2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) | 978-1-6654-0476-1/20/$31.00 ©2021 IEEE | DOI: 10.1109/ElConRus51938.2021.9396473

Research of Modern Routing Systems

Evgeniy Belikov
Department of Computer Systems and Technologies
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russian Federation
evgeniy.belikov.02@mail.ru

Polina Afonichkina
Department of Computer Systems and Technologies
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russian Federation
polina.afoni4k1na@gmail.com

***Abstract*— This research is devoted to the study and analysis of modern methods of building routes. The article describes the data provided by the Open street map, google maps API and ways of working with them, as well as various routing algorithms used in Graphhopper, Yandex.Maps, and other routing services. In the course of the study, based on the material studied about working with maps, graphs, and routing services, a web application was created. It is aimed at solving the problem of optimizing the route for the needs of the user.**

Keywords - Open street map; Mapbox Valhalla API; graphs; geocoding; mapping algorithms

I. Introduction

Today the cities grow quicker and quicker, infrastructure in them becomes complicated therefore it becomes harder and harder to be guided without the aid of interactive maps. However, sometimes even on them is expensive on time to look for at the same time useful and fast routes. One of solutions of this problem is presented in this article. In the course of application programming various articles were considered:

Dijkstra's algorithm[6]

This algorithm is used for creation of routes on columns of maps of the real world. From initial top with an initial weight of 0 the weight of a way to all adjacent (new) tops is considered. That gets out of new tops, the weight of a way to which the smallest. Now it is the weight of this new top, and the previous point in way to it is the point of the beginning of a way is considered. Then this top is accepted to initial, and from it the distance to all tops, adjacent to it, is looked for, and weight to them pays off with addition of weight of initial (new) top. Thus, for each top the total minimum weight of a way to it from the initial point and the previous top in this way remains. As soon as the final point of a route is reached and processed, it is possible to restore the shortest route to it.

A. *A* algorithm*[7]

This algorithm mostly is applied in computer games. Its main difference from Dijkstra's algorithm is the absence of the initial count. All weight in this algorithm are set as the work of the constant price of movement of coefficient of cost of transition to a new point. This coefficient, in turn, pays off depending on distance to a final point, own initial coefficient (for example, a relief), and a method of transition to this point (for example, on diagonal or on a straight line). Thus weight pay off for all adjacent with given tops, and that for which weight will be minimum gets out of them. Further again pay off the weight of all adjacent with new points (except for what already visited), and iteration repeats, until the final point won't be considered.

B. *Floyd's algorithm*[8]

This algorithm finds the shortest way from each knot to everyone. 2 matrices on the basis of a contiguity matrix are created: matrix of scales and matrix of history. After the first and only start of this algorithm these matrices are filled in such a way that they contain information on the shortest ways from each top in everyone.

C. *Genetic algorithm*[1]

This algorithm is applied in cases when it isn't possible to find the only right decision because of task volume therefore it is necessary to find rather good. A striking example of this class of tasks is the traveling salesman problem. It consists in creation of the shortest route for the closed round of several points when the sequence of a round of these points isn't essential. The algorithm randomly makes several options of the sequences of tops between two set points of the beginning and end. Then randomly for several sequences the mutation or crossing becomes. Thus, the total amount of the sequences increases. After that all sequences are placed in ascending order of the total length of a way, and the last of them leave so that the total amount of the sequences is equalized with initial. Iteration repeats, rather good solution won't be found yet.

D. *The Comparison of Three Algorithms in Shortest Path Issue*[2]

The comparative analysis of a Dijkstra's algorithm, Floyd's algorithm, and Bellman-Ford's algorithm is given in this article. High-speed parameters of algorithms, and also the fields of their application are considered by the author. From this article follows that Dijkstra's algorithm, despite in applicability in columns with negative scales of edges, is better than others is suitable for the solution of a problem of search of the shortest way on the road count at the expense of the speed.

E. *Hierarchical Shortest Pathfinding applied to Route-Planning for Wheelchair Users*[3]

In this article modern modifications of algorithms A* and Dijkstra's by means of creation of hierarchies and reductions on columns are considered. Such decision allows reducing considerably time of search of the shortest way despite sharply increased scale of amount of the processed data.

F. *Introduction to graph theory* [4]

In this book the bases of the theory of counts necessary for understanding above the mentioned algorithms are described.

G. *A Neural Network for Shortest Path Computation*[5]

In this article methods of search of the shortest way in the column through neural network, and also prospects of this technology are described.

II. Research Materials And Methods

From all considered algorithms for routing on the card Dijkstra's algorithm and its modern modifications is optimum therefore during creation of the project the routing services using similar algorithms in the basis were considered.

Way of selection of POI:

Any modern card is set of three main objects: points, edge and ground. Each of these objects has a certain set of the tags placed on the hierarchy developed under them and indicating any given features of this object. These tags are used for identification of necessary points of interest for the user.

A large number of trial and error methods of interesting routes, here some of them was considered:

1). To change weight on the route graph under preferences of the user by using a neural network. Then build routes on it using Dijkstra's algorithm.

The main advantage of this method is the quality and individuality of the data provided to the user. However, it is problematic to implement the conclusion of alternative shortest routes. Also, this form requires a large amount of memory and high performance from the server.

2). To construct a straight line between points of the beginning and end of a route. Then in the sector which radius is equal to this straight line and the corner of a turn is set by the author and doesn't change, to find by means of Overpass API of coordinate of all POI entering this sector. After that through three of these tops to build broken lines from the beginning of a route up to the end and to look for the shortest of all possible options. Then through the received points to construct a route by means of Mapbox Valhalla API and to remove to the user an interesting route along with the shortest.

This method is also simple in realization, doesn't demand creation of the special count. However, it limits the maximum quantity of interesting points on a route and wastes more time for creation of a route.

3). To place indexes of necessary POI in a matrix of distances. After that to construct the shortest route between the set points. Then to include in it tops, adjacent to tops of the shortest way, with the necessary indexes, so far total length doesn't exceed the set percent from the shortest route. As soon as the "interesting" route approaches the set limit, to send the list of tops of a "interesting" route to API of routing service (For example, Graphhopper) and to remove to the user the turned-out route along with the shortest.

This method is simpler in realization, demands smaller memory sizes and computing power. Receiving and processing of a necessary matrix of distances remains the only problem. this way concedes above-mentioned in identity of the services provided to the user.

After the analysis of all listed ways the decision to unite ways 2 and 3 was made. Thus, the final algorithm means creation of the direct piece connecting the end and the beginning of a route. Then in a circle which radius is equal to a half of length of a piece and the center is his middle, all points decide on the tags set by the user. After that by means of Mapbox Valhalla API the matrix of distances for a start and the end of a route and also for points with the necessary tags is under construction. The received matrix is processed by means of the following algorithm (fig.1):

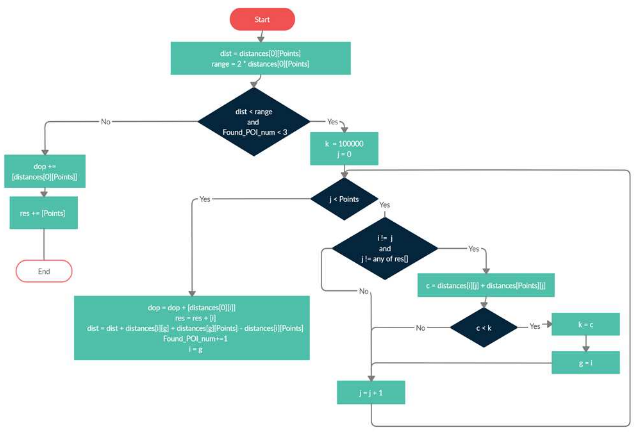


Fig.1. POI selecting algorithm

As a result there are 5 points through which it is necessary to lay a route.

The following step is sorting of points so that the route was built through them consistently. That is the point is closer to start, the earlier the route has to pass through it (fig.2):

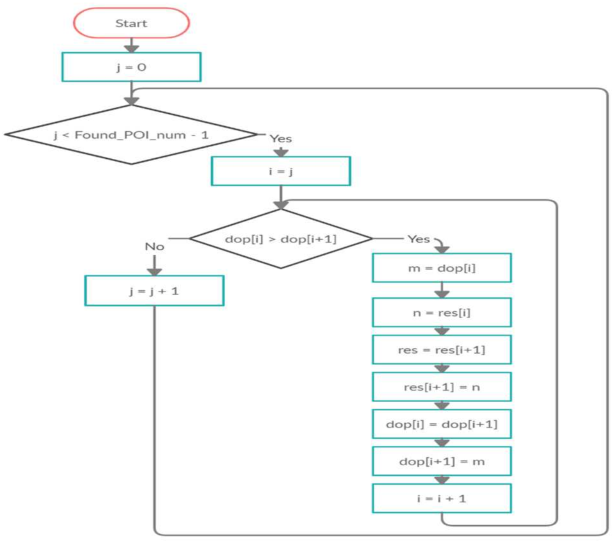


Fig.2. POI sorting algorithm

Coordinates of these points move in inquiry of Mapbox Valhalla API and then the returned geometry is brought to the card for the user.

III. Results

This approach allows to use service worldwide. Selection of tags is carried out irrespective of the region in which the user wants to construct a route.

As a result of an experiment, data on existence of necessary POI in a search radius for this route were obtained and processed (fig.3, fig.4):

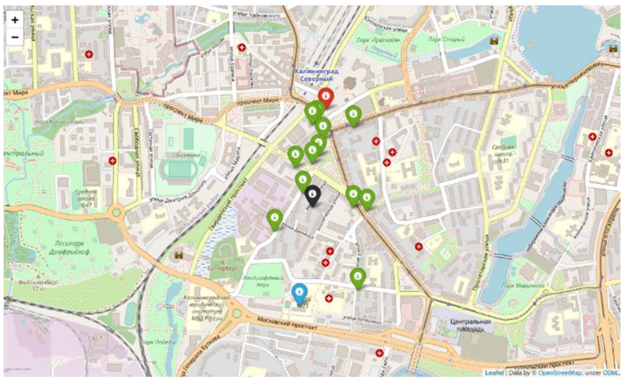


Fig.3. POI selection in Kaliningrad

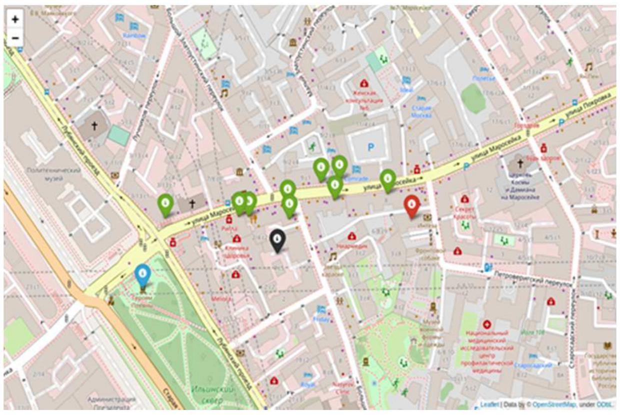
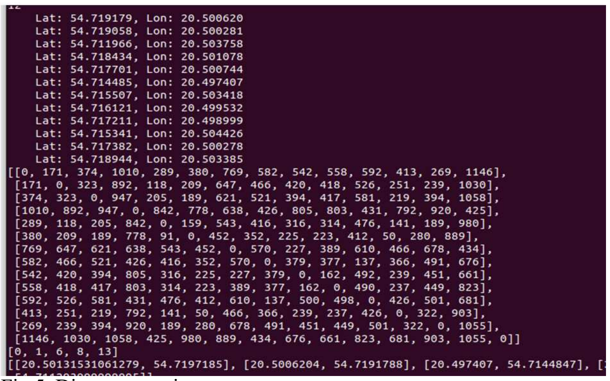


Fig.4. POI selection in Moscow

On the submitted cards (fig.3, fig.4) red and blue markers noted points of the beginning and the end of a route. The black marker displays the center of a circle in which search in tags is run. Green markers marked all found interest points.

Further the received points are compared in a matrix of distances which is processed by the above-mentioned code (fig.5):



```
    Lat: 54.719179, Lon: 20.500620
    Lat: 54.719058, Lon: 20.500281
    Lat: 54.711966, Lon: 20.503758
    Lat: 54.718434, Lon: 20.501078
    Lat: 54.717701, Lon: 20.500744
    Lat: 54.714485, Lon: 20.497407
    Lat: 54.715507, Lon: 20.503418
    Lat: 54.716121, Lon: 20.499532
    Lat: 54.717211, Lon: 20.498999
    Lat: 54.715341, Lon: 20.504426
    Lat: 54.717382, Lon: 20.500278
    Lat: 54.718944, Lon: 20.503385
[[0, 171, 374, 1010, 289, 380, 769, 582, 542, 558, 592, 413, 269, 1146],
 [171, 0, 323, 892, 118, 209, 647, 466, 420, 418, 526, 251, 239, 1030],
 [374, 323, 0, 947, 205, 189, 621, 521, 394, 417, 581, 219, 394, 1058],
 [1010, 892, 947, 0, 842, 778, 638, 426, 805, 803, 431, 792, 920, 425],
 [289, 118, 205, 842, 0, 159, 543, 416, 316, 314, 476, 141, 189, 980],
 [380, 209, 189, 778, 91, 0, 452, 352, 225, 223, 412, 50, 280, 889],
 [769, 647, 621, 638, 543, 452, 0, 570, 227, 389, 610, 466, 678, 434],
 [582, 466, 521, 426, 416, 352, 570, 0, 379, 377, 137, 366, 491, 676],
 [542, 420, 394, 805, 316, 225, 227, 379, 0, 162, 492, 239, 451, 661],
 [558, 418, 417, 803, 314, 223, 389, 377, 162, 0, 490, 237, 449, 823],
 [592, 526, 581, 431, 476, 412, 610, 137, 500, 498, 0, 426, 501, 681],
 [413, 251, 219, 792, 141, 50, 466, 366, 239, 237, 426, 0, 322, 903],
 [269, 239, 394, 920, 189, 280, 678, 491, 451, 449, 501, 322, 0, 1055],
 [1146, 1030, 1058, 425, 980, 889, 434, 676, 661, 823, 681, 903, 1055, 0]]
[0, 1, 6, 8, 13]
[[20.50131531061279, 54.7197185], [20.5006204, 54.7191788], [20.497407, 54.7144847], [
```

Fig.5. Distance matrix

For this matrix the sequence of numbers of points is result [0, 1, 6, 8, 13]. Then numbers are compared with initial coordinates which presented in the top part of a screenshot. The route is built using the obtained coordinates (fig.6, fig.7).

The route constructed by Mapbox Valhalla API (fig.6):

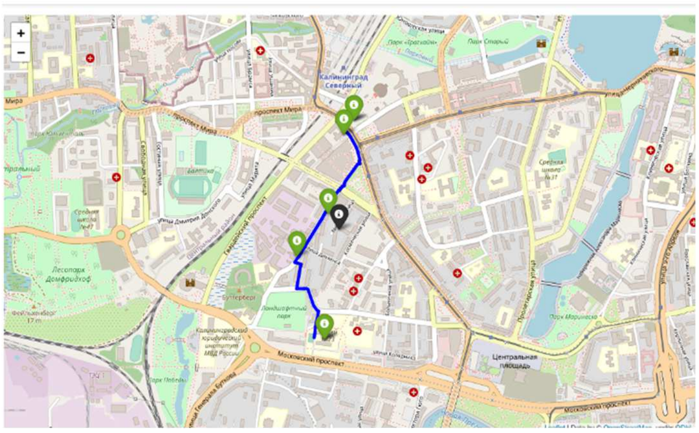
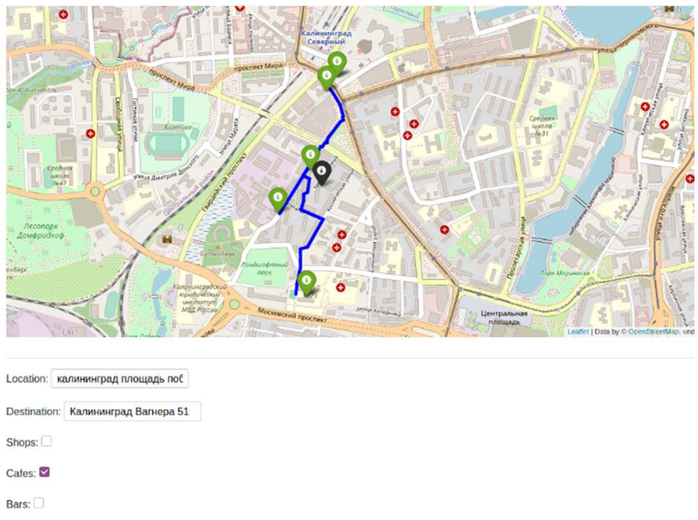


Fig.6. The Mapbox Valhalla route

The route constructed by Graphhopper API:



Location: калининград площадь поб

Destination: Калининград Вагнера 51

Shops: ☐

Cafes: ☑

Bars: ☐

Fig.7. The graphhopper route

At this stage it becomes obvious that Mapbox Valhalla is more suitable for a pedestrian routing in Russia.

IV. Discussion And Conclusions

As a result of research of the principles of work of modern route services on their basis the web application was created. Also, the comparative analysis of accuracy of the returned routes during which for implementation of the application Mapbox Valhalla API was chosen was carried out.

It should be noted that from all possible types of geometry on the card this way is capable to select only points that can affect its efficiency and opportunities. Thus, at further development of a research it is necessary to find a way of the solution of an above-mentioned problem. The problem of optimization of routes doesn't lose the relevance. On the contrary, together with growth of the traffic necessary for a covering of needs of modern people, the need for fast search of optimum routes not only on distance, but also on time increases. There is a wish to allocate the option of the Dijkstra's algorithm which is widely used for optimization of automobile routes accelerated by means of levels hierarchies and preservation of some already constructed optimum routes. Thus, the research can continue the development in the sphere of new genetic algorithms, auxiliary counts for routing, and other modern methods of an optimization of routes. It is necessary to pay attention not only to pedestrian bilateral columns, but also to the directed columns of highways.

REFERENCES

[1] Y. Sharma, S. C. Saini, and M. Bhandhari, "Comparison of Dijkstra ' s Shortest Path Algorithm with Genetic Algorithm for Static and Dynamic Routing Network," Int. J. Electron. Comput. Sci. Eng., 2012.

[2] X. Z. Wang, "The Comparison of Three Algorithms in Shortest Path Issue," 2018, doi: 10.1088/1742-6596/1087/2/022011.

[3] S. Yang and A. K. Mackworth, "Hierarchical shortest pathfinding applied to route-planning for wheelchair users," 2007, doi: 10.1007/978-3-540-72665-4_46.

[4] V. I. Voloshin, Introduction to graph theory. 2009.

[5] F. Araújo, B. Ribeiro, and L. Rodrigues, "A neural network for shortest path computation," IEEE Trans. Neural Networks, 2001, doi: 10.1109/72.950136.

[6] CP-Algorithms. Available at: https://cp-algorithms.com/graph/dijkstra.html

[7] Pathfinding: A* Search Algorithm. Available at: https://medium.com/@karlmatthes/pathfinding-a-search-algorithm-d77400c89888

[8] Floyd Warshall Algorithm. Available at: https://medium.com/dev-genius/floyd-warshall-algorithm-f004a01ae40e