UDC 004.8

Artur Morozov, Artem Volokyta.

COMPARATIVE ANALYSIS OF HEURISTIC ALGORITHMS USAGE IN THE TRAVELING SALESMAN PROBLEM SOLVING IN TOPOLOGIES

This article discusses the issue of comparing the use of different heuristic algorithms to solve the salesman problem in different topologies. The example in a fully connected topology is given. We analyze different existing topologies and different heuristic algorithms, and conduct a statistical comparison of the results of using these algorithms to solve this problem with different input data.

Key words: euristic algorithms, traveling salesman problem, topologies, fully connected topology.

Fig.: 14. Tabl.: 1. Bibl.: 4.

Target setting. Firstly, the problem of solving the traveling salesman problem today is relevant in many areas of human activity, for example, in logistics. Secondly, the possibilities of computers are still limited. Thus, the choice of the more suitable heuristic algorithm for a particular situation is highly relevant.

Actual scientific researches and issues analysis. There have been many articles and papers in recent years on the problem of the salesman, but the clear criteria for the use of a particular heuristic algorithm are insufficiently studied.

Uninvestigated parts of general matters defining. Despite the large number of articles on similar topics, they usually do not define clear criteria when and what kind of the studied algorithms should be used. The problem of defining clear criteria when and what kind of the studied algorithms should be used remains little investigated. Moreover, this issue is even more relevant due to the fact that the study data usually differs depending on the software and hardware used in the research.

The research objective. The first purpose of this paper is to provide a comparative analysis of the use of heuristic algorithms to solve the problem of the traveling salesman. The second one is to find clear criteria for which it makes sense to use one of the studied heuristic algorithms. This article will focus on the study of heuristic algorithms such as ant, genetic and greedy, as well as the full search algorithm as the most obvious alternative.

The statement of basic materials.

The overview of existing topologies

Hypercube is one of the most used and the simplest topologies. This topology is a special case of mesh topology. There are different types of hypercube depending of dimension (fig. 1) :

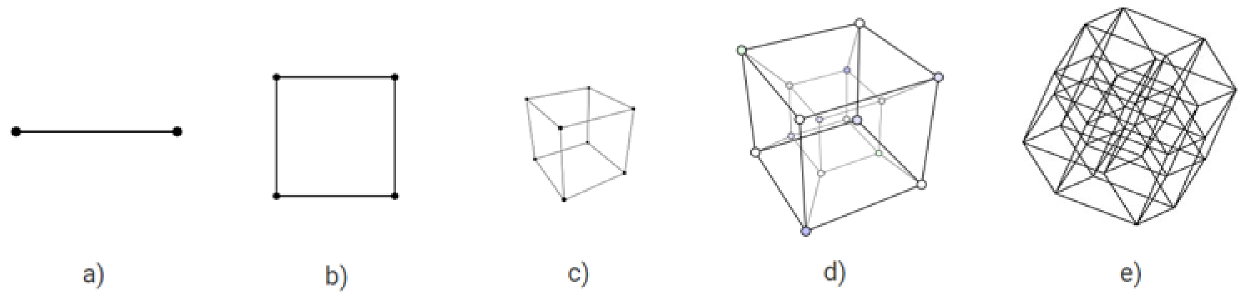

Fig. 7. Types of a hypercube : a – one-dimensional hypercube, b – two-dimensional hypercube, c – three-dimensional hypercube, d – tesseract, e - penteract

A hypercube refers to the static topologies, i.e., it can only have one direct path between two nodes, which is also fixed. It also means that the topology will not change during the task proceeding.[1] Here are some basic characteristics of a hypercube : Diameter : m; Number of connections : $mN/2$; Connectivity : $m-1$; Network size : $m(N-2^m)$; Node order : m; Bisection width : m.

Star is the simplest topology type. In this topology, all existing vertices are not connected to each other, but are connected to some node located in its center. The connectivity in this topology is really minimal, and all vertices, except the central one, are completely equal. All nodes in the star have the first order. The central node is often also called the hub. There are different types of a star (fig. 2) :

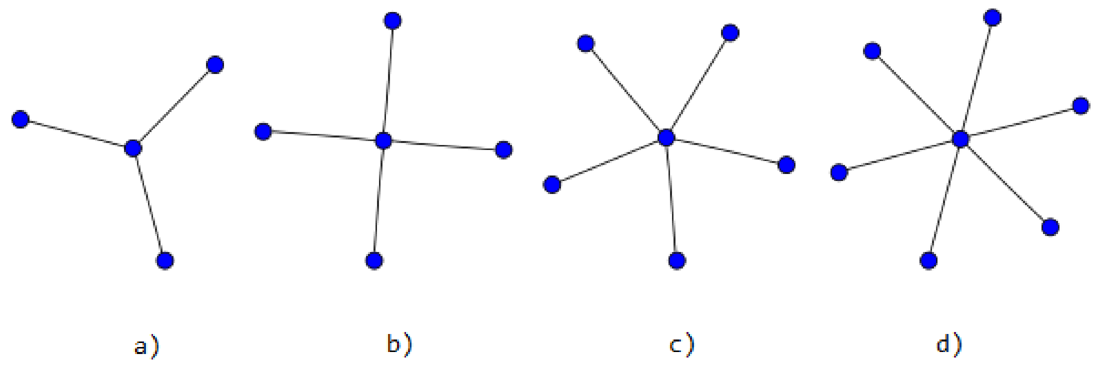

Fig. 8. Types of a star : a – claw or star with three ribs, b – star with four ribs, c – star with five ribs, d – star with six ribs [1]

The star topology has the following characteristics : Diameter : 2; Number of connections : $N-1$; Connectivity : 1; Network size : N; Node order : 1; Bisection width : 1. A star refers to the static topologies. [1]

Tree is an hierarchical topology. It means that the higher-level nodes in it are connected to the lower-level nodes. This connection is established by stellar connections. Thus a combination of stars is formed, and therefore this topology is also sometimes called a hierarchical star.

The name of the topology came to us from graph theory. The accepted name of the first node in the tree is the root. The following nodes, if they are at the higher level, are called parent. The nodes of the lower level are called child nodes.

Therefore, each child node that also has a connection and forms a hierarchy with lower-level nodes will be called the parent node for these nodes.

There are two common types of tree – binary (fig. 3 – a, b, c) and n-ary :

The tree topology has the following characteristics : Diameter : $2(log_2N - 1)$; Number of connections : $N - 1$; Connectivity : 1; Network size : N; Node order : 3; Bisection width : 1. A tree refers to the static topologies. [1]

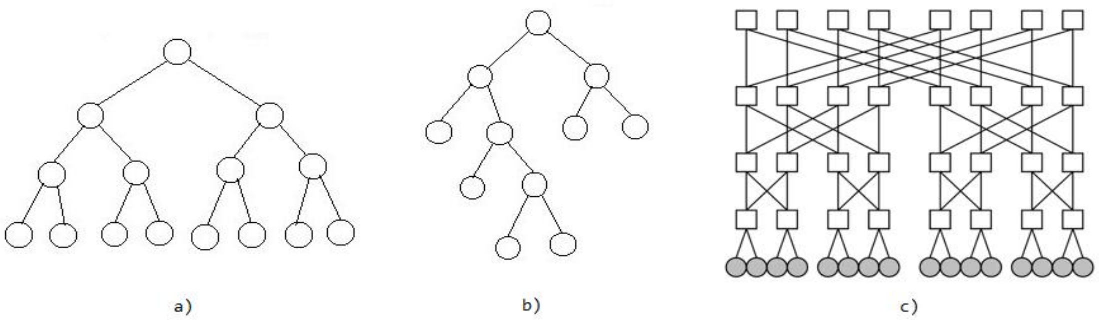

Fig. 9. Types of a tree : a – full binary tree [1],
b – incomplete binary tree, c – «fat» tree

Ring is a topology in which the connected nodes together with the edges of the connection look like a ring. In the simplest form of a ring topology, each node is connected to only the other two. The examples of a ring topology are in fig. 4.

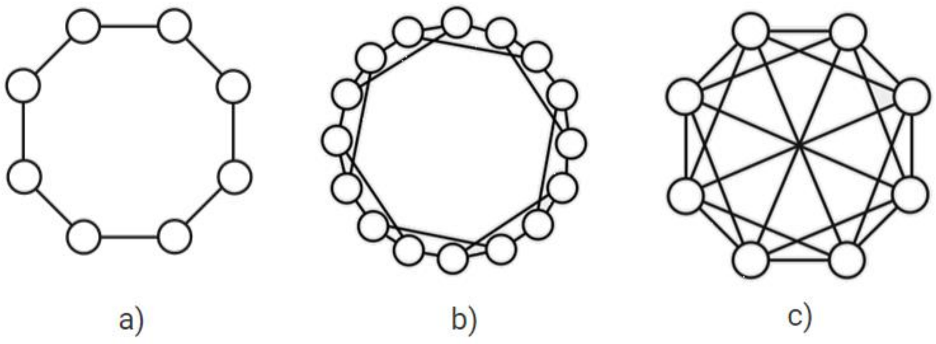

Fig. 10. Types of a ring : a – simple ring,
b - chordal ring, c - cyclic shift ring [1]

The ring topology has the following characteristics : Diameter : $N / 2$; Number of connections : N; Connectivity : 2; Network size : N; Node order : 2; Bisection width : 2; A ring refers to the static topologies. [1]

Mesh is a topology whose nodes form connections in the form of a one-dimensional or multidimensional mesh (fig. 5 – a).

Also each edge of such a mesh topology must be located parallel to its axis and connect two adjacent nodes located along this axis.

The mesh is a complicated variant of some other topologies, such as a hypercube.

A multidimensional mesh, which is also cyclically combined in several dimensions, is called a torus topology (fig. 5 – b).

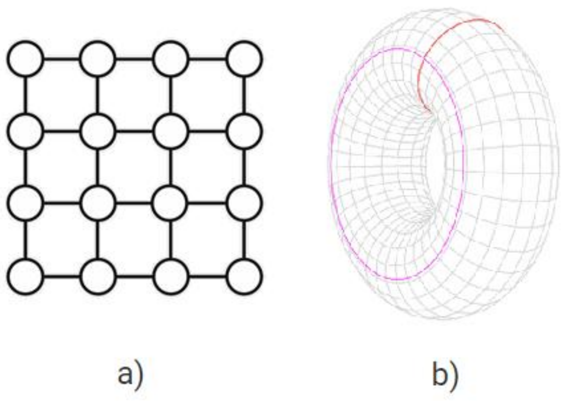

Fig. 11. Types of a mesh : a – classic mesh [1], b – Torus mesh

The mesh topology has the following characteristics :Diameter : $2(m-1)$; Number of connections : $2(\mathrm{N}-\mathrm{m})$; Connectivity : 2; Network size : N; Node order : 4; Bisection width : $\sqrt{N}$. A mesh refers to the static topologies. [1]

Fully connected topology is a topology in which all nodes are directly connected to each other. This topology is also often referred to as "maximum grouping" and "click topology". A fully connected topology allows transfer information with minimal transfer costs. An example of a fully connected topology is in the fig. 6.

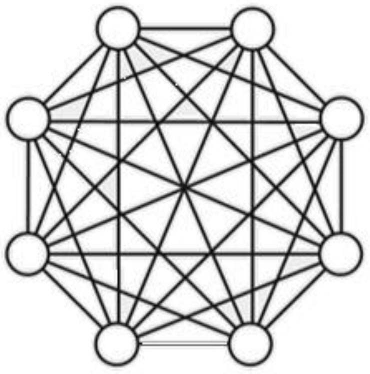

Fig. 12. Fully connected topology [1]

A fully connected topology has the following characteristics : Diameter : 1; Number of connections :$N(N-1)/\,2$; Connectivity : $N-1$; Network size : N; Node order : $N - 1$; Bisection width :$N^2/4$. A fully connected topology refers to a three-dimensional static topologies. [1]

Conclusion of an overview of existing topologies

In this paragraph, we have reviewed different topologies with all their advantages and disadvantages. accordingly Due to their differences, current topologies are usually used under the different conditions and provide different results.

To provide research in this article, a fully connected topology was chosen, since it is used in many places of human activity, and therefore it is relevant enough.

The overview of selected heuristic algorithms

Genetic algorithm uses the mechanism of evolution such as a natural selection. It means that more perspective individuals have more chances for survival and reproduction. Due to the transfer of genetic information from parents to children, children inherit the main qualities from their parents. So the descendants of more perspective parents will also be more perspective, and the percentage of their presence in population will be increased. After a change of several tens or hundreds of generations, the average fitness of individuals strongly increases. [2]

The block diagram of a genetic algorithm is shown in the fig. 7.

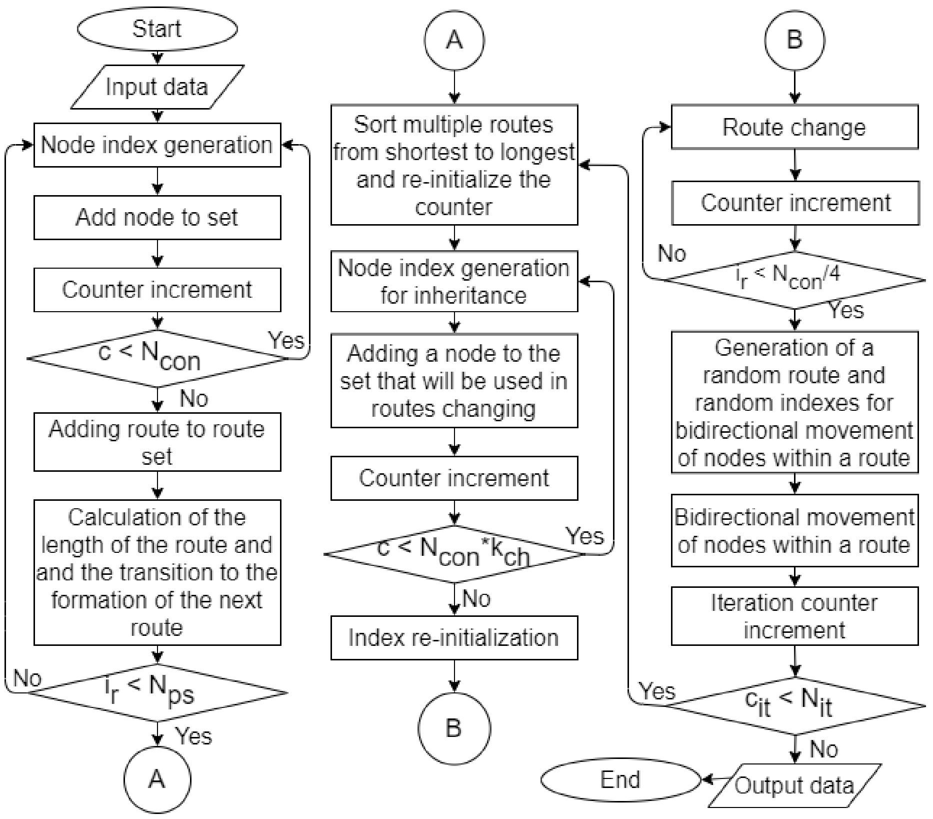
Start
Input data
Node index generation
Add node to set
Counter increment
c < N_con
Yes
No
Adding route to route set
Calculation of the length of the route and and the transition to the formation of the next route
No
i_r < N_ps
Yes
A
A
Sort multiple routes from shortest to longest and re-initialize the counter
Node index generation for inheritance
Adding a node to the set that will be used in routes changing
Counter increment
c < N_con*k_ch
Yes
No
Index re-initialization
B
B
Route change
Counter increment
No
i_r < N_con/4
Yes
Generation of a random route and random indexes for bidirectional movement of nodes within a route
Bidirectional movement of nodes within a route
Iteration counter increment
Yes
c_it < N_it
No
End
Output data

Fig. 13. Block diagram of a genetic algorithm [2]

Input data for a genetic algorithm : G_{rv} – the set of route variants between graph nodes; N_{ps} – the size of a set G_{rv}; c_{it} – the index of the node in route, can have value in diapason : [1, N_{ps}]; i_r – the index of a route, can have value in diapason : [1, N_{con}]; c – counter variable;

Greedy algorithm is about finding of local optimal solutions and building a solution of the global problem from them. The main advantage of this algorithm is the solution search time. The main disadvantage of greedy algorithm is that the solution in most cases is not optimal.The block diagram of a greedy algorithm is shown in the fig. 8.

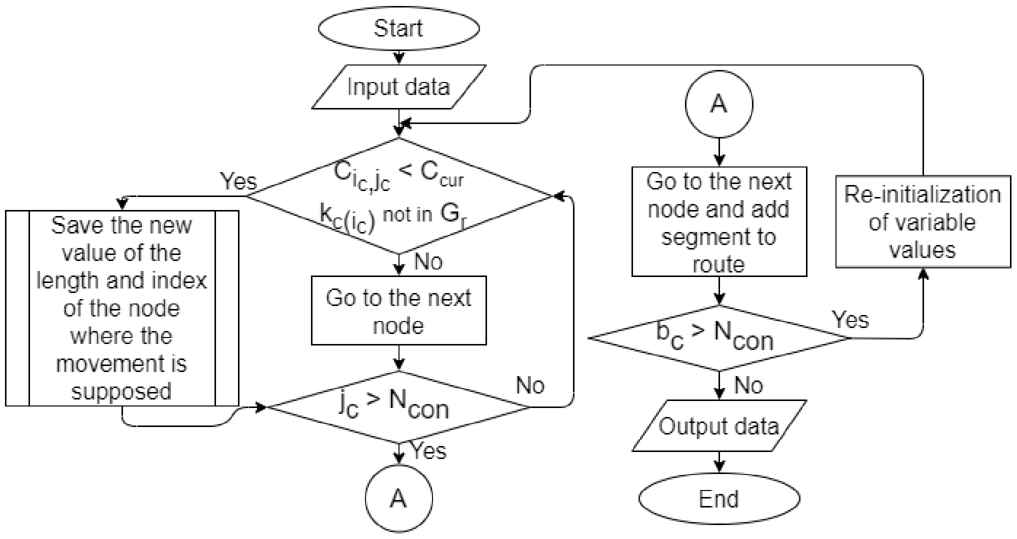
Start
Input data
$C_{i_c,j_c} < C_{cur}$
$k_c(i_c)$ not in G_r
Yes
No
Save the new value of the length and index of the node where the movement is supposed
Go to the next node
$j_c > N_{con}$
No
Yes
A
A
Go to the next node and add segment to route
Re-initialization of variable values
$b_c > N_{con}$
Yes
No
Output data
End

Fig. 14. Block diagram of a greedy algorithm

Input data for a greedy algorithm : K_c – the set of congestions of graph nodes; b_c – counter of graph nodes; i_c – index of node – current start point; j_c – index of node – current end point; c_{cur} – the most favorable value of the price of the rib for now; n_{cur} – index of node with the most favorable value of the price of the rib; G_r – nodes in a result route;

Brute force algorithm is about finding a solution by enumerating all possible solutions, and finding among these options one that satisfies the given requirements. The main advantage of this method is the highest accuracy of the result. But the main disadvantage is the amount of time needed to find all possible solutions, which grows with the increase of the number of vertices of the graph, which may require much more time than human life to find a solution. In our research, we used an upgraded brute force algorithm that stops calculations after the path length becames bigger than the best one. This upgrade allowed us to increase the number of vertices in graph at which it is possible to measure the execution time of this algorithm. [4]

Ant algorithm allows us to find approximate solutions of the traveling salesman problem. The main idea of the ant algorithm is to simulate the behavior of an ant colony when searching for food. Passing the route from one node of the graph (home) to the other (food source), ants (agents) leave a trace of pheromone after themselves, that is, they increase the significance of the rib, which decreases over time. The significance of the rib on the route depends on the length of the route and on the number of agents that have passed along the route. [3]

The block diagram of an ant algorithm is shown in the fig. 9.

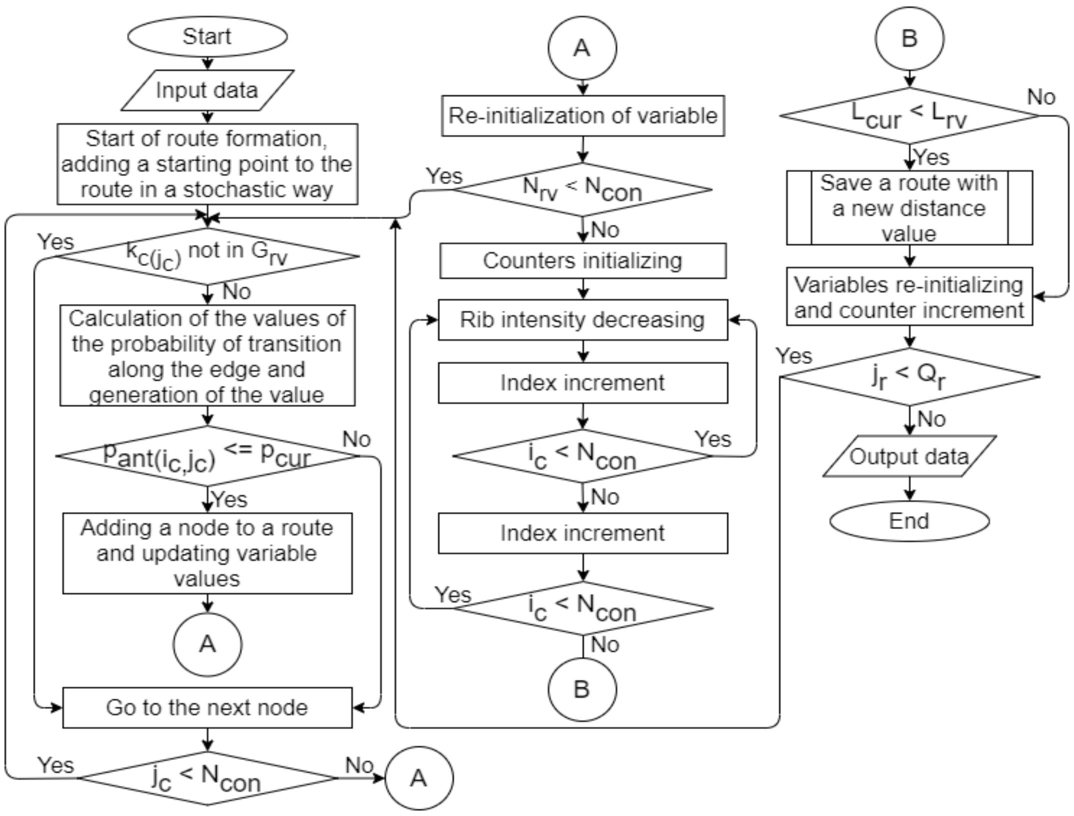
Start
Input data
Start of route formation, adding a starting point to the route in a stochastic way
Yes
$k_{c(j_c)}$ not in G_{rv}
No
Calculation of the values of the probability of transition along the edge and generation of the value
$p_{ant(i_c,j_c)} <= p_{cur}$
No
Yes
Adding a node to a route and updating variable values
A
Go to the next node
Yes
$j_c < N_{con}$
No
A
A
Re-initialization of variable
Yes
$N_{rv} < N_{con}$
No
Counters initializing
Rib intensity decreasing
Index increment
$i_c < N_{con}$
Yes
No
Index increment
Yes
$i_c < N_{con}$
No
B
B
$L_{cur} < L_{rv}$
No
Yes
Save a route with a new distance value
Variables re-initializing and counter increment
Yes
$j_r < Q_r$
No
Output data
End

Fig. 15. Block diagram of an ant algorithm [3]

As we can see on the block diagram, a route finding with the help of an ant algorithm consists of a multiple steps. Modeling the behavior of ants is related to the distribution of pheromone on the path - the edge of the graph in the traveling salesman problem. In this case, the probability of including the edge of the route for a single ant is proportional to the number of pheromone on this edge, and the amount of delayed pheromone is proportional to the length of the route. The shorter the route, the larger the pheromone will be delayed on the ridges, therefore, more than the number of ants will include its synthesis of their own routes. Modeling this approach using only positive feedback leads to premature convergence - most ants move in a path-optimal manner. This can be avoided by modeling negative feedback due to the evaporation of the pheromone. In this case, if the pheromone evaporates quickly, it leads to losing memory of the colony and forgetting good decisions, on the other hand, and for a long time to evaporate, it may be possible to obtain a stable local optimal solution.

Experiment. The next step of the investigation is a practical one. For his purpose we have written a program on Java. Then this program was launched on a device with these characteristics :

- Operation system : Windows 10
- CPU : Intel Core i7 – 6700HQ, 2.6 GHz
- RAM : 16.0 GB

In a written program we can specify the number of nodes in our future graph. These nodes will be placed in a fixed-size square that we set. All coordinates of the nodes are generated using the equally probable distribution law of a random variable. Thus, the topology structure is generated each time in a new way with new edge weights.

During investigation via the written program, we save the amount of time spent on execution and the length of the route found by each of the algorithms with the different inputted values.

The above tests for the brute force algorithm were performed only for the number of nodes in the graph less than or equal to 14. Further research with an increase of the number of nodes was performed without using a brute force algorithm, because its operation time grows exponentially. Thus, it would take too long to wait for the program to finish using this algorithm, and at the same time it would be obvious what kind of results it would show.

The results of the experiment with different values of the number of nodes of the graph are shown in the table 1.

Table 7

Results of the route finding

N	***Genetic algorithm***		***Greedy algorithm***		***Brute force algorithm***		***Ant algorithm***	
	Length	**Time**	**Length**	**Time**	**Length**	**Time**	**Length**	**Time**
5	852.195	3.138	852.195	0.001	852.195	0.001	852.195	0.001
6	1326.11	3.573	1398.256	0.001	1326.11	0.001	1326.11	0.001
7	1175.952	3.601	1175.952	0.001	1175.952	0.001	1175.952	0.003
8	1578.639	3.688	1665.621	0.001	1578.639	0.004	1578.639	0.011
9	1610.778	3.798	1722.655	0.001	1610.778	0.013	1610.778	0.009
10	1898.781	4.207	1925.52	0.001	1878.153	0.095	1878.153	0.013
11	2125.621	4.346	2017.228	0.001	2017.228	0.865	2094.74	0.015
12	2155.34	4.398	2587.219	0.001	2155.34	7.855	2155.34	0.01
13	2124.53	4.765	2124.534	0.001	2124.53	85.621	2124.534	0.017
14	2456.225	5.054	2720.253	0.001	2425.925	1355.92	2511.78	0.014
15	2591.591	5.427	2722.668	0.001	-	-	2630.005	0.015
30	3327.89	7.213	4269.43	0.001	-	-	3527.867	0.035
50	4122.437	9.659	5125.537	0.001	-	-	4590.65	0.095
100	6937.558	17.367	7578.976	0.001	-	-	7528.049	0.382
200	11120.53	37.293	11056.96	0.002	-	-	10527.65	1.918

There are charts of average time spent on calculations per each algorithm depending of quantity of nodes in the graph in the fig. 10.

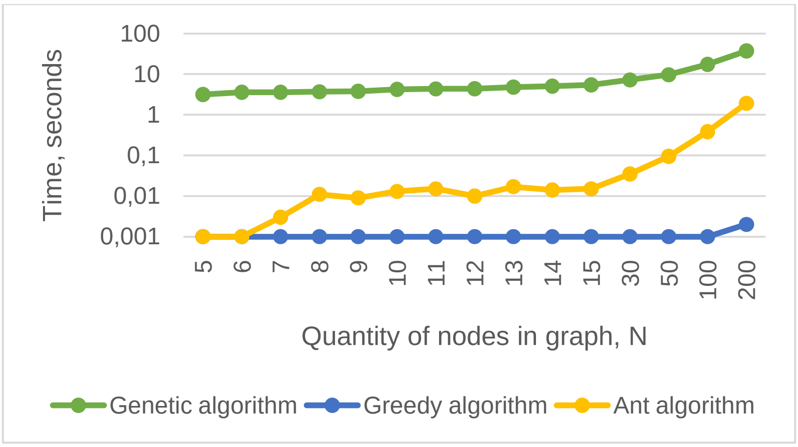

Fig. 16. Time spent on calculations per each algorithm depending of quantity of nodes in graph

After analyzing these charts, we can see that with an increase of the number of nodes in the graph, the time required to search for a route grows at very different rates for different algorithms.

Thus, for example, for a graph with 14 nodes, the time required for calculations using the brute force algorithm is 1355 seconds, while for a greedy algorithm it is only 0.001. At the same time, both ant and genetic algorithms show a satisfactory time compared to the brute force algorithm. For this reason, a brute force algorithm was not used in the experiments with bigger number of nodes in graph.

Then we can see the charts of the deviation of the length of the calculated route in the genetic, greedy and ant algorithms in relation to the length value calculated by the brute force algorithm in the fig. 11.

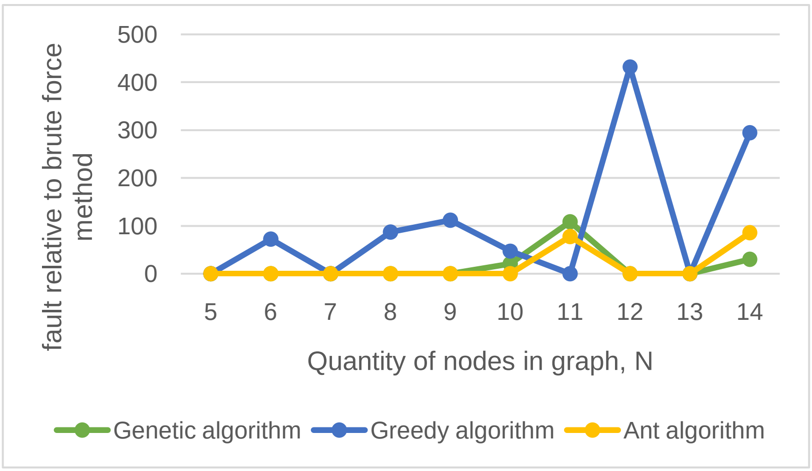

Fig. 11. Fault of each algorithm relative to brute force method depending of quantity of nodes in graph

After analyzing the charts in the fig. 11, we can say that the greedy algorithm proved to be the worst here. The scatter of the results that it produces is too unstable, so sometimes the fault in its calculations becomes too big. From the other side, when the number of nodes in the graph less than or equal to 14, we can see that both the ant and the genetic algorithm achieve a good result. This is especially true of the genetic algorithm. We can see that it proved to be the best for this quantity of nodes in graph.

As we can see from the table 1, when the number of nodes is less than 12, the brute force algorithm is the best one to use to resolve our problem. The result of its usage we can see in fig. 12.

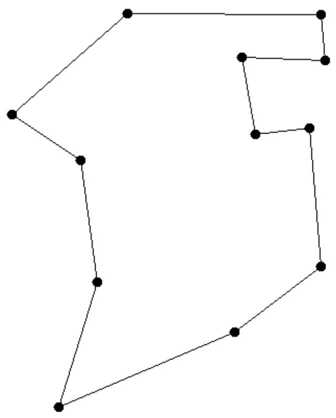

Fig. 12. Result of brute algorithm usage on graph with N = 12

Next, when the number of nodes is greater than or equal to 12, two algorithms are better than another ones. They are genetic and ant algorithms. The result of a genetic algorithm usage we can see in fig. 13.

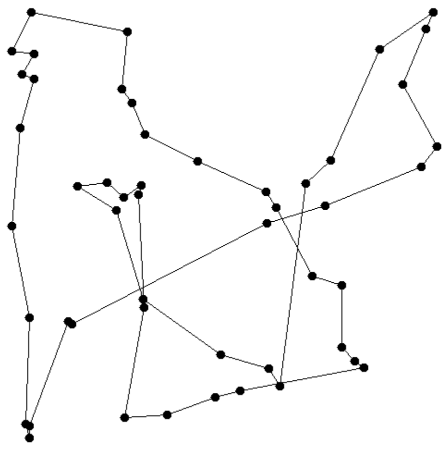

Fig. 17. Result of genetic algorithm usage on graph with N = 50

When the number of nodes is 100 or bigger, the best algorithm is the ant one (fig. 14). But a genetic algorithm also shows good results.

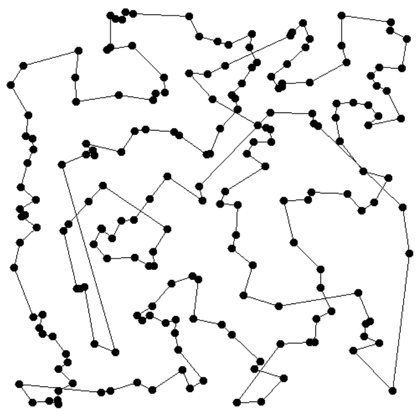

Fig. 18. Result of ant algorithm usage on graph with N = 200

As we can see, an ant algorithm gives us good result with an as short as possible path. For this quantity of nodes, a genetic algorithm result was also satisfying and had the length with the value close to the one that an ant algorithm gave us.

Conclusions. After the experiment and provided analysis, we can make some conclusions. Firstly, we found out that the criteria for choosing an algorithm for calculating the route in the topology will depend on the number of nodes N and the available machine time T. Based on this, we can also make the following conclusions:

- If $N < 12$, the route search is best carried out using the brute force algorithm, which allows us to get the exact solution in time $T < 1s$;
- If $12 \leq N < 100$, the search for the route is best carried out using genetic or ant algorithms that allow solutions to be obtained for the time $T_1 < 17s$, and $T_2 < 0.4s$ respectively;
- If $100 \leq N < 200$, the search for the route is best carried out using ant and genetic algorithms that allow us to get a solution for the time $T_1 < 2s$, and $T_2 < 37s$ respectively. And also the result will be much more stable than in case of using the greedy algorithm;
- We should not underestimate the greedy algorithm, it can be used if $N > 12$, when it is very important for us to reduce the amount of machine time used.

References

1. Cilker, B., Orlov, S. (2011). Organization of Computers and Systems: Book for High Schools -2nd edition.(p. 688).

2. Kureichik, V. (1998). Genetic algorithms - 2nd edition (pp. 5 – 11).

3. Kureichik, V., Kazharov, A. (2008). About some modifications of an ant algorithm – 4th edition (pp. 7-11).

4. Levitin, A. (2006). Introduction to The Design & Analysis of Algorithms (p. 141).

AUTHORS

Artur Morozov – 4th year student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

E-mail: ar2r.pv@gmail.com

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

E-mail: artem.volokita@kpi.ua