SURVEY OF ALGORITHMS SEARCH OF SHORTEST WAYS FOR DESIGN OF GEOLOCATION INFORMATION SYSTEMS

[1]Ismailov O.M., [2]Mirzakhalilov C.
[1,2]Tashkent university of Information Technologies
https://doi.org/10.5281/zenodo.7729386

Abstract. *This paper analyzes the existing algorithms for finding the shortest paths in graphs. Based on Dijkstra's algorithm, an application is being developed for a geolocation information system that allows calculating the shortest path between two points. The purpose of the application is to serve as a complement to various services, find the shortest path or the necessary institutions, navigate in an unknown area and get to the destination faster online.*

Keywords: *geolocation systems, Dijkstra's algorithm, routing algorithms, graphs, geolocation information systems.*

Introduction. Since the 1980-90ss, the use of geographic information systems has increased significantly. One of the most popular applications is the search for shortest paths. Computing shortest paths in a geometric environment is a fundamental topic in computational geometry and has applications in many other areas. The problem of processing geometric queries about shortest paths is related to the construction of an efficient data structure for quickly responding to operational queries about shortest paths connecting any two query points in a geometric setting. This problem is a generalization of the well-studied problem of calculating the geometric shortest path connecting only two given points.

To date, there are a huge number of applications in the world that use geolocation systems. Some of the world-famous companies that use location-based apps for optimal functionality include Airbnb, UberEats, Foursquare, or the Apple Maps app that displays a "Allow AppX access to device location" notification. All of these location-based apps can detect objects, buildings, services, and businesses around you. Even dating app Happn uses location technology to match people. Also, geolocation is an integral component of applications focused on logistics, delivery and other service industries. APP solutions use these locations based features in apps like Alfred Ibiza, HYPR and Nuwbii.

The ongoing COVID pandemic situation, the war between Russia and Ukraine, the economic sanctions imposed on each other by countries contribute to a number of difficulties that this industry is facing these days. But despite the obstacles encountered, many companies are looking for options to get their consumers. The analysis of the market for systems using location-based location services (LBS) and real-time location systems (RTLS) is growing rapidly. According to MarketsandMarkets, the market for geolocation services will grow at 17% per year over the next five years. Marketers predict that if the market value in 2021 was about $17.8 billion, by 2025 this figure will increase to $39.2 billion, which means an estimated increase in approximately 50% percentage.

At the same time, there is a significant breakthrough in the use of geolocation systems in medicine. Medical services now rely on convenient services and geolocation platforms (for example, patients can find a doctor nearby with a certified medical application, detect the geolocation of critically ill patients, etc.).

The main directions of research and analysis of algorithms for finding the shortest path. The main task of geolocation systems is related to solving the problem of determining the location and building an effective connection of any two points in the geometric plane. The solution to the problem of finding the shortest path is based on graph theory, which has applications in problems with communications, transport and electronics. This problem plays a central role in the design and analysis of systems. Therefore, it has become the subject of extensive research.

There are various formulations of the shortest path problem:

The problem of the shortest path to a given destination. It is required to find the shortest path to a given destination vertex t that starts at each of the vertices of the graph (except t). By changing the direction of each edge belonging to the graph, this problem can be reduced to the problem of a single initial vertex (in which the search for the shortest path from a given vertex to all the others is carried out).

The problem of the shortest path between a given pair of vertices. It is required to find the shortest path from a given vertex u to a given vertex *V*.

The problem of the shortest path between all pairs of vertices. You need to find the shortest path from every vertex *u* to every vertex *V*. This problem can also be solved using the algorithm for solving the single source vertex problem, but it is usually faster.

In various formulations of the problem, the role of the edge length can be played not only by the lengths themselves, but also by time, cost, expenses, the amount of resources expended (material, financial, fuel and energy, etc.) or other characteristics associated with the passage of each edge. Thus, the problem finds practical application in a large number of areas (informatics, economics, robotics, geography, etc.).

The shortest path problem is a fundamental method of route discovery in computer networks. Routing is one of the important problems in the field of packet-switched computer networks. The use of shortest path algorithms reduces the overall network setup costs. There are various algorithms for solving the shortest path problem. Network routing algorithms define routes from a source host to a destination host for communication. In circuit-switched networks, as in telephone networks, between source and destination nodes. The routing algorithms for such networks generate a route based on the state of the network such as statistics, link weights, and so on. The task of routing algorithms is to specify how the network should respond quickly to changes in network topology.

The routing algorithm can be static or dynamic. With static routing, the path used by the sessions of each origin-destination pair is fixed regardless of traffic conditions. Most large packet networks use dynamic routing, where paths change from time to time in response to congestion. The routing algorithm must change its routes and direct traffic around the congestion point [1-4].

Geometric shortest path problems also play an important role in many practical applications (for example, geographic information systems, operations research, plant layout, robotics, and transportation.

Much research work has been done over the past two decades on solving geometric shortest path problems, especially for planar and three-dimensional (3D) conditions (see review [Mitchell 1996]). To calculate shortest paths in a plane with polygonal obstacles, Hershberger and Suri [Hershberger 1993] developed an optimal *O (n log n)* time algorithm (where *n* is the total number of obstacle vertices). For computing shortest paths in 3D space with polyhedral obstacles, Canny

and Reif [Canny, 1987] showed that the problem is NP-difficult, and several efficient approximation algorithms have been found (eg [Choi, 1994]).

Along with these scientific works, there are theoretical studies of the possibility of using Genetic Algorithms (GA) to solve the general version of the shortest path problem with the highest possible probability of success in a reasonable time (Davis, Lingras, 2003)

It is very gratifying that at present a number of these studies find their practical application in projects for the implementation of geometric algorithms.

However, from the point of view of practical applications, many theoretically efficient geometric algorithms for finding the shortest paths are not without significant drawbacks. One such potential drawback is that these theoretical solutions are often quite environmentally specific. For example, many path algorithms make extensive use of the structures of particular geometries, and hence their performance is highly dependent on certain conditions. Of course, algorithms of this type can be very efficient for some systems, but are of little use for other types of applications and among those using geometric algorithms. Which makes them specific to a particular type of application. However, it is also possible that algorithms that are too specific to the environment may be difficult to extend to the more complex environments that system users are likely to have to deal with in practical applications. For example, in actual route planning for people with cardiovascular disease or patients with mobility difficulties, the shape of the obstacles can be very complex, and the paths can go up and down different landscapes. Therefore, the cost of a path may depend on many factors (eg, distance, shape and types of sections traveled, slopes and directions of various subpaths), and environment-dependent algorithms may have limited practical applicability.

In view of the above, it is considered necessary to apply easy-to-implement, but sufficiently effective algorithms for medical systems of geometric path planning, taking into account the specifics of the terrain (for example, distances, shapes and types of sections traveled, slopes and directions of various subpaths).

Statement of the problem of determining the shortest path from one source. The geometric version of the shortest path problem can be formulated as follows: for a given weighted graph $G = (V, E)$ ($where\ V - \{v_0, \dots, v_n\}, n > 1, E - \{e_0, \dots, v_m\}, m > 1$) and selected vertex *s*, find the shortest weighted path between s and any other vertex in the graph. If we take the weight of each edge equal to 1, then this statement of the problem reduces to finding paths with the smallest length. Edgar Dijkstra (1956) proposed an algorithm for solving a weighted graph version of this problem, provided that the edges do not have negative edge weights. In his algorithm, it does not matter whether the graph is oriented or undirected. The result of the algorithm is a list of the shortest (weighted) distances to each vertex. If the output should include a set of paths to each vertex, then the algorithm can be modified to capture this information.

The idea of the algorithm is to maintain a temporary set of vertices *T* with the property that correctly determines the shortest path from *s* to each vertex in *T*, and iteratively increase this set. This set can be thought of as a known set, because for any vertex in this set, the shortest path from *s* to that vertex is known to be correct. Initially, *T* will contain only the original vertex *s*, since the distance from s to s is 0. At each iteration, a new vertex is added to *T*. When the size of the known set is |*V*|, the algorithm stops.

The algorithm also maintains, for each vertex *v* not in *T*, a temporary shortest distance $d = (v)$ from s. The value of $d = (v)$ is the weight of the shortest path from *s* to *v* that, with the

exception of *v*, passes only through the vertices of T. The algorithm can detect that $d = (v)$ is too large, and it will decrease when this happens. We call $d = (v)$ estimated distance.

After initializing the set *T* and recording the initial calculated distances $d = (v)$ to each node, the algorithm enters the loop. At each iteration of the loop, the vertex $v \in V\text{-}T$ with the minimum $d = (v)$ is added to the vertex *T*, and the distances $d = (v)$ to each vertex $v \in V\text{-}T$ are updated. Since the set *T* only starts at a vertex s in it, and at each iteration a vertex outside of T is added to it, the algorithm must execute exactly$|V - 1|$times, and after$|V - 1|$ iterations of the loop, all vertices are added to *T*, and the algorithm terminates.

In the description of the following algorithm, it is first assumed that the cost of each edge is denoted by the cost function $c(v, w)$, which is defined on each edge *(v,w)* in the set of edges *E*. This function needs to be extended to the function $c'(v, w)$, which is defined on all pairs of vertices $v,\ w \in V$ as follows, even if there is no edge *(v,w)* in *E*. This function is defined as follows:

$$c'(v,w) = \begin{cases} 0 & if\ v = w \\ c(v,w) & if\ v \neq w \wedge (v,w) \in E \\ \infty & if\ v \neq w \wedge (v,w) \notin E \end{cases} .$$

In other words, the cost of an edge *(v,w)* is infinite if that edge does not exist. In a real implementation, a special value may denote the absence of an edge. The cost of all other edges is simply the weight of the edge itself.

Algorithm. For the sake of clarity, on the example of the classical algorithm of Edgar Dijkstra (Edgar Dijkstra, 1956), we will study in detail the functioning of algorithms for finding the shortest path.

Let a graph be given:

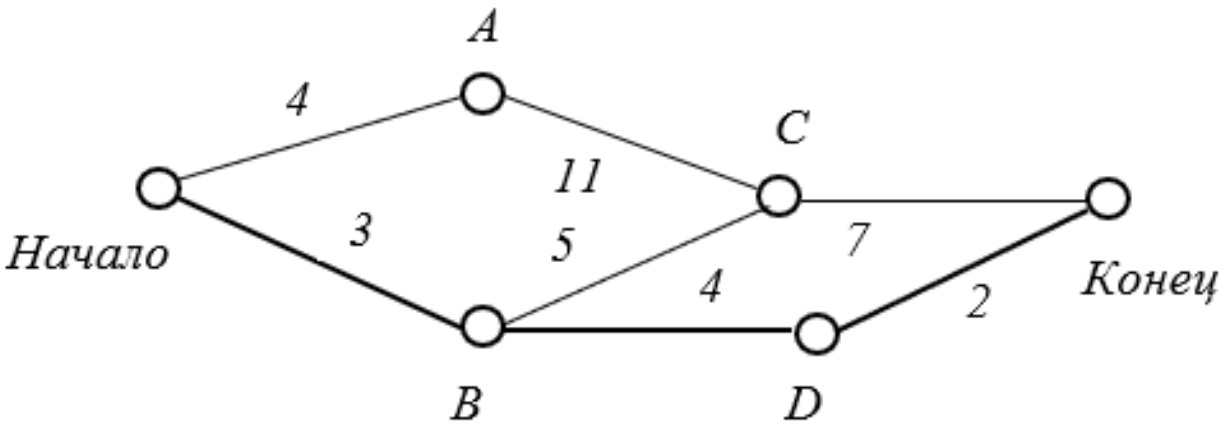

Fig.1. An example of an undirected graph.

Let's study how this Dijkstra algorithm works with graphs.

Each edge $d(v)$ is assigned *T* - time $(where\ T - \{t_1, \dots, t_y\}, y > 1)$ of travel in minutes. Dijksta's algorithm is used to find a path from the start point to the end point in the shortest time *T*.

Step 1. Consider any vertex v as the original one. In the graph under consideration, “Start” as will be denoted as the initial vertex, and the distance to all other vertices is equal to ∞-infinity.

Initially, the distances between the path from "Beginning" to "End" are unknown. To determine the path, you first need to find out the v vertices along which the path is laid, which are

directly related to the initial vertex. From Figure 6. It can be seen that the two vertices "A" and "B" are directly connected to the "Start" vertex.

The distance between peaks "A" and "B" can be calculated using the following formula:

$$d(v_{start}, v_B) = d(v_{start}) + c(v,w) < d(v_{start}) = (0+4) < \infty = 4 < \infty. \quad (1)$$

As $4 < \infty$ updated $d(v)$ from ∞ to 4.

Step 2. In the next step, it calculates how long it takes to get to the next neighbor, the "Start" vertex.

$$d(v_{start}, v_B) = d(v_{start}) + c(v,w) < d(v_{start}) = (0+3) < \infty = 3 < \infty.$$

Therefore, the value $d(v_B)$=3. Now let's replace the value of ∞-infinity indicated earlier as the path distance to the vertices A and B by the value 4 and 3, respectively. So, we have found the shortest path from "Start" to vertex A and from "Start" to vertex B. Now let's compare all d(v) vertices except vertex $d(v_{start})$. Since vertex $d(v_B)$ has the smallest value $d(v_B) < d(v_A)$, therefore, vertex $d(v_B)$ is selected.

Step 3. Since the vertex $d(v_B)$, is selected, we consider the path to the nearest neighbors of the vertex $d(v_B)$, to $d(v_c)$, and $d(v_d)$, respectively. The path from $d(v_B)$, to $d(v_{start})$ will not be considered, since the length of the path *c(v,w)* was calculated at step 2 of the algorithm.

In accordance with formula (1), the length of the path from the vertices $d(v_B)$ to $d(v_C)$ and $d(v_D)$ calculated as follow:

$$d(v_B, v_C) = d(v_B) + c(v,w) < d(v_B) = (3+5) < \infty = 8 < \infty,$$
$$d(v_B, v_D) = d(v_B) + c(v,w) < d(v_B) = (3+4) < \infty = 7 < \infty.$$

Since $8 < \infty$ и $7 < \infty$, let's replace the value of ∞-infinity denoted earlier as the path distance to the vertices C and D by the value 8 and 7, respectively. We match the vertices $d(v_C) < d(v_D)$, therefore, at the next step, we are looking for the path to the nearest neighbors of the vertex $d(v_D)$.

Step 4. Calculate the distance path to the nearest neighbors of the vertex $d(v_D)$. The length of the path from the top $d(v_D)$ to the "End" in accordance with the formula (1):

$$d(v_D, v_{\text{Конец}}) = d(v_D) + c(v,w) < d(v_D) = (7+2) < \infty = 9 < \infty,$$

will be $9 < \infty$.

As a result of the algorithm, the shortest path from the “Start” to the “End” of the graph is calculated, passing through the vertices $d(v_{start})$, $d(v_B)$, $d(v_D)$ and $d(v_{end})$ ($fig.2.$):

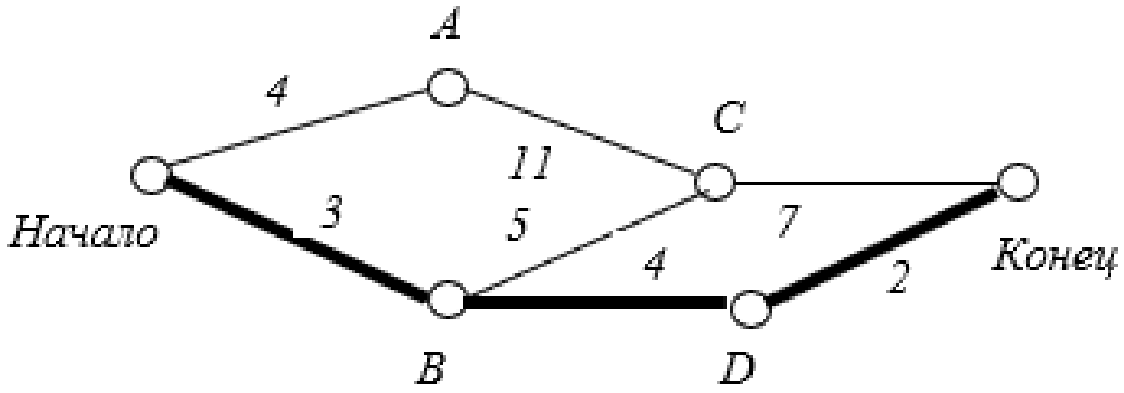

Fig.2. The result of Dijkstra's algorithm with an undirected graph.

The time complexity of Dijkstra's algorithm is $O(V)^2$, but with a minimum priority queue it drops to $O(V + E \log V)$.

The practical implementation of the Dijkstra algorithm for geolocation information systems can be done in C++, Java or Phyton programming language. However, given the fact that geolocation systems must function quickly and reliably on the Internet, we implement the algorithm in the Phyton programming language (Fig. 3.). In recent years, Python has become one of the popular programming languages all over the world. Developers around the world rely on Python for reliable, efficient, and intuitive programming solutions.

In addition, Python is a good option for web development (backend), is the leading language for data analysis and machine learning. In terms of simplicity, Python is much easier to use and has a great support system when it comes to AI and ML frameworks.

Pulumi (multi-language, multi-cloud open source development platform) has a long history of technical support for the Python programming language, Cloud (Programming the Cloud with Python) and Internet of Things (IoT) technologies.

```python
import sys

class Graph(object):
    def __init__(self, nodes, init_graph):
        self.nodes = nodes
        self.graph = self.construct_graph(nodes, init_graph)

    def construct_graph(self, nodes, init_graph):
        '''
        This method makes sure that the graph is symmetrical. In other words, if there
        '''
        graph = {}
        for node in nodes:
            graph[node] = {}

        graph.update(init_graph)

        for node, edges in graph.items():
            for adjacent_node, value in edges.items():
                if graph[adjacent_node].get(node, False) == False:
                    graph[adjacent_node][node] = value

        return graph

    def get_nodes(self):
        "Returns the nodes of the graph."
        return self.nodes

    def get_outgoing_edges(self, node):
        "Returns the neighbors of a node."
        connections = []
        for out_node in self.nodes:
            if self.graph[node].get(out_node, False) != False:
                connections.append(out_node)
        return connections

    def value(self, node1, node2):
        "Returns the value of an edge between two nodes."
        return self.graph[node1][node2]
```

Fig. 3. Program code of Dijkstra's algorithm.

Approbation of Dijkstra's algorithm. To demonstrate the operation of Dijkstra's algorithm, let's make a visual problem on the example of routing.

We set two points on the map, where we denote the starting point in green, and the end point in red. The blue vertex points through which the route must pass. Next, we will build a graph based on them, taking the real distances and paths between all the vertices (Fig. 4.).

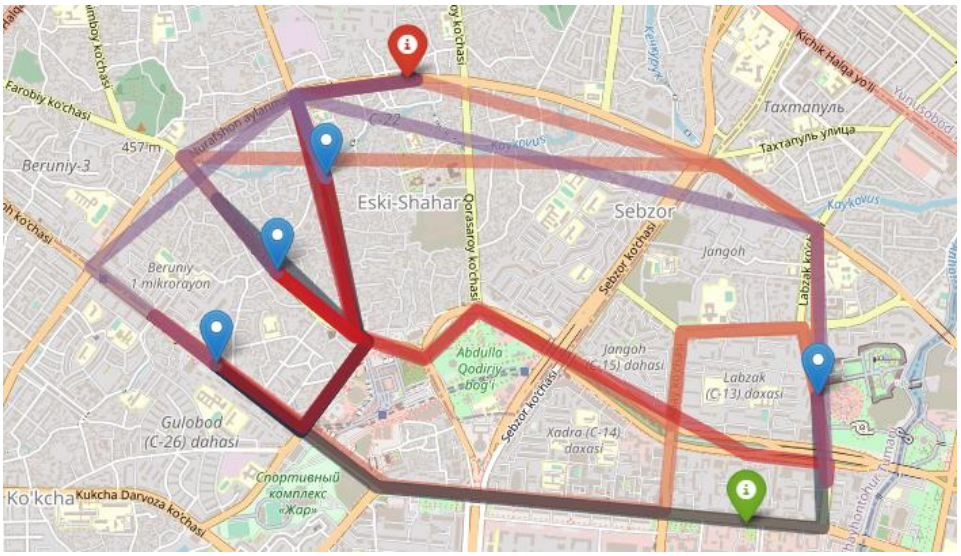

Fig. 4. The result of establishing a route of two points through the specified vertices.

In the next case, we create a directed graph. We start the algorithm that will go through all the vertices and prioritize the weights on the minimum values. At the end of the path normalization function, we get the following result:

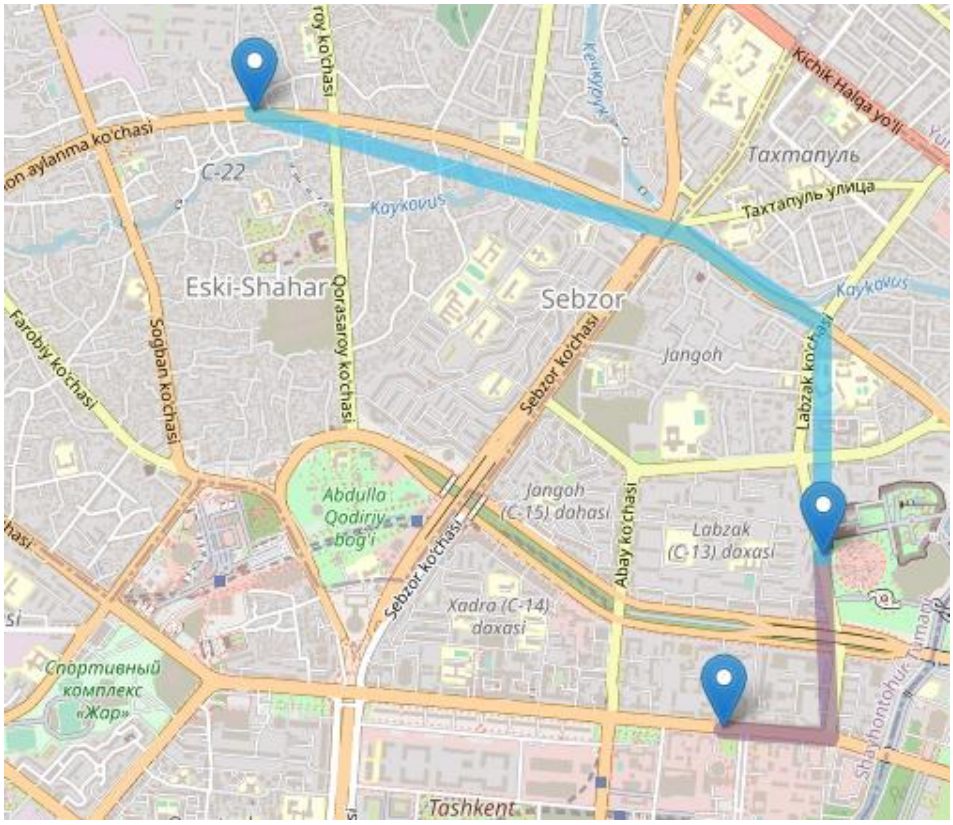

Fig 5. The result of the search for an algorithm with a priority on the minimum values.

This route will be the most optimal of all passing through the given peaks.

Conclusion. Research has shown that despite the many algorithms available for finding the shortest paths for designing geolocation information systems, more efforts should be made to develop more universal algorithms and software. For example, it would be very useful if some

tools were available for computer-aided design of computer programs for geometric shortest path problems. Users can use the "Geometric Problem Specification Language" to describe the problems they want to solve. The toolbox will then process this task specification program from the user and generate target programs in the desired programming language for the required geometric calculations (eg. Lex and Yacc to create compiler components). Such tools will greatly increase the accessibility of geometric algorithms to many users. The basis for developing such high-level design tools for geometric computing is to make geometric algorithms available not only on paper but also in software.

It is expected that further research on the development of new algorithms and software for geometric path planning problems will have a great impact on the theoretical study and practical application of computational geometry.

REFERENCES

1. Paige R, Kruskal C. Parallel algorithms for shorest path problem. IEEE Transactions on Computer, vol.C34, 1985, pp.14-20.
2. Muhammad Nazam Mqbool. Shortest Path Algorithms: Comparative Study and Analysis. URL.:https://www.academia.edu/40605005/Shortest_Path_Algorithms_Comparative_Study_and_Analysis.
3. Selim, H., Zhan, J. Towards shortest path identification on large networks. *J Big Data* 3, 10 (2016). https://doi.org/10.1186/s40537-016-0042-7.
4. Joel Minot. Telecommunication network and searching arrangement for finding the path of least cost. URL:https://patents.google.com/patent/US5481604A/en.
5. Aidin Parsakhoo, Mohammad Jajouzadeh. Determining an optimal path for forest road construction using Dijkstra's algorithm. Journal of Forest Science, 62((No. 6)), 2016: p.264–268.
6. Mogens Graf Plessen, Alberto Bemporad. Shortest Path Computations under Trajectory Constraints for Ground Vehicles within Agricultural Fields. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016, p. 1733- 1738.
7. Agam Mathur , Mayuresh Jakhotia , Anish Lavalekar , Nikita Magar. Shortest Path Finding Algorithms for Real Road Network. Computer Department, VIIT, Pune University, IJLTEMAS Volume III, Issue X, October 2014, p.53-56.
8. Pooja Singal, R.S.Chhillar. Dijkstra Shortest Path Algorithm using Global Positioning System, *International Journal of Computer Applications (0975 – 8887) Volume 101– No.6, September 2014, p13-18.*
9. Ke Deng, Xiaofang Zhou. Expansion-Based Algorithms for Finding Single Pair Shortest Path on Surface. Web and Wireless Geographical Information Systems, 2005, Volume 3428 ISBN : 978-3-540-26004-2.
10. Hofmann, W.B., Legat, K. & Weiser,M. (2003). *Principles of Positioning and Guidance*. Springer Vienna, Austria. 427p.
11. Hamid Ali Abed Alasadi[1,a,*], Mohammed Talib Aziz[2,b], Mohammed Dhiya[2,c] and Ahmed Abdulmajed. A Network Analysis for Finding the Shortest Path in Hospital Information System with GIS and GPS Journal of Network Computing and Applications. Clausius Scientific Press, Canada, Vol 5, Issue 1, 2020: 10-22.

12. YOON MI KIM. Dijkstra Algorithm: Key to Finding the Shortest Path, Google Map to Waze. Jun 13, 2019. URL:https://medium.com/@yk392/dijkstra-algorithm-key-to-finding-the-shortest-path-google-map-to-waze-56ff3d9f92f0.
13. Eshmuradov D. E., Elmuradov T. D., Turaeva N. M. Methods of Presentation of Aeronautical Information //Design Engineering. – 2021. – С. 12173-12181.
14. Eshmuradov D. E. et al. The Need To Use Geographic Information Systems In Air Traffic Control //Turkish Journal of Computer and Mathematics Education (TURCOMAT). – 2021. – Т. 12. – №. 7. – С. 1972-1976.