**Artem Volokyta, Victor Steshyn,
Liudmyla Mishchenko, Viktoria Masimyk,
Kostyantyn Tykhonov.**

USING SURPLUS CODE TO THE GENETIC ALGORITHM

The article considers the method of finding the minimum of a multimodal function using a genetic algorithm with the excess code 0/1 / -1 in gene mutations.

Keywords: genetic algorithm, redundant code~~,~~ multimodal function.

Acute problem. The issue of optimization of the genetic algorithm is very important in solving mathematical problems that require significant calculations [1]. Particular attention is paid to mutations and gene crosses. And this requires an optimal approach to the development of an already known algorithm. The article proposes the implementation of a genetic algorithm using the excess code 0/1 / -1 [2,3,4] during gene mutation, which will significantly increase the efficiency of this algorithm.

Target setting. An important part of the algorithm implementation is its efficiency and performance. The genetic algorithm is used to solve many problems, such as finding the extremum of a multimodal function, optimizing dynamic systems, the salesman problem, and others. In such tasks, it is important to optimize the solution, and, above all, efficiency. Therefore, a method is proposed to increase the efficiency of the genetic algorithm through the use of redundant code.

Actual scientific researches and issues analysis. Currently, a significant number of problems in the different areas such as IT, economics, mathematics, management and others that can be solved by a genetic algorithm have been studied and described. However, most publications focus on the application of natural genetic evolution to various professional fields. There are works that describe the application of the genetic algorithm to specific mathematical problems, such as the Diophantine equation, function optimization, selection of optimal values of weights for the neural network, which minimize the error value.

Uninvestigated parts of general matters defining. Today, the issue of optimization of mathematical calculations in the genetic algorithm is acute. Therefore, the possibility of using redundant code 0/1 / -1 to speed up calculations remains unexplored.

The research objective. The aim of the study is to apply a genetic algorithm to the problem of finding the minimum of the multimodal function of many variables using the redundant code 0/1 / -1.

The statement of basic materials. The proposed article considers the mechanism of finding the minimum of a function using a genetic algorithm. There are five main steps to accomplishing this.

At the first stage, the population is initialized. That is, the number, bit rate and number of genes in each individual are set. The next step is to calculate the value of the quality criterion, ie fitness function, for each individual population. Thus, in the third stage, their suitability for crossing is determined. The value of fitness is directly proportional to the probability of crossing. The fourth stage is to perform crossbreeding and mutation for each of the selected individuals. There is an exchange of chromosomes (bits) between the corresponding genes. Thus, a new generation is formed, for which the execution of the algorithm from the second point is repeated. At this stage, the algorithm can complete its work if one of the following conditions is met:

- the expected optimal value is reached, possibly with the set accuracy;
- there is no improvement in the value already achieved for some time;
- the specified execution time or number of iterations is over.

The main stages of the genetic algorithm

1. Construction of chromosomes.

In a given interval, random values are selected that will be used to calculate the fitness function in the first generation.

2. Execution of fitness function

The generated chromosomes are substituted into a given equation and the value of the function is calculated. The results of these calculations are called fitness.

3. Selection

In the proposed solution, the selection is performed by the method of roulette. Based on the formula $V(c_i) = P(c_i) \cdot 100\%$, where $V(c_i) = P(c_i) \cdot 100\%$, and where $F(c_i)$- the value of the fitness function, $P(c_i)$- the probability of chromosome selectionc_i, N - population size.

As a result of selection, the parent population is formed, the number of which corresponds to the current value.

4. Crossing

The probability of crossing is usually very high, ie it occurs almost always. The process of crossing begins with selected as a result of selection and determined by the crossover for further transformations of chromosomes that make up the parent population. From it, the chromosomes combine to form a new population of offspring.

The process is implemented randomly, according to the probability P (c). Next, a position for gene crossing is randomly selected in each parent pair. And according to this point, a pair of offspring is obtained: the first chromosome consists from the beginning to the point of crossing of the genes of the first parent, and then the genes of the second; the second chromosome consists from the beginning to the point of crossing of the genes of the second parent, then - the genes of the first.

5. Mutation

Mutation of chromosomes occurs in populations of offspring formed by crossing. Basically, the mutation process occurs according to the Gray code, ie one of the genes on the chromosome will be different by a bit.

In the proposed algorithm, the mutation occurs similarly, except for the number T, which is present in the process of gene change. This manifests itself as an additional variant of chromosome gene replacement. That is, there are two possibilities for gene mutation at 0/1 or T, where T is the value of the excess code.

Excess code 0/1 / -1. To optimize the algorithm and speed up the calculations, it is proposed to use redundant code. That is, the usual binary code is supplemented by an additional value of -1, which is denoted by the letter T. All digits in this code combination are equivalent, including bit weights and other properties. However, such a code representation also includes negative numbers. The main feature of redundant code is that the same number can be represented in several ways. For example, the number 3 in the three-bit redundancy code can be represented as 011 or 10T, or 1T1.

Application of the algorithm without the use of redundant code. The task of finding the minimum in a given function was chosen for experimental research. All stages of the genetic algorithm are performed according to the steps described above, except for the mutation process. The excess code was not used during the mutation, so the finite state machine is a hypercube.

According to the implementation of the genetic algorithm, the following initial data are obtained, described in table 1.

Table 1

The initial data of the genetic algorithm for 36 generations

Generation number	The best fitness	Generation number	The best fitness	Generation number	The best fitness	Generation number	The best fitness
1	212	10	761	19	452	28	113
2	884	11	648	20	450	29	74
3	882	12	761	21	452	30	41
4	884	13	761	22	290	31	20
5	882	14	648	23	225	32	5
6	882	15	545	24	221	33	18
7	882	16	650	25	164	34	5
8	882	17	648	26	164	35	5
9	761	18	545	27	113	36	0

Total number of generations before we find minimum of the function is 36. We have same values of the best fitness in some first generations, which indicates the imperfection of the chosen approach

Finding the best fitness of each generation for the hypercube function is shown in Fig. 1.

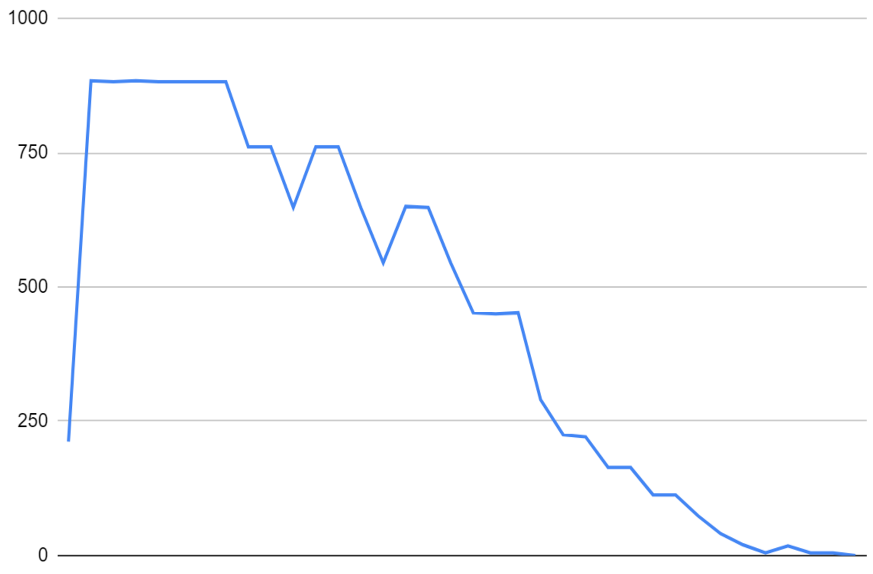

Fig. 1. Graph of the dependence of the value of the best fitness (y-axis) on each generation (x-axis) for the hypercube

Chart on picture 1 is better displaying calculation problem of the algorithm in first stages where we have same results for the best fitness and we need some time to change this situation and continue finding fitness that will be different from previous values.

Implementation of redundant code. Excess code affects all stages of the genetic algorithm. Its is changing the topology, and it also affects the mutation process that uses the Gray code. In the proposed implementation, the addition of a negative state showed positive results in the modeling of the genetic algorithm. State "-1" speeds up the process of finding the optimal solution by forming additional connections.

Genetic algorithm using redundant code. As mentioned earlier, experimental studies were conducted on the problem of finding the minimum in a given function. The implementation of the proposed method differs in the process of mutation. Excess code is used during the mutation, in which case the finite state machine is the de Bruijn topology.

According to the work of the proposed implementation of the genetic algorithm, the following initial data are obtained, described in table 2.

Table 2

The initial data of the proposed implementation of the genetic algorithm for 13 generations

Generation number	The best fitness	Generation number	The best fitness
1	36	8	2
2	338	9	2
3	338	10	2
4	2	11	2
5	90	12	2
6	2	13	0
7	2		

In compression we easily could see that using of redundant code solve same task in less number of steps that could give us best performance with system that will support such calculations natively**.**

Finding the best fitness of each generation for the de Bruijn function is shown in Fig. 2.

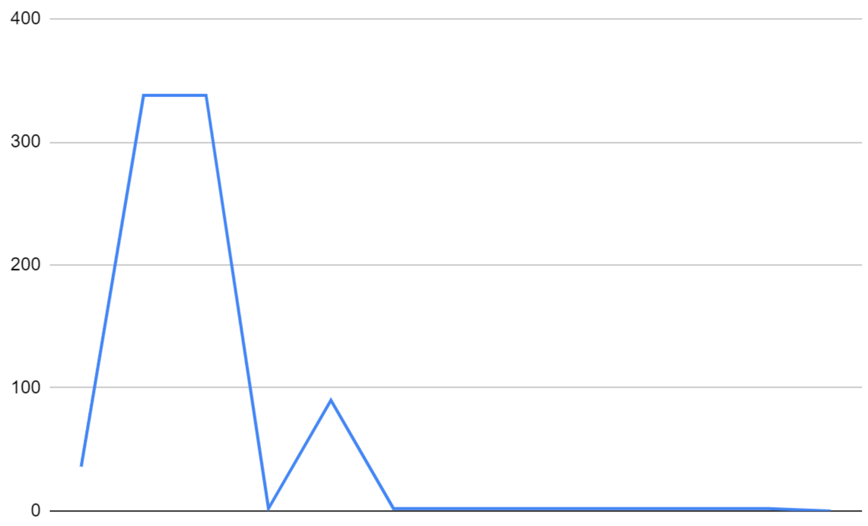

Fig. 2. Graph of the dependence of the value of the best fitness (y-axis) on each generation (x-axis) for de Bruijn

Chart on picture 2 displaying much better results in less steps, that is obvious in compression with previous approach. We could clearly see the benefits of this approach on this chart.

Conclusions. The method of realization of genetic algorithm with use of redundant code is offered in the work. The algorithm is used to solve the problem of finding the minimum of a multimodal function of many variables using the code 0/1 / -1.

After analyzing the results of the proposed solution, we can conclude that using this method was able to increase the efficiency of GA. After all, the use of redundant code has helped reduce the number of generations by almost three times (from 36 to 13).

The implementation of the genetic algorithm with the redundant code came much faster to the correct answer. We got much better fitness in the 6th generation, and the algorithm without redundant code took 32 generations to get closer to similar results.

All the data obtained indicate that the use of redundant code can significantly speed up the calculation using a genetic algorithm.

However, the software and hardware solutions used in the work do not allow the calculation of values of less than zero at the same speed as the values of greater than zero. The proposed algorithm is only a simulated ideal situation. To achieve such results, it is necessary to use specialized software solutions that allow you to perform operations with negative values at the same speed as with values greater than zero.

References

1. Генетичні алгоритми. Ключові поняття і методи реалізації. Update date: 17.05.2020. URL: http://www.znannya.org/?view=ga_general (request date : 10.02.2020).

2. O. Honcharenko, H. Loutskii, P. Rehida, A. Volokyta Using excess code to design fault-tolerant topologies. //Technical sciences and technologies -2019-V.1(15), pp. 134-144.

3. H. Loutskii, A. Volokyta, P. Rehida, O. Honcharenko, B. Ivanishchev and A. Kaplunov, "Increasing the fault tolerance of distributed systems for the Hyper de Bruijn topology with excess code,"// IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, -2019- pp. 1-6, doi: 10.1109/ATIT49449. 2019.9030487.

4. Olexandr G., Rehida P., Volokyta A., Loutskii H., Thinh V.D. Routing Method Based on the Excess Code for Fault Tolerant Clusters with InfiniBand./ Hu Z., Petoukhov S., Dychka I., He M.// Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, Springer, Cham-2019-V.938.

AUTHORS

Artem Volokyta (supervisor) – associate professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Victor Steshyn – assistant professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Liudmyla Mishchenko – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Viktoriia Maksymuk – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Tykhonov Kostyantyn – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”