TELEINFORMATICS REVIEW
NO. 3-4, 2018

Method of determining the longest simple chain in a graph with the use of a genetic algorithm

Łukasz MIELNICZUK, Łukasz STRZELECKI

Institute of Teleinformatics and Automation, Faculty of Cybernetics,
Military University of Technology
ul. Gen. W. Urbanowicza 2, 00-908 Warszawa
lukasz.mielniczuk@wat.edu.pl, lukasz.strzelecki@wat.edu.pl

ABSTRACT: This paper discusses the issue of determining the longest simple chain in a graph by using a heuristic algorithm - a genetic algorithm. A method enabling the effective determination of the longest chain in any connected, undirected graph without loops.

KEYWORDS: graph theory, Hamiltonian chain, heuristic algorithm, genetic algorithm

1. Introduction

According to the definition by B. Korzan [2], there may be multiple longest simple chains in a graph, although unlike maximum simple chains, they are all of the same length. This means that each longest chain is a maximum chain, while the reverse is false. If the given graph contains a Hamiltonian chain, it is at the same time the maximum and longest chain of the graph.

The problem of finding the longest simple chain in a graph can be reduced to the problem of finding a Hamiltonian chain or the longest simple chain that is not a Hamiltonian chain. Unlike the shortest chain problem, where algorithms exist that enable this problem to be solved in polynomial time, the problem of finding a Hamiltonian chain is an NP-complete problem.

It must be noted that there are multiple algorithms for finding the longest chain in a graph. They differ in the method used, and in their computational complexity. They are both analytical algorithms, such as the Roberts-Flores algorithm [3], and heuristic algorithms, an example being the ant colony algorithm.

The purpose of this paper is to present a method for heuristic determination of the longest chain in a graph using a genetic algorithm.

2. Graph properties

The graph definition applied in this paper follows the interpretation used by B. Korzan [2]. For the purposes of order and clarity, it is also given in this section. The definitions of *path, chain, simple chain* and *Hamiltonian chain* are given as well - they too follow the interpretation used by B. Korzan.

As this paper only refers to simple graphs, it is possible to adopt the definition of a graph as an ordered pair $G = \langle W, U \rangle$. Issues related to using the method described herein for directed graphs will be addressed in another paper.

From a practical standpoint, a graph can therefore be a model of a situation where we deal with two distinct sets, W and U, of elements of an object, with elements $u \in U$ representing distinct bonds between certain elements $w \in W$, which means that element u binds two elements $\{x, y\} \subseteq W$. We therefore refer to set W as the graph node set, and to set U as the graph edge set.

Node degree $s(x)$ is the number of graph edges that are incident to node x. Node degree is therefore equal to the number of all incoming and outgoing arcs, edges and loops that connect to the given node [2].

Path $G = \langle W, U \rangle$ in a graph is any alternating sequence of nodes $x_{i_s} \in W$ and edges $u_{i_s} \in U$ leading to children x_{i_s}, taking the form of:

$$\{x_{i_0}, u_{i_1}, x_{i_1}, u_{i_2}, x_{i_2}, u_{i_3}, \dots, x_{i_{l-1}}, u_{i_l}, x_{i_l}\},$$

such that for every $s \in \{1, \dots, l\}$ there exists edge connecting nodes x_s and x_{s+1}. The number of edges l present in the sequence defining the path is called the path length.

A specific case of path is a *chain*, which is a path connecting two graph nodes in such a manner that all edges along this path are different. A chain $Ł_i(x_{i_0}, x_{i_1})$ with an initial node x_{i_0} and terminal node x_{i_1} is therefore any path

$\{x_{i_0}, u_{i_1}, x_{i_1}, u_{i_2}, x_{i_2}, u_{i_3}, \dots, x_{i_{l-1}}, u_{i_l}, x_{i_l}\}$ that meets this condition. Chain length is the number of the chain's edges, therefore the maximum chain length in a graph with m edges is m.

A *simple chain* is a chain where all nodes are different. This means that the maximum simple chain in a graph with n nodes can have a length no greater than n-1. A maximum simple chain is any simple chain that is not part of another simple chain in the graph[1]. This definition implies that there may exist multiple maximum simple chains of different lengths in a graph.

A *Hamiltonian chain* is a simple chain that includes all nodes of a graph, thus if in a simple graph there is a longest simple chain $Ł(x_o, x_l) = \{\, x_o,\ u_1,\ x_1, \dots,\ x_{l-1},\ u_l,\ x_l \,\}$ with a length of $l \geq 2$ that meets the condition $s(x_o) + s(x_1) \geq l + 1$, then this chain is a Hamiltonian chain.

3. Principle of genetic algorithms

As issues related to the genetic algorithm will be used later in the paper, the mechanism of its functioning is shown below. This description has been prepared according to the classic interpretation of genetic algorithms, as presented by D. Goldberg [1].

Genetic algorithms are based on the mechanisms of natural selection and inheritance. They combine two principles that apply in nature - a greater survival chance of fitter organisms and their systematic development caused by information exchange (hybridisation) and individual mutation.

The algorithm assumes the existence of a certain initial population of individuals, which correspond to solutions to a problem. The initial population is generated randomly. The solutions are then given scores according to pre-defined criteria. During the next step, another population of individuals is generated, combining the characteristics of the best solutions from the preceding step. The analogy to the world of nature is clearly visible here.

The algorithm effectively utilises the experience of the previous populations to determine a new search area, for which the quality of the solutions is expected to be higher. The solutions in the given population that are scored higher than the others, are more likely to form a so-called parent pool for

[1] This means that a maximum simple chain in its entirety cannot be a fragment of another, longer chain.

the new generation. The next iteration of the algorithm will be performed on a dataset whose quality should be higher than in the preceding steps. The genetic algorithm searches for the best solution across a specific number of iterations, a pre-defined time, or until the optimum solution is found.

3.1. Basic operations used in a genetic algorithm

When performing a task using a genetic algorithm, you should begin by generating a set of random solutions that form the initial population. These solutions may take various forms - for example, numbers or tables. Solutions are also frequently presented as binary vectors.

The population size can be chosen freely and needs to be specified when solving the task. A population that contains a small number of individuals may make finding the best solution impossible, while an excessive number of individuals will significantly extend the search duration.

The prepared datasets forming the population are subsequently subjected to operations corresponding to processes occurring to chromosomes. These are: reproduction, hybridisation and mutation.

During the reproduction process, individual datasets forming the solution are reproduced and hybridised with one another. The first stage of this process is solution evaluation. Function f, which is an objective function (in biology, it is referred to as the fitness function), is used for this purpose. This function is a certain measure of the utility of the solution whose value we wish to maximise – the better the solution, the higher the value of the objective function.

Once all solutions have been given scores, the so-called selection needs to be performed, where individuals to be reproduced and form the parent pool for the next solution population are selected. Selection may be performed according to one of many strategies. The basic strategy is the so-called roulette method, where the probability of an individual being selected for the reproduction pool depends on the evaluation function score - the higher the value of the objective function for the given solution, the greater the probability it will be selected for the next generation.

The first step is to select the number and locations of hybridisation points. This is done based on the strategy chosen. The basic strategy assumes that the number of points is determined first, after which they are arranged at random. It is also possible, as in the selection process, to use mathematical distributions, according to which the hybridisation points are selected.

Next, divide the solutions into groups within which hybridisation will be performed. The simplest method is to select pairs of individuals, then hybridise them with each other, although hybridising more solutions at the same time may be employed as well.

Using hybridisation exclusively may cause potentially good solutions to be eliminated by narrowing the search set. To avoid this effect, so-called mutation is used when searching for the optimum solution. This involves randomly changing values in certain solutions. For bit sequences, this can be a change of values at specific positions in the sequence from 0 to 1 or vice versa, for tables it may, for example, be changing the values of selected elements in the table.

Figure 1 below shows a flowchart of a genetic algorithm.

4. Developing a method for finding the longest chain in a graph using a genetic algorithm

This section describes the principles adopted for the original method, and how the task was performed.

4.1. The principles

It was assumed that the solution needs to enable finding the longest path in any connected, undirected and loopless graph. This subsection describes the method developed. It contains information on the principles adopted, the adopted graph presentation method, and the objective function developed. The method's functioning is also presented using a sample graph.

4.1.1. The adopted graph presentation method

Following an analysis of the available solutions, it was decided that a transition matrix-based graph presentation method [4] would be used to achieve the aim of this paper. It was decided to introduce several additional principles, which would enable the method to be simplified and better adapted to the problem to be solved.

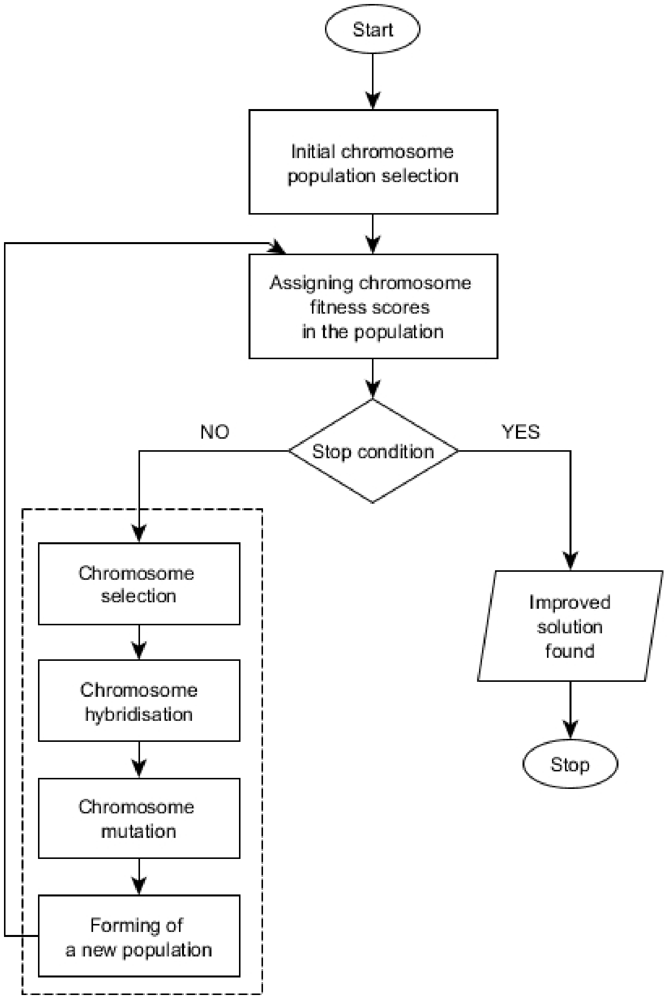

Start
Initial chromosome population selection
Assigning chromosome fitness scores in the population
NO
Stop condition
YES
Chromosome selection
Chromosome hybridisation
Chromosome mutation
Forming of a new population
Improved solution found
Stop

Fig. 1. Genetic algorithm flowchart

One principle is to analyse only loopless graphs. The main diagonal A of the transition matrix determines the presence of edges whose source and closure is a single node. Assuming that no loops are present in the graph in question, analysing the values of elements lying on the main diagonal of the matrix is not necessary, which enables the algorithm's computational complexity to be reduced.

Another simplification stems from the properties of undirected graphs. An edge connecting two nodes in an undirected graph is interpreted as a pair of antisymmetric arcs, which is reflected in the transition matrix. It can be observed that elements located above the main diagonal are symmetrical to elements located below the main diagonal, as shown in Figure 2.

Taking the above into account, a simplified matrix representation (derived directly from the transition matrix) can be adopted for undirected graphs, as shown in Figure 3.

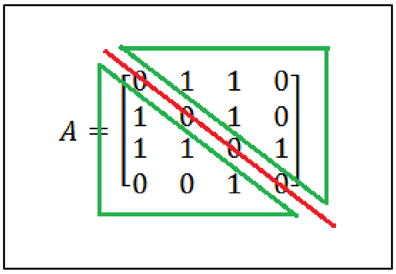


Fig. 2. Symmetry of transition matrix elements for undirected graphs

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\downarrow$$

$$B = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 3. Zeroing of elements located below the diagonal

Individuals that constitute the solutions for the genetic algorithm are commonly represented as bit vectors. It was therefore assumed that the graph in question will also be presented in this form. For this reason, matrix A was transformed to vector B in the manner described above. An example of transforming the transition matrix to a bit vector is shown in Figure 4.

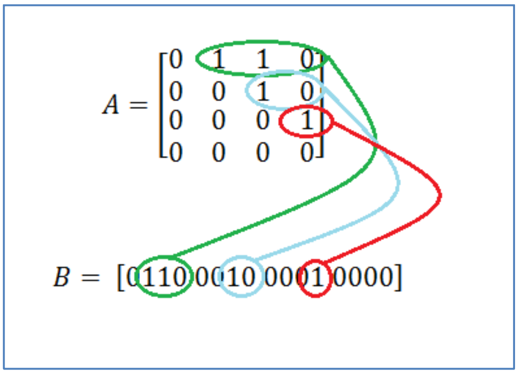

$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
$B = [0110\ 0010\ 0001\ 0000]$

Fig. 4. Transformation of the transition matrix of an undirected graph to vector form

It must be noted that the operation shown above is reversible, i.e. it enables building a simplified matrix, and then a transition matrix based on a bit vector (Figure 5). This property will be used in the original solution.

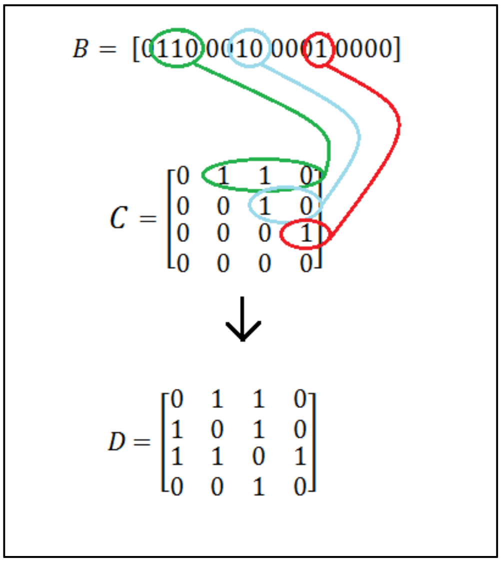

$B = [0110\ 0010\ 0001\ 0000]$
$C = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
$D = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Fig. 5. Transformation of an undirected graph vector to the matrix form

4.1.2. Objective function

The objective function is a crucial element of the genetic algorithm. It is used to assign scores to the randomly selected solutions, and consequently the algorithm's effectiveness depends on its correct functioning. This function should assume the highest values for the most valuable solutions. For the purposes of the method in question, an objective function with the following formula was developed:

$$f(x) = s_x - (k_x - s_x)$$

where:

s_x – length of the longest simple path for solution x,

k_x– number of graph edges for solution x.

The values assumed by the objective function expressed with this formula are higher, the greater the s_x value is, or in other words, the greater the power of the set of nodes between which the path runs[2] is. Additionally, by subtracting the difference $k_x - s_x$, which is the difference between the number of edges in the graph and the length of the longest path, from the result, we promote those solutions for which the graph contains the lowest number of edges. Taking the above into account, it can be assumed that the function enables the correct evaluation of the selected solutions, assigning higher scores to those that contain the longest simple path with a greater number of nodes.

4.2. Practical application of the method

This subsection presents the way this method is used to determine the longest path for the undirected graph shown in Figure 6.

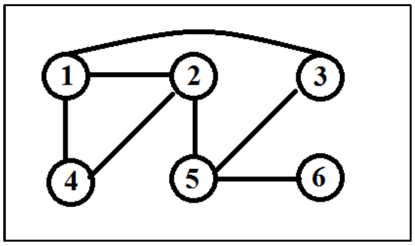


Fig. 6. Visualisation of a 6-node undirected graph

1) Preparation of the node transition matrix

[2] Any algorithm capable of performing this task can be used to determine the set in question. It must be noted that the paper does not analyse the effects of computational complexity of the selected algorithm on the total complexity of the proposed solution.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

2) "Zeroing" of elements of matrix A, located below the diagonal

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3) Transformation of matrix B to the vector form

C = [011100 000110 000010 000000 000001 000000]

4) Generating n vectors of length $l = 36$, constituting the first population. Population size $n = 6$ was assumed.

x_1 = [011111 000000 010110 010111 010100 001110]

x_2 = [000011 010100 011010 010110 000110 010100]

x_3 = [001101 011000 010000 010100 100100 001110]

x_4 = [000000 010000 011010 000100 110000 011110]

x_5 = [011111 011110 110010 010011 011110 111000]

x_6 = [010111 011110 000110 000000 010100 000000]

5) Assigning scores to the resulting solutions

a) Solution normalisation[3] based on vector B. The purpose of this operation is to obtain the value of 1 only at such positions in vectors x_1 - x_6 at which they also occur in vector B - which ensures that an appropriate edge exists in the graph in question.

x_{1n} = [011100 000000 000010 000000 000000 000000]

[3] Normalisation for the solution discussed means the preliminary preparation of solution vectors for further analysis by setting to zero those positions in the vectors for which no edges exist in the analysed graph.

x_{2n} = [000000 000100 000010 000000 000000 000000]

x_{3n} = [001100 000000 000000 000000 000000 000000]

x_{4n} = [000000 000000 000010 000000 000000 000000]

x_{5n} = [011100 000110 000010 000000 000000 000000]

x_{6n} = [010100 000110 000010 000000 000000 000000]

b) Assigning scores to the solutions using objective function f_x.

Tab. 1. Lengths of the longest graph paths for the randomly selected solutions

Solution	Length of the longest path
x_{1n}	4
x_{2n}	2
x_{3n}	3
x_{4n}	2
x_{5n}	4
x_{6n}	5

Solutions x_{6n}, x_{1n} and x_{5n}, which indicate the longest paths (among the selected solutions), received the highest scores.

c) Selection of solutions to participate in reproduction, using one of the strategies. For the roulette strategy, solutions with the highest objective function values have the highest probability of becoming the parent pool of the next generation. In the case in question, solutions x_{6n} and x_{1n} were randomly selected to be used in reproduction twice, while x_{5n} and x_{3n} will be used once.

d) Hybridisation of the selected solutions. A single hybridisation point was assumed, and the hybridisation point was selected using the roulette method and is $r = 3$. The solutions were divided into fragments at the 3rd position in the sequence.

x_{1n} = [011100 000000 000010 000000 000000 000000]

x_{2n} = [011100 000000 000010 000000 000000 000000]

x_{3n} = [001100 000000 000000 000000 000000 000000]

x_{4n} = [010100 000110 000010 000000 000000 000000]

x_{5n} = [011100 000110 000010 000000 000000 000000]

x_{6n} = [010100 000110 000010 000000 000000 000000]

Next, hybridisation pairs were randomly selected:

$$x_{1k} - x_{3k}$$
$$x_{2k} - x_{6k}$$
$$x_{4k} - x_{5k}$$

The following solutions were obtained:

y_1 = [011100 000000 000000 000000 000000 000000]

y_2 = [011100 000110 000010 000000 000000 000000]

y_3 = [001100 000000 000010 000000 000000 000000]

y_4 = [010100 000110 000010 000000 000000 000000]

y_5 = [011100 000110 000010 000000 000000 000000]

y_6 = [010100 000000 000010 000000 000000 000000]

e) Solution mutation was performed. One solution y_3 was randomly selected for mutation. Mutation was performed at three randomly selected positions: $a = 4, b = 6$, and $c = 20$.

After the mutation, vector y_3 has the following form:

y_{3m} = [001001 000000 000010 010000 000000 000000]

f)

$$y_1, y_2, y_{3m}, y_4, y_5, y_6$$

Repeat steps 4-7 for the resulting vectors. Repeat the steps for the assumed number of iterations or until the optimum solution is found.

g) After the requisite number of iterations were performed, the optimum solution z_{opt} in bit vector form was found, based on which a simplified matrix, and subsequently a transition matrix were built (these operations were the reverse of steps 1-3).

z_{opt} = [011000 000100 000010 000000 000001 000000]

$$\downarrow$$

$$C = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\downarrow$$

$$D = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

h) Based on transition matrix D, the optimum solution was selected - the longest path in the analysed graph. It is shown in Figure 7.

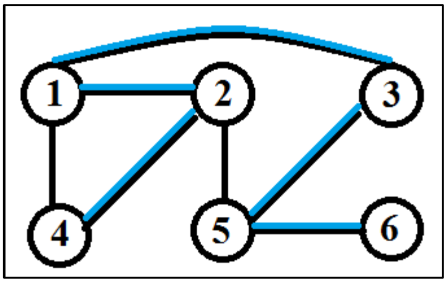


Fig. 7. Visualisation of the longest path in the analysed graph

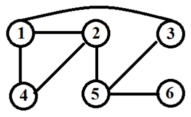
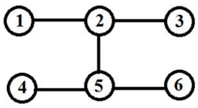
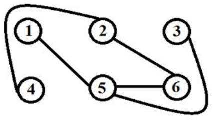
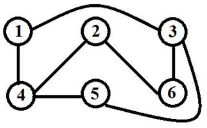
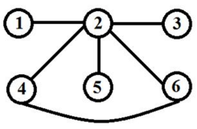
5. Acceptance tests

In order to confirm the effectiveness of the method developed, it was subjected to acceptance tests. The results obtained with its use were compared to results obtained by using an existing analytical algorithm - the Roberts-Flores algorithm [3].

The tests were performed on five 6-node graphs. The results are shown in Table 2.

The results confirm the effectiveness of the original method. For each test graph, the longest existing path was found. In some cases, it was not a path identical to that found by the Roberts-Flores algorithm. This stems from the fact that there may be multiple longest paths in a graph. One must remember that searching for a solution using a genetic algorithm is based on randomness, so different paths may be obtained after multiple attempts to use the method for a given graph.

Tab. 2. Comparison of the results obtained using the original method with the results obtained using another algorithm

No.	Analysed graph	Path found by the original method	Length of the path found by the original method	Path found by the R-F algorithm	Length of the path found by the R-F algorithm
1		4-2-1-3-5-6	6	3-1-4-2-5-6	6
2		1-2-5-4	4	1-2-5-4	4
3		4-2-6-5-1	5	1-5-6-2-4	5
4		1-3-6-2-4-5	6	1-3-6-2-4-5	6
5		1-2-4-6	4	1-2-4-6	4

6. Conclusion

The method of finding the longest chain in a graph using a genetic algorithm, presented in the paper, is an alternative to existing analytical methods (such as the Roberts-Flores algorithm). Presenting the graph as a transition matrix enabled it (after performing the appropriate transformations) to be presented as a bit vector, which is the usual form of presenting specimens that constitute solutions for genetic algorithms. An objective function enabling the solutions generated to be properly assessed was also developed.

Acceptance tests performed on the original method, which involved comparing the results obtained with its use to results obtained from an analytical algorithm (Roberts-Flores), confirmed its effectiveness.

It must be emphasised that further development of the original method is possible, which will enable the algorithm's computational complexity and processing time to be reduced. This can be achieved, among other ways, by preparing a more effective objective function that enables better evaluation of the resulting solutions, which will be beneficial for the quality of subsequent populations and thus will reduce the algorithm's processing time. It is also possible to develop a different method of graph presentation, which may enable reducing the solution's computational complexity.

References

[1] GOLDBERG D., *Algorytmy genetyczne i ich zastosowania*, Warszawa 2003.

[2] KORZAN B., *Elementy teorii grafów i sieci. Metody i zastosowania*, Warszawa 1978.

Online sources

[3] JAWORSKI J., *Algorytmy teorii grafów. Algorytm Robertsa – Floresa.* http://www.staff.amu.edu.pl/~jaworski/agrsyll6.pdf.

[4] WAŁASZEK J., *Reprezentacja grafów w komputerze.* http://eduinf.waw.pl/inf/alg/001_search/0124.php.

Metoda wyznaczania najdłuższego łańcucha prostego w grafie wykorzystująca algorytm genetyczny

STRESZCZENIE: W artykule rozpatrzono problem wyznaczania najdłuższego łańcucha prostego w grafie wykorzystującą algorytm heurystyczny – algorytm genetyczny. Przedstawiono metodę umożliwiającą efektywne wyznaczanie najdłuższego łańcucha w dowolnym grafie spójnym, nieskierowanym, bez pętli.

SŁOWA KLUCZOWE: teoria grafów, łańcuch Hamiltona, algorytm heurystyczny, algorytm genetyczny

Received by the editorial staff on: 26 June 2018