VERSITA
Information Technology and Management Science
doi: 10.2478/itms-2013-0012
2013 / 16

Suitability Analysis of Routing Algorithms for Web-based Transportation Planning

Davis Jaunzems[1], Arnis Lektauers[2], [1-2] *Riga Technical University*

***Abstract* – The routing of vehicles in complex urban or regional systems is a task that needs to be solved in numerous transportation-oriented applications. This paper describes the process and approaches used for routing algorithm analysis emphasizing transportation planning in interactive web-based planning, analysis and simulation solutions. The objectives of the presented research are to examine existing routing algorithms used for finding the shortest path between the nodes in transport networks and to analyse the implementation possibilities and efficiency of routing algorithms in web-based geographical information systems.**

***Keywords* – Routing algorithms, transportation planning, web-based geographical information systems**

I. Introduction

The development progress of private and public transportation networks as well as infrastructure leads to an increase in demand for efficient methods and algorithms to solve the actual tasks and problems of transportation planning and control. Transportation systems are increasingly complex systems incorporating diverse travel modes and services; therefore, the need to integrate and efficiently operate these systems poses a challenge to planners and operators [1]. By using new technologies and applications, as well as development assistance and evaluation tools prior to field implementation in transportation systems, it is possible to find a solution to this complex problem area.

One of the actual tasks that needs to be solved in numerous transportation-oriented applications is the routing of vehicles or persons. The basic concept of routing algorithms is to model the specific problem in a suitable graph and to compute the shortest path to solve it. While it is simple to come up with an algorithm that just solves the problem, it is much more difficult to engineer an efficient routing algorithm.

This paper describes the process and approaches used for routing algorithm analysis emphasizing transportation planning in interactive web-based planning, analysis and simulation solutions. The objectives of the presented research are to examine existing routing algorithms used for finding the shortest path between the nodes in transport networks and to analyse the implementation possibilities and efficiency of routing algorithms in web-based geographical information systems.

II. Routing Algorithms for Transportation Planning

A. Classification of Routing Algorithms

Based on the graph theory, multiple routing algorithms can be used for solving the shortest path problem. This paper focuses on four well-known methods classified as the shortest path algorithms [2]:

- Dijkstra;
- Bellman–Ford;
- Floyd–Warshall;
- A* (A star).

The existing shortest path algorithms have different structure, functionality and working principles depending on the used graph types, edge directions, edge weights, graph density and other parameters. Some algorithm functionalities are overlapping in the case when they are derived from the same base algorithm. For example, the A* algorithm is an extended version of Dijkstra, adding possibility to use heuristics to determine the order of node searching [3]. The Bellman–Ford [4] and Floyd–Warshall [5], [6] algorithms are used to compute all possible paths in the graph. This functionality can be used in the so-called graph pre-processing. However, the searching of all possible paths requires more computational resources than single path searching does. This aspect is considered when algorithms are compared in the analysis carried out in this study.

The research presented in this paper focuses on the analysis of the necessary computational resources – time and memory – needed for algorithms to find the shortest path in diverse contextual situations. The theoretical behaviour of each algorithm can be calculated using the big *O* notation [7]. This gives only the worst- and best-case scenario results using graph size measurements. In transportation planning, the route properties between sources and destinations can change multiple times as the route passes through dense regions like cities and through highways that are considered scattered regions. Therefore, the analysis results of routing algorithms using real-life geographical data may significantly differ from the theoretically estimated results.

B. Hierarchical Routing Algorithms

Transport networks can be modelled as graphs consisting of nodes and edges connecting the nodes [8]. Large scale routing is a complex task that requires considerable amount of computation using graph data structures. A specific graph functionality is required to achieve effective routing, for example, a procedure is there needed that searches the graph for connected nodes in relation to a single entry node. As the graph density grows, the number of edges between nodes increases and, therefore, the shortest path search requires more computational resources. The following list gives an overview of two possible graph data processing and storage approaches:

- A graph is pre-processed and later used in a real-time application; usually the information is stored in the

random access memory (RAM). In such a case, routing is fast but the data processing capabilities are limited by the available system resources.

- A set of nodes and vertices are stored in a graph-supported database. Large size graph data can be easily stored and retrieved. However, routing is slower and is limited by the existing database system resources and restrictions.

Additional research of hierarchical information storage methods was conducted by the authors of the paper to find performance improvements for a routing system. Two routing capable hierarchical graph methods were analysed – Transit Node Routing [9] and Contraction Hierarchies [10].

Transit Node Routing is based on the method of Highway Hierarchies. In the road network, graph routes that lead to other places are there identified. These routes act as connection points for less active roads. By using such a route separation method, multiple road layers can be identified. The main highways are located in a top layer, which provides routes between cities. Going deeper in the hierarchy, smaller routes are stored that provide information about city streets. This method greatly decreases the graph size and reduces the amount of computation needed for finding the shortest path in large-scale graphs.

As an alternative to the Transit Node Routing, there is the Hierarchical Contraction method. This method contracts graph nodes together, therefore, reducing the total node and edge count. As a result of contraction, shortcuts between nodes are created, but the calculated shortest path value is still available. The increase in overall system performance can be observed when the shortest path algorithms are executed on contracted graphs.

To achieve a lower level of computation time and resource usage when searching for the shortest paths, in graph-based data structures the data pre-processing is required. One of the shortcomings of graph pre-processing is the necessity to use static geographical data. The existing systems are not capable of working with dynamic data. However, to support computationally effective transportation planning it is required that routing systems provide the latest actual road information.

III. Suitability Analysis of Routing Algorithms

The theoretical analysis of routing algorithm execution times does not fully provide information about the algorithm suitability for transportation planning using real-life data. While building graphs using cartographic information, several special use cases can be examined. In cities and around them, graph vertices are located relatively dense and there are multiple edges between two single nodes. On the other hand, rural area vertices are scattered and fewer edges can be there identified. The observations for suitability analysis were made by the authors of the paper by constructing a road graph using publicly available data of Latvia's territory.

C. Analysis Model

This paper focuses on the routing algorithms for application in web-based cartographic transportation planning systems. Since such systems require the processing of a large amount of data, the execution time is one of the most essential indicators. Execution time provides information about the time required for an algorithm to find the shortest path. Time parameter can also be used to measure the complexity of a graph structure – the more nodes and edges exist in a graph, the more time is required for its construction. For obtaining valid results, the algorithm execution and graph construction are separated processes.

Such routing algorithms as Floyd–Warshall calculate all the possible routes in a graph and store this information in a matrix format. The Dijkstra algorithm calculates the single shortest path between two nodes. To compare routing algorithms, it is necessary that all possible routes are calculated with each of the analysed algorithms. The information about the shortest paths is stored in a unified data structure – matrix. Calculation of all shortest paths provides more accurate analysis information about algorithm suitability for pre-processing operations.

Routing algorithms use edge values (costs) between vertices to determine the shortest path. In this analysis, edge values represent the distance between nodes calculated using the Haversine equation [7]:

$$haversin\ d\ r\ = haversin\ \phi_2 - \phi_1 + \cos\ \phi_1\ \cos\ \phi_2 \times haversin\ \lambda_2 - \lambda_1 \quad (1)$$

where

- $haversin$ is the haversine function:
  $haversin\ \theta\ = \sin^2\ \theta\ 2\ =\ 1 - \cos\theta\ \ 2$ (2)
- d is the distance between two nodes;
- r is the radius of the world;
- ϕ_1, ϕ_2 is the latitude of point 1 and point 2;
- λ_1, λ_2 is the longitude of point 1 and point 2.

For correct suitability analysis of routing algorithms, multiple graph scenarios should be tested. For this task, two experimental environments are created: the first one is the road network graph using all available Latvia's road information and the second one is random structure graphs using geographical data of Latvia's cities. A graph containing road and city graph elements of Latvia is shown in Fig. 1.

Fig. 1. Latvia's road and city map

OpenStreetMap [11], open source geospatial web database containing data of territory of Latvia, was used to construct routing algorithm analysis models. The database consists of more than 4 million records. The information about 76 cities is extracted to construct random structure graphs, and 119590 nodes and 112600 edges are used in the construction of road network graph based scenarios. The database is stored in the *PostgreSQL* database management system using spatial extension *PostGIS*.

Analysis models were created with the programming language Java and open source graph library *JGraphT* [12]. This library supports the creation of graph data structures and implementation of built-in Dijkstra, Bellman–Ford and Floyd-Warshall routing algorithms. However, a few additional changes were introduced to this library:

- the calculated shortest path routes were stored in a matrix data structure to use the same data storage format across all algorithms;
- the library did not provide A* algorithm implementation; therefore, a custom implementation of A* algorithm was created and added to the *JGraphT* library.

The graph model analysis process consists of two phases - initializing or graph constructing and routing or searching for the shortest path. In each phase, the memory consumption and time consumption are calculated and then stored for further analysis.

In the first experimental environment, Latvia's road network graph model is used to determine and compare real-life route accuracy against other cartographic routing system results. The graph construction takes most of the time because all information should be extracted from the database and stored in a temporary real time data structure. Routing takes then least of the time when all routes are calculated and stored in a matrix after the search for the first route.

In the second experimental environment, cities are used as random graph nodes and edges between them are determined in a random order. 100-degree levels of graph density are used, where the value of 100 means the densest graph (all nodes are mutually connected) and 1 means the most scattered graph (only a few edges exist). In each density level, 100 random graphs are constructed and all shortest paths are determined. The computational resource estimation and graph parameter results are then stored for each graph for the further analysis.

All routing modes are tested on a computer system with *Intel Core i7-3970X 3.5GHz* CPU, 32GB RAM and *Windows 8* OS.

D. Analysis Results

Results of a random structure graph represent algorithm performance in the cases of different graph densities. 100 density degrees are tested, where density is calculated by dividing the sum of vertex edges by the total vertex count. 10000 shortest paths between all graph nodes were calculated using previously described routing algorithms. Additionally, the diameter of each graph was calculated. The diameter is the longest possible path between two nodes. Figure Fig. 2 shows the relation of the graph diameter to the graph size.

The results show that by decreasing the edge count between nodes, the diameter size increases, but such a tendency does not apply to all graphs. As the edge count decreases, more nodes become disconnected from the graph and, therefore, the graph diameter decreases. In Table I, the average computation results across different density levels can be seen.

The A* algorithm uses heuristics to find the shortest paths. In this paper, edge heuristic values are calculated based on the node edge count. The more edges are connected to a node, the higher the probability that a path going through this node will lead to the resulting node.

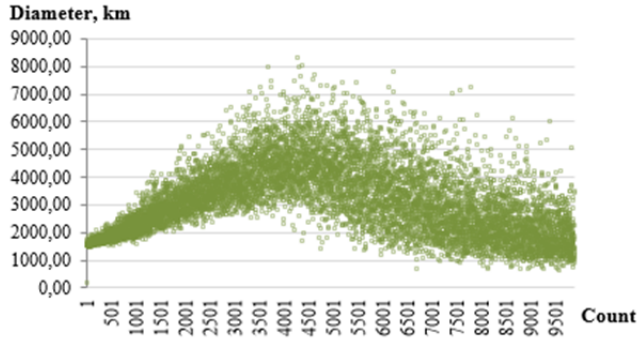

Fig. 2. Graph diameter range across multiple densities

TABLE I

RANDOM GRAPH COMPUTATION RESULTS

Density degree	Average density	Max shortest paths	Creation time, ms	Memory usage, MB
1	37.5	2850	199.56	679.89
5	7.80	2850	27.87	52.986
25	1.82	2775	22.35	7.054
50	0.76	1187	21.21	10.933
100	0.33	38	20.76	4.725

In the first analysis scenario, routing algorithms were tested for finding the single shortest path based on a random node selection. The average computation time can be seen in Fig. Fig. 4.

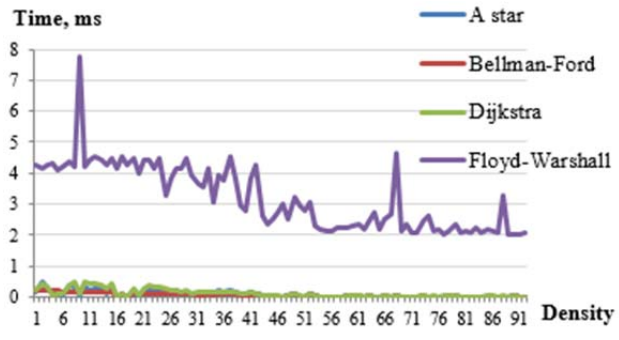

Fig. 3. Computation times required for the search of all the shortest paths

The Floyd–Warshall algorithm requires time between 3 to 8 milliseconds to find the shortest path. The Bellman– Ford algorithm calculates all the possible shortest paths from the starting node and the time required for such an operation is comparable to Dijkstra and A* algorithms. In a few occasions this algorithm shows even faster results than other algorithms. All algorithms except the Floyd–Warshall one show search time in the range of 0.5 milliseconds. As the graph becomes more scattered, the algorithm execution time, as well as the possible count of roads decreases. In addition, the difference between execution times becomes insignificant.

Also, the analysis routing algorithms were tested by finding all the shortest paths based on a random node selection. In each situation, a new graph was generated because all results should be stored in matrixes when the first search was completed. The model was designed in such a way that the algorithms would use the shortest path data stored in a matrix rather than recalculating data each time. The Floyd–Warshall algorithm is specially designed for such situations and shows the fastest execution time. In fact, it shows the shortest execution time in comparison with other algorithms that can be clearly seen in Fig. Fig. 3. The A* algorithm requires the longest time to find all the shortest paths. This is associated with a fact that this algorithm requires additional computation of a heuristic value for each graph edge.

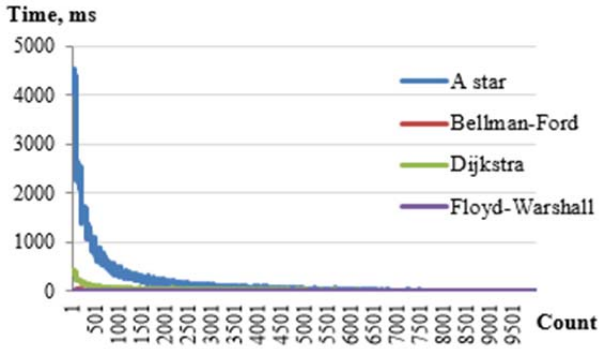

Fig. 4. Computation times required for the search of single shortest path

As the graph density decreases, the algorithm execution time also decreases. When the node to the edge ratio becomes less than 1.0, execution times become similar for all algorithms. These results show that for a scattered or small graph the algorithm execution times are similar and they all are equally suitable for routing purposes. These results also show that Floyd–Warshall algorithms are better suited for data pre-processing of large graphs. Analysis results of the existing Latvia's road network graph represent the overall routing behaviour in large scale graphs based on static geographical information. Unlike random graph-based tests, the routing algorithms on the existing road network are executed only after the graph construction. The test graph consists of 119590 nodes and 112600 edges with density ratio of 0.94. The average graph construction time was 426.47 seconds. Despite the large number of nodes and edges, all the calculations of shortest paths provided the same results.

In all shortest path searching cases, the Dijkstra algorithm showed the fastest execution time. In this analysis no pre-processing of paths was made, therefore the Floyd-Warshall algorithm showed the slowest execution time as it required computing around 7 billion different paths. The results showed that if the target nodes were located more closely, A* algorithm would have shown better results than the Bellman–Ford one, but it still would have required more computation time than the Dijkstra algorithm because additional heuristic calculations would be required.

The retrieved shortest paths between cities in a road graph were compared to other cartographic web routing systems – *Google Maps* and *Bing Maps*. The Dijkstra routing algorithm was used for the path comparison. No additional heuristics were calculated, only the graph edge directions were identified. In some routes, the difference between other routing systems was only 0.02%, but in some cases it was even 24.97%. These results characterize the data accuracy of the open source geographical road database *OpenStreetMap* in different places in the territory of Latvia.

IV. Web-based Transportation Routing Software Prototype

For better understanding of routing algorithm suitability in advanced transportation planning information systems, a routing software prototype was developed. The main functions of the developed prototype are:

- digital geographical data rendering into visual vector-based images;
- geographical information presentation in a cross - platform environment;
- usage of previously analysed routing algorithms for shortest path calculations;
- visualization of transportation routes on the top of cartographical base information.

E. Geographical Information Visualization

The research presented in this paper uses *OpenStreetMap* geographical data of Latvia's territory. The analysis of the performed algorithm proved that the existing data was accurate and could be compared to other existing geographical information system (GIS) data.

The prototype uses *PostgreSQL* database for storage of the geographical data extracted from the *OpenStreetMap* using *osm2pgsql* utility program. The database was installed on a separate virtual *Ubuntu* 12.04 server. The data contains more than 4 million geographical points converted in the WGS-84 coordinate reference system.

To obtain easily understandable geographical data representation, an additional data styling was needed for visualization purposes. The *TileMill* software was used for data visualization and styling. This program uses *Cascading Style Sheets* (CSS) language to create vector-based images. After creating a multi-layer map, this software provides an option for data export in specific *mbtiles* format specifically designed for storage of a large number of image tiles.

A decision was made to store map tiles in the *Mapbox* geographical cloud platform.

The transportation routing prototype was built using modern web browser technologies. For map tile serving and route representing, the open source *Javascript* based library *Leaflet* was used. This library provides built-in functionality that detects the coordinates the user tries to use, and this information together with the *Mapbox* platform APIs makes it possible to serve exactly the necessary tiles. *Leaflet* also provides a multi-layer architecture that allows putting in additional information on the top of the base map.

The prototype allows the user to choose two Latvia's cities and then determine and visualize the shortest path between them using the previously analysed routing algorithms. As described previously, a network graph of Latvia's roads and cities using geographical data was constructed and stored on a separate virtual server. An additional functionality for the prototype was also implemented to enable city detection in the graph structure. The shortest paths were computed using the Dijkstra algorithm as it required the smallest number of computation resources. The communication between a web browser and the routing application server was performed using *HTML5* web socket technology. The shortest path was returned to the browser as a set of coordinates in web-supported *GeoJSON* format displayed on the top of a geographical map. A graphical example of the shortest path between such two cities as Riga and Liepaja using a routing prototype can be seen in Fig. Fig. 5.

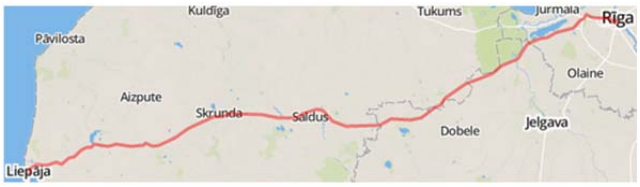

Fig. 5. The shortest path between Riga and Liepaja

F. Prototype Testing

During the testing process, the transportation routing software prototype was compared to other existing cartographic routing solutions. The road network graph analysis that was previously described already showed path length calculation results similar to the results of other systems. However, this information did not provide a specific visual insight into roads used for the shortest path determination. The prototype provides visual confirmation that in most cases the resulting roads are very similar but as Fig. **Error! Reference source not found.** shows in some cases they are different.

This can be explained by the lack of detailed information about algorithms used in the compared routing solutions.

The prototype testing also revealed that geographical information used in route construction was not precise at city road cross-points. As Fig. Fig. 7 shows, some corners that are crossed result in a shorter path than it actually is. These mistakes should be further corrected if such information is essential in navigation systems.

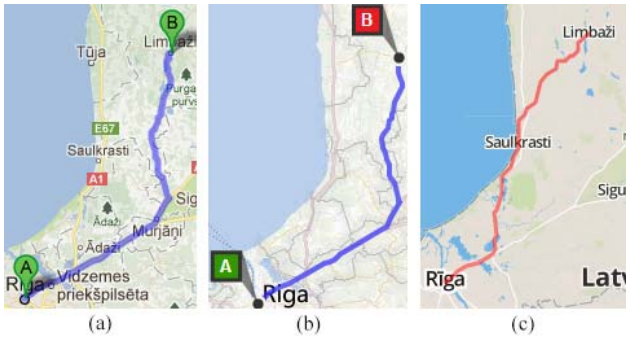

Fig. 6. The shortest path: (a) Google Maps; (b) Bing Maps; (c) implemented prototype

Fig. 7. The shortest path with cutting corners at road cross-points

The developed transportation routing software prototype allows not only for better understanding of routing algorithm suitability in real-life situations but also for visualizing algorithm execution results. Visualization provides additional information about geographical data usage, errors and actual road data used in the shortest paths.

V. Conclusion

In the presented research, routing algorithm suitability was analysed in the context of transportation planning. Different kinds of real-life scenarios using real geographical data were observed and results were comparatively analysed. From the results gathered and analysed in the current research, the following main conclusions can be drawn:

- The existing routing algorithms provide different performance results in dense and scattered environments. In dense environments, the Bellman–Ford algorithm shows the fastest execution time during the searching process of single shortest path, but in scattered situations– the Dijkstra's algorithm requires the least amount of computation time.
- The road data pre-processing requires more resources during a graph construction phase, but can greatly decrease the shortest path search time. Hierarchical routing algorithms can decrease search time from milliseconds to microseconds. The Floyd–Warshall algorithm shows the fastest pre-processing time in all possible determination scenarios of the shortest path.
- The Dijkstra algorithm without specific heuristic data shows results similar to other existing routing solutions. In some situations, the difference between routing results was only 0.02%.

For better result suitability for web-based transportation planning, a routing software prototype was developed. The results gathered from the prototype show visual information about roads used to determine the shortest paths. The obtained results are similar to other routing solutions, but in some occasions they differ, which can be explained by the lack of detailed information about the used algorithms and difference in geographical data.

REFERENCES

[1] T. Toledo, O. Cats, W. Burghout, and H. N. Koutsopoulos, "Mesoscopic Simulation for Transit Operations," *Transp. Res. Part C Emerg. Technol.*, vol. 18, no. 6, pp. 896–908, Dec. 2010.

[2] D. Joyner, M. Van Nguyen, and D. Phillips, *Algorithmic Graph Theory and Sage*, Version 0.8-r1991, 2013, p. 304. [E-book] Available: https://code.google.com/p/graphbook/.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Syst. Sci. Cybern. IEEE Trans.*, vol. 4, no. 2, pp. 100–107, 1968.

[4] R. Bellman, "On a Routing Problem," *Q. Appl. Math.*, vol. 16, pp. 87–90, 1958.

[5] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345–, 1962.

[6] S. Warshall, "A Theorem on Boolean Matrices," *J. ACM*, vol. 9, no. 1, pp. 11–12, 1962.

[7] M. J. d. Smith, M. F. Goodchild, and P. A. Longley, *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*, 3rd Revise. Matador, 2009, p. 516.

[8] K. Gutenschwager, A. Radtke, S. Volker, and G. Zeller, "The shortest path: Comparison of different approaches and implementations for the automatic routing of vehicles," in *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 2012, pp. 1–12.

[9] P. Sanders and D. Schultes, "Engineering Highway Hierarchies," in in *Algorithms – ESA 2006 SE - 71*, vol. 4168, Y. Azar and T. Erlebach, Eds. Springer Berlin Heidelberg, 2006, pp. 804–816.

[10] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," in in *Experimental Algorithms SE - 24*, vol. 5038, C. McGeoch, Ed. Springer Berlin Heidelberg, 2008, pp. 319–333.

[11] OpenStreetMap, "Atvērtā Wiki pasaules karte." [Online]. Available: http://www.openstreetmap.org. [Accessed: Sep. 25, 2013]

[12] JGraphT, "Java mathematical graph-theory objects and algorithms," 2013. [Online]. Available: http://jgrapht.org. [Accessed: Sep. 25, 2013]

Davis Jaunzems is a master's student majoring in information technology at Riga Technical University.

Since 2009 he has been working in software development companies and at the same time working on projects focusing on large data processing and analysis. He has also participated in many events aimed at building prototypes with the latest technologies. His professional interests include mobile technologies, data analysis and geographical information systems.
E-mail: davis.jaunzems@gmail.com

Arnis Lektauers, Dr.sc.ing., is an Assistant Professor at the Department of Modelling and Simulation of Riga Technical University (RTU). His main professional interests include the development of interactive hybrid modelling and simulation algorithms with an application to complex systems analysis and the research of industrial, economic, ecological and sustainable development problems. A. Lektauers is the Secretary of Latvia Section of the Institute of Electrical and Electronics Engineers (IEEE), a member of the Council of RTU Faculty of Computer Science and Information Technology, and a member of Latvian Simulation Society, System Dynamics Society and European Social Simulation Association (ESSA). He is the author of 1 textbook and more than 30 papers in scientific journals and conference proceedings in the field of Information Technology.
E-mail: arnis.lektauers@rtu.lv

Dāvis Jaunzems, Arnis Lektauers. Maršrutēšanas algoritmu piemērotības analīze tīmeklī bāzētai transporta plānošanai
Transporta līdzekļu maršrutēšana sarežģītās pilsētvides vai reģionālajās sistēmās ir uzdevums, kuru nepieciešams risināt dažādiem pielietojumiem, piemēram, privātā vai sabiedriskā transporta tīklu detalizētas analīzes modeļos. Šajā rakstā ir aprakstīts maršrutēšanas algoritmu analīzes process un paņēmieni transporta plānošanai tīmekļa bāzētās kartogrāfiskajās sistēmās. Piedāvātā pētījuma mērķis ir izpētīt eksistējošos maršrutēšanas algoritmus īsākā ceļa meklēšanai transporta tīklos un analizēt maršrutēšanas algoritmu implementācijas iespējas un efektivitāti tīmekļī bāzētās ģeogrāfiskajās informācijas sistēmās. Balstoties uz grafu teoriju, īsākā ceļa meklēšanai var tikt izmantoti dažādi maršrutēšanas algoritmi. Šajā rakstā galvenā uzmanība ir pievērsta četru metožu analīzei, kas ir klasificējami kā īsākā ceļa algoritmi: Deikstras, Bellmana-Forda, Floida-Varšala, A* (A zvaigznes). Piedāvātais pētījums fokusējas uz skaitļošanas laika un datora atmiņas resursu analīzi, kas nepieciešami maršrutēšanas algoritmiem īsākā ceļa noteikšanai atšķirīgās kontekstuālās situācijās. Definētā pētījuma mērķa sasniegšanai ir izstrādāta un notestēta interaktīva tīmeklī bāzēta kartogrāfiskā sistēma ar iebūvētām maršrutēšanas iespējām. Balstoties uz publiski pieejamiem Latvijas teritorijas ceļu tīkla datiem, ir veikti vairāki eksperimenti, notestējot dažādus īsākā ceļa atrašanas scenārijus un nosacījumus. Izstrādātais prototips ļauj noteikt katra analizētā maršrutēšanas algoritma piemērotības nosacījumus, kā arī nepieciešamās darbības tīmeklī bāzēta transporta plānošanas procesa optimizācijai.

Давис Яунземс, Арнис Лектауэрс. Анализ пригодности алгоритмов маршрутизации для транспортного планирования в глобальной сети
Маршрутизация транспортных средств в сложных городских или региональных системах является задачей, которую нужно решить в разных приложениях, например, в детализированных моделях анализа сетей личного или общественного транспорта. В данной статье описан процесс и методика анализа алгоритмов маршрутизации для планирования транспорта в веб-базированных картографических системах. Целью представленного исследования является изучение существующих алгоритмов маршрутизации для нахождения кратчайшего пути в транспортных сетях и для анализа возможности и эффективности реализации алгоритмов маршрутизации в веб базированных географических информационных системах. Базируясь на теории графов, для нахождения кратчайшего пути можно использовать разные алгоритмы маршрутизации. В данной статье главное внимание обращено к анализу четырёх методов, которые классифицированы как алгоритмы кратчайшего пути: алгоритмы Дейкстры, Беллмана–Форда, Флойда–Уоршелла, A* (A star). Представленное исследование фокусируется на анализе требуемых вычислительных ресурсов времени и памяти, которые необходимы для реализаций алгоритмов во время нахождения кратчайшего пути в разных контекстуальных ситуациях. Для достижения поставленной цели исследования создан и протестирован прототип интерактивной веб базированной картографической системы с интегрированными возможностями маршрутизации. Базируясь на публично доступных данных дорожной сети на территории Латвии, было проведено несколько экспериментов по изучению разных сценариев и условий для нахождения кратчайшего пути. Созданный прототип позволяет определить условия пригодности каждого проанализированного алгоритма маршрутизации, а также необходимые действия для оптимизации процесса веб базированного транспортного планирования.