9] begin search from certain initial point of a searchable space. Afterwards, solution is improved step by step. The disadvantage of such technique is that it becomes necessary to calculate first or higher rank derivative of a target function. Besides, search may end at local minimum. Genetic algorithms search simultaneously in several directions [10-17]. Transition from one population to another gives us ability to avoid getting at local minimum. For today, from the non-classic algorithms that are used to search global extremum of function of multiple variables, the most popular are genetic algorithms. In these algorithms genetical principles of population's development are simulated. In the process of formation of genotype, new individual is formed with crossover of parents' chromosomes and random change of genotype, known as mutation in a nature.

The work offers genetic algorithm of solution of a problem of optimization:

$$f(x) \rightarrow min, x \in R_n \quad (1)$$

Genetic algorithms represent one type of heuristic algorithms that were developed as a result of observation of living organisms. In these algorithms everything is similar to process of development of biologic populations. They represent laws such as natural selection, inheritance of genetic information and organism's ability to adapt to living environment. The idea to copy processes of evolution and selection has originated in 70s of the past century and it belongs to Holland [10].

2. The General Scheme of Genetic Algorithm

The algorithm offered for solution of this problem implies the general principles of genetic algorithm. This algorithm can be described non-formally in the following way:

Algorithm 1. Formation of initial population

Step 1. Determine size of initial population: *P*;

Step 2. Select scheme of encoding: binary encoding;

ISSN 1512-1232

Step 3. Determine necessary quantity of ranks *k* for encoding data with precision of rank *k*0.

Step 4. Select decimal numbers in given range $x = (x_1, x_2, \dots, x_p)$;

Step 5. Encode array *x* in binary array *s*;

Step 6. Convert binary array *s* into Grey code.

Let us describe realization of these steps. Let us assume size of initial population is *P*. We select scheme of binary encoding and afterwards, begin selection of elements of initial population in given interval. Let us also assume that precision of representation of numbers is *k*0 decimal rank after comma. Interesting interval is [*a, b*]. Maximal number of bits that are used to write these numbers into binary code is *lbit*. We can use 32-rank or 16-rank encoding, but in our work we use *lbit*=32. Number of bits that are necessary for real encoding can be calculated with the following pseudo-code:

$\delta = (b - a) \cdot 10^{k0}, \quad k = k0$

$\boldsymbol{for}(\boldsymbol{int}\ i = k;\ i < lbit;\ i++)$

$\boldsymbol{if}((2^{k-1} < \delta)\&(\delta \le 2^k - 1))\ \boldsymbol{break};$

$\boldsymbol{else}\ k++;$

Our numbers will occupy bits *k*, i.e. first ranks from rank *lbit*. This bit *k* can be filled with any sequence of 0s and 1s, but other higher rank bits should be filled with 0s.

Pseudo code for generation of these numbers can look like this:

$p0 = 2^k / RAND\_MAX$

$\boldsymbol{unsigned\ long}\ popul[P];$

$\boldsymbol{for}(\boldsymbol{int}\ i = 0;\ i < S;\ i++)$

{

$\boldsymbol{if}(p0 == 0)\ popul[i] = rand()\%2^k;$

$\boldsymbol{else}\ popul[i] = rand()\%p0;$

}

(**Note:** here and in the following algorithms indices of arrays are written in brackets).

We get number *P* of encoded chromosomes in array *popul*[].

Reverse conversion or conversion of row of bits into true numbers, i.e. decoding is possible with the following pseudo code:

$\boldsymbol{for}(\boldsymbol{int}\ i = 0;\ i < \mathrm{P};\ i++)$

{ $\boldsymbol{float}\ popul10[i] = a + popul[i] * ((b - a)/(2^k - 1));$

}

In array *popul*10[] we get decimal values of chromosomes.

ISSN 1512-1232

3. Selection with Technique of Inbreeding or of Outbreeding

We must select individuals from previous population (with certain rule) and then crossover them to get next population from existing one. Afterwards, selected parent individuals will be replaced with successor, crossovered individuals.

Selection implies determination of criterions to choose individuals for crossover. The work offers technique of inbreeding, i.e. one parent is selected randomly and another one is maximally "similar" to already selected individual. Similarity is determined with Heming's distance. It is a quantity of different bits in corresponding row of two individuals s^r and s^p:

$$d_h(s^r, s^p) = \sum_{i=1}^{n} |s^r - s^p| \tag{2}$$

Let us assume that two individuals s^r and s^p are similar. In case of inbreeding, if

$$d_h(s^r, s^p) = min\{d_h(s^r, s^p), \forall r = 1,2, \dots S, p = 1,2, \dots, S, r \neq p\} \tag{3}$$

in case of outbreeding:

$$d_h(s^r, s^p) = ma\ x\{d_h(s^r, s^p), \forall r = 1,2, \dots S, p = 1,2, \dots, S, r \neq p\} \tag{4}$$

when two parents for crossover are selected from current population with technique of inbreeding or outbreeding, we use single-point crossover.

4. Crossover

Let us assume that at the increment of selection, numbers of selected parent chromosomes are j_1 and j_2. Crossover algorithm looks like this:

Algorithm 2. Crossover

Step 1. Chromosomes with numbers j_1 and j_2 are decoded. We converted them into Grey code when initial population was selected. Now their values are written into arrays *vf* and *vm*;

Step 2. Take a random number from range $\vartheta \in [1, lbit]$, where lbit is a number of bits that is necessary to encode chromosomes;

Step 3. Replace elements of arrays *vectf* and *vectm* with indexes from ϑ to *lbit*;

Step 4. Encode "crossovered" arrays *vectf* and *vectm* at the places of chromosomes with numbers j_1 and j_2 into positional binary code. Afterwards, once again convert them from binary to Grey code.

We can *gf* decoded and written into any array *vf*:

```
for(int i = 0; i < lbit; i++)
    vf[i] = 0;
unsigned long c0,c; int c1;
for (int j = 0; j < lbit; j++){
```

ISSN 1512-1232

$c0 = 1;$

$c = c0 << (lbit - j - 1);$

$c1 = gf \ \& \ c;$

$\boldsymbol{if}\ (c1! = 0)\ vf[j] = 1;$ }

where *lbit* is a quantity of bits that is necessary to encode chromosome.

After crossover is finished, we get new population that differs from previous one, because parent chromosomes are replaced with child chromosomes, but formation of new population is not finished yet. One or more chromosomes should mutate. Mutation operator is described below.

5. Mutation

Step of mutation is realized after crossover. We describe mutation algorithm for function of two variables. One or more genes change in the process of mutation.
If size of population is *P*, then number of bits necessary to encode chromosome into populations k_1 and k_1 is (x, y) correspondingly. *i*th chromosome ins encoded into arrays xch and ych.

Algorithm 3. Mutation

Step 1. Generate some random number from range $rn \in [0,P]$. Chromosome with number rn is subject to mutation;

Step 2. Generate some random number from range $rn \in [0,100]$. If $r < 50$, then gene $xch[rn]$ is subject to mutation, but if $r \geq 50$, then gene $ych[rn]$ is subject to mutation;

(**Note:** If number of variables is different, then mutable gene is determined with the same rule according to percentage value of r. For instance, if we have four variables (x_1, x_2, x_3, x_4), the following choices are available: if $r \leq 25$, then select x_1, if $25 < r \leq 50$, then select x_2, if $50 < r \leq 75$, then select x_3 and if $75 < r \leq 100$, then select x_4);

Step 3. On the basis of mutated gene, generate random number $rk \in (0,k_1)$ or $rk \in (0,k_2)$. Here k_1 and k_2 are number of bits necessary to generate genes $xch[rn]$ and $ych[rn]$;

Step 4. Convert selected gene $xch[rn]$ (or $ych[rn]$) into Grey code. Let us assume it is g;

Step 5. Decode g into binary vector v;

Step 6. Invert $v[bit - rk - 1]$ and write 0 instead 1 or write 1 instead 0 into it;

Step 7. Encode binary vector v into gene $xch[rn]$ or $ych[rn]$ according to which gene was mutated.

We can use the following operator to transfer the $xch\ [rn]$ gene into the gray code:

$g = xch[rn]\text{^}(xch[rn] >> 1);$

ISSN 1512-1232

Afterwards, *g* should be written into *lbit* rank of array *v*. For this purpose, we use program with the following fragment:

```
for(int i = 0; i < lbit; i++)
    v[i] = 0;
    for (int j = 0; j < lbit; j++){
      c0 = 1;
      c = c0 << (lbit - j - 1);
      c1 = g & c;
      if (c1! = 0) v[j] = 1;
                        }
```

at last, rank *rk* is mutated by means of the following operator:

```
if (v[bit - rk - 1] == 0) v[bit - rk - 1] = 1; else v[bit - rk - 1] = 0
```

After mutation we get new population.

6. Determination of fitness function and estimation of individuals' fitness from resulting population

We must calculate values of corresponding target function and their probabilities to estimate fitness of individuals of population. After selection of initial population, we can calculate values of corresponding target function $f(x_j)$, j=1,2,...,P.

Let us denote:

$$fmin = min\{f(x_1), f(x_2), \ldots, f(x_p)\} \quad (5)$$

At every following iteration we compare *fmin* with new minimal value and remember the smallest number. When iterations are finished, it will be the best value.

Now we must check condition of completion of genetic algorithm. If condition is met, then solution of our problem will be the best individual from the resulting population. Otherwise, steps of algorithm, i.e. selection, crossover and mutation are repeated until condition of completion of genetic algorithm is met. In our work condition of completion of genetic algorithm is pre-defined number of iterations.

Algorithm is approved for test functions and it gives good results.

7. Results

The results of algorithm for test functions (with two variables) are described in the following tables.

ISSN 1512-1232

Table 1. The results of calculation for test functions, when selection occurs using inbreeding:

Test function	*Function value*	x_1	x_2	*Computation time*	*Number of iterations*
Eason's function	*0.999876*	*3.14934*	*3.13864*	*75.086*	*1822*
Rsatrigin's function	*-0,0082*	*-0.00598*	*0.002391*	*66.31*	*907*
Six-hump camel black function	*-1.03142*	*0.09688475*	*-0.711739*	*90.007*	*1075*
Three-hump camel black function	*-0.00091212*	*-0.019783*	*-0.005187*	*62.865*	*3053*
Rozenbrok's function	*0.00106851*	*1.011310*	*1.019681*	*69.251*	*121*
Shaffer's function	*-3.2167e-005*	*-0.0029778*	*0.0048542*	*105.791*	*2859*
Bird's function	*-106.004* *-106.682*	*-1.57427* *4.69286*	*-3.05588* *3.17597*	*63.773* *89.744*	*1299* *608*
Peaks function	*-6.54949065*	*0.2411*	*-1.6212*	*119.88*	*1489*

Table 2. The results of calculation for test functions, when selection occurs using outbreeding:

Test function	*Function value*	x_1	x_2	*Computation time*	*Number of iterations*
Eason's function	*0.981604*	*3.23051*	*3.20592*	*64.732*	*159*
Rsatrigin's function	*-0.0122073*	*0.00533752*	*0.00574875*	*16.618*	*27*
Six-hump camel black function	*-1.02989*	*0.0839157*	*-0.340912*	*20.541*	*34*
Three-hump camel black function	*-0.000218384*	*-0.0108719*	*0.00203705*	*393.301*	*2127*
Rozenbrok's function	*-0.000263523*	*1.00698*	*1.01547*	*15.169*	*103*
Shaffer's function	*-0.00124076*	*-0.00694275*	*-0.034523*	*223.4*	*3372*
Bird's function	*-105.971* *-71.9911*	*-1.57634* *4.94802*	*-3.20608* *0.0407124*	*49.235* *41.937*	*341* *237*
Peaks function	*-6.5399*	*0.215019*	*-1.64497*	*94.744*	*1679*

Optimal meanings for these functions, obtained using different algorithms and basically taken from virtual test library[18], are given in the following table:

ISSN 1512-1232

Table 3. Optimal meanings of functions taking from the internet

Test function	***Function value***	x_1	x_2
Eason's function	0	π	π
Rsatrigin's function	0	-0.00598	0.002391
Six-hump camel black function	-1.0316	0.0898	-0.71126
Three-hump camel black function	0	0	0
Rozenbrok's function	0	1	1
Shaffer's function	0	0	0
Bird's function	−106.764537	-1.57427 4.69286	-3.05588 3.17597
Peaks function	*−6.551133332*	*0.228279999*	*−1.625531*

We offer [18] other genetic algorithm for solution of a problem (1) that uses the same scheme of encoding, but different genetic operators. Test problems show that both algorithms consistently give satisfactory results. Let us show diagram to compare these two methods and also compare results of calculation for several functions.

Tables shows that both methods give almost similar results. Values obtained from us (table 1-2) mainly differs from the known values (table 3) by 10^{-3}, only in some cases the difference is $10^{-1.}$

Below given a new function (Figure 1):

$$f(x,y) = -5\,sin\left(\sqrt{x^2+y^2}\right) - 2cos\left(\sqrt{x^2+y^2}\right) + 11ln\left|sin\left(\frac{x}{20}\right) + x - 0.1\right| \qquad (6)$$

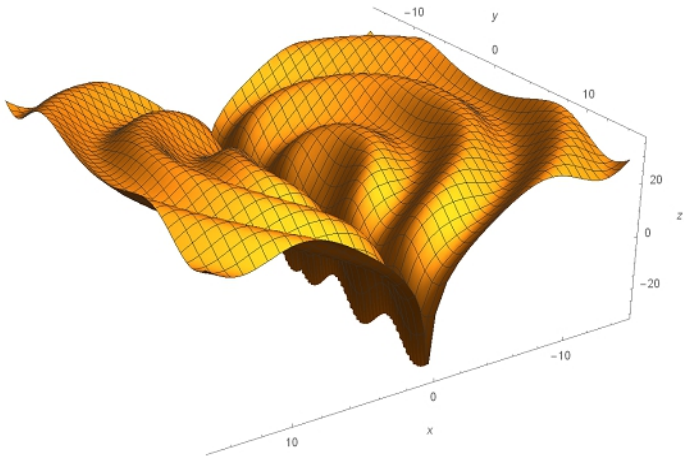

Figure 1. Function

For this function, the minimum points, calculated using our algorithms for the section x∈ [-0.2,0,2], y∈ [-14,14], are:

ISSN 1512-1232

$x = 0.0952381123777199, y = 7.47200188766424, \quad f(x,y) = -434.647859$

$x = 0.09523811237771995, y = -13.753580664706421, \quad f(x,y) = -450.777548$

$x = 0.09523811237771995, y = 13.756176085314774, \quad f(x,y) = -434.647860$

$x = 0.09523811237771997, y = -7.470056314844304, \quad f(x,y) = -434.647839$

8. Conclusion

These algorithms are working reliable and for some tasks for which using classical methods can not give the result, above mentioned algorithms find the solution. The speed of work of algorithms corresponds to speeds, acceptable for genetic algorithms.

References:

[1] Polyak Boris T. Introduction to Optimization. Optimization Software. New York : Translated from the Russian. With a foreword by Dimitri P. Bertsekas. Translations Series in Mathematics and Engineering. Optimization Software, Inc., Publications Division, 1987.

[2] Bhaya A, Pazos F and Kaszkurewicz E. The Controlled Conjugate Gradient Type Trajectory-Following Neural net for Minimization of Nonconvex Functions. Spain, Barcelona : s.n., 2010. Proceedings of the IEEE 2010 International Joint Conference on Neural Networks (IJCNN). 1-8.

[3] Brown G.W. Iterative Solution of Games by Fictitious Play. Activity Analysis of Production and Allocation. Issue: 1, s.l. : New York: Wiley, 1951,13: 374–376.

[4] Brown G.W, Neumann J.von . Solutions of Games by Differential Equations, Contributions to the Theory of Games. s.l. : TPrinston University Press, 1950, Annals of Mathematics Studies, 24:73-79.

[5] Cabot A, Engler H, Gadat S. On the Long time Behavior of Second Order Differential Equations with Asymptotically Small Dissipation. s.l. : Trans. Amer. Math. Soc., 361(11), 2009. 5983–6017.

[6] Gelashvili K, Alkhazishvili L, Khutsishvili I, Ananiashvili N. On the Modification of Heavy Ball Method. ISSN 1512-0007, Tbilisi : A. Razmadze Mathematical Institute, 2013, 161.

[7] Goudou X, Munier J. The Gradient and Heavy Ball with Friction Dynamical Systems: The Quasiconvex Case. s.l. : Math. Program., Ser. B, 2009,173-19. 116.

[8] Hajba Tamas. Optimizing Second-order Differential Equation Systems. 2011, Electronic Journal of Differential Equations, 2011, 44: 1–16.

[9] Paul-Emile Maingé. Asymptotic Convergence of an Inertial Proximal Method for Unconstrained Quasiconvex Minimization. Issue 4, December 2009, Journal of Global Optimization, 45.

[10] Holland J.H. Adaptation in Natural and Artificial Systems, Ann Arbor: University of Michigan Press, 1975.

[11] Koza John R. Genetic Programming: on the Programming of Computers by means of Natural Selection, A Bradford book, The MIT Press, London, 1992.

[12] Mitchell M. An Introduction to Genetic Algorithms. Cambridge, MA: The MIT Press, 1996.

ISSN 1512-1232

[13] Pan ZJ, Kang LS An Adaptive Evolutionary Algorithm for Numerical Optimization.(1997) In: Yao X, Kim JH, Furuhashi T ed. Simulated Evolution and Learning. Lecture notes in artificial intelligence. Berlin, Germany,: Springer-Verlag. 27-34.

[14] Randy L. Haupt, Sue Ellen Haupt, "Practical Genetic Algorithms"-2nd ed. Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 2004.

[15] Rutkowska D., PiliĔski M., Rutkowsli L., Sieci Neuronowe, Algorytmy Genetyczne i systemy rozmyte, Wydawnictwo Naukowe PWN, Warszawa-àódĨ, 1999.

[16] Xing L.N , Chen Y.W, and Cai H.P. "An Intelligent Genetic Algorithm Designed for Global Optimization of Multi-minima Functions,"Applied Mathematics and Computation,178:355–371,2006.

[17] Virtual Library of Simulation Experiments: Test Functions and datasets. Optimization Test Problems.- URL: https://www.sfu.ca/~ssurjano/optimization.html

[18] Ananiashvili N. Solution of Problem of Unconditional Optimization by Means of Genetic Algorithm. Issue 4, 2015, International Journal of Engineering and Innovative Technology (IJEIT), 5: 2277-3754.

The article received: 2020-10-14