Multimodal Shortest Path Algorithm for Carsharing Systems with Operation Area Constraint

Wesam Herbawi and Stefan Landsbek
moovel Lab
moovel Group GmbH
Stuttgart, Germany
Email: firstname.lastname@moovel.com

Abstract

This paper studies the problem of computing the shortest path in free floating carsharing systems with operation area constraint. In this type of systems, The shortest path, between a given source and destination points, consists of a walking part from the source to some carsharing vehicle, then a driving part, probably, followed by a final walking part to the destination. The return of carsharing vehicles is limited by an operation area constraint represented as geographical multipolygon.

In this work we propose a multimodal shortest path algorithm that takes the operation area constraint into consideration to solve the shortest path problem in free floating carsharing. The proposed algorithm has been tested using real world carsharing data. Experimentation results showed that the algorithm was able to successfully solve the problem and cope with the different problems settings in reasonable time suitable for online applications.

1 Introduction

A carsharing system consists of a fleet of vehicles distributed throughout the city and available to the system users for rental on a short notice and usually for a short period for the duration of their trips. This kind of systems is gaining more and more interest and widespread recently.

The types of car sharing systems vary according to the constraints imposed by the system operator. Free floating one-way carsharing (simply called free floating carsharing) is the most flexible type of carsharing where the user can rent a vehicle and return it at any place within the operation area of the carsharing system operator. The operation area is usually defined as a multipolygon with included and excluded geographical areas. Example free floating carsharing systems are car2go (www.car2go.com) and DriveNow (www.drive-now.com).

Finding the shortest path in such systems is a multimodal shortest path problem with the modes walk and drive. It includes finding a walking path from the source to a nearby vehicle, then a driving path to the destination point. Often the destination point is not directly accessible by driving mode. It could be outside the operation area of the carsharing operator or it could be within a pedestrian zone. In both cases, a final walking path has to be computed.

The multimodal shortest path problem has been widely studied. In [Paj09, HJ13, YL12], different solution approaches for the multimodal shortest path problem have been proposed considering different modes including walk, drive and public transport. The driving mode was supposed to be possible everywhere on the street network as long as the street types allow for driving. This is because the driving mode was thought for private car or for a taxi. However, to the best of our knowledge, no work has considered the multimodal shortest path problem for carsharing where the driving mode cannot be started and ended everywhere on the street network, nonetheless, limited by the availability of vehicles and by the operation area constraint.

In this work, we integrate the operation area con-

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

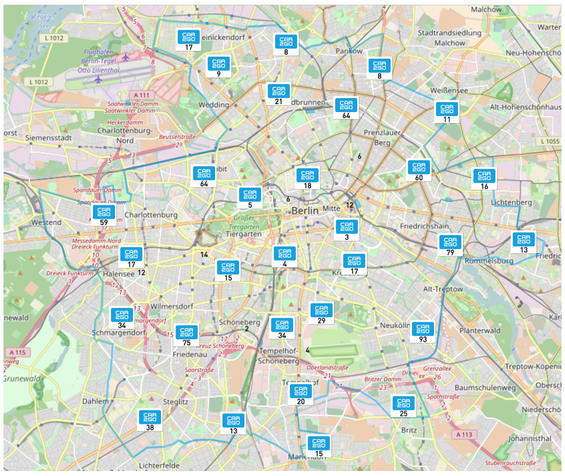


Figure 1: Sample free floating carsharing system settings. A set of vehicles distributed through the city along with a multipolygon operation area constraint. Data is taken from car2go. Blue icons are car2go vehicles (clustered for visualization purposes) and the blue multipolygon represents the main part of the operation area.

straint in the graph representation of the street network and propose a multimodal shortest path algorithm to answer shortest path queries of the form walk-carsharing-walk. The algorithm takes the operation area constraint into consideration while computing the multimodal shortest path.

The rest of the paper is organized as follows. We describe the problem in more details in Section 2. The proposed algorithm is explained in Section 3 followed by the experimentation and results in Section 4. Finally, we outline the conclusions of the paper.

2 Problem description and modeling

Typically, a free floating carsharing system has a set of vehicles on specific geolocations and a geographical multipolygon defining the operation area of the carsharing system as shown in Figure 1. Carsharing rental can be started everywhere where a vehicle is available and can be ended only within the operation area defined by the multipolygon. The task is to find the shortest path on the street network between a given source s and target t points (later will be denoted nodes) using the modes walk and drive. The mode walk can be used everywhere on the street network as long as the street types allow for walk mode. A switch from walk mode to drive mode can be triggered only if a carsharing vehicle is reached. Obviously the walk mode cannot be simply stopped after reaching a carsharing vehicle. It might be necessary to walk a bit farther to reach another carsharing vehicle that might result in the shortest path. Such a vehicle might be located on a street where no detour is required to head toward the destination. This problem is a multimodal shortest path problem where a switch between different modes of transport is required. The shortest path might include a walk to some carsharing vehicle followed by a drive segment and ends with another walk segment. The operation area constraint is a new component to this type of problems and to our knowledge, this is the first work to consider such constraint. In the following, we provide the modeling of the different components of the problem.

The street network is represented as a diagraph $G = (V, E), E \subseteq V \times V$. We assume that the set of nodes V consists of integer values in the range $[0, |V|)$. We use the functions w and $d \colon E \to \{true, false\}$ to denote if an edge $e \in E$ is accessible by walk and drive modes respectively. Edges E are annotated with travel time for both modes walk and drive. We use the functions $time_w$ and $time_d \colon E \to \mathbb{R}_{\geq 0}$ to denote the travel times of an edge $e \in E$ for the modes walk and drive respectively with the following hold $w(e) \oplus (time_w(e) = \infty)$ and $d(e) \oplus (time_d(e) = \infty)$ i.e. if an edge e is not accessible for some mode, then the travel time of that mode on e is infinity.

For each carsharing vehicle, we find the geographically closest node $v \in V$ to get the set $C \subseteq V$ of nodes that will represent the carsharing vehicles in our graph G. New nodes are added to V if necessary, to represent vehicles that are located close to long edges and far from the end nodes of the edges. Using the operation area multipolygon, we generate the set of nodes that are geographically withing the operation area. This set is denoted by $O = \{v \in V \mid v \text{ is geographically within operation area}\}, C \subseteq O \subseteq V$.

3 The proposed algorithm

To solve the multimodal shortest path problem in free floating carsharing, we propose an extension of Dijkstra algorithm [Dij59], that alternates between the modes walk and drive, and takes the operation area constraint into consideration. Algorithm 1 is a pseudocode representation of the proposed algorithm.

Algorithm 1 follows the basic structure of Dijkstra algorithm. However, the different modes of travel and the operation area constraint have to be taken into consideration. The mode switch (from walk to drive and vice versa) and its trigger has to be handled in the algorithm. In comparison to single mode shortest path problem, in multimodal shortest path, the same node could have different predecessors at the same time (namely, one per travel mode) as shown in Figure 2. This happens, for example, when the system user has to walk in some direction to reach a vehicle and then drive back the same way. This behavior will

Algorithm 1: Multimodal Operation Area Aware Dijkstra

```
   begin
      Input: $G = (V, E)$ source $s \in V$ target $t \in V$
             vehicle nodes $C \subseteq V$ Operation area
             nodes $O \subseteq V$
      Output: Multimodal shortest path tree
              rooted at $s$
1     Q ← Priority Queue
2     Q.insert(s,0)
3     $d[i] \leftarrow \infty, i \in 0, 1, .., 2|V|$
4     $d[s] \leftarrow 0$
5     while $!Q.isEmpty()$ do
6        v ← Q.dequeue()
7        if $v = t$ or $((v \bmod |V| = t)$ and $t \in O)$
          then
8           Stop
9        for each $e = (v \bmod |V|, u) \in E$ do
10          if $v < |V|$ or $v \bmod |V| \in O$ then
11             $tmpD \leftarrow d[v] + time_w(e)$
12             if $tmpD < d[u]$ then
13                if $d[u] = \infty$ then
14                   $Q.insert(u, tmpD)$
                  else
15                   $Q.decreaseKey(u, tmpD)$
16                $d[u] \leftarrow tmpD$
17                $pred[u] \leftarrow v$
18          if $v \in C$ or $v \geq |V|$ then
19             $tmpD \leftarrow d[v] + time_d(e)$
20             if $tmpD < d[u + |V|]$ then
21                if $d[u + |V|] = \infty$ then
22                   $Q.insert(u + |V|, tmpD)$
                  else
23                   $Q.decreaseKey(u + |V|, tmpD)$
24                $d[u + |V|] \leftarrow tmpD$
25                $pred[u + |V|] \leftarrow v$
```

result in some nodes, where the same node is reached at different durations from different predecessor nodes and despite that the different predecessors have to be accepted as valid predecessors. In single mode shortest path, this behavior is not allowed. If node x is reached through predecessor y with duration d_1, then no predecessor for x will be accepted with duration $d_2 \geq d_1$. An edge base traversal Dijkstra will manage to solve the problem depicted in Figure 2. However, it will fail in handling the case where the same edge has to be reached both by walk and drive even if the drive mode results in less duration. Walking a bit farther might result in the shortest path as explained earlier in this section.

At the early stages of the algorithm, it behaves as a typical Dijkstra exploring G in the walk mode (lines $10 - 17$). A mode switch from walk to drive is triggered once a node $v \in C$ is settled (removed from the priority queue). This means a carsharing vehicle is reached, and from there one can start driving mode (line 18). Once we start exploring G in drive mode, we might face the problem depicted in Figure 2. Node b has already been reached through a with duration $time_w((a, b))$. Once the node c is settled, we can start exploring in mode drive. Now the algorithm tries to reach node b through node c with a total duration of $time_w((a, b)) + time_w((b, c)) + time_d((c, b)) \geq time_w((a, b))$. In a single mode shortest path, this step is not allowed as b has already been reached with less duration through a. However, in our case, this step should be allowed as the vehicle has to be reached first and the shortest path might be the one with the vehicle driving back the walk path. The algorithm handles that by making local graph copies. For each node $v \in V$ that is to be explored in drive mode, the algorithm makes a copy of v and assign it the value $v + |V|$. Now the new node $v + |V|$ can be reached with a higher duration compared to its walk mode node v (lines 20 $-$ 25). Note that the node $v + |V|$ preserves all the attributes and outgoing edges as $(v + |V|) \bmod |V| = v$. Now, any node $v \geq |V|$ indicates that the algorithm is exploring the driving mode. Hence, the condition at line 18 indicates that we can explore nodes in driving mode either if the settled node is a vehicle node $v \in C$ or it is already a driving mode node $v \geq |V|$.

The operation area constraint is enforced by the set O. It is used to define a valid transition from the drive mode to the walk mode in line 10. A transition from drive to walk is allowed only if the settled drive node is within the operation area $(v + |V|) \bmod |V| = v \in O$. This means that a carsharing vehicle can be returned only within the operation area. The set O is also used to define a valid algorithm stop condition at line 8 where the solution is found. For a valid stop condition, the settled node v has to be in a walk mode or the

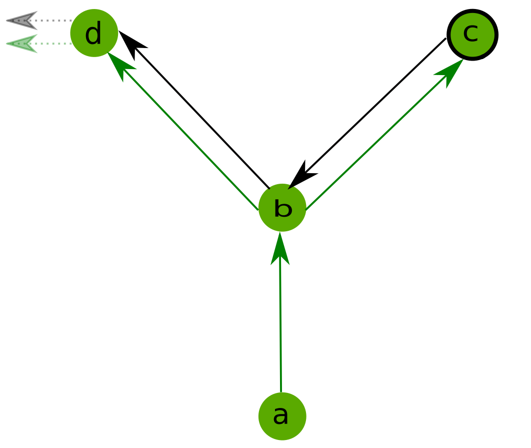


Figure 2: Multiple predecessors problem. Node b is reached by walk mode (green) through a and by drive mode (black) through c

destination has to be within the operation area, $t \in O$, if v is in drive mode. It could happen that $v \geq |V|$ and $v \mod |V| = t$ but $t \notin O$. This means the destination is reached in drive mode but it is outside the operation, yet solution is not found.

What the algorithm conceptually does is shown in Figure 3. It splits the graph into two graphs, one for walk and another for drive. The two graphs are bidirectional connected at vehicle nodes and one directionally connected, from drive to walk, at nodes within the operation area. The algorithm does that on the fly and locally. Instead of splitting the whole graph into two graphs, the algorithm only splits the part of the graph that is visited during the search. The visited part is usually small compared to the size of the graph as carsharing trips are usually short distance. Also, this approach is more suitable to the highly dynamic nature of the problem as vehicles updates typically happen at least once a minute. The algorithm has the same computational complexity, $O(E \log V)$, as Dijkstra. It operates in the same way as Dijkstra but on a larger graph with a maximum graph size factor of 2 if the whole graph is visited during the search.

4 Experientation and Results

We have tested the proposed algorithm using real world carsharing data provided by car2go. The car2go dataset contains 945 vehicles operating in Berlin in an operation area as shown in Figure 4. The Openstreet map (OSM) data of Berlin is used to build the street network graph. Our algorithm is implemented as an extra module plugged to Graphhoper (graphhopper.com). Experiments are performed on a computer with 4G RAM and 2.5GHz, 64bit dual core processor.

Figure 5 shows the results of the algorithm for different carsharing use cases. In 5(a), the source and

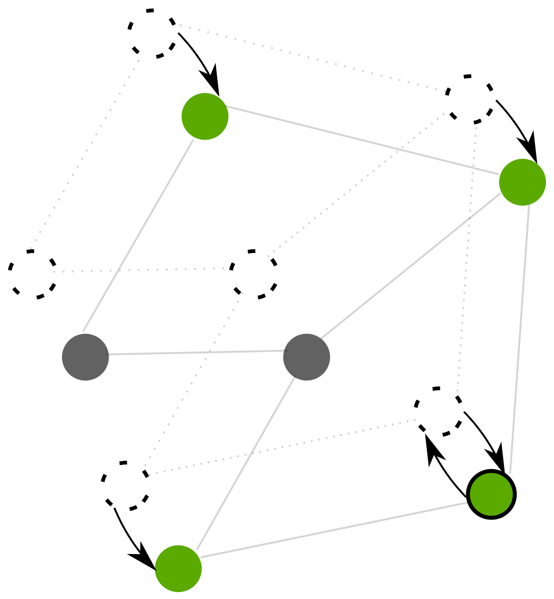


Figure 3: Pictorial representation of the algorithm behavior. Green and Gray nodes are nodes inside and outside the operation area respectively. Dashed nodes are the node copies. The node with black stroke represents a vehicle node.

destination points are within the operation area and the destination is reachable by drive mode. The shortest path consists of a walking part to reach a vehicle followed by a drive part. No final walk is required as the destination is reachable by drive mode. The destination point is outside the operation in 5(b) and therefore the destination has to be reached walking as the vehicle has to be returned within the operation area. 5(c) shows the result of the algorithm when the source is within one of the polygons defining the operation area and the destination is close to another polygon. The vehicle is returned in the polygon close to the destination and a walking part to the destination is computed. A single mode walking result is computed in 5(d) as the end points are close to each other. Often, carsharing providers exclude parts of the operation and, hence, it is not allowed to return the vehicle in such excluded parts. 5(e) shows the result when the destination is within an excluded area.

Table 1 summarizes the average runtime and number of settled nodes for different scenarios of carsharing trips. For each scenario, 10 different trips, between randomly selected source and destination points, have been computed. The trips are categorized as short and long distance. We consider trips of an average distance 7-8km as short and trips of an average distance 23-27km as long. While both, short and long distance trips in our study, are considered very short trips for typical single mode routing, still a 23km trip in carsharing systems is considered a long one. The first sce-

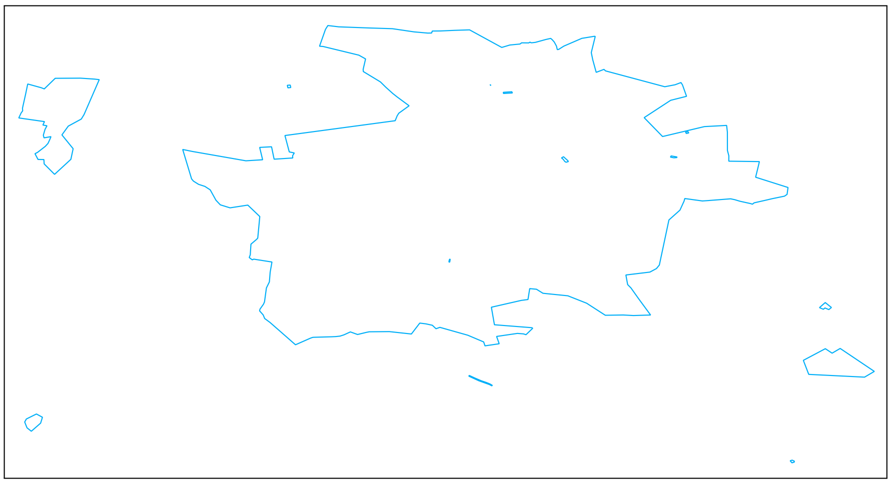


Figure 4: Operation area of car2go in Berlin

Table 1: Experimentation Results. runtime in milliseconds, number of settled nodes, total trip distance of all modes and the walk distance

	drive				walk-drive				drive-walk			
	runtime	settled	total distance	walk distance	runtime	settled	total distance	walk distance	runtime	settled	total distance	walk distance
Short	20	16551	7732	0	19	15761	7962	1039	60	51767	6975	1062
Long	61	49561	23514	0	60	50698	27230	1015	110	95402	23987	1060

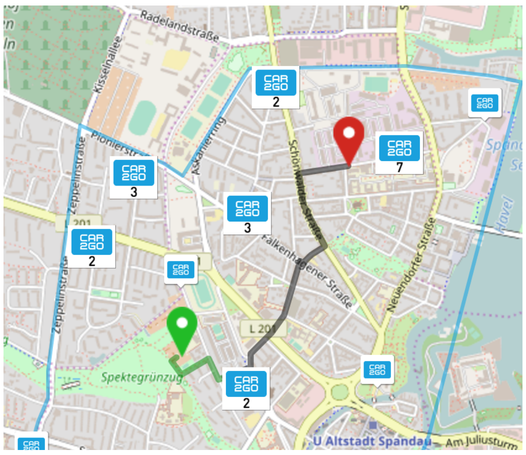
nario of carsharing trips is the drive only trip where a carsharing vehicle is available directly at the source and can be returned at the destination. No walk is required. The second scenario is the walk-drive where some walk is needed to reach the vehicle and the destination can be directly reached by the vehicle. In the last scenario, drive-walk, the vehicle is directly available at the source but some walk is needed at the end to reach the destination.

We notice that there is no considerable difference in the runtime and the number of settled nodes between the scenarios drive and walk-drive. However, a considerable higher runtime and number of settled nodes for drive-walk on both short and long trips. This is because, when the destination is not directly reachable by the vehicle, either because of being in a pedestrian zone or outside the operation area, the algorithm settles high number of drive mode nodes. It could happen that the algorithm reaches the destination in drive mode but cannot stop the search as it is outside the operation area. The algorithm then has to explore more walk nodes to reach the destination. As the walk mode is slower than the drive mode, walk mode nodes get less priority in the priority queue, as they have higher duration, and the algorithm tends to settle more and more drive mode nodes. In other words, having a walk at the end results in longer trip duration till the destination is reached. The algorithm settles all nodes with duration less than the duration needed to reach the destination and therefore it settles larger number of drive nodes as driving is usually faster and takes less duration. This behavior does not happen when the walking part is at the beginning of the trip even that it results in longer trip duration. This is because, a walk at the beginning of the trip means that a vehicle is not reached yet and drive mode is not active. So the algorithm mainly expands walking nodes.

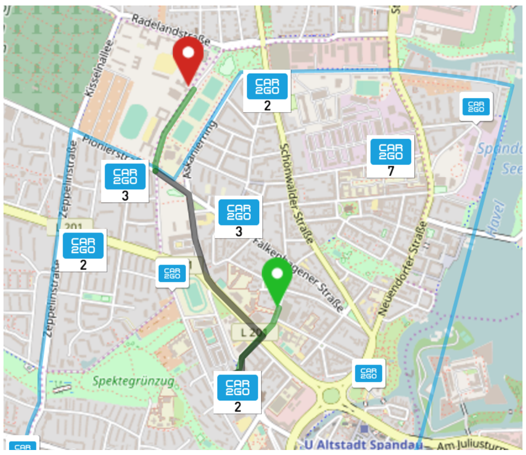
5 Final remark

In this work we have proposed an algorithm for the multimodal shortest path problem in free-floating carsharing systems with operation area constraint. The algorithm has been tested, under different trip scenarios, using real world carsharing data. Test results showed that the algorithm was able to efficiently solve the problem.

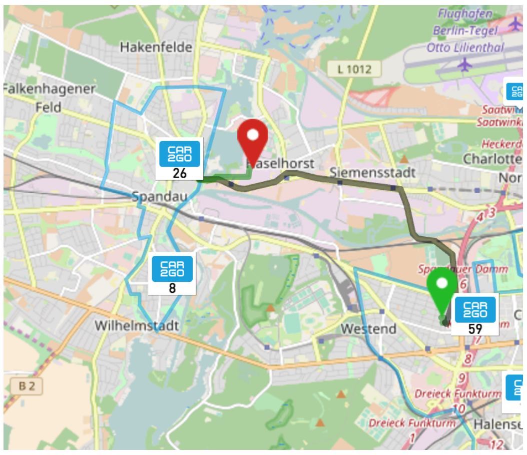
An interesting extension of this work is to combine the carsharing routing with public transport. This is very interesting especially if the destination is far from the operation area, then a last mile public transport could be taken instead of walking. This will affect the return point near the borders of the operation area depending on the availability of public transport stops and trips.



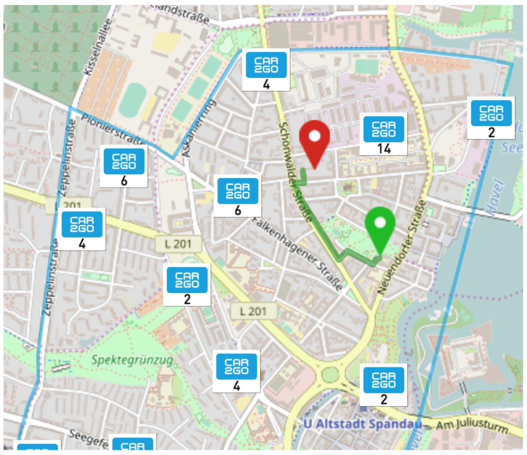
(a) source and destination are within the operation area



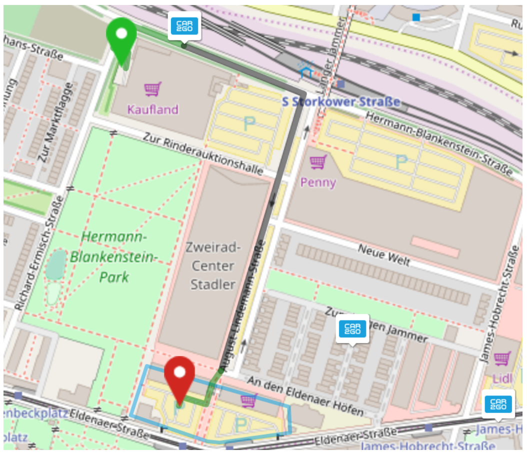
(b) destination is outside the operation area



(c) destination is outside the operation area and close to a polygon other than the polygon of the source



(d) close source and destination



(e) destination is in excluded polygon

Figure 5: Algorithm results for different carsharing use cases based on different positions of the source (green) and destination (red). Green is walk and black is drive.

References

[Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[HJ13] Jan Hrncir and Michal Jakob. Generalised time-dependent graphs for fully multimodal journey planning. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 2138–2145. IEEE, 2013.

[Paj09] Thomas Pajor. Multi-modal route planning. *Universität Karlsruhe*, 2009.

[YL12] Haicong Yu and Feng Lu. A multi-modal route planning approach with an improved genetic algorithm. *Advances in Geo-Spatial Information Science*, 1:0, 2012.