UDC 004.051:681.5 **doi: 10.26906/SUNZ.2024.2.144**

O. Skakalina, A. Kapiton

National University “Yuri Kondratyuk Poltava Polytechnic”, Poltava, Ukraine

COMPARATIVE ANALYSIS OF THE APPLICATION OF HEURISTIC ALGORITHMS FOR SOLVING THE TSP PROBLEM

Abstract. The need to solve the traveling salesman problem (TSP) often arises when solving practically significant optimization problems, such as problems in the field of economics, logistics in the widest range of applications, in chains of technical programs. Quite often, the specifics of these problems require obtaining a solution that is as close to the exact value as possible. But the TSP problem is NP-complex, that is, its exact solution can be obtained only in exponential time. Therefore, it is not efficient to solve the TSP problem by the full search algorithm in the presence of a large number of vertices of the graph. However, there are various heuristic algorithms that allow finding a rational solution to this problem with a large number of vertices in a time acceptable for the relevant subject area. In this work, the problem of the traveling salesman is defined as a mathematical programming task of finding the shortest path for the movement of a traveling salesman (traveling salesman), the goal of which is to visit all the objects involved in the task in the shortest time and with minimal costs. Appropriate adaptations of the heuristic algorithms, namely the genetic algorithm and the ant colony algorithm, were developed in the MATLAB environment. A computational experiment was performed on the same input sample, a comparative analysis of the performance of two heuristic algorithms, and the effectiveness of the use of heuristic algorithms for solving NP-complex problems was proven.

Keywords: genetic algorithms, NP-complex problem, TSP-problem, ant colony algorithm, MATLAB.

Introduction

At the current stage, many effective algorithms have been developed for solving the traveling salesman problem (TSP). However, the problem remains NP-complete, which means that its solution in the general case takes time, which grows exponentially with the number of cities. Therefore, genetic algorithms are usually used to solve the problem for large data sets.

The importance of the decision of the Central Committee can be highlighted in the following areas:

• *Transport:* TSP is widely used in transport logistics to optimize freight transport routes. This allows you to reduce transportation costs, increase the efficiency of the use of vehicles and reduce the negative impact on the environment.

• *Sales:* TSP is used to optimize product distribution routes. This allows companies to reduce shipping costs, improve customer service and increase sales.

• *Production:* TSP is used to optimize routes for the delivery of raw materials and finished products at enterprises. This allows companies to reduce production costs, improve resource efficiency and reduce production waste.

In addition, TSP is used in other areas, such as:

• *Information technologies:* TSP is used to optimize data delivery routes in computer networks.

• *Service:* TSP is used to optimize customer service routes.

• *Entertainment:* TSP is used in some games such as chess and go.

• *Computer science:* TSP is used for testing optimization algorithms. It is a good example of a complex problem for which there are many effective algorithms.

• *Other spheres of activity:* TSP is also widely used in other spheres of activity. For example, it can be used to optimize customer service routes, travel routes, flight schedules, etc.

The development of technologies and the globalization of the economy make the solution of TSP even more urgent. For example, the growth of trade volumes between countries requires the search for the most efficient routes for the delivery of goods. And the development of unmanned aerial vehicles opens up new opportunities for the application of TSP in logistics. The solution to this problem is of particular importance in the current state of the country. Because calculations of optimal flight routes of military UAVs are of decisive importance.

The solution of TSP is a complex mathematical problem. However, thanks to the development of computer technologies, effective algorithms have been developed that allow finding practically optimal solutions for problems with a large number of cities.

1. Analysis of recent research and publications

The traveling salesman problem is defined as a mathematical programming task of finding the shortest path for a traveling salesman (traveling salesman), the goal of which is to visit all the objects involved in the problem in the shortest time and with minimal costs. In graph theory, it is finding a route that connects two or more nodes, using the appropriate criterion of optimality. The task of the wandering merchant is to find the most optimal path that passes through the marked cities at least once. In the conditions of the problem, the indicator of the profitability of the route (the cheapest, the shortest, the least time-consuming, etc.) and the corresponding matrices of costs, distances, etc. are indicated. As a rule, it is indicated that the path should include each city of the route only once, under this condition the choice is made between the so-called Hamiltonian cycles. The traveling salesman problem (TSP) is an NP-complex discrete optimization problem. There are no fast polynomial algorithms for its solution. In terms of graph theory, this problem is defined as

follows: the shortest path should be found that passes through the defined nodes of the graph at least once with a subsequent return to the node at the beginning of the route [1].

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem that consists in finding the shortest route that passes through all cities at least once. Formally, the task of a traveling salesman can be formulated as follows:

Given:

set of cities V={1,2,…,n};

distances between cities dij, where i,j∈V.

Find:

a route R passing through all cities V and having a minimum length of:

i=1∑n−1di,i+1+dn,1=R∈Rmin(i,j)∈R∑dij

where R is the set of all routes passing through all cities of V.

For the first time, the TSP problem was formulated in 1930 and today it is one of the most studied optimization problems in information and communication technologies [2-8]. Despite the fact that the problem is computationally complex (belongs to the class of NP-complex), many heuristic and exact algorithms are known, which were used to solve practical problems.

Research on solving the traveling salesman problem can be conditionally divided into two directions [2-8].

1. Development of accurate algorithms that work fast enough only for small-sized problems;

2. Development of "non-optimal" or heuristic algorithms that provide approximate solutions in reasonable time.

One can also highlight the search for special cases of the TSP problem ("sub-problems") for which either better or exact heuristics are possible.

It is also important to distinguish between symmetric and asymmetric TSP tasks. For the symmetric case (usually called simply TSP), for all distances in D the equality dij = dji holds, that is, it does not matter whether we move from i to j or vice versa, the distance is the same. In the asymmetric case (called ATSP), the distances are not equal for all pairs of cities. Problems of this kind arise when we are not dealing with spatial distances between cities, but, for example, with the cost or time required to travel between places, where the price of a plane ticket between two cities may be different. In this case, we will consider a symmetric version of the TSP problem.

It is quite obvious that the task can be solved by going through all the travel options of the traveling salesman and choosing the optimal one. The thing is that the number of possible routes N grows very quickly with the number of cities to visit n N=n!.

For example, for n = 100, the number of options will be represented by a 158-digit number. A modern computer capable of processing a million operations per second will struggle with the task for about 144 years.

It is considered proven that there is no exact TSP solution algorithm that has polynomial complexity (that is, has an asymptotic estimate of the algorithm execution time T(n) = O(nα)) of the complexity of execution. Therefore, the TSP problem for any large n becomes almost unsolvable for exact algorithms.

TSP is a problem that is hard for NP and is so easy to describe and so hard to solve. In this case, you should give up trying to find an exact solution to the traveling salesman's problem and focus on finding an approximation - even if not optimal, but at least close to it. Due to the great practical importance of the task, approximate solutions will also be useful [2-8].

The Evolutionary Algorithm (EA) is an algorithm first proposed by Charles Darwin in 1859. It provides solutions for various optimization problems. It copies the process of evolution that occurs in nature, i.e. mutation, recombination and natural selection. GA (Genetic Algorithm) is a type of evolutionary algorithm. GA provides a solution to a problem in the form of strings of numbers and applies operators such as mutation and recombination. It starts from the seed and finds the most appropriate generation of the population using these operators.

The ACO (Ant Colony Optimization) algorithm is also a heuristic algorithm first discovered by Marco Dorigo that provides an optimal solution by simulating the way ants find food. ACO is a type of AI (swarm intelligence) technique.

The PSO (Particle Swarm) algorithm, first discovered by Kennedy and Eberhart, also provides an optimal solution to the problem and is inspired by flocks of birds, fish, and herds of animals. This mimics how they find their food environment and follows an information sharing approach.

In addition to presenting TSP as a combinatorial optimization problem, TSP can also be formulated as a theoretical problem of graph theory.

Let us present TSP in the formulation of the graph theory problem. Graph theory defines the problem as finding the Hamiltonian cycle with the least weight for a given complete weighted graph. This kind of problem formulation is widespread in engineering applications and some industrial problems such as machine planning, cellular manufacturing and frequency assignment problems can be formulated as TSP.

Let a weighted graph G = (V, E) be given, in which cities correspond to the set of vertices V = {1, 2,. . . , n} and each edge ei ∈ E has a corresponding weight wi representing the distance between the cities it connects. If the graph is not complete, missing edges can be replaced by edges with very long distances.

The goal of solving the TSP problem is to find a Hamiltonian cycle, that is, a cycle that visits each node on the graph exactly once, with the smallest possible weight for the given graph. This formulation naturally leads to procedures involving the search for minimal tree frames for a given graph.

TSPs can also be represented as integer and linear programming programs. The formulation of integer programming (IP) is based on the assignment problem with the additional constraint of no subtours [9].

The branch-and-bound method is a discrete optimization algorithm used to find the optimal solution to a problem in which only discrete variable values are possible. The method works by building a decision tree,

in which each node represents a possible solution to the problem. The branch-and-bound method starts with the root node of the tree, which represents the initial solution to the problem. Then, the algorithm uses a cut-off criterion to discard nodes that cannot contain an optimal solution. The cutoff criterion is a function that estimates the probability that a node contains an optimal solution [10].

If the node is not discarded, then the algorithm branches it into two new nodes, which represent two possible options for continuing the solution of the problem. Then, the algorithm repeats this process for each of the new nodes.

The branching process continues until an optimal solution to the problem is found or until the entire decision tree is explored.

The branch-and-bound method is an effective algorithm for solving many discrete optimization problems. However, it may not be effective for problems with very large decision trees [10]. Here is an example of how the branch-and-bound method can be used to solve the traveling salesman problem. The task of the traveling salesman is to find the shortest route that passes through all the cities visited by the traveling salesman.

To solve this problem, we can build a decision tree in which each node represents a possible route of the traveling salesman. Then, we can use a cut-off criterion to discard nodes that cannot contain an optimal route.

One of the cutoff criteria that can be used for the traveling salesman problem is the following. If the length of the route from a node to its parent node is greater than or equal to the length of the shortest known route, then the node can be discarded.

This cutoff criterion works because if the length of the route from a node to its parent node is greater than or equal to the length of the shortest known route, then that node cannot be a continuation of the shortest known route. By using this cutoff criterion, we can limit the size of the decision tree to be examined. This can significantly improve the efficiency of the algorithm.

The branch-and-bound method is used to solve many discrete optimization problems, including:

- Task of a traveling salesman;
- Maximum flow problem;
- Placement problem;
- Timetable problem;
- Backpack problem;
- Assignment task.

The branch-and-bound method is a powerful tool for solving discrete optimization problems. It can be used to solve a wide range of problems, including problems with very complex constraints [10]. Dijkstra's algorithm is a widely used algorithm for finding the shortest path in a weighted graph from one vertex to all others. It was developed by Edsger Dijkstra in 1959 and is characterized by its simplicity and efficiency.

Working principle:

1. Input data:

A weighted graph G = (V, E), where V is the set of vertices and E is the set of edges, each edge is assigned a weight (price).

The initial vertex s from which to find the shortest paths.

2. Initialization:

For each vertex v in G, we store two values:

dist[v]: Estimated distance from initial vertex s to v (initially infinite except dist[s] = 0).

prev[v]: predecessor of v on the shortest path (None initially).

3. The main cycle:

- We select an unvisited vertex v with the minimum dist[v] score.
- Mark v as visited.
- For each unvisited neighbor w of vertices v:
- new_dist = dist[v] + weight(v, w): we calculate the estimated distance to w through v.
- If new_dist < dist[w]: update the estimated distance dist[w] and the predecessor prev[w] to v.

4.Output:

dist[v] contains the distance from the starting vertex s to each vertex v.

prev[v] allows you to recover the shortest path from s to v, moving along the predecessors.

Advantages:

- Ease of implementation.
- Guaranteed to find the shortest path.
- Effective for graphs with non-negative weights.
- Disadvantages:
- Inefficient for graphs with large negative weights.
- Cannot find all shortest paths if the graph has cycles with negative weights.
- Algorithms of combinatorial optimization are classified by such an indicator as the accuracy of obtaining a solution.

2. Statement of the research problem

Genetic algorithms are a family of computational models inspired by evolution. These algorithms work on the principle of encoding a potential solution to a specific problem based on a simple chromosomal data structure, by recombining these structures while preserving critical information. Genetic algorithms are often considered as an optimizer of functions, although the range of problems of their application is much wider. The implementation of the genetic algorithm begins with the formation of a population of (usually random) chromosomes. Then these structures are evaluated and their reproductive capabilities are identified, that is, those chromosomes that are the best solution to the target problem. Thus, from generation to generation, useful traits spread throughout the population, and bad ones gradually disappear. Thanks to the crossing of the most adapted individuals, more promising areas of the search space are inherited. Eventually, the population will converge to the optimal solution to the problem. GA finds approximate optimal solutions in a fairly short time, which is an obvious advantage of this method. Thus, the genetic algorithm is a heuristic search algorithm that is used to solve optimization and modeling problems by randomly selecting, combining, and varying the parameters sought using mechanisms that resemble biological evolution.

"Genetic" algorithms began to be called later, until in 1975 Holland called them reproductive plans and considered them primarily as adaptation algorithms. But the shift in emphasis in the interpretation of the concept of "adaptation", which the author talks about in the preface of 1992, very accurately conveys the state of ambiguity, trying, on the one hand, to give a fairly general and uncontroversial concept of adaptation, and on the other hand, to distinguish between the concepts of adaptation and optimization, adaptation and evolution, adaptation and learning.

Genetic algorithms are search algorithms that are based on the concepts of natural selection. In nature, individuals with better survival traits exist for a longer period of time, as they have a better chance of producing offspring with similar genetic material. Over time, the entire population will consist of a large number of genes from superior individuals and a smaller number from weaker ones. The mistake of older theorists, such as Jean-Baptiste Lamarck, was that the environment influenced the individual personality. That is, the environment will force the individual to adapt to it. The molecular explanation of evolution proves that this is biologically impossible. The species does not adapt to the environment, rather, only the strongest survive. This is how natural selection works in genetic algorithms. The genetic algorithm differs from other search methods in that it searches among a set of points and works with a set of parameters, rather than the parameter values themselves. It also uses objective function information without any gradient information. Whereas traditional methods use gradient information. Because of these features of the algorithm, they are applied to various optimization functions, parameter estimation, and machine learning programs [11].

At this stage of development, there is no specific strategy for building a solution, there is a huge number of individual implemented algorithms that are not very similar to each other. However, the operation of all these algorithms can be presented in the form of a traditional scheme of operation of these algorithms (Fig. 1). There can be several criteria for stopping the search for a solution: time frames, the number of created populations, and a decrease in the improvement of the fitness function compared to previous iterations.

The potential of genetic algorithms is difficult to overestimate. When it is almost impossible to find a solution by conventional methods, they are the obvious way out of the situation. In various forms, genetic algorithms are applied to scientific and technical problems. They can be used in building computational frameworks such as state machines and sorting networks. They are often used in the design of neural networks and robot management, in modeling the development of processes in various subject areas, in game strategies, scheduling, logistics, cutting tasks, in the development of artificial life, and in many other areas [12]. However, the most popular application of genetic algorithms is the optimization of multiparameter functions.

So, for example, experiments are being conducted to create robots working in a team for the purpose of demining the territory. It is based on a multilayer neural network, which is directly responsible for controlling the robots and, in addition, transmits and receives signals to other team members. In other words, in addition to parametric optimization, the hybrid neuromodel performs an implicit creation of a communication language between robots. All parameters of the neural network are optimized using genetic algorithms. The results of such hybridization of various methods and algorithms of artificial intelligence show a clear advantage of a team of sapper robots that exchange information with each other over robots that operate without any communication. As a result, this technology can save many human lives in the long run, and hopefully the sapper's profession will not be so relevant.

In recent years, the scientific direction of Natural Computing has been intensively developed, which combines mathematical methods that incorporate the principles of natural decision-making mechanisms. These mechanisms ensure the efficient adaptation of flora and fauna to the environment for millions of years.

One of these methods is ant algorithms, which are based on the principles of self-organization of an ant colony. Despite the disjointed behavior of each of its representatives, it forms a highly organized system consisting of a large number of "agents" - ants, and thanks to this, it is able to solve complex tasks that exceed the capabilities of each individual element.

Studying the behavior of an ant colony is interesting for computer science because it provides insight into the disjoint organization, which is very useful for solving complex optimization problems.

The idea behind the ant algorithm is to model the behavior of ants related to their ability to find the shortest path from the anthill to the food source and adapt to changing conditions by finding a new shortest path. When moving, an ant marks its path with a pheromone, and this information is used by other ants. This is the basic rule of conduct for each representative of the colony in case the old route becomes unavailable. If an ant encounters an obstacle while moving, it will go around it to the left or right with equal probability. The same will happen on the way back. However, those ants that choose the shortest path will go through it faster and in a few moves it will be stronger. If we model the process of such behavior on a graph, the edges of which are all kinds of movement paths, then for some time the path most enriched with pheromone will be the shortest, which will solve the problem [13].

3. Basic material and results

The first version of the ant algorithm was proposed by Marco Dorigo in 1992. This scientist, in fact, proposed a mathematical model of the behavior of ants looking for paths from a colony to a food source and is metaheuristic optimization.

An ant colony is considered as a multi-agent system in which each agent (ant) functions autonomously according to very simple rules. In contrast to the almost primitive behavior of agents, the behavior of the entire system turns out to be surprisingly reasonable [14].

The basis of the "social" behavior of ants is self-organization - a set of dynamic mechanisms that ensure the achievement of a global goal by the system as a result of low-level interaction of its elements. A fundamental feature of such interaction is the use of only local information by system elements. Self-organization is the result of the interaction of the following five components:

a. randomness;
b. multiplicity;
c. positive feedback;
d. negative feedback;
e. objective function.

These components are the key properties of the ant algorithm. Let us consider them in more detail using the example of the movement of ants for food from an anthill (Fig. 1) [14].

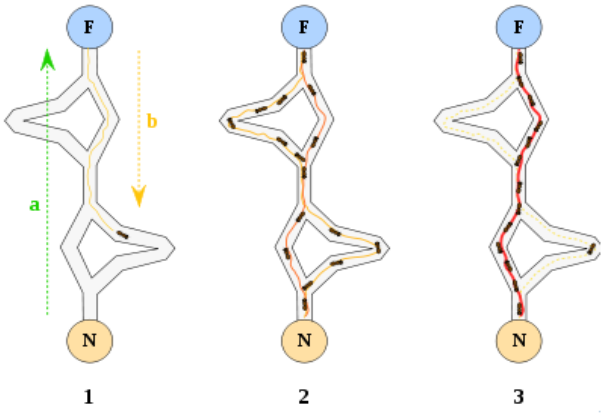

Fig. 1. Movement of ants for food

When an ant moves from an anthill (point F) to food (point N), pheromone is deposited throughout the entire path (obviously, the shortest path is marked by the green arrow – (a)). At the same time, the greater the density of the pheromone, the shorter the path, accordingly, the ant will leave less pheromone on a long section of the path. The longer the path, the faster the pheromone evaporates. Over time (if an approximately equal number of ants move along all sections), the ants will leave the most pheromone on the shortest section of the path. Thus, most ants will choose the shortest path - that is, they will leave even more pheromone on it, which will reduce the probability of moving along other routes (although this probability will remain, as you can see, one ant still moves along a different part of the path).

Accident. From the above example, it is possible to reveal the random nature of the movement of ants. Indeed, at the first stage, ants focus on all kinds of options for constructing their path. Even when the density of the pheromone helps to choose the optimal (shortest) route, there remain "distrustful individuals" who do not stop trying to find more optimal routes.

Multiple use. Of course, with one ant or even dozens of individuals, it is unlikely to be able to choose the optimal path. It takes many, many attempts to find the shortest path or at least warn others not to go in that direction. A pheromone and positive feedback serve as such a warning [14].

Positive feedback. Zoologists call it stigmergy. Stigmerges are a time-dispersed type of interaction, when one subject of interaction changes some part of the environment, and others use information about its state later, when they are in its vicinity. Thus, positive feedback is a kind of "collective memory" based on pheromone. Using this "memory" (based on trial and error) you can find the right solution.

Negative feedback. Eventually, the pheromone evaporates, which allows the ants to adapt their behavior to changes in the external environment. The distribution of pheromone in the space of movement of ants is a kind of dynamic variable of the global memory of the anthill. Any ant at a fixed moment in time can perceive and change only one local cell of this global memory [15].

Objective function. The most important element of the algorithm is the objective or fitness function to be optimized (for an ant, this is the shortest route). But you can successfully solve other similar tasks. Do not forget that the ultimate goal of the algorithm is optimization (in the language of mathematics, it means finding a global or acceptable maximum or minimum). The algorithm is heuristic, that is, it does not guarantee an exact solution, but only an approximate or acceptable one.

Condition of the problem. It is necessary to find the shortest route that starts in the starting city and ends in it. The route must pass through all cities only once.

The ant algorithm will be implemented to find the shortest route around all regional centers of Ukraine, i.e. 25 cities, the beginning of the route is in the city of Vinnytsia.

The developed algorithm will work with city coordinates and calculate results in degrees. The ant algorithm was programmed in the MATLAB system [16]. For the task with 25 regional centers of Ukraine, the algorithm without elite ants after 300 iterations found the optimal route with a length of 5216.18 kilometers in 14.16 seconds only in one case out of five. The solution can be improved by simply increasing the number of iterations to 1-2 thousand. The graph of the optimization of the objective function for each iteration is shown in Fig. 2.

The conducted experiments show that the population of solutions never degenerates to one common route for all ants. On the contrary, the algorithm continues to synthesize new, possibly better solutions. So, in any city for an ant there are several promising alternatives for continuing the route. information with the results is given in Table 1.

Table 1 – **Performance results of the ant algorithm with different iterations**

Iterations	Path in degrees	Distance in km	Algorithm operation time
100	50.85	5660.68	6.36 s.
200	49.51	5511.37	9.18 s.
300	46.86	5216.18	14.16 s.

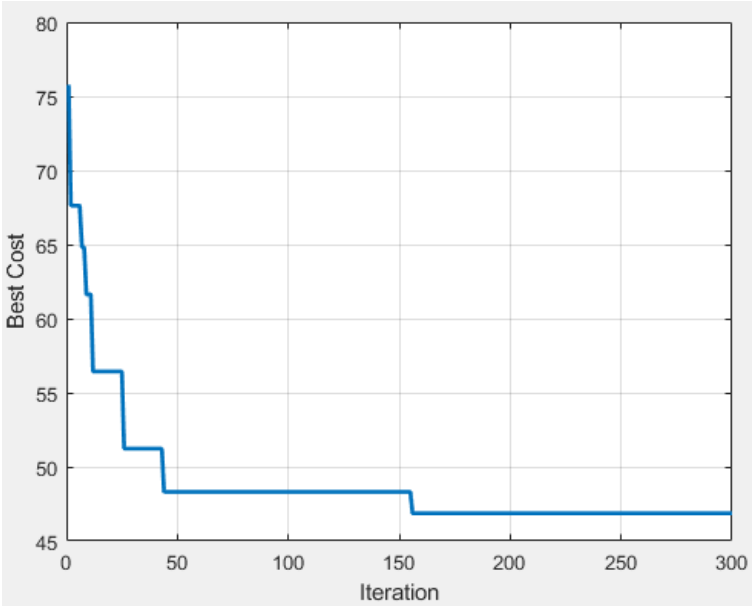

Fig. 2. Optimal current solutions of the ant algorithm

The condition of the problem for the operation of the genetic algorithm remains exactly the same as for the ant colony algorithm. It is necessary to find the shortest route that starts in the starting city and ends in it. The route must pass through all cities only once. It is necessary to go around 25 regional centers of Ukraine.

The selection process is carried out by creating chromosomes that have a low target function, a high probability of being selected, or a high probability value. The selection process used in this algorithm is roulette wheel selection.

After the selection process, the next process is the crossover process. The chromosomes used as parents are chosen randomly, and the number of chromosomes that undergo crossover is affected by the crossover rate parameter.

The mutation process used in this algorithm is reverse mutation. This mutation process is carried out by breaking the chromosomes at two points of intersection, and the subchromosomes recombine in the reverse order.

The number of chromosomes that undergo mutations in the population is determined by the mutation rate parameter.

The structure of chromosomes. Genes are part of a chromosome, where one chromosome consists of several interconnected genes. Chromosomes represent individuals, in other words, chromosomes are the same as individuals.

In the traveling salesman problem, the chromosomes formed contain genes that represent the serial numbers of all existing cities. The number of genes in each chromosome is equal to the number of cities. Each city serial number can appear in a chromosome only once.

The initial initialization of the population is carried out to obtain the initial solution of the genetic algorithm problem. This initialization is performed randomly for as many chromosomes/populations as desired. For example: each of the 25 cities has its own coordinates, you need to make a path from them, each city can be used once. So, the initial population with 3 chromosomes and 25 genes can look like this:

Chromosome[1]=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25];
Chromosome[2]=[23 8 11 20 14 22 12 19 7 1 24 10 2 18 9 17 4 6 21 3 5 25 15 13 16];
Chromosome[3]=[21 8 25 12 10 6 20 17 24 9 19 23 11 5 14 3 2 22 7 16 18 13 4 1 15].

So, the above chromosomes represent several possible ways of traveling through cities by a traveling salesman. This path is the structure of chromosomes.

The implemented genetic algorithm for the task with 25 regional centers of Ukraine after 300 iterations found the optimal route with a length of 5146.57 kilometers in 2.44 seconds only in one case out of five. In this case, the solution did not improve after 300 iterations, so there is no point in putting more. The graph of the optimization of the objective function for each iteration is shown in Fig. 3. The conducted experiments show that the solution is at most 300 iterations away and usually it is the same. The algorithm does not continue to synthesize new, possibly better solutions, provided that they exist at all. Full information with results is given in Table 2.

Table 2 – **Genetic** algorithm results with different iterations

Iterations	Path in degrees	Distance in km	Algorithm operation time
100	46.9	5220.35	1.7 c.
200	46.24	5146.57	2.14 c.
300	46.24	5146.57	2.44 c.

Fig. 4 shows the path found in 300 iterations, although the path found in 200 iterations is also a quasi-optimal solution. So, according to the route of the best results, we have the following sequence (chromosome) of visits to distribution sites (Table 3).

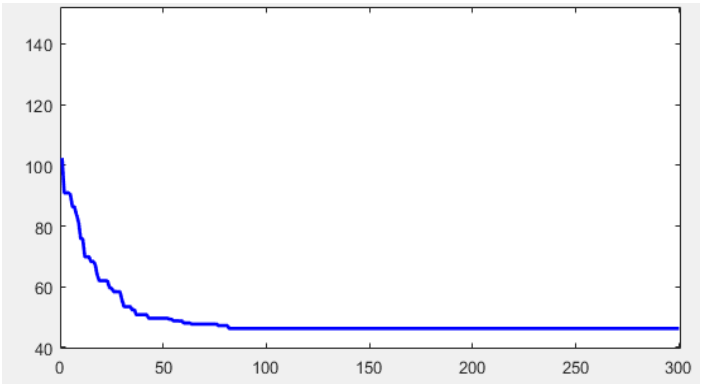

Fig. 3. Best current genetic algorithm solutions

Fig. 4. The optimal path according to the genetic algorithm

Table 3 – **The best of the traveling salesman's routes found**

№	From which city to depart	Which city to go to
1	Vinnitsa	Odesa
2	Odesa	Mykolaiv
3	Mykolaiv	Kherson
4	Kherson	Simferopol
5	Simferopol	Zaporizhzhia
6	Zaporizhzhia	Dnipro
7	Dnipro	Donetsk
8	Donetsk	Luhansk
9	Luhansk	Kharkiv
10	Kharkiv	Sumy
11	Sumy	Poltava
12	Poltava	Kropyvnytskyi
13	Kropyvnytskyi	Cherkasy
14	Cherkasy	Chernihiv
15	Chernihiv	Kyiv
16	Kyiv	Zhytomyr
17	Zhytomyr	Rivne
18	Rivne	Lutsk
19	Lutsk	Lviv
20	Lviv	Uzhhorod
21	Uzhhorod	Ivano-Frankivsk
22	Ivano-Frankivsk	Ternopil
23	Ternopil	Chernivtsi
24	Chernivtsi	Khmelnytskyi
25	Khmelnytskyi	Vinnitsa

In order to draw a conclusion about the effectiveness of the ant colony algorithm and the genetic algorithm, it was decided to compare their work with each other to solve the same task of a traveling salesman to visit all regional centers of Ukraine.

A comparison of the number of iterations, the length of the best route, and the running time of the algorithm is shown in Table 4.

Table 4 – **Comparison of the work of genetic and ant algorithms**

Number of iterat-ions	Ant algorithm		Genetic algorithm	
	Best route length (km)	Algorithm operation time	Best route length (km)	Algorithm operation time
100	5660.68	6.36 c.	5220.35	1.7 c.
200	5511.37	9.18 c.	5146.57	2.14 c.
300	5216.18	14.16 c.	5146.57	2.44 c.

Conclusions

The results of the algorithms show that the ant algorithm takes more time to solve the problem. The ant algorithm can find solutions for 25 cities in 300 iterations.

This is due to the fact that the results of the ant colony algorithm converge longer to a certain value (minimum solution) compared to the genetic algorithm.

In this study, the genetic algorithm can provide a better solution because the genetic algorithm in each run presents a better average optimal solution. If the population is large, this will require many iterations. On average, each run of the genetic algorithm implementation process presents a better solution than the ant one, where in each run the solution graph of the genetic algorithm salesman problem is relatively stable with low solution fluctuations compared to the ant colony algorithm.

Both algorithms can be classified as good and able to present an optimal solution to the traveling salesman problem, which is sufficient for most practical problems.

The simulation results for a 25-city path show the optimal travel route by choosing the best path without intersecting routes.

REFERENCES

1. The Traveling Salesman Problem: A Computational Study, D.L. Applegate, R.E. Bixby, V. Chvátal & W.J. Cook (2006). [Electronic resource]. URL: https://www.math.uwaterloo.ca/tsp/book/contents.html (Date accessed 11/01/2023).
2. Mathematical methods of operations research: a textbook/ E. A. Lavrov, L. P. Perhun, V. V. Shendryk and others. – Sumy: Sumy State University, 2017. – 212 p.
3. Hassan M. H. Mustafa, Ayoub Al-Hamadi, Mohamed Abdulrahman, Shahinaz Mahmoud, Mohammed O Sarhan On Comparative Analogy between Ant Colony Systems and Neural Networks Considering Behavioral Learning Performance// Journal of Computer Sciences and Applications. 2015, vol. 3 No. 3, 79-89.
4. Biological bases of ant colonies - [Electronic resource]. URL: http://posibniki.com.ua/post-prikladni-sistemi-kolektivnogo-intelektu-swarm-intelli-gence (Date of application 10/15/2023).
5. Rukundo, O., Cao, H. Advances on image interpolation based on ant colony algorithm. SpringerPlus 5, 403 (2016) - [Electronic resource]. URL: https://springerplus.springeropen.com/articles/10.1186/s40064-016-2040-9
6. Korte B., Vygen J. Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics) 6th ed., New York, 2018, 455 p.
7. Divya M. A Comparison of Ant Colony Optimization Algorithms Applied to Distribution Network Reconfiguration// International Journal of Engineering Research & Technology, Volume 3, Issue 01, 2016. – pp. 1-4.
8. Hahsler M., Hornik K. TSP – Infrastructure for the Traveling Salesperson Problem// Journal of Statistical Software, December 2007, Vol. 23, Issue 2, 2007. – pp. 1-21.
9. Genetic algorithms. Key concepts and implementation methods. znannya.org : website. URL: http://www.znannya.org/view_ga_general (access date: 10/3/2023).
10. Sathya N., Muthukumaravel A. A review of the Optimization Algorithms on the Traveling Salesman Problem. Indian Journal of Science and Technology, Vol 8(29), DOI: 10.17485/ijst/2015/v8i1/84652, November 2015.
11. Genetic Algorithm Tom V. Mathew Assistant Professor, Department of Civil Engineering, Indian Institute of Technology Bombay, Mumbai-400076.
12. Ivanova E.A. "The possibility of applying genetic algorithms in solving scheduling problems" // Colloquium-journal. 2018. No. 3-1 (14). P. 36-38.
13. Dorigo, M. Ant algorithms and stimergy / M. Dorigo, E. Bonabeau, G. Theraulaz // Future Generation Computer Systems. — 2000. — No. 16. — P. 851-871.
14. Chivilikhin D., Ulyantsev V. Inferring Automata-Based Programs from Specification With Mutation-Based Ant Colony Optimization / Proceedings of the 16th Genetic and Evolutionary Computation Conference companion. - ACM, 2014. - p. 68.
15. Alexandrov A., Sergushichev A., Kazakov S., Tsarev F. Genetic Algorithm for Induction of Finite Automation with Continuous and Discrete Output Actions / Proceedings of the 2011 GECCO Conference Companion on Genetic and Evolutionary Computation. NY. : ACM. 2011, p. 778.
16. Getting Started with MATLAB. Version 6.5. The MathWorks, Inc., 2002.

Received (Надійшла) 06.02.2024
Accepted for publication (Прийнята до друку) 03.04.2024

Порівняльний аналіз застосування евристичних алгоритмів для розв'язання задачі TSP

О. В. Скакаліна, А. М. Капитон

Анотація. Необхідність вирішення задачі комівояжера (TSP) часто виникає при вирішенні практично значущих оптимізаційних задач, таких як задачі в області економіки, логістики в найширшому діапазоні застосувань, в ланцюгах технічних програм. Досить часто специфіка цих задач вимагає отримання рішення, максимально наближеного до точного значення. Але задача TSP є NP-складною, тобто її точний розв'язок можна отримати лише за експоненціальний час. Тому розв'язувати задачу TSP за алгоритмом повного пошуку за наявності великої кількості вершин графа неефективно. Однак існують різноманітні евристичні алгоритми, які дозволяють знайти раціональне рішення цієї задачі з великою кількістю вершин за час, прийнятний для відповідної предметної області. У даній роботі задача комівояжера визначається як задача математичного програмування знаходження найкоротшого шляху руху комівояжера, метою якої є відвідування всіх об'єктів, задіяних у задачі в найкоротший термін і з мінімальними витратами. Відповідні адаптації евристичних алгоритмів, а саме генетичного алгоритму та алгоритму мурашиної колонії, розроблені в середовищі MATLAB. Проведено обчислювальний експеримент на вхідній вибірці, проведено порівняльний аналіз продуктивності двох евристичних алгоритмів і доведено ефективність використання евристичних алгоритмів для розв'язування NP-комплексних задач.

Ключові слова: генетичні алгоритми, NP-комплексна задача, TSP-задача, алгоритм мурашиної колонії, MATLAB.