Acta et Commentationes, Exact and Natural Sciences ISSN: 2537-6284
Volume 2(16), 2023, Pages 128–142 E-ISSN: 2587-3644
CZU: 004.8:519.1 DOI: 10.36120/2587-3644.v16i2.128-142

Dedicated to Professor Alexandru Şubă on the occasion of his 70th birthday

Application of genetic algorithm to solving the optimization problem of locations graph vertices in the line

Liubomir Chiriac, Natalia Lupashco, and Maria Pavel

Abstract. This article examines genetic algorithms that are built on the "survival of the fittest" principle enunciated by Charles Darwin. By applying genetic algorithms to solving optimization problems, it is not always possible to guarantee the determination of the global optimum in polynomial time. This fact does not occur because only brute force search methods allow us to find the global optimum. Instead the genetic algorithm allows selecting good decisions, in a reasonable time, compared to other well-known deterministic or heuristic search engine optimization algorithms. The authors of this article develop an algorithm of solving the optimization problem of locations graph vertices in the line.

2010 Mathematics Subject Classification: 05C85, 05C90.
Keywords: genetic algorithm, optimization problem, location problem, graphs algorithms.

Aplicarea algoritmului genetic la rezolvarea problemei de optimizare privind amplasarea vârfurilor grafului în linie

Rezumat. Acest articol examinează algoritmii genetici care se bazează pe principiul "supravieţuirii celui mai adaptat" enunţat de Charles Darwin. Prin aplicarea algoritmilor genetici la rezolvarea problemelor de optimizare, nu este întotdeauna posibil să se garanteze determinarea optimului global în timp polinomial. Acest fapt nu se întâmplă deoarece numai metodele de căutare prin forţă brută ne permit să găsim optimul global. În schimb, algoritmul genetic permite selectarea unor decizii bune, într-un timp rezonabil, în comparaţie cu alţi algoritmi de optimizare a motoarelor de căutare deterministe sau euristice bine cunoscute. Autorii acestui articol dezvoltă un algoritm de rezolvare a problemei de optimizare a amplasării vârfurilor grafurilor în linie.

Cuvinte-cheie: algoritm genetic, problema de optimizare, problema de localizare, algoritmi de grafuri.

1. History of the development of the evolutionary calculus

The approach regarding the application of evolutionary principles (evolutionary computation) in the automated solving of problems dates back long before the emergence and development of modern computers.

As early as 1948, Alan Turing introduced a new approach applied to problem-solving called evolutionary or genetic approach. Subsequently, in the 1960s, Dr. Lawrence Jerome Fogel (March 2, 1928 - February 18, 2007), a pioneer in evolutionary computation, along with Wlash (later David B. Fogel, born on February 2, 1964), introduced and developed the concept of evolutionary programming. During the same period, Holland focused on genetic algorithms. Hans-Paul Schwefel (born on December 4, 1940), a German computer scientist and emeritus professor at the University of Dortmund, and Ingo Rechenberg (November 20, 1934 - September 25, 2021), a German researcher and professor in the field of bionics, launched and developed evolutionary strategies as alternative methods for automated problem-solving. Later, in the 1990s, J. R. Koza developed genetic programming, a new technique for searching solutions.

Therefore, ***evolutionary computation*** is a field of modern computer science with a strong emphasis on mathematics, inspired by the natural evolutionary process. The fundamental concept underlying evolutionary computation is the interconnection between natural evolution and the trial-and-error problem-solving technique [1].

In the context of the above, evolutionary computation is currently an important research field in computer science. As known, this field derives from the natural evolutionary process. The algorithms that emerge and develop in this field are called evolutionary algorithms, and they include significant and promising subdomains such as:

- Evolutionary programming;
- Evolutionary strategies;
- Genetic programming;
- Genetic algorithms.

2. Genetic Algorithms

The fact that mathematics and computer science are widely applied in various sciences, including biology, is a well-known and appreciated phenomenon. However, the reciprocity of this relationship does not always occur. For instance, in modern science, there are not many instances where mechanisms, concepts and basic notions from biology that are widely and efficiently used in mathematics and computer science.

In this context, the genetic algorithm is an eloquent and convincing example. Genetic algorithms represent adaptive heuristic search techniques that are implemented based on the principles of natural selection and genetics.

The mechanisms of the Genetic Algorithm are similar to natural evolution and rely on the principle stated by Charles Darwin, ”survival of the fittest”, meaning that the most well-adapted individuals, not necessarily the strongest or most intelligent, survive.

Thus, the Genetic Algorithm represents a computer-mathematical model that mimics the evolutionary biological model to solve search and optimization problems. The Genetic Algorithm is determined by a set of elements representing a population, consisting of chromosomes (binary strings), and a set of genetic operators (selection, crossover, and mutation) that influence the population's structure.

Genetic algorithms are commonly used for problems where finding the optimal solution is not simple or at least inefficient due to the characteristics of probabilistic search. Genetic algorithms encode a possible solution to a specific problem in a unique data structure called a ”chromosome” and apply genetic operators to these structures to maintain critical information. The implementation process of genetic algorithms starts with an ”initial set of possible solutions” to the examined problem (usually chosen randomly) referred to in the literature as ”population” [2], [3].

Each individual in the ”examined population” represents a potential solution to the problem and is called a ”chromosome”, which is a string of symbols, typically expressed as a string of bits. The examined chromosomes evolve over successive iterations, symbolically called generations. In each generation, these chromosomes are evaluated, using fitness measures.

To generate the next population (generations), the most ”efficient” or ”best” chromosomes from the current generation are selected. New chromosomes are formed, using one of the three (or even all three) central genetic operators: selection, crossover and mutation.

Selection ensures the process from the following perspective: certain chromosomes from the examined (current) generation are copied, depending on their fitness value, in accordance with the problem requirements into the new generation. This indicates that chromosomes with high significance have a high probability of contributing to the formation of the new generation.

The genetic operator crossover represents the process by which, based on two individuals (chromosomes) from the current population, two individuals (chromosomes), called descendants, are formed for the next population. Mutation is the genetic operator that represents the process through which a chromosome from the current population is modified and saved in the new population.

Genetic algorithms have been successfully applied to a variety of NP-complete problems that require global optimization of the solution and, in this regard, there is no iterative method for resolution [4], [5].

In genetic algorithms, the individuals in a population are represented by chromosomes with encoded sets, task parameters, for example, solutions otherwise called points in the search space or search points. In some works, individuals are called organisms. In this sense, we will clarify the meaning of the following biological concepts from the perspective of computer science.

Darwin's concept of evolution is adapted to the functioning of the genetic algorithm to find solutions to a problem expressed through the fitness function (objective function or adaptation function).

The fitness function represents a measure of the adaptability of a given individual within each generation. This characteristic allows the evaluation of the adaptation degree of individuals in the population and selects the most adapted ones, i.e., those with the highest values of fitness function, following the evolution principle of the survival of the fittest [6], [7].

Thus, selection represents the choice of individuals with the best aptitude for reproduction (sorting by the value of the objective function). The better an individual's fitness is, the greater the chances of crossing and passing on its genes to the next generation are. The crossover operator is analogous to biological reproduction and crossover and usually it is applied to individuals in the intermediate population. Two individuals are selected from the intermediate population, and certain portions of their two chromosomes are exchanged.

In simple terms, mutation can be defined as a small random modification of the chromosome to obtain a new solution. Mutation is used to maintain and introduce diversity into the genetic population. Mutation is the part of the Genetic Algorithm related to "exploring" the search space [4].

3. Introductory Concepts

Genetic Algorithms are algorithms of evolutionary computation, inspired by Darwin's Theory of Evolution. In 1960, Ingo Rechenberg (November 20, 1934 - September 25, 2021) introduced the idea of evolutionary computation in a work titled "Evolution strategies". Rechenberg, a German researcher and professor in the field of bionics, was a pioneer in the fields of evolutionary computation and artificial evolution. In the 1960s and 1970s, he invented several optimization methods known as evolution strategies (in German, Evolutionsstrategie). His research team successfully applied these algorithms to optimization problems, including the aerodynamic design of wings. These were the first serious technical applications of artificial evolution, an important component of bionics and artificial intelligence [8].

In 1975, John Henry Holland (February 2, 1929 - August 9, 2015), an American scientist and professor of psychology and electrical engineering and computer science at the University of Michigan, introduced and analyzed a mathematical model that, through adaptive procedures, relied on a mechanism of natural selection and genetic evolution called genetic algorithm. He was a pioneer in what became known as genetic algorithms [1], [9].

Genetic algorithms are used often in cases where the optimal solution involves searches among all combinations, permutations or probabilistic arrangements, a very complex and sometimes inefficient process. They implement specific data structures called ”chromosomes” to encode, through genetic operators, the possible solution to a particular problem while retaining important information.

Usually, to solve a problem using a genetic algorithm, the so-called ”population” is identified, constructed randomly based on the ”initial set of possible solutions”. Each individual or ”chromosome” (a string of characters expressed as a sequence of bits) in the examined ”population” represents a possible solution to the problem. Through consecutive iterations, the evolution of the ”chromosomes” occurs at the ”generation” level, each of which is validated by an evaluation function called fitness. Using one of the three main genetic operators (*selection*, *crossover*, and *mutation*) new ”chromosomes” identified from the current generation as the most ”efficient” are generated for the future population. Thus, just like in biology, the most ”powerful chromosomes”, with a higher probability, *are selected* from the given generation to transmit their characteristics (values of the ”evaluation function”, according to the requirements of the problem), to the next generation, ensuring the perpetuation of the entire process. Using the *crossover* genetic operator combines information from two individuals (”parents”) from the current population to generate one or more descendants. *Mutation*, on the other hand, allows the random modification of a gene or a small section of the ”chromosome” to ensure diversity in the future population.

The success of genetic algorithms is ensured by their implementation in solving a series of NP-complete problems, whose solutions cannot be identified through iterative methods, but rather by obtaining the optimal solution globally.

In genetic algorithms, individuals in a population are represented by chromosomes with encoded sets of task parameters, e.g., solutions, otherwise called points in the search space (search points). In some works, individuals are called organisms.

In this sense, the following biological concepts, borrowed by computer scientists from the perspective of genetic algorithms, will be clarified:

Chromosomes: Ordered sequences of genes.

Gene: Also called a property, sign, or detector, is the atomic element of the genotype, especially of chromosomes.

Genotype: The set of chromosomes of a given individual. Consequently, individuals in a population can be genotypes or unique chromosomes (in a common case when the genotype consists of a single chromosome).

Phenotype: A set of values that correspond to a specific genotype or set of task parameters (solution, search space point).

Allele: The value of a specific gene, also defined as the property value or property variant.

Locus: The position indicating the location of a specific gene in a chromosome (chain). The set of gene positions represents loci.

Genome: The totality of the genetic material of an organism or species, determining the development, functioning and transmission of hereditary traits from one generation to another.

Individual: A unique entity that has a specific set of chromosomes inherited from parents. In genetic algorithms, an individual represents a possible solution or a combination of parameters that can be optimized over time by selection and recombination processes and is evaluated within a specific problem.

Population: A group of individuals sharing a common set of genetic characteristics that occupy a certain type of environment. Genetic variation within populations is important for adaptation to environmental changes.

Mapping: An essential evaluation function that assigns a numerical value to each individual in the given population, reflecting the quality or appropriateness of that solution within the optimization problem. The mapping process is also called morphogenesis.

A crucial concept in genetic algorithms is the function that measures the degree of adaptability known as the fitness function.

The fitness function is a measure of the adaptability of a given individual within each generation. This characteristic allows the evaluation of the adaptation degree of individuals in the population and the most adapted individuals, those with the highest values of the fitness function, are selected in accordance with the evolutionary principle of the survival of the fittest.

The fitness function got its name directly from genetics. It has a strong impact on the functioning of genetic algorithms and must have precision and correct definition. In optimization problems, the fitness function is usually optimized (maximized or minimized) and is called the objective function.

At each iteration of the genetic algorithm, the fitness (adaptation degree) of each individual in a particular population is estimated using the fitness function. Based on this, the next generation (population of individuals) is generated, constituting the possible set of solutions to the examined problem [1], [4].

4. Implementation of the Genetic Algorithm

In the specialized literature, the Genetic Algorithm involves a series of steps (originally proposed by John Henry Holland) intended to conclude with an optimal solution to the examined problem. Below, we will outline the steps regarding the implementation of the Genetic Algorithm.

***Step 1*:** Creating/generating the initial population.

***Step 2*:** Evaluating the fitness function value of each individual (a mechanism used to measure and evaluate the state of a chromosome).

***Step 3*:** Selection - considering the characteristics of each individual, during this stage, some individuals may reproduce more frequently than others.

***Step 4*:** Crossover.

***Step 5*:** Mutation.

***Step 6*:** Replacing the old population of chromosomes with the new population of chromosomes.

Step 7*:** Finding the best solutions (but if the optimization criteria are not met, then the method requires returning to ***Step 2 and ultimately selecting the best individual as the final solution).

Genetic algorithms generate a new population composed of individuals with better and more adapted characteristics to the environment than those of the previous population. The logical scheme related to the implementation of the Genetic Algorithm is presented in Figure 1.

The process begins by initializing a random genetic pool through the creation of a set of chromosomes according to a predefined template, where the values of all genes are randomly selected for each chromosome. These initial chromosomes correspond to the individuals in the initial population. Typically, the number of individuals (and implicitly chromosomes) in the population remains constant at different generations, although this is not always the case. The Genetic Algorithm starts with a set of permissible solutions called the ”population” (created arbitrarily, as mentioned earlier), each of which represents a potential solution to the problem, called a ”chromosome” [10].

Once the population is established, it evolves towards better solutions through various genetic processes (*selection*, *crossover*, *mutation*) that lead to a better fitness function value, used to evaluate the state of each chromosome.

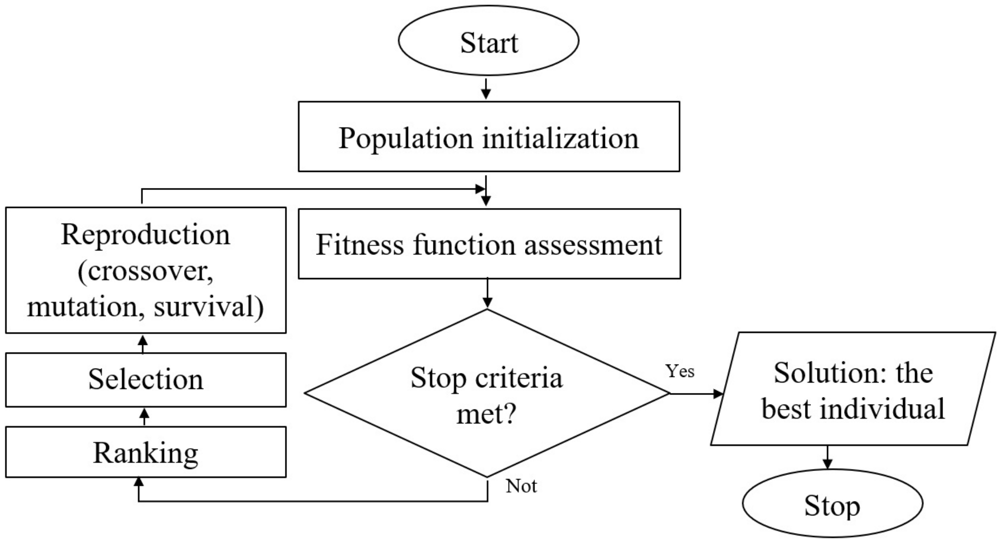

Start
Population initialization
Fitness function assessment
Reproduction (crossover, mutation, survival)
Selection
Ranking
Stop criteria met?
Yes
Not
Solution: the best individual
Stop

Figure 1. Logical Scheme of the Genetic Algorithm

5. Genetic algorithm for solving the optimization problem of locations graph vertices in the line

The problem of optimal placement of vertices of an undirected graph on a linear grid is a classic problem that requires knowledge in mathematics, computer science, and, evidently, genetic algorithms. In this section, the authors propose an algorithm that solves this problem under certain conditions.

Problem Statement: Given a graph G, where $n = |G|$ is the number of vertices of graph G. The goal is to find the best placement of the vertices of graph G on a linear grid after performing a genetic algorithm for k iterations (k generations), where $k < n$. It is assumed that the distances between the vertices of the graph are equal.

Solution: The total length of the edges of graph G is calculated according to the following formula:

$$L(G) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{i,j} a_{i,j} \tag{1}$$

where $n = |G|$ is the number of vertices, $d_{i,j}$ represents the distance between vertices of graph G, v_i and v_j, on the examined line. The distance in this case is measured in the number of edges of the graph, $a_{i,j}$ is the corresponding element of the adjacency matrix (0 or 1). In other words, the task is to find $\min L(G)$ after changing k generations (after performing k iterations).

The main goal of placement algorithms is to minimize the total length of the edges of a graph or hypergraph. Let us formulate this placement problem as an optimization problem. Formula (1) is selected as the objective function, which needs to be minimized.

Notations. The index i for chromosomes $C_1^i, C_2^i, \dots, C_k^i$ represents the number of the generation to which chromosomes numbered $1, 2, \dots, k$ belong.

The output will be the positions of each vertex. The genetic algorithm applied consists of the following steps:

***Step 1*:** Create the initial population consisting of k chromosomes, each composed of n elements (vertices of the graph):

$$C_1^i, C_2^i, \dots, C_k^i, \text{ where } i = 0, 1, 2, \dots, n-1.$$

***Step 2*:** Place the vertices of the graph on the linear grid according to the values of the examined chromosomes: $C_1^i, C_2^i, \dots, C_k^i$.

Step 3*:** Calculate the length of each chromosome $C_1^i, C_2^i, \dots, C_k^i$. This involves calculating the number of horizontal segments that connect the vertices of the graph, according to the placement made in ***Step 2, for each individual chromosome. Obtain lengths:

$$L_1(C_1^i), L_2(C_2^i), \dots, L_k(C_k^i).$$

***Step 4*:** Calculate the total sum of the edges of the graph according to the placement of the vertices on the grid determined by that particular population of chromosomes:

$$S(i) = L_1(C_1^i) + L_2(C_2^i) + \dots + L_k(C_k^i), \ i = 0, 1, 2, \dots, n-1 \tag{2}$$

where i is the number of the iteration or generation of the population.

***Step 5*:** Choose the ”fittest” chromosome (with the smallest length) from the population $C_1^i, C_2^i, \dots, C_k^i$. Let this chromosome be C_r^i, where $1 \le r \le k$.

Step 6*:** Apply the ***inverse mutation genetic operator on chromosome C_r^i after the first element. In other words, in the first iteration, the first element of the chromosome remains in place and the other elements are written in reverse order, starting with the last one, which will already be in the second position in the chromosome representation. In the second iteration, the first and second elements remain intact and the other elements are written in reverse order, starting with the last one, which will already be in the third position. And so on for each iteration. Denote the newly obtained chromosome after performing the inverse mutation by C_{rm}^i.

***Step 7*:** Calculate the length of the newly obtained chromosome C_{rm}^i.

Step 8: Identify and eliminate the weakest chromosome (with the maximum length) from the first generation of chromosomes, which is subsequently replaced by the chromosome C^i_{rm}.

Step 9: Build the next generation of chromosomes (which already includes the new chromosome C_{rm} and excludes the weakest chromosome) and then proceed to ***Step 2***.

Step 10: The process of the genetic algorithm stops after performing k iterations or, in other words, after constructing k generations of chromosomes. At each iteration, calculate $S(0), S(1), \ldots, S(k)$, where for each sum, the condition

$$S(m) \geq S(m+1),\ m = 0, 1, \ldots, k \tag{3}$$

is satisfied. The best placement of the vertices of the graph is obtained after completing the last iteration, in which the last genetically modified chromosome C^i_{rm} represents the solution $\min L(G)$.

6. Example of genetic algorithm application for solving the optimization problem of locations graph vertices in the line

Problem: Find the best placement of the vertices of graph G in Figure 2 on a line after performing three iterations of the genetic algorithm. The graph G and the initial population consisting of 3 chromosomes are given below.

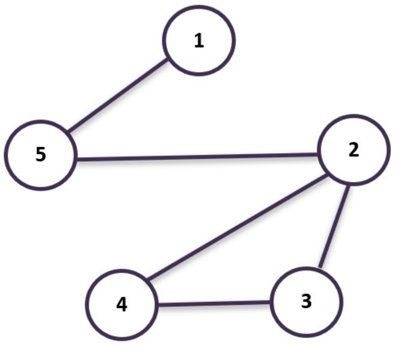


Figure 2. The examined graph, $n = 5$, and $m = 5$.

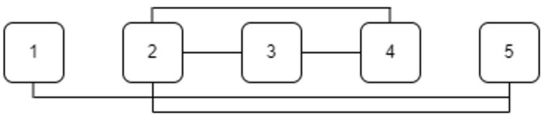
Solution: Place the vertices of the graph on the line according to the initial population of chromosomes to calculate the length of each chromosome.

Therefore, we get the graphical representation of the chromosomes.

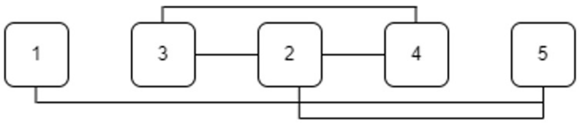
Table 1. Initial Population of Chromosomes (Generation 0)

Chromosome C_1^0	1	2	3	4	5
Chromosome C_2^0	1	3	2	4	5
Chromosome C_3^0	5	2	3	4	1

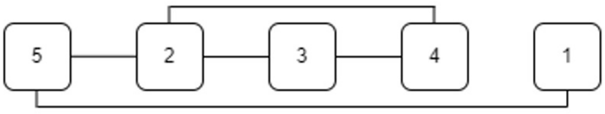
Chromosome C_1^0



Chromosome C_2^0



Chromosome C_3^0



We calculate the number of horizontal segments between the vertices of the graph (between chromosome elements). We obtain:

$$L_1(C_1^0) = 1 + 4 + 4 + 2 = 11;$$
$$L_2(C_2^0) = 1 + 3 + 4 + 2 = 10;$$
$$L_3(C_3^0) = 2 + 3 + 3 + 1 = 9.$$

Calculate the total sum of the edges of the graph according to the placement of the vertices on the grid determined by that particular population of chromosomes:

$$S(0) = L_1(C_1^0) + L_2(C_2^0) + L_3(C_3^0) = 11 + 10 + 9 = 30.$$

Among the examined chromosomes, chromosome C_3^0 has the minimum length, which is 9, so it is the fittest. Consider chromosome C_3^0 selected.

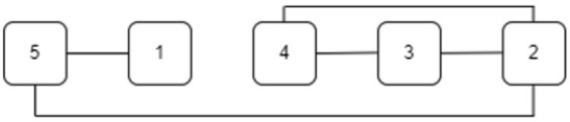
Chromosome C_3^0	5	2	3	4	1

Apply the inverse mutation genetic operator on chromosome C_3^0 to obtain a new chromosome, which we will denote as C_1^1.

Chromosome C_1^1	5	1	4	3	2

In other words, in the first iteration, the first element of the chromosome (5) remains in place, and the other elements are written in reverse order, starting with the last one, which will already be in the second position in the chromosome representation.

Chromosome C_1^1



Calculate the length of chromosome C_1^1:

$$L_1(C_1^1) = 2 + 1 + 3 + 3 = 9.$$

Thus, from the initial generation of the population, we replace the less fit chromosome C_1^0 with the more fit chromosome C_1^1 with a length of 9. Chromosome C_2^0 will be denoted as C_2^1, and chromosome C_3^0 will be denoted as C_3^1. Thus, we obtain Generation 1 of the population of chromosomes.

Table 2. Population of Chromosomes (Generation 1)

Chromosome C_1^1	5	1	4	3	2
Chromosome C_2^1	1	3	2	4	5
Chromosome C_3^1	5	2	3	4	1

The length of each chromosome is:

$$L_1(C_1^1) = 2 + 1 + 3 + 3 = 9;$$

$$L_2(C_2^1) = 1 + 3 + 4 + 2 = 10;$$

$$L_3(C_3^1) = 2 + 3 + 3 + 1 = 9.$$

The total sum of the edges of the graph according to the placement of the vertices on the grid determined by that particular population of chromosomes in the first iteration is:

$$S(1) = L_1(C_1^1) + L_2(C_2^1) + L_3(C_3^1) = 9 + 10 + 9 = 28.$$

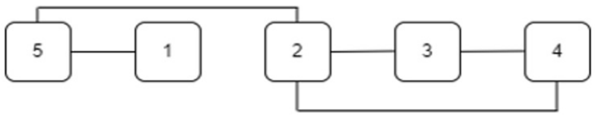
Among the two chromosomes with a length of 9, we select chromosome C_1^1 as the fittest from Generation 1.

Chromosome C_1^1	5	1	4	3	2

Apply the inverse mutation genetic operator on chromosome C_1^1 to obtain a new chromosome C_2^2. In other words, in iteration 2, the first and second elements of the chromosome (5, 1) remain intact, and the other elements are written in reverse order, starting with the last one, which will already be in the third position in the chromosome representation. Thus, we obtain:

Chromosome C_2^2	5	1	2	3	4

Chromosome C_2^2



Calculate the length of chromosome C_2^2. We have

$$L_2(C_2^2) = 2 + 1 + 2 + 2 = 7.$$

Further, in Generation 1 of the population, we have two chromosomes with a maximum length of 9, considered the least fit: C_3^1 and C_2^1. We replace less fit chromosome C_2^1 with fitter chromosome C_2^2 with a length of 7. Thus, we obtain Generation 2 of the population of chromosomes.

Table 3. Population of Chromosomes (Generation 2)

Chromosome C_1^2	5	1	4	3	2
Chromosome C_2^2	5	1	2	3	4
Chromosome C_3^2	5	2	3	4	1

The length of each chromosome is:

$$L_1(C_1^2) = 2 + 1 + 3 + 3 = 9;$$

$$L_2(C_2^2) = 2 + 1 + 2 + 2 = 7;$$

$$L_3(C_3^2) = 2 + 3 + 3 + 1 = 9.$$

The total sum of the edges of the graph according to the placement of the vertices on the grid determined by that particular population of chromosomes in the second iteration is:

$$S(2) = L_1(C_1^2) + L_2(C_2^2) + L_3(C_3^2) = 9 + 7 + 9 = 25.$$

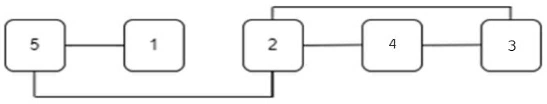
Choose the best chromosome from Generation 2. Clearly, we need to choose the chromosome of minimum length, which is C_2^2:

Chromosome C_2^2	5	1	2	3	4

Apply the inverse mutation genetic operator on chromosome C_2^2 to obtain a new chromosome C_3^3. In other words, in iteration 3, the first, second and third elements (5, 1, 2) of the chromosome remain in place, and the other elements are written in reverse order, starting with the last one, which will already be in the fourth position in the chromosome representation. Thus, we obtain:

Chromosome C_3^3	5	1	2	4	3

Chromosome C_3^3



Calculate the length of chromosome C_3^3:

$$L_3(C_3^3) = 2 + 1 + 2 + 2 = 7.$$

Further, in Generation 2 of the population, we select the chromosome with the maximum length equal to 9, considered the least fit, either C_1^2 or C_3^2. We replace the less fit chromosome C_3^2 with fitter chromosome C_3^3 with a length of 7. Thus, we obtain Generation 3 of the population of chromosomes.

Table 4. Population of Chromosomes (Generation 3)

Chromosome C_1^3	5	1	4	3	2
Chromosome C_2^3	5	1	2	3	4
Chromosome C_3^3	5	1	2	4	3

The length of each chromosome is:

$$L_1(C_1^3) = 1 + 1 + 3 + 2 = 7;$$
$$L_2(C_2^3) = 1 + 1 + 2 + 2 = 7;$$
$$L_3(C_3^3) = 1 + 1 + 3 + 2 = 7.$$

The total sum of the edges of the graph according to the placement of the vertices on the grid determined by that particular population of chromosomes in the third iteration is:

$$S(3) = L_1(C_1^3) + L_2(C_2^3) + L_3(C_3^3) = 7 + 7 + 7 = 21.$$

The evolution of lengths in the case of generations 0, 1, 2, 3 is as follows:

$$S(0) = 30 > S(1) = 28 > S(2) = 25 > S(3) = 21.$$

The best placement of the vertices of the graph on the line is obtained in the last iteration, in which the last genetically modified chromosomes C_2^3 and C_3^3 represent the solution with $\min L(G) = 7$ and $S(3) = 21$.

References

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

[2] Mitchell, Melanie. *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 1996.

[3] Mitchell, Melanie. Genetic Algorithms: An Overview. *Complexity*, 1995, vol. 1, no. 1, 31–39.

[4] Russell, Stuart J., Norvig, Peter. *Artificial Intelligence: A Modern Approach*. Second Edition. Prentice Hall, 2003.

[5] Beasley, David, Bull, David R., Martin, Ralph R. An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, 1993, vol. 15, no. 2, 58–69.

[6] Dumitrescu, Dan. *Algoritmi genetici şi strategii evolutive - Aplicaţii în inteligenţa artificială şi în domenii conexe*. Cluj-Napoca: Editura Albastră, 2000.

[7] Емельянов, В.В., Курейчик, В.В., Курейчик, В.М. Теория и практика эволюционного моделирования. Москва: Физматлит, 2003.

[8] Goldberg, David Edward. *Genetic algorithms in search, optimization and machine learning*. Addison -Wesley: Reading, MA, 1989.

[9] Garey, Michael R., Johnson, David S. *Computers and Intractability: A Guide to NP-completeness*. New York: W.H. Freeman and Company, 1978.

[10] Oltean, Mihai. *Proiectarea şi implementarea algoritmilor*. Cluj-Napoca: Comp. Libris Agora, 2000.

Received: October 20, 2023 *Accepted: December 15, 2023*

(Liubomir Chiriac, Natalia Lupashco, Maria Pavel) "Ion Creangă" State Pedagogical University,
5 Gh. Iablocikin st., Chişinău, MD-2069, Republic of Moldova