Ridesharing for Carsharing Service Provider: Driver and Pedestrian Route Matching

Alexey Kashevnik[1], Nikolay Teslya[1,2], Sergey Mikhailov[1,2], Mikhail Petrov[1,2], Anton Shabaev[3], and Andrey Krasov[4]
[1]ITMO University, St.Petersburg, Russia
[2]SPIIRAS, St.Petersburg, Russia
[3]Petrozavodsk State University (PetrSU), Petrozavodsk, Russia
[4]Bonch-Bruyevich St. Petersburg State University of Telecommunications, St.Petersburg, Russia;
{alexey.kashevnik, teslya, sergey.mikhailov, mikhail.petrov}@iias.spb.su, ashabaev@petrsu.ru, krasov@inbox.ru

***Abstract*—Nowadays there are a lot of car sharing service provider have been appeared in Russian market as well as in the world. Car sharing become popular since a person can have comfortable and quick transportation means in the city. This transportation means allow him/her to reach destinations without owning a private vehicle. However, sometimes for the car sharing user it is not possible to find a car in nearest location. In this case the car sharing company can propose customers who goes to the same destination to share the ride together with the travel expenses. The paper proposes an approach to matching of driver and pedestrian routes based on predefined constraints such as detours and additional time the customers are going to spend for ride sharing.**

I. Introduction

Last years, there are a lot of car sharing companies appeared in the roads of big Russian cities as well as in the world [1]. Yandex Drive, YouDrive, Delimobil, EasyRide, Car5, BelkaCar, colesa.com and etc. At the moment most of companies have large car parks as well as YouDrive aggregates car parks from different car sharing service providers. Along with that at outskirts sometimes for the car sharing users is not possible to find the car in walking distance. One of the possibilities to solve this problem and keep customers for car sharing companies as well as reduce traffic jams in popular routes is the possibility to share the rides by the two or more car sharing users that will share the expenses for car utilization.

The ridesharing is another strategy of sharing a car. In this strategy, owner of the car shares his/her trip with passengers based on the close located points of origin and destination. Therefore it is also knows as carpooling, lift-sharing or covoiturage [2]. The strategy allows to share travel cost as well as car maintenance cost between participants. In the same time it increases efficiency of car usage, since one car is used to transport at least two people, therefore reducing carbon emission by person, saving parking lots and road network from being overloaded [3]. Ridesharing became very popular in the US since government of many states provides a privileges to drives who cares more than one passenger by allowing to use special line marked as “2 or more persons in vehicle” or “high occupancy vehicle line” usually used by buses.

The paper proposes a task definition of the ridesharing feature task for car sharing company, algorithms to match the driver path with the pedestrian location and destination points as well as evaluation of the proposed algorithm. The proposed algorithm is based on the previous author’s research in this area [4], [5], [6], [7]. The main research novelty of this paper is adaptation of the developed earlier approach to the car sharing service providers as well as to show the possibility to its utilization.

The rest of the paper is organized as follows. Related work in the topic of ridesharing is presented in Section 2. Section 3 is devoted to task definition. A driver and pedestrian route matching algorithm is presented in Section 4. Evaluation of the developed algorithm is provided in Section 5. Main results are summarized in Conclusion.

II. Related Work

Nowadays, the next period of interest in ridesharing is expecting. It is associated with the intensive development of data processing, transfer technologies, and computing capacities, which can simplify the search for fellow travelers.

There are a lot of possible schemes of ridesharing implementation used by people in different countries:

- Public forums and communities. The information about upcoming or ongoing trips is shared through an open Web portal by drivers and passengers. The registration is free and shared trips are available for all users. The shared information consists of origin and destination points, time of travel (start and/or end time of the trip), personal information about fellow-traveler (gender, age, interests), trop cost, etc. The examples are: *eRideShare.com* [8], *PickupPal* [9], *Zimride* [10], *RideshareOnline* [11], *rideshare.511.org* [12], *CarJungle* [13];
- Private Web-services. In contrast to public forums private servers can be used only if user have an invitation. Usually this schemes is used by organizations to provide private space for their employees where their trips can be shared. For example, *Zimride* provides a private interface for universities and companies;
- Mobile applications. The scheme is similar to public and private services but access to trips is provided by mobile application for Android or iOS. Some of them provide base level automation to find fellow-travelers by trip parameters

and user preferences. The examples are *PickupPal* [9] and *Avego* [14];

- Using agents to find fellow-traveler (for example taxi companies);
- Simple pick-up (not pre-arranged points e.g. on bus stops).

Mobile applications are usually built based on the client-server architecture and search for fellow traveler is conducted on the server side to reduce power consumption of mobile device. The approach presented in the paper is based on the idea of ubiquitous computing where all devices are participating in calculations. The idea is implemented through the decentralized smart space infrastructure. Decentralization provides increasing of the service stability and performance.

The main task to be solved during the development of the ride sharing support system is to search for meeting points for the pedestrian and driver. Different approaches are offered for the selection of drivers and passengers, among which is an agent-based approach. In this approach drivers and pedestrians are represented as agents engaged in bidding when choosing a pair [15]. The selection of potential pairs is initially based on the analysis of the driver's path and the selection of potential passengers from pedestrians within a predetermined radius, after which the agents begin bidding to find a price that suits everyone. It is also proposed to formulate the problem of searching for drivers and passengers as an integer linear programming problem in which it is necessary to find a solution that satisfies a number of conditions [16]. The cost function is defined as the sum of the delay times of the driver and pedestrian, including the waiting time at the meeting point.

The use of meeting points that are not related to the pedestrian's starting and ending points, although it complicates the task of selecting drivers, has advantages, the main of which is to increase the potential pairs of passengers and drivers [17]. Additionally, the solution found at the meeting point is often more optimal in terms of overall costs compared to fixed meeting points [17].

There are also studies aimed at integrating existing transport networks with ridesharing ideas. In particular, the paper [18] presents a model for integrating an existing public transport network with ridesharing. It is noted that a feature of this model is the need to take into account a fixed schedule of public transport when planning a trip. At the same time, the mobility of residents of suburban areas in which public transport routes are not able to cover all the required areas is significantly increased with introducing the ridesharing model.

Separately, it should be noted the study [19], in which the main emphasis is on the model for calculating the cost of a joint trip. The case of joint trips in the early hours of the day is considered. There are three roles for system users - a single driver, a ridesharing driver, a ridesharing rider. Costing schemes are based on optimizing travel times and travel distances. Based on these parameters, a model is built, according to which the system optimum is calculated to equalize the wishes of the driver and pedestrian in the cost of the trip.

In addition, the safety of joint travel remains an important issue. At the same time, security means not only the protection of personal data in the applications used, but the physical safety of passengers and drivers. The economy of shared consumption presupposes universal trust among everyone, and its maintenance through information services. So, in [20], a detailed analysis of health threats to passengers and drivers of Uber and Lyft companies providing ride sharing services in their applications is carried out. Among the causes of possible danger, the malicious actions of drivers and passengers, the use of drugs and alcohol by drivers, which significantly increases the probability of an accident, the unsatisfactory technical condition of the vehicle, are highlighted. All these problems are solved only by building control by the service owners with a timely response to complaints from ridesharing participants. To ensure route privacy when searching for matches, it is also proposed to use the Secure Multiparty Computation technique and its extension - Private Set Intersection (PSI) [21]. For this mechanism to work, the driver and passenger provide isochrones (many points with an equal time to reach them), among which matches are calculated.

However, most of mentioned services provide platforms for finding fellow travelers offline, rather than real time mobile services for generate rideshare plans. In everyday life people often make spontaneous decisions and are keener to pursue own schedules. Thus, the service should enable dynamic formation of carpools based on existing carsharing cars, active drivers and on current situation and people preferences. In this paper, dynamic ridesharing service implementation for mobile devices is described.

III. Ridesharing Task Definition

Common ridesharing scenario is shown in Fig. 1. A car sharing user who cannot find an acceptable car tries to find the drivers who goes with the same route. Both users, pedestrian and driver, should specify their preferences for the system: maximum delay acceptable to him/her to wait, maximum detour that his/her route will be extended, social interests (including favorite music, in-cabin temperature), and etc.

Then the matching algorithm described in the Section IV is implemented. Algorithm is aimed to match the driver path with the pedestrian location and destination points. The goal of algorithm is to determine list acceptable drivers for the pedestrian based on driver and pedestrian preferences. For every acceptable driver the meeting points are determined that are satisfied to determined detours. Then the pedestrian can choose the driver that is most preferable for joint ride. After that the driver gets a request. It he/she accepts the request the pedestrian gets a notification that he/she should reach the meeting point. The price for the ride is divided between the participating people.

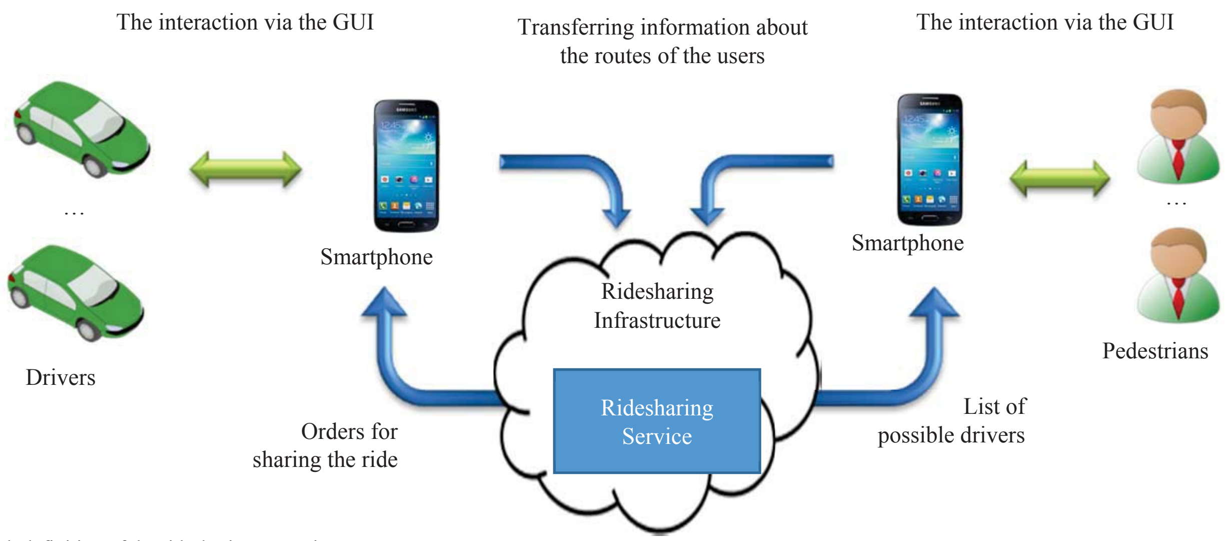
The interaction via the GUI
Transferring information about the routes of the users
The interaction via the GUI
...
Smartphone
Smartphone
...
Drivers
Ridesharing Infrastructure
Pedestrians
Ridesharing Service
Orders for sharing the ride
List of possible drivers

Fig. 1. Task definition of the ridesharing scenario

IV. Algorithm For Matching of the Driver and Pedestrian Routes

While searching for matching path of driver and pedestrian for the ridesharing the following problem has to be solved: is it possible to start joint trip for users based on the provided information about origin and destination, and taking into account restrictions from each user. If “Yes” the best points for meeting have to be calculated based on local preferences of driver and potential passenger as well as meeting time. To solve this task the algorithm has been proposed that finds a matching between driver and pedestrian path acceptable for both of them.

Let consider *A* as origin and *B* as destination point of the pedestrian's path. For the driver let consider *C* as origin and *D* as destination point of the driver's path. The shortest path between driver’s points is acquired from navigation service and is shown schematically with solid line on Fig. 2. The Fig 2. also shows that the driver and pedestrian are moving almost in the same direction and the driver can change his route to give the pedestrian a ride. The simplest situation, when meeting points are fully matched with pedestrian origin and destination is indicated on Fig 2 with the dotted line *CABD*. A more complex situation raises in case of searching for a meeting point in some distance from driver’s and pedestrian’s paths that satisfies both the driver and the pedestrian restrictions. One of the possible solutions is presented on the Fig. 2 with the dash-dot line: meeting points of driver and pedestrian are shown with *E* and *F* points and the resulting joint path is presented with *CEFD* line.

The main stages of the algorithm for finding matching between driver and pedestrian routes is presented below:

```
FOR driver IN driver_list DO
  FOR pedestrian IN pedestrian_list DO
  matching =
  find_match(driver.path,pedestrian.path);
  // according to the above scheme
  cd = constraint_checking(driver,
  matching);
  cp = constraint_checking(pedestrian,
  matching);
    IF cd && cp THEN
      share(driver, match);
      share(pedestrian, match);
  ENDFOR;
ENDFOR;
```

The meeting points have to meet the following restrictions:

- Pedestrian’s meeting points should be not far from origin and destination points that half of maximum allowed detour distance (showed by dotted circle around points *A* and B on Fig. 2).
- The driver's detour should not exceed the maximum allowed detour.

During the meeting points finding the following goal functions have to be considered:

- Driver’s path length should tends to minimum;
- Waiting time for driver and passenger should both tends to minimum value;
- Distance between origin and meeting point as well as between pick-off and destination point for pedestrian should tends to minimum value.

It has been calculated that the presented algorithm has exponential complexity. In this case the matching can work quickly for the short amount of drivers and pedestrians but in case of the system scalability the matching time will be unsatisfactory for operation in real time. To reduce the operation time authors propose to use the heuristics that are aimed at reducing search space of drivers and pedestrians. There are two heuristics have been proposed: distance heuristic and sector heuristic.

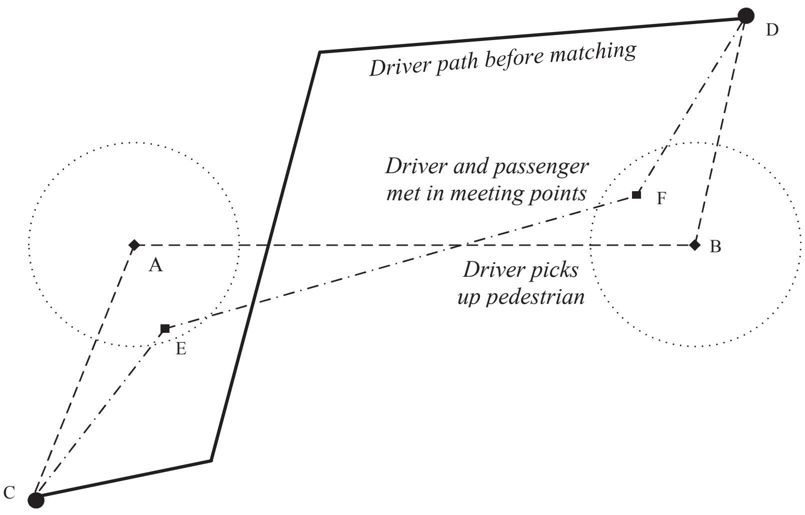
Driver path before matching
Driver and passenger met in meeting points
Driver picks up pedestrian
A
B
C
D
E
F

Fig. 2. Matching driver and pedestrian paths

Distance heuristic is aimed at formation of set of drivers that have a physical possibility to meet the pedestrian (the nearest distance of pedestrian locations and driver path less than their predefined detours as well as the nearest distance of pedestrian destination to the driver path). Formally, it selects meeting points in the intersections of the circles of radius P_{Detour} around the pedestrian location and destination points with the circles of radius D_{Detour} around the points of the driver route.

$$(pp^x_{locat} - dp^x_i)^2 + (pp^y_{locat} - dp^y_i)^2 \le (P_{\det our} + D_{\det our})^2, \quad (1)$$

$$(pp^x_{dest} - dp^x_i)^2 + (pp^y_{dest} - dp^y_i)^2 \le (P_{\det our} + D_{\det our})^2, \quad (2)$$

where pp_{local}, and pp_{dest} is the pedestrian location and destination points, dp_i a set of points of the driver path, and P_{Detour}, D_{Detour} represent the pedestrian and driver detours.

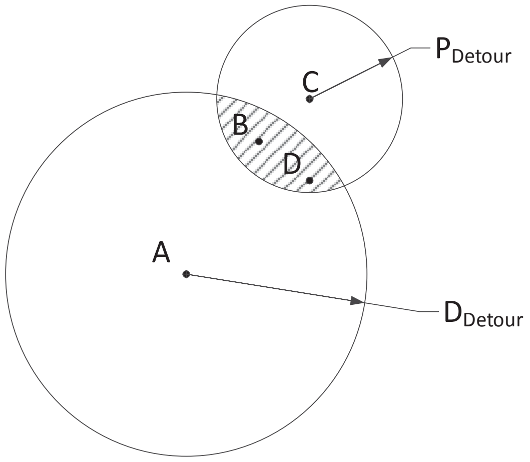

Fig. 3. Distance heuristic graphical representation

The second heuristics is aimed at determine the subset of possible meeting points for the pedestrian in according to the following sector (see Fig. 4 and Fig. 5). In Fig. 4 the situation is showed when one meeting point is acceptable (point C). In this case the angle of the sector is determined by the formula (3).

$$\left[\theta - \frac{\pi}{4}, \theta + \frac{\pi}{4}\right], \quad (3)$$

that is corresponded to points L and M of the pedestrian path in Fig. 4 and angle θ is defined in the following way:

$$\theta = arctg\left(\frac{C^y - A^y}{C^x - A^x}\right). \quad (4)$$

Location point (A) of the pedestrian path can be a potential meeting point. In general case several meeting points should be considered (e.g. L and M in Fig. 5). In this case additional possible meeting points should be included to the possible meeting point set (point L, M, N).

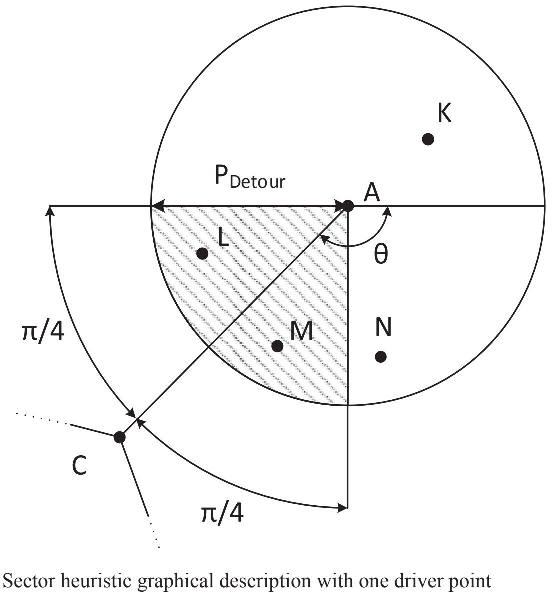

Fig. 4. Sector heuristic graphical description with one driver point

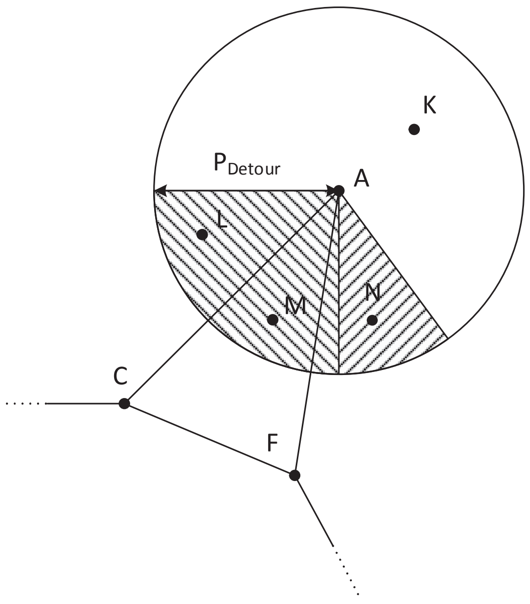

Fig. 5. Sector heuristic graphical description with two driver's points

The presented sector heuristic reduce the search space but in case of unbalanced map some with rivers and lakes some possible meeting points can be lost. Both heuristics requires the following constrains that should be met to work effectively.

- Large number of drivers should participate in the ridesharing. Heuristics are aimed at filtering points. If the drivers count will be small the situation can occur that the pedestrian will not find a driver. However if the small amount of drivers are participated in the ridesharing the matching algorithm can be used without the heuristics.
- In case the drivers will specify the large amount of detour (taxi drivers start to participate in ridesharing) the heuristics will not reduce the search space.
- Balanced map (uniform distribution of roads on the map rivers and lakes absence). Unbalanced maps caused of the meeting point losses due to the sector heuristics utilization.

VI. Implementation

The presented in the paper algorithm and heuristics have been tested on random pedestrians and drivers datasets (see, Table I) that includes coordinates of location and destination points of a pedestrian as well as driver paths. Experiments of the algorithm testing show that heuristics help to reduce the time of search in more than *1.5* times. The following hardware have been used for the experiments: Intel Pentium 4, 1.6 GHz, RAM: DDR1 512 Mb.

TABLE I: Results of the experiments

Drivers	Pedestrian s	Matching time, sec
1	1	0,0135
5	5	0,0316
10	10	0,0641
20	20	0,2248
40	40	1,5462
60	60	2,2416
80	80	3,4725

Relationship between matching time and number of drivers and pedestrians has been shown in Fig. 6. It can be concluded from the experiments that algorithm operation time for 100 simultaneous drivers and pedestrians is about 4 seconds that is applicable time to provide results to the user that try to find a driver for ridesharing online.

To increase performance of proposed algorithms the following features was considered. The first feature is the consideration road graph specific while searching for potential meeting points. The ordinal spatial query allows to find the nearest points within a certain radius, without regard to their reachability. In other words, the distance between the points is calculated on a spheroid reflecting the shape of the Earth, while the topology of the roads is not taken into account, and points may reach the circle bounded by the selected radius, even if it is longer than the limit specified by the user.

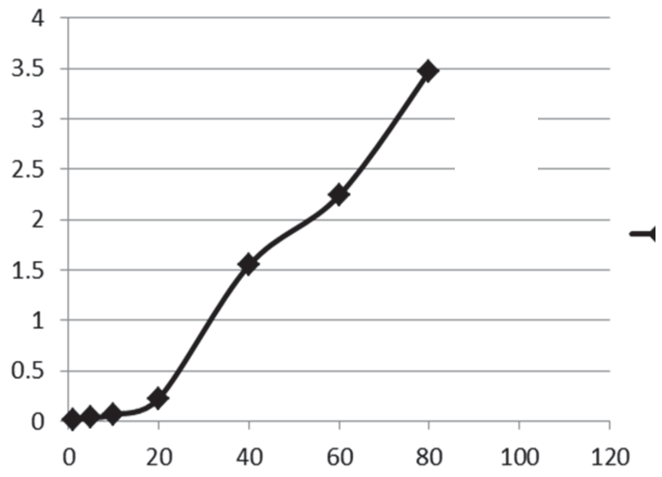

Fig. 6. Relationship between matching time and number of drivers and pedestrians in the service

Checking each found point with Dijkstra's algorithm or A * requires serious time consuming, and therefore, to select points, we use the pgRouting library function - pgr_drivingDistance - which allows you to build a polygon containing points whose path to which does not exceed the specified value. Fig. 7 shows an example of a situation where the use of the pgr_drivingDistance function provides points exclusion that falling into a circle of radius Distance centered at the user's current location (point A), but the actual path to which from this point is greater than the Distance parameter (points G, F, I). As a result of the function, only four potential meeting points remain (A, D, L, H). The second feature to increase performance is the calculation of routes between one start and several end points. At the stage of enumerating options for a joint trip, it is required to calculate many driver paths to potential meeting points, many combinations of paths from potential pick-on points to pick-off points and many paths from pick-off points to the next point of the driver’s path.

The use of Dijkstra or A * algorithm at each stage requires significant computational and time costs, and therefore optimization of the calculation of path combinations at each stage is required. The possibility of such optimization is provided in the pgRouting library and implemented in the pgr_kdijkstraCost function. This function allows to find the path length from a selected point to an array of points in one pass of the Dijkstra algorithm. Moreover, the more points are indicated in the array, the greater the gain is obtained from using this function. Runtime comparison for eight destinations with and without filters is shown in the Fig. 8.

VII. Conclusion

The paper proposes the ridesharing feature utilization for the popular in the market at the moment car sharing service providers. If the car sharing user cannot find a car nearby, he/she can use the ride sharing feature instead of a taxi. In compared the taxi the feature significantly reduces transportation costs for the pedestrian as well as for the driver who participate in the ridesharing. We propose the algorithm for matching of the driver routes with the pedestrian location and destination points. For the proposed algorithm we propose two heuristics that significantly reduced time complexity and allow to implement the algorithm for real-time ridesharing. Evaluation shows that the proposed algorithm can be used in real time to finding the matching between drivers and pedestrians routes.

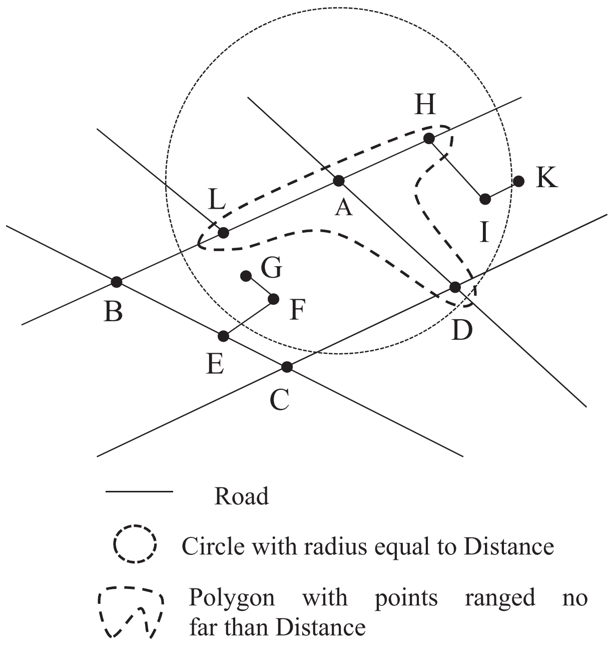
H
K
L
A
I
G
F
B
D
E
C
Road
Circle with radius equal to Distance
Polygon with points ranged no far than Distance

Fig. 7. The result of the pgr_drivingDistance function

Acknowledgment

The algorithm has been developed in scope of the project financed by Government of Russian Federation (Grant 08-08). Proposed heuristics have been developed in scope of RFBR project # 18-37-00337. Implementation of the algorithm has been done in scope of RFBR project # 18-37-00377.

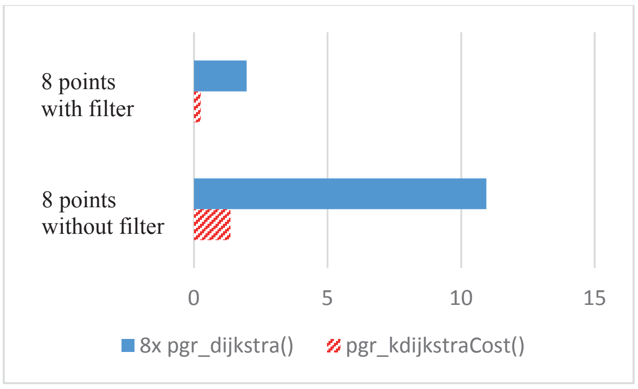

Fig. 8. Comparison of distance calculation time for eight points using Dijkstra's algorithm and using kDijkstra's algorithm

References

[1] K. Münzel, L. Piscicelli, W. Boon, K. Frenken, “Different business models – different users? Uncovering the motives and characteristics of business-to-consumer and peer-to-peer carsharing adopters in the Netherlands”, *Transportation Research Part D: Transport and Environment*, vol. 73, 2019, pp. 276-306.
[2] W. Abrahamse and M. Keall, “Effectiveness of a web-based intervention to encourage carpooling to work: A case study of Wellington”, *Transport Policy*, vol. 21, 2012, pp. 45–51.
[3] A. Cho, A. Yasar, L. Knapen T. Bellemans, D. Janssens, and G. Wets, “A conceptual design of an agent-based interaction model for the carpooling application”, *In the 1-st Internationsl Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications*, vol. 10, 2012, pp. 801–807.
[4] N. Teslya, A. Kashevnik, and A. Pashkin, “Context-Based Access Control for Ridesharing Service”, *Proceedings of the 14th Conference of Open Innovations Association FRUCT*, Espoo, Finland, 2013, pp. 156–163.
[5] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, and S. Laizane, “Smart Space-based Ridesharing Service in e-Tourism Application for Karelia Region Accessibility: Ontology-based Approach and Implementation”, *8th International Joint Conference on Software Technologies*, Reykjavik, Iceland, 2013, pp. 591-598.
[6] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, “OpenStreetMap-Based Dynamic Ridesharing Service”, *Information Fusion and Geographic Information Systems*, Saint Petersburg, Russia, 2013, pp. 103–118.
[7] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, “Smart Logistic Service for Dynamic Ridesharing”, *Internet of Things, Smart Spaces, and Next Generation Networking*, St. Petersburg, Russia, 2012, pp. 140–151.
[8] eRideShare.com. URL: http://erideshare.com, last access date 15.09.2019.
[9] PickupPal. URL: http://www.pickuppal.com, last access date 15.09.2019.
[10] Zimride. URL: http://www.zimride.com, last access date 15.09.2019.
[11] RideshareOnline. URL: http://www.rideshareonline.com, last access date 15.09.2019.
[12] Rideshare 511. URL: http://rideshare.511.org, last access date 15.09.2019.
[13] CarJungle. URL: http://www.carjungle.ru, last access date 15.09.2019.
[14] Avego. URL: http://www.avego.com, last access date 15.09.2019.
[15] M. Nourinejad and M. J. Roorda, “Agent based model for dynamic ridesharing,” Transp. Res. Part C Emerg. Technol., vol. 64, pp. 117–132, 2016.
[16] J. Alonso-mora et al., “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment,” Proc. Natl. Acad. Sci., vol. 115, no. 3, pp. E555–E555, Jan. 2018.
[17] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, “The benefits of meeting points in ride-sharing systems,” Transp. Res. Part B Methodol., vol. 82, pp. 36–53, 2015.

[18] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, “Enhancing urban mobility: Integrating ride-sharing and public transit,” Comput. Oper. Res., vol. 90, pp. 12–21, 2018.

[19] Y. Liu and Y. Li, “Pricing scheme design of ridesharing program in morning commute problem,” Transp. Res. Part C Emerg. Technol., vol. 79, pp. 156–177, 2017.

[20] M. Feeney, “Is Ridesharing Safe?,” Policy Anal., no. 767, pp. 1–15, 2015.

[21] U. M. Aïvodji, S. Gambs, M. J. Huguet, and M. O. Killijian, “Meeting points in ridesharing: A privacy-preserving approach,” Transp. Res. Part C Emerg. Technol., vol. 72, pp. 239–253, 2016.