UDC 004.7

Yevheniia Zubrych,
Oleksandr Podrubailo

GENERATION OF THE SHORTEST ROUTE BASED ON THE VISIBILITY OF THE INTERMEDIATE POINTS

Євгенія Зубрич,
Олександр Подрубайло

ПОБУДОВА НАЙКОРОТШОГО МАРШРУТУ НА КАРТІ З УРАХУВАННЯМ ОБЛАСТЕЙ ВИДИМОСТІ ПРОМІЖНИХ ПУНКТІВ

This paper analyzes the existing routing solutions and identifies their disadvantages. Based on the obtained results, an algorithm for constructing the shortest route on the map is proposed.

Key words: route planning, tourist routes, area of object's visibility.
Fig.: 3. Tabl.: 1. Bibl.: 9.

У даній роботі було проаналізовано існуючі рішення для географічної маршрутизації, виявлено їх недоліки. Розроблено алгоритм для побудови маршруту з урахуванням областей видимості проміжних пунктів.

Ключові слова: планування маршрутів, туристичні маршрути, області видимості об'єктів.
Рис.: 3. Табл.: 1. Бібл.: 9.

Relevance of research topic. Today, automated systems for building and visualizing routes are used in many industries. Route planners have become a part of everyday life. Tourism is the most urgent area. The positive impact of information technology on the dynamics of the local and international tourist flow has led to the transformation of the tourism industry from a service-oriented to a diversified field of activity aimed at meeting the diverse needs of millions of individual tourists [1].

The real-time planners used for tourist purposes are especially in demand.

Formulation of the problem. Existing algorithms are focused at finding the fastest and shortest route. However, using systems that generate only the shortest route, the tourist risks not to see many of showplaces in the particular area.

Analysis of recent research and publications. With the growing demand for navigation systems and route planners, the number of researches in this area has also increased. There are many systems that built the shortest path between the two points.

Most of them can be divided in two main types of existing routing applications: offline, or onboard planners and online or web planners. Numerous applications of both types exist, this chapter will only name a few very popular ones and several that are in some way related to this project.

TomTom is a large international company offering stand-alone navigation devices. Their devices are some of the most popular, mainly due to the intuitive interface, speed and accuracy of route calculations. Devices can count on routes for traveling on a car, bike or on foot. Unfortunately, their routing algorithm and data sources are not open to developers. TomTom has also recently released an online version of its route planner, but this version does not have many features that offers a built-in version [2]. Among other things, it lacks the ability to plan routes by bike or on foot. Also, neither an online version nor a built-in version has a Ukrainian localization.

Google has launched its own routing service, called Google Maps [3]. This is a very fast free service. Obviously, Google performs some preprocessing or caching to make their routing service so fast, but the details and routing algorithm remain secret. Google Maps is available in Ukrainian, but the Google Maps API does not allow to modify the algorithm or consider the priorities of the objects when constructing the route.

Via Michelin is an online routing service based on the maps of the well-known Michelin roadmap issuer [4]. The service is free and works fast, but again, the information about the algorithms is confidential. Via Michelin is not available in Ukrainian, builds only the shortest routes and does not always work exactly for cycling routes.

YourNavigation.org is a demo website for the YOURS project. The purpose of the project is to create a routing website based on OpenStreetMap (OSM) data using other open source applications. It uses an open source routing mechanism called Gosmore [5]. The service is not very fast, and its website mentions that Gosmore is not intended to generate routes longer than 200km. Planner gives the ability to choose between two types of route: the shortest or fastest, and one of the nine modes of transport, but it is only available in English.

OpenRouteService.org is another online service that uses OSM data [6]. As well as YourNavigation.org, this is a non-profit service. The service uses the A * algorithm and is slower on long routes than other analogues. OpenRouteService.org is available in Ukrainian, can configure the types of roads, the type of route (fastest or shortest), the type of vehicle and its type of fuel. However, as well as other similar services, it does not take into account the types of intermediate points of the route and does not allow them to be configured as a priority when constructing a route relative to the user's preferences.

The results of comparing the five routing services are shown in Table 1. Unfortunately for the best performing applications the routing algorithms were confidential. The two that did reveal their routing algorithm both used A*. It is obvious that non-commercial routing services tend to stick with less complex algorithms, to limit the time that needs to be invested. Commercial applications have a lot depending on the performance of their service and can afford to invest more time and money in order to increase performance. It was found that most of the existing solutions do not have the Ukrainian language version and none of them supports the construction of tourist routes.

Selection of unexplored parts of the general problem. Nowadays, systems that can generate route based on the types of objects on the map are unexplored and not investigated. However, it would be useful to modify the route in such a way that it was, on the one hand, short, on the other hand, covered as many objects of the given type as possible.

Table 1

Comparison of existing services

Name	***Algorithm***	***Data source***	***Selecting start and destination***	***Supports Ukrainian language***	***Supports priorities of intermediat e points***
TomTom	*confidential*	Own data	Addresses	No	No
Google Maps	*confidential*	Own data	Both addresses and select on a map	Yes	No
Via Michelin	*confidential*	Michelin maps	Both addresses and select on a map	No	No
YourNavig ation.org	A*	OSM	Both addresses and select on a map	No	No
Open Route Service.org	A*	OSM	Both addresses and select on a map	Yes	No

Setting objectives. Thus, the actual task is to develop a system that when constructing a route will consider its length and the number of objects covered by the given type. Also, when calculating the route, it is necessary to reflect the limits of visibility of each of the points, because it would be enough that the path would be passed by objects, and not through them.

Presentation of the main material

1. The task of finding the shortest path

The problem of finding a route on a map can be represented in a simple form by using a graph, as in Figure 1. To find the shortest path from, e.g., A to H, one would need to find the collection of edges that connect A to H through any other nodes, for which the sum of the weights is as low as possible. Throughout this paper, the length of the shortest path from some node s to some node t will be referred to as the distance from s to t.

Consider the weighted graph $G = (\mathrm{V}, \mathrm{E})$, where V – is a finite set of nodes and E - is a set of edges between these nodes. Number of nodes $|V|$ we denote as n , and the number of edges $|E|$ - m. Each edge e has a weight $w\ (e)$. The path is determined by the sequence of nodes $(v_1, \dots, v_k)$, for which $(v_{\mathrm{i}}, v_{i+1}) \in E$ and $1\ < i\ < k$. A node connected to a specific node v with some edge is called a neighbor v.

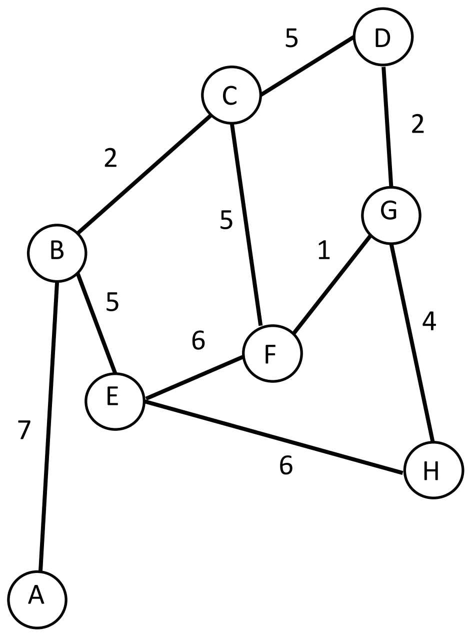


Fig. 1. Weighted graph

If the starting node is a vertex $s \in V$ and a finite node $t \in V$ then the shortest path is defined as the path $(s, \dots, t)$, which has the minimum sum of the weights of all edges in the path. The length of the shortest path from s to node v is defined as $g\ (v)$ and is also called the distance from s to v.

2. Algorithm A*

When traveling to a certain destination, it usually does not make sense to look for a path in the opposite direction. Therefore, an algorithm that prefers the vertices in the direction to the destination appeared and first visits them, unlike the Dijkstra algorithm, which searches in all directions of space. The algorithm does not search in the direction of the target node. Hart, Nilsson and Raphael [7] introduce the A* algorithm, which adds heuristics to Dijkstra, making it more directional to the final vertex. Instead of the weight of node v, we use the estimate of the shortest path $\hat{g}(v)$ from the initial vertex to the finite, which runs through the node v. For this purpose, a function (1) is introduced that represents the shortest path from s to t passing through v, with $\hat{g}(v)$ being the distance from s to v, and $h(v)$ is the distance from v to t.

$$f(v) = g(v) + h(v) \tag{1}$$

Also, estimates of the values of functions were introduced:

$$\hat{f}(v) = \hat{g}(v) + \hat{h}(v) \tag{2}$$

The estimate of the distance from s to v, $\hat{g}(v)$ is determined in the same way as in the Dijkstra algorithm, it is the shortest path from s to v, found at the current iteration. Estimation of the distance from v to t, $\hat{h}(v)$ is determined heuristively. This function can be any function and often its definition is a separate task. Euclidean distance from v to t is most often used as heuristic function for route planning. The heuristics should be admissible, that is, they do not have to overestimate the cost of the route; the estimate of the path should be in the range $[0;\ k]$ where k is the actual distance, and the motoonon, ie, for each vertex v and adjacent vertex $v`$ it must be inequality (3), where $c(v, v`)$ is the actual distance between v and $v`$:

$$h(v)\ \leq\ c(v, v`) + h(v`) \tag{3}$$

3. Calculation of the weight of the edge

In order to solve the problem, it is proposed to introduce the following definition of the function $\dot{w}(e)$:

$$\dot{w}(e) = k(e) * w(e), \tag{4}$$

where $k(e)$ – the function which defines the value of the coefficient k for each edge e.

The value of k should be based on types of points, which areas of visibility cover that edge e. Moreover $k(e) \in (0,1]$, accordingly $\dot{w}(e) \in (0, w(e)]$. Therefore, it will be possible to take into account the type of intermediate point when constructing the route and to achieve an increase in the probability of entering the priority object to the resulting route, because at identical distances, the vertex, the edge to which passes through the scope of the priority object will have a smaller value $\dot{w}(e)$. It means that when choosing a vertex v from the set X such that $\hat{g}\ (v){:}= \min_{u\in X} \hat{g}\ (u)$ [9], where $X{:}\ =$ set of nodes v for which the path from s to v is not defined, the vertex with the smallest $\dot{w}(e)$ will be selected.

For each edge e_i passing through the visibility of the vertex v, $\dot{w}(e_i)$ will be equal to the edge weight, multiplied by the coefficient $k(v)$: $\dot{w}(e_i) = w(e_i) * k(v)$, and for others edges $\dot{w}(e_i) = w(e_i)$, since for them $k = 1$. The calculated value $\dot{w}(e_i)$ can then be used in the algorithms for constructing the route instead of the distance between the vertices that connect the edge e_i, that is, its length.

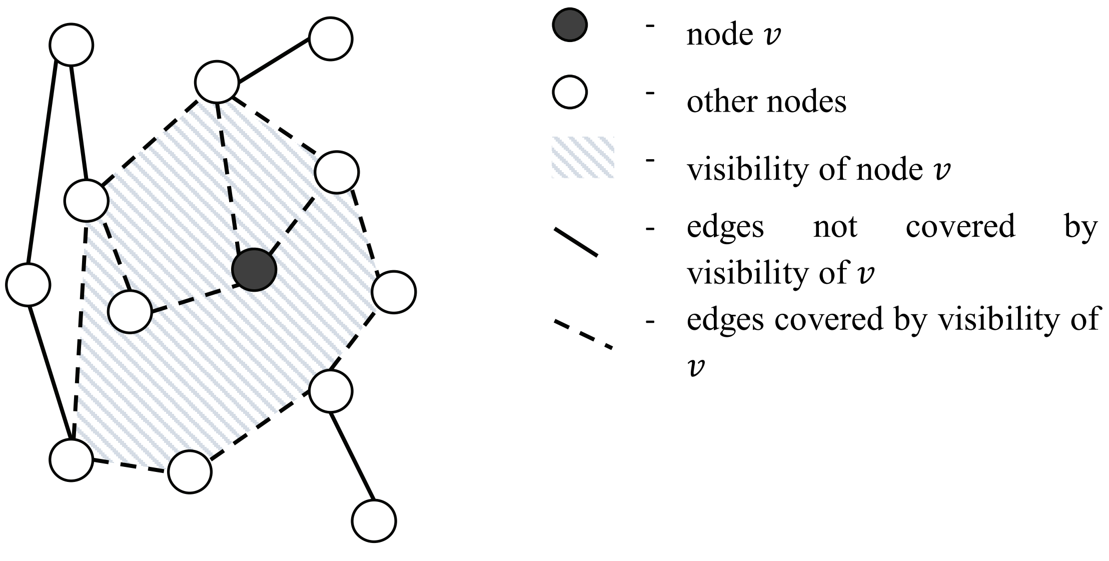

● - node v
○ - other nodes
▨ - visibility of node v
- edges not covered by visibility of v
- edges covered by visibility of v

Fig. 2. Visibility of the node

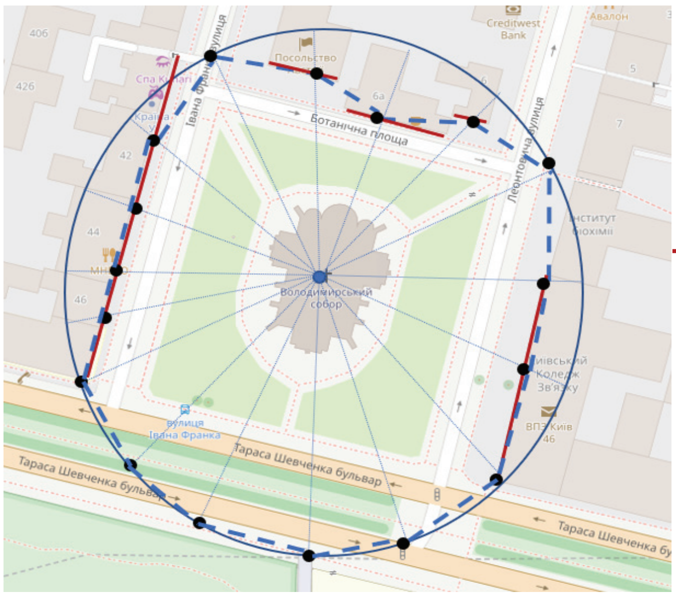
4. Calculation of the object's visibility on the map

When calculating the route, it would be necessary to determine the scope of visibility of each of the priority objects, that is, the set of points from which this object will be visible on the locality. This is especially important in the context of a dense building of the city, because the neighboring buildings can overlap the view of the object.

As an input to the system it is better to use OSM maps in XML format. All map objects in OSM XML format are divided into 2 types: <node>, which consist only one point and are described by parameters such as latitude, longitude and id (node id), as well as ways (<way >), which can consist of 2 or more dots (up to 2000). The way may be open or closed. These types of objects are interconnected by relations (<relation>). Buildings, or rather, their boundaries are usually described by means of closed way. This means that it is possible to get the coordinates of the boundaries of buildings located within a radius from point x (lat, lon) from OSM XML maps.

By the method of ray tracing [8] it is possible to construct a polygon, which would determine the scope of the object. The point of the start of all rays may be the center of the object, but for the sake of greater accuracy, it is necessary to take two more distant points from each other that are at the boundary of the object, for which the polygon is calculated, because some buildings may be too long.

Figure 3 shows an example of constructing a polygon of an visibility area. In order to construct a polygon, the point x of the rays is selected (in the picture - the center of the object). A bounding circle is constructed with radius r, which is the maximum distance from which the object can be seen. After that n rays with start in x are throwed in all directions.



- - bounds of the visibility area
- - ray from the center of the object
- 🟥 - blocked surfaces
- ● - the point of intersection of the ray with a blocking surface or a limiting circle
- ○ - limited circle with radius r

Fig. 3. Calculation of the object's visibility

Each of the rays intersects either the limiting circle or the blocking surface. The blocking surface mean the first boundary of the building, which occurs on the path of the ray. A plurality of intersection points forms a polygon, or the area of visibility.

5. Calculation of the coefficient *k*

Most objects in OSM XML maps have metadata other than coordinates and IDs. For example, the <amenity> tag may be one of the most useful for identifying an object type. This tag is used to designate public infrastructure objects: banks, schools, hospitals, cinemas, theaters, fountains, etc. Also, such tags as <*tourism*>, <*building*> with value "*architecture*" would be useful in systems oriented on tourist routes building. If it needed the route to go through parks and green areas, it be useful to find all objects marked with the <*leisure*> tag with the value of "*park*".

Once all the priority objects are found, it is necessary to determine the level of their importance, since the monuments are different, and have varying degrees of influence on the route. For example, for matching with each of the conditions you can count points: +3 for the tag <*historic*>, +4 for <*leisure*> with the value of "*park*", +2 for a link to Wikipedia with an article about this object, +1 for each additional tag <*name*>.

The sum of the points should be converted into a coefficient with a value within $(0;\ 1]$ and matched to each visibility area with the value of the factor of the object around which this area was constructed.

Conclusions. This paper analyzes the existing routing solutions and identifies their disadvantages. Based on the obtained results, an algorithm for constructing the shortest route on the map is proposed. Using this algorithm to modify Dijkstra and A* algorithms, it's possible to create a system for constructing tourist routes.

References

1. Мельниченко С. «Інформаційні технології в туризмі: теоретичні та практичні аспекти» / Мельниченко С., Запоріжжя: Вісник Запорізького національного університету №2(6). – 2010. – С.129.

2. Marcin Wojnarski «TomTom Traffic Prediction for Intelligent GPS Navigation» / M. Wojnarski, IEEE International Conference on Data Mining Workshops – 2010 – C.20-21.

3. Gabriel Svennerberg "Google Maps API 3" / G. Svennerberg, Apress – 2010 – C.157-160.

4. ViaMichelin Navigation. User Manual [Electronic source] / ViaMichelin, URL: http://enav.download.viamichelin.com/nav/tel/manuels/gbr/User_Manual_GBR_VMN_New_Edition_v7.pdf (Accessed: 01.05.2019).

5. YourNavigation.org About [Electronic source] / YOURS, URL: https://wiki.openstreetmap.org/wiki/YOURS/ (Accessed: 01.05.2019).

6. Sebastian Schmitz «New Applications based on collaborative geodata – the case of Routing» / Sebastian Schmitz, Proceedings of XXVIII INCA international congress on collaborative mapping and space technology – 2008 – C.1-7.

7. Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. «A formal basis for the heuristic determination of minimum cost paths» / Hart, Peter E., Nils J. Nilsson, and Bertram Raphael., IEEE transactions on Systems Science and Cybernetics 4.2 – 1968 – C.100-107.

8. Spencer G.H., M.V.R.K. Murty «General ray-tracing procedure» / Spencer G.H., M.V.R.K. Murty, JOSA 52.6 – 1962 – C.672-678.

9. Knuth, Donald E. «A generalization of Dijkstra's algorithm» / Knuth, Donald E, Information Processing Letters 6.1 – 1977 – C.1-5.

Autors

Oleksandr Podrubailo – research assistant, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: alex.podrubailo@gmail.com

Подрубайло Олександр Олександрович – асистент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Yevheniia Zubrych – student, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: evg.zubrich@gmail.com

Зубрич Євгенія Сергіївна – студентка, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Євгенія Зубрич,
Олександр Подрубайло

ПОБУДОВА НАЙКОРОТШОГО МАРШРУТУ НА КАРТІ З УРАХУВАННЯМ ОБЛАСТЕЙ ВИДИМОСТІ ПРОМІЖНИХ ПУНКТІВ

Актуальність теми дослідження. Сьогодні автоматизовані системи побудови та візуалізації маршрутів використовуються у багатьох галузях, а планувальники маршрутів стали невід’ємною частиною повсякденного життя. Особливо затребуваними стали в наші дні і продовжують активно розвиватися системи планування маршрутів, що працюють в режимі реального часу.

Постановка проблеми. Існуючі алгоритми здебільшого направлені на пошук найшвидшого та найкоротшого маршруту. Проте використовуючи системи, що генерують лише найкоротший маршрут турист ризикує не побачити частину пам’яток даної місцевості.

Аналіз останніх досліджень і розробок. З ростом попиту на навігаційні системи та планувальників маршрутів, збільшилась і кількість досліджень у цій галузі. Насамперед з’явилось чимало систем, що будують найкоротший шлях між двома точками. У даній роботі було проаналізовано існуючі рішення для географічної маршрутизації, виявлено такі їх недоліки як відсутність підтримки української мови та побудови туристичних маршрутів.

Виділення недосліджених частин загальної проблеми. На даний момент недослідженими та нереалізованими є системи які б при побудові найкоротшого маршруту враховували типи об’єктів, що є на мапі, та класифікували їх за тематикою.

Постановка завдання. Актуальною є розробка системи, що при побудові маршруту буде враховувати його довжину та кількість охоплених об’єктів заданого типу. Також при розрахунку маршруту необхідно врахувати межі видимості кожного з пунктів, адже достатньо, щоб шлях проходив повз об’єкти, а не крізь них.

Викладення основного матеріалу. У даній роботі будо запропоновано алгоритм побудови найкоротшого шляху, що базується на областях видимості проміжних пунктів. Визначено поняття області видимості, розроблено метод її розрахунку на основі координат об’єкта, відносно якого ця область будується та координат будівель, що знаходяться не далі, ніж деяка відстань r від центру об'єкта.

Висновки. Було проаналізовано існуючі рішення побудови найкоротших маршрутів та виявлено їх недоліки. На основі отриманих результатів було запропоновано алгоритм побудови найкоротших маршрутів. Використовуючи даний алгоритм для модифікації алгоритмів Дейкстри та A*, можливе створення системи побудови туристичних маршрутів.

Ключові слова: планування маршрутів, туристичні маршрути, області видимості об’єктів.