DE GRUYTER OPEN

JAISCR, 2014, Vol. 4, No. 4, pp. 287 – 299
10.1515/jaiscr-2015-0015

A PRIORI APPROACH OF REAL-TIME RIDESHARING PROBLEM WITH INTERMEDIATE MEETING LOCATIONS

Kamel Aissat[1] and Ammar Oulamara[2]

[1]*Loria laboratory, University of Lorraine, Nancy, France*
Kamel.Aissat@loria.fr

[2]*University of Lorraine, Ile de Saulcy, Metz, France*
Ammar.Oulamara@univ-lorraine.fr

Abstract

Ridesharing is a mobility concept in which a trip is shared by a vehicle's driver and one or more passengers called riders. Ridesharing is considered as a more environmentally friendly alternative to single driver commutes in pollution-creating vehicles on overcrowded streets. In this paper, we present the core of a new strategy of the ridesharing system, making it more flexible and competitive than the recurring system. More precisely, we allow the driver and the rider to meet each other at an intermediate starting location and to separate at another intermediate ending location not necessarily their origins and destinations, respectively. This allows to reduce both the driver's detour and the total travel cost. The term "*A priori approach*" means that the driver sets the sharing cost rate on the common path with rider in advance. An exact and heuristic approaches to identify meeting locations, while minimizing the total travel cost of both driver and rider are proposed. Finally, we analyze their empirical performance on a set of real road networks consisting of up to 3,5 million nodes and 8,7 million edges. Our experimental results show that our heuristics provide efficient performances within short CPU times and improves the recurring ridesharing approach in terms of cost-savings.

1 Introduction

Ridesharing is a mobility concept in which a trip is shared by a vehicle's driver and one or more passengers called riders. The idea is based on a better use of private vehicles and concerns people who are willing to intelligently ride in order to save money. Ridesharing is also considered as a more environmentally friendly alternative to single driver commutes in pollution-creating vehicles on overcrowded streets.

The effective use of new communication capabilities, including mobile technology and global positioning system (GPS), enabled the emergence of dynamic or real-time ridesharing systems. It consists in automatically and instantly matching riders-drivers through a network service by using a smartphone both as a geolocation and a communication device. The ability of a dynamic ridesharing system to successfully provide an instant matching depends (i) on the characteristics of the environment in terms of density of users of that service, traffic patterns and availability of roadway and transit infrastructure (ii) on the efficiency of implemented algorithms to tackle the underlying decision problems, such as optimal instant matching of drivers and riders and efficient route-planning algorithms.

In the recurring ridesharing problem, when an offer is matched with a demand, the driver picks-up the rider at his starting location, drops him off at his ending location and continues to his target location.

This approach is the most implemented in ridesharing services, but unfortunately, lacks flexibility and misses some possible matchings. However, in order to get more opportunities of ridesharing, a rider can accept two different intermediate meeting locations. One for pick-up and another for the drop-off. Specifically, the rider travels by his own to the first intermediate location, where he will be picked up by the driver and dropped off at the second intermediate location, where he will continue to his ending location. These two intermediate locations will be determined in a way that the total travel cost is minimized, and the detour costs of the driver and the rider remain reasonable. The majority of ridesharing services ignore possible intermediate stops, if they are not mentioned explicitly beforehand by the rider or the driver. Thus, solving the problem of matching riders and drivers with intermediate locations requires developing efficient algorithms.

In this paper we consider the dynamic ridesharing problem in which a driver and a rider want to travel from their origins to their destinations. The driver may pick up the rider at an intermediate location and then to drop him off at another intermediate location. The driver sets the sharing cost rate on the common path with the rider in advance. The objective is to identify the best intermediate locations that minimize the total travel cost of both the driver and the rider, while ensuring that their detour costs remain reasonable.

(a)

(b)

Figure 1: Figure [a] represents the shortest paths of the rider (green path) and the driver (red path) before matching. Figure [b] represents the new paths of the driver and the rider after the matching with two new intermediate locations (intermediate meeting location are pointed with two black arrows and the common path is represented with blue path).

The paper is organized as follows. In section 2, we provide a concise survey on different problems and approaches related to ridesharing. In section 3, we detail the model of ridesharing with two intermediate locations. In section 4, we provide exact and heuristic approaches. In Section 5, we analyze the performance of our algorithms. Finally, concluding remarks are given in Section 6.

2 Background

In recent years, there has been a growing interest in ridesharing systems both in academia and industry (see [1] and [2]). The ridesharing process is the following: a user indicating his status (i.e. driver or rider) generates a request through his smartphone. The request is sent to a central server. Potential drivers/riders are then contacted. Those might accept or reject the request. If several drivers/riders have accepted the request, then the rider/driver might choose which one he prefers. In addition to the user's status, other informations must be specified. More precisely, if the user is identified as a driver, then he specifies if he's willing to take a single rider or may be willing to take multiple riders. On the other hand, if he's identified as a rider, he specifies if he wants to ride with a single driver or may accept to ride with multiple drivers and be transferred from one driver to another. Hence, four variants can be dis-

tinguished, namely, single-driver and single-rider, single-driver and multiple-riders, multiple-drivers and single-rider, and finally multiple-drivers and multiple-riders. Table 1 shows the different cases that might arise when a user enters the ridesharing system.

Table 1: Variants of ridesharing system

	Single rider	Multiple riders
Single driver	Simple matching between driver/rider	⋆ One pick-up and drop-off location ⋆ Multiple pick-up and drop-off locations
Multiple drivers	Rider is transferred between drivers	⋆ Routing drivers ⋆ Routing riders

Single-driver and Single-rider problem. In this variant, each driver shares his trip with at most one rider. So, for a given rider (driver), the system determines the best driver (rider) among a set of drivers (riders) to share his trip. In [10], the authors consider the problem of single-driver and single-rider, the objective is to determine the driver's offer that can be used to serve the rider's request with the minimal detour. In [7], the authors study the matching optimization problem in which given a set of potential users, each user can be either a driver or a rider, and the objective is to minimize the vehicle-kilometers. The decision consists in assigning a role to each user and finding the optimal matching of drivers with riders. The authors formulate the problem as a general network flow problem with additional constraints. The decision consists in assigning a role to each user and finding the optimal matching of drivers with riders.

Single-driver and Multiple-riders problem. For the variant single-driver and multiple-riders, two cases may arise:

(i) the case where the driver is willing to pick-up each rider at his origin and drop him off at his destination. In this case, the system determines a list of riders and an order of pick-up and drop-off of riders. The authors in [4] propose both exact and heuristic methods. Their approaches are based on two integer programming formulations. In [11], the authors develop a local search based heuristic. In [12], the authors model the ride-matching problem with time windows as an optimization problem and they propose a genetic algorithm to solve it. The objective is to minimize the total travel time and the travel distance of the drivers, the total travel time of the riders and to maximize the number of the matches.

(ii) the case where the driver wants to pick-up all riders at the same location and drop them off at the same location. The system determines a list of riders, the best pick-up location among their origins and the best drop-off location between their destinations. To the best of our knowledge, no previous study was devoted to tackle this variant in the literature.

Multiple-drivers and Single-rider problem. The problem of single-rider and multiple-drivers, in which a rider is transferred between drivers is studied by [13]. This particular problem of ridesharing is called "multi-hop ridesharing problem". The objective is to find a route that minimizes the total travel costs and the number of transfers. The problem is modeled as a multi-objective shortest path problem on a time-expanded graph [14]. Recently, a fast algorithm to generate the shortest path satisfying the requirements of multi-hop ridesharing was presented by [15]. The authors consider a high number of stations in the road network where riders may switch from one vehicle to another. Their model was solved by exploiting time-expanded graphs that are used for routing in transportation networks.

Multiple-drivers and Multiple-riders problem. This problem consists in determining simultaneous routing for the drivers and the riders. More precisely, a driver can take several riders and at the same time a rider can ride with multiple drivers. Authors in [5] propose a genetic algorithm to solve this problem with a limitation on the number of drivers, that can be matched with a rider along his trip to two.

The problem of ridesharing with intermediate locations for pick-up and drop-off offers more flexibility and increases the chance to obtain a route planning minimizing total costs and/or total travel time. In fact, by allowing the rider to reach a new pick-up location and to drop him off at another drop-off location not necessarily his destination, allows to reduce the driver's detour and to generate more cost-savings.

To the best of our knowledge, the only work that considers the intermediate locations is addressed in [16]. The authors propose exact and heuristic methods to find the pick-up and drop-off locations in a multi-modal transportation network. Their objective is to minimize the total travel time for both the driver and the rider. Despite the good performance of their heuristic, it can be considered as very time-consuming in the case of real-time ridesharing. In our model, instead of minimizing the total travel time, we minimize the total travel cost that, in our study, is supposed to be time-independent (i.e. the cost of travel between two locations is always fixed and does not depend on time). Furthermore, we consider additional constraints that ensure that the ridesharing incurred cost of the driver (rider) is more attractive than the incurred cost when he travel alone. We suggest heuristic approaches for reducing the number of shortest path computations that are based on the exact pruning algorithm. These heuristic solutions are computed efficiently using a constant number of shortest path computations and provide almost exact results in practice.

3 Problem description and notation

The road network is represented by a weighted graph $G = (V, E)$, where V is the set of nodes and E the set of edges. Nodes model intersections and edges depict street segments. With each edge $(i, j) \in E$ two weights are associated. In our model, $w(i, j)$ represents the traveling cost and $\tau(i, j)$ the traveling time. A path in a graph G is represented by a vector $\mu = (u, \ldots, v)$ of nodes in which two successive nodes are connected by an edge of E. The length $w(\mu)$ of path μ is given by the sum of the weights of all edges in μ. A shortest-path between a source node u and a target node v is the path with minimal weight among all paths from u to v. In the following, a shortest-path (in terms of cost) between node u and node v will be represented by $u \to v$. We consider an offer and a demand of ridesharing represented by $o = (s, t)$ and $d = (s', t')$, respectively, where s and s' (t and t') are the starting (ending) locations of the driver and the rider, respectively. An edge (i, j) has a nonnegative traveling cost $w_k(i, j)$ depending on the fact that the edge is used by driver ($k = o$) or by rider ($k = d$). In the recurring ridesharing, an offer o is matched with a demand d, if and only if there exists a shortest-path $\gamma_{od} = s \to s' \to t' \to t$ in graph G, such that the driver's detour cost (i.e $w_o(\gamma_{od}) - w_o(s \to t)$) remains reasonable, see (1)

$$w_o(\gamma_{od}) - w_o(s \to t) \leq \varepsilon \cdot w_o(s' \to t') \qquad (1)$$

The value of ε can be seen as a sharing cost rate between the rider and the driver on the common path $s' \to t'$. The term $\varepsilon \cdot w_o(s' \to t')$ in (1) is the reward that rider provides to driver. More precisely, it represents the maximal detour that still guarantee an incentive to the driver for pick up the rider at his starting location s' and to drop him off at his ending location t'. Clearly, if $w_o(\gamma_{od}) - w_o(s \to t) > \varepsilon \cdot w_o(s' \to t')$, the driver does not accept the demand d. The authors in [10] show that the reasonable choice of ε is at most 0.5.

In the case of two intermediate locations, if the rider accepts to travel on his own to intermediate location r_1 with a cost $w_d(s' \to r_1)$, where he will be picked up by the driver and dropped off at another intermediate location r_2, the rider will continue his travel from r_2 to t' on his own. Thus, the driver and the rider will share $w_o(r_1 \to r_2)$. Therefore, the rider's cost is $w_d(s' \to r_1) + \varepsilon \cdot w_o(r_1 \to r_2) + w_d(r_2 \to t')$.

In a nutshell, we consider the general problem of ridesharing with two intermediate locations r_1 and r_2 in which driver picks up the rider at the intermediate starting location r_1 and drops him off at the intermediate ending location r_2. The rider travels on his own from s' to r_1 and from r_2 to t' (see Figure 2).

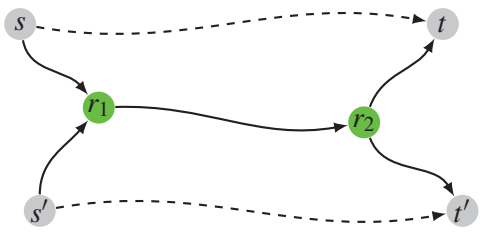


Figure 2: The solid lines symbolize the paths of the driver and the rider in order to form a joint trip, whereas, the dashed ones stand for the shortest paths of the driver and the rider.

In the following, we extend the definition given in [10].

Definition 1 *Let $\varepsilon > 0$ (fixed by the driver), we say that an offer $o = (s,t)$ and a demand $d = (s',t')$ form a reasonable fit if and only if there exist two intermediate locations r_1 and r_2 ($r_2 \neq r_1$) such that:*

$$w_o(s \to r_1) + w_o(r_1 \to r_2) + w_o(r_2 \to t) - w_o(s \to t) \leq \varepsilon \cdot w_o(r_1 \to r_2) \quad (2)$$

$$w_d(s' \to r_1) + w_o(r_1 \to r_2) + w_d(r_2 \to t') - w_d(s' \to t') \leq (1-\varepsilon).w_o(r_1 \to r_2) \quad (3)$$

Applying the ε on the shared path gives both the driver and the rider an incentive to meet in r_1 and to separate in r_2. The constraint (2) ensures that the incurred cost of the driver using rideshare service is less than the incurred cost when he travel alone. The same reasoning is applied for the rider in the constraint (3). In the following, we use the term *global-path* $\langle s,s',r_1,r_2,t,t'\rangle$ to describe the concatenation of paths $s \to r_1$, $s' \to r_1$, $r_1 \to r_2$, $r_2 \to t$ and $r_2 \to t'$. i.e. $\langle s,s',r_1,r_2,t,t'\rangle = s \to r_1 \oplus s' \to r_1 \oplus r_1 \to r_2 \oplus r_2 \to t \oplus r_2 \to t'$.

A shortest *global-path* between source nodes s, s' and target nodes t, t' is the *global-path* with minimal weight $w(s,s',r_1,r_2,t,t')$ of any *global-path* from s, s' to t, t', where

$$w(s,s',r_1,r_2,t,t') = w_o(s \to r_1) + w_d(s' \to r_1) + w_o(r_1 \to r_2) + w_o(r_2 \to t) + w_d(r_2 \to t') \quad (4)$$

The objective is to determine the best intermediate locations r_1 and r_2 that minimize the shortest *global-path* such that constraints (2) and (3) are satisfied.

4 Solution approach

In this section, we propose two solving approaches for our model. The first approach is an enumerative approach, the second approach is a heuristic one. These approaches are based on computing several shortest paths.

Route planning in road networks is solvable by Dijkstra's algorithm [17]. Specifically, with Dijkstra's seminal algorithm being the baseline, a number of techniques pre-process the static input graph to achieve drastic speedup. Contraction Hierarchies (CH) is a speedup-technique that has a convenient trade-off between preprocessing effort and query efficiency. Road network with millions of nodes and edges can be preprocessed in a few minutes while queries run in about a hundred microseconds. Transit Node Routing (TNR) is one of the fastest speed-up techniques for shortest path distance queries in road networks. By preprocessing the input road network even further, it yields almost constant-time queries, in the sense that nearly all queries can be answered by a small number of table lookups.

Based on the matching constraints previously defined, the two approaches start by limiting the search space for the intermediate locations. Especially, given a driver offer $o = (s,t)$ and a rider demand $d = (s',t')$, we determine two subclasses $\mathcal{C}_1$ and $\mathcal{C}_2$ of nodes. Class $\mathcal{C}_1$ ($\mathcal{C}_2$) contains all nodes candidate to be an intermediate starting (ending) location.

Definition 2 *Let $N^{\uparrow}(s)$, $N^{\uparrow}(s')$, $N^{\downarrow}(t)$ and $N^{\downarrow}(t')$ be the sets of nodes, such that,*

$$N^{\uparrow}(s) = \{v \in V \mid w_o(s \to v) + (1-\varepsilon) \cdot w_o(v \to t) \leq w_o(s \to t)\}$$

$$N^{\downarrow}(t) = \{v \in V \mid (1-\varepsilon) \cdot w_o(s \to v) + w_o(v \to t) \leq w_o(s \to t)\}$$

$$N^{\uparrow}(s') = \{v \in V \mid w_d(s' \to v) \leq w_d(s' \to t')\}$$

$$N^{\downarrow}(t') = \{v \in V \mid w_d(v \to t') \leq w_d(s' \to t')\}$$

The set $N^{\uparrow}(s)$ represents the potential intermediate starting locations of the driver. Indeed, if v as the intermediate starting location, the best situation for the driver is to share the cost $w_o(v \to t)$ with the rider, then if the node v does not respect this constraint, it will never satisfy the constraint of cost detour.

On the other hand, the set $N^{\downarrow}(t)$ represents the potential intermediate ending locations of the driver. Taking v as the intermediate ending location, the best situation for the driver in this case is to share the cost $w_o(s \to v)$ with the rider.

The same reasoning is applied to the rider in $N^{\uparrow}(s')$ and $N^{\downarrow}(t')$. The case where the driver and the rider have the same cost, the sets $N^{\uparrow}(s')$ and $N^{\downarrow}(t')$ will become

$$N^{\uparrow}(s') = \{v \in \mathrm{V} \mid w_d(s' \to v) + \varepsilon \cdot w_o(v \to t') \leq w_d(s' \to t')\}$$

$$N^{\downarrow}(t') = \{v \in V \mid \varepsilon \cdot w_o(s' \to v) + w_d(v \to t') \leq w_d(s' \to t')\}$$

The following lemma characterizes the set of potential intermediate locations.

Lemma 1 *A node $v \in V$ is a potential intermediate node if and only if $v \in \mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ where $\mathcal{C}_1 = N^{\uparrow}(s) \cap N^{\uparrow}(s')$ and $\mathcal{C}_2 = N^{\downarrow}(t) \cap N^{\downarrow}(t')$.*

4.1 Enumerative approach

Given two intermediate starting and ending locations, it is easy to check if these locations form *a reasonable fit*, i.e. satisfying constraints (2) and (3). The enumerative approach lists all possible pairs of intermediate locations (r_1, r_2) in $\mathcal{C}$ and selects the pair of intermediate locations (r_1^*, r_2^*) with minimal *global-path* cost and satisfying constraints (2) and (3).

To this aim, we compute (i) two backward one-to-all Dijkstra algorithm, with t and t' as target nodes, respectively, (ii) two forward one-to-all Dijkstra algorithm, with s and s' as source nodes, respectively, and finally (iii) one forward one-to-all Dijkstra algorithm, for each intermediate location r in $\mathcal{C}_1$ as source node (except s and s') to all nodes of $\mathcal{C}_2$. The detailed method is given in Algorithm 1.

Algorithm 1 Enumerative method

1: **Input:** Graph $G(V,E)$, source nodes s, s' and target nodes t, t'
2: **Output:** Intermediate locations r_1^* and r_2^*, *global-path* $GP(s,s',r_1^*,r_2^*,t,t')$.
3: Initialization: $w(GP_{\min}) \leftarrow \infty$.
4: Compute two backward one-to-all Dijkstra algorithm, with t and t' as target nodes in order to determine $\mathcal{C}_1$
5: Compute two forward one-to-all Dijkstra algorithm, with s and s' as source nodes in order to determine $\mathcal{C}_2$
6: **for each** r_1 in $\mathcal{C}_1$ **do**
7: Compute one forward one-to-all Dijkstra algorithm, with r_1 as source node
8: **for each** r_2 in $\mathcal{C}_2$ **do**
9: Compute $w(s,s',r_1,r_2,t,t')$ as described in equation (4)
10: **if** $w(s,s',r_1,r_2,t,t') < w(GP_{\min})$ **and** $r_1 \neq r_2$ **and** constraints (2) and (3) are verified **then**
11: $GP_{\min} \leftarrow w(s,s',r_1,r_2,t,t')$.
12: $(r_1^*, r_2^*) \leftarrow (r_1, r_2)$.
13: **end if**
14: **end for**
15: **end for**

Even if the enumerative approach provides a solution in quadratic complexity time, the computation time can be too long for large instances with real road networks. In particular, when the proposed approach is used in real-time ridesharing problem, a solution should be provided in few seconds. In the following section, we propose heuristic approaches with a lower-time complexity.

4.2 Heuristic approaches

We propose two efficient heuristics for intermediate selection that minimize the objective (4) and satisfy constraints (2) and (3).

To do so, we define two metrics based on travel cost which will be used to separate a class $\mathcal{C}$ into two disjoint subclasses. Thus, class $\mathcal{C}_1$ will contain only potential intermediate starting locations and class $\mathcal{C}_2$ will contain only potential intermediate ending locations. Next, we compute for each node v in $\mathcal{C}_2$ the optimal global path having node v as intermediate ending location. Once global paths have been enumerated, we select the *global-path* with minimal cost that satisfies constraints (2) and (3). In the following, we provide a detailed description of the components of our heuristics.

Closest nodes metric ($\mathcal{M}_1$)

The first metric $\mathcal{M}_1$ classifies the nodes of $\mathcal{C}_1 \cap \mathcal{C}_2$ either in $\mathcal{C}_1$ or in $\mathcal{C}_2$, based on their locations relative to sources nodes s, s' and target nodes t, t'. Indeed, if a node is closer to the source nodes than the target nodes, then this node is removed from $\mathcal{C}_2$, otherwise, the node is removed from $\mathcal{C}_1$.

More formally, given a node $v \in \mathcal{C}_1 \cap \mathcal{C}_2$, if $w_o(s \to v) + w_d(s' \to v) \leq w_o(v \to t) + w_d(v \to t')$ then v is removed from class $\mathcal{C}_2$, otherwise, v is removed from class $\mathcal{C}_1$ (see Figure 3).

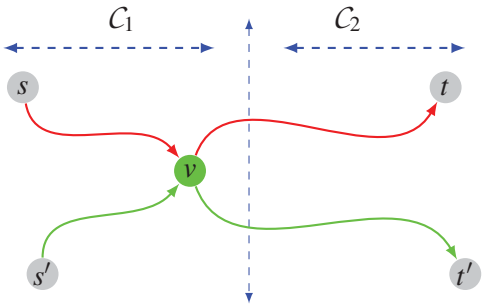


Figure 3: Node classification according to $\mathcal{M}_1$. The node v is close to the origins of rider and driver, so it is classified in $\mathcal{C}_1$, and removed from $\mathcal{C}_2$. Red lines correspond to driver path, while green lines correspond to rider path.

Global path's total cost Metric ($\mathcal{M}_2$)

The second metric $\mathcal{M}_2$ improves the quality of metric $\mathcal{M}_1$ by estimating the cost of *global-path* from s, s' to t, t'. More precisely, given a node $v \in \mathcal{C}_1 \cap \mathcal{C}_2$, if $w_o(s \to v) + w_d(s' \to v) + \alpha \leq w_o(v \to t) + w_d(v \to t') + \beta$, then v is removed from class $\mathcal{C}_2$, otherwise, v is removed from class $\mathcal{C}_1$, where α and β are given as follows:

$$\alpha = \max \begin{cases} \min\limits_{k \neq v} w_o(v,k) + \min\limits_{r \neq v} \{w_o(r \to t) + w_d(r \to t')\} & (5a) \\ \min \begin{cases} w_o(v \to t') + w_o(t' \to t) & (5b) \\ w_o(v \to t) + \min\limits_{r \neq v} w_d(r,t') & (5c) \end{cases} \end{cases}$$

$$\beta = \max \begin{cases} \min\limits_{r \neq v} \{w_o(s \to r) + w_d(s' \to r)\} + \min\limits_{k \neq v} w_o(k,v) & (6a) \\ \min \begin{cases} w_o(s \to s') + w_o(s' \to v) & (6b) \\ w_o(s \to v) + \min\limits_{r \neq v} w_d(s',r) & (6c) \end{cases} \end{cases}$$

If v is a starting intermediate node, α is a lower bound of all paths of the driver and the rider from the starting intermediate location v to their destinations with at least a common sub-path.

It is easy to see that $w_o(v \to u) \geq \min_{k \neq v} w_o(v,k)$ and $w_o(u \to t) + w_d(u \to t') \geq \min_{r \neq v} \{w_o(r \to t) + w_d(r \to t')\}$, then equation (5a) is a valid lower bound.

In equations (5b) and (5c), the ending intermediate node is the target node t' and a node k that belongs to the shortest path $v \to t$, respectively (see Figure 4). β is a lower bound of all paths of the driver and the rider from their starting locations to the intermediate ending location v with at least a common sub-path.

Equations (6a), (6b) and (6c) are symmetric lower bounds corresponding to equations (5a), (5b) and (5c), respectively, when v is the ending intermediate node (see Figure 5).

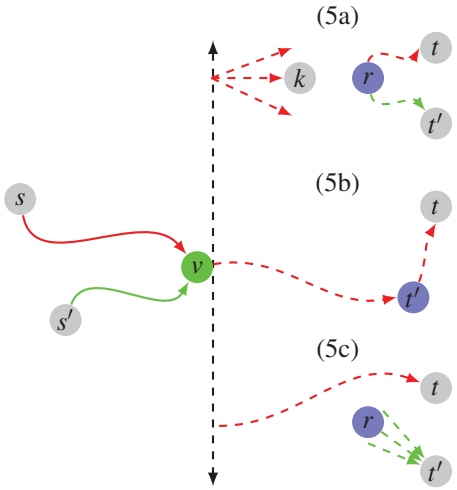


Figure 4: Estimation of *global-path* cost taking v as intermediate meeting location. Red lines correspond to driver paths, while green lines correspond to rider paths. The dashed lines correspond to the estimation of remaining cost from meeting location v to their destinations (α).

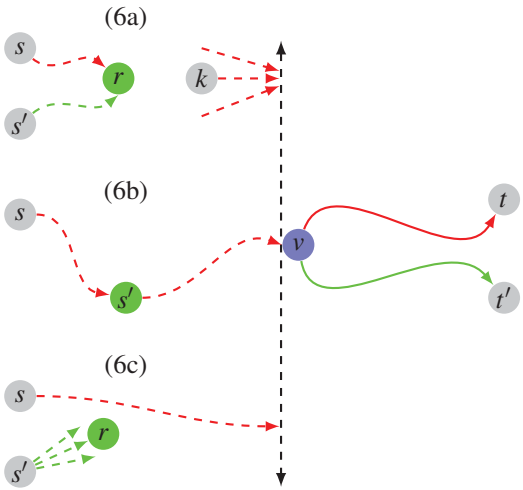


Figure 5: Estimation of *global-path* cost taking v as intermediate separate location. Red lines correspond to driver paths, while green lines correspond to rider paths. The dashed lines correspond to the estimation of remaining cost from their origins to the separate location v (β).

After constructing the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the next step determines the shortest *global-path* with a given intermediate ending node. For this purpose, let us define a new graph $G'(V' = V \cup \{s^\star\}, E' = E \cup E^\star)$, where a specific node $s^\star$ is added to graph G and for each node $u \in \mathcal{C}_1$, an arc $(s^\star, u) \in E^\star$, having a cost $w(s^\star, u) = w_o(s \to u) + w_d(s' \to u)$, is added to graph G. The arc $(s^\star, u)$ represents the driver and the rider moves out from their starting locations to the intermediate node u. Note that the considered arcs' costs in the set E is the driver cost (see Figure 6).

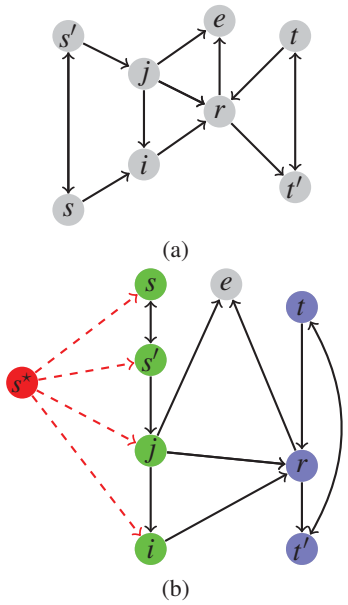


Figure 6: Sub-figure (a) represents an example of a graph G consisting of 8 nodes. The sub-figure (b) represents the new transformed graph G' that consists of two disjoint classes $\mathcal{C}_1$ (green nodes) and $\mathcal{C}_2$ (blue nodes). The node e belongs neither to $\mathcal{C}_1$ nor to $\mathcal{C}_2$.

We denote by $\Phi(v) = (s^\star, \phi(1), \ldots, \phi(k), v)$ a dipath in graph G', where $s^\star$ and v are, respectively, the initial and the final node. Then the following facts hold:

Fact 1 *Any dipath* $\Phi(v) = (s^\star, \phi(1), \ldots, \phi(k), v)$ *in* G' *verifies that* $\phi(1) \in \mathcal{C}_1$.

Fact 2 *Given an optimal dipath* $\Phi(v) = (s^\star, \phi(1), \ldots, \phi(k), v)$, *having* $v \in \mathcal{C}_2$, *with a cost* $w(\Phi)$, *a global-path* $GP(s^\star, s', \phi(1), v, t, t')$ *can be derived from the latter dipath, where* $\phi(1)$ *and* v *correspond to the intermediate starting and ending nodes, respectively, and* $w(GP(s^\star, s', \phi(1), v, t, t')) = w(\Phi(v)) + w_o(v \to t) + w_d(v \to t')$.

A straightforward consequence is the following.

Corollaire. 1 *Let* Ω *be the set of the shortest global paths, such that for each* v *in* $\mathcal{C}_2$, *one shortest global path having* v *as an intermediate ending location is added to* Ω. *Then,* Ω *can be computed in* $O(|V'|\log|V'| + |E'|)$*-time by using the forward one-to-all Dijkstra algorithm with Fibonacci heap from* $s^\star$ *to all nodes in* $\mathcal{C}_2$ *in graph* G'.

Finally, the last step selects the solution with the minimal *global-path* cost that satisfies constraints (2) and (3). The detailed approach is given in Algorithm 2.

Algorithm 2 Heuristic approach with metric $\mathcal{M}_i$

1: **Input:** Graph $G(V, E)$, source nodes s, s' and target nodes t, t'.
2: **Output:** Intermediate locations r_1^* and r_2^*, *global-path* $GP(s, s', r_1^*, r_2^*, t, t')$.
3: Initialization: $\mathcal{C} = \emptyset$, $\mathcal{C}_1 = \emptyset$ and $\mathcal{C}_2 = \emptyset$.
4: Compute two backward one-to-all Dijkstra algorithm with t and t' as target nodes.
5: Compute two forward one-to-all Dijkstra algorithm with s and s' as source nodes.
6: Compute one forward one-to-all Dijkstra algorithm from s' with driver cost.
7: Compute one backward one-to-all Dijkstra algorithm from t' with driver cost.
8: **for** All r in V **do**
9: If $r \in N^\uparrow(s)$ AND $r \in N^\uparrow(s')$ then $\mathcal{C}_1 = \mathcal{C}_1 \cup \{r\}$.
10: If $r \in N^\downarrow(t)$ AND $r \in N^\downarrow(t')$ then $\mathcal{C}_2 = \mathcal{C}_2 \cup \{r\}$.
11: If r in $\mathcal{C}_1 \cap \mathcal{C}_2$, then test the condition of metric $\mathcal{M}_i$ and remove r from the appropriate class $\mathcal{C}_1$ or $\mathcal{C}_2$.
12: **end for**
13: Add node $s^\star$ to graph G and connect $s^\star$ to all nodes of class $\mathcal{C}_1$ with cost $w(s^\star, u) = w_o(s \to u) + w_d(s' \to u)$.
14: Compute the forward one-to-all Dijkstra algorithm on the new constructed graph from $s^\star$ to all nodes in $\mathcal{C}_2$.
15: Select the shortest path in the new constructed graph that satisfies constraints (2) and (3) with two intermediate nodes $r_1^* \in \mathcal{C}_1$ and $r_2^* \in \mathcal{C}_2$.

Complexity. In order to reduce the computational time of our heuristics, we start by computing and saving two backward one-to-all Dijkstra's algorithm with t and t' as target nodes, and two for-

ward one-to-all Dijkstra's algorithm with s and s' as source nodes. We also compute one forward one-to-all Dijkstra's algorithm from s' with driver cost, and one backward one-to-all Dijkstra's algorithm from t' with driver cost. These computations allow us to determine the class $\mathcal{C}$ by computing $N^{\uparrow}(s)$, $N^{\uparrow}(s')$, $N^{\downarrow}(t)$, $N^{\downarrow}(t')$ and the cost of the edges in E'. These steps are done in $O(6(|V|\log|V|+|E|))$-time. Furthermore, the computation of metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ can be done in $O(|V|)$-time. Thus, a collection of *global-path* is obtained by solving a forward one-to-all Dijkstra's algorithm from $s^{\star}$ to all nodes of $\mathcal{C}_2$ in $O(|V'|\log|V'|+|E'|)$-time. Finally, the selection of the best *global-path* that satisfies constraints (2) and (3) can be done in the same time when a collection of *global-paths* is calculated. Thus, it follows that the proposed heuristic can be computed in $O(6(|V|\log|V|+|E|)+|V|+(|V'|\log|V'|+|E'|))$-time. When the driver and the rider have the same cost, the complexity of our heuristic is reduced to $O(4(|V|\log|V|+|E|)+|V|+(|V'|\log|V'|+|E'|))$. In the following, we denote by $\mathcal{HM}_1$ and $\mathcal{HM}_2$ the heuristic versions that use metric $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

4.3 Time windows constraint

The solving approach proposed in the previous section can be easily generalized when departure time windows of the rider and the driver are specified. In fact, in reality, for each offer $o=(s,t)$ and demand $d=(s',t')$, we specify the departure time windows that will be denoted by $\mathcal{TW}_o(s)=[t^{o\_s}_{\min},t^{o\_s}_{\max}]$, $\mathcal{TW}_d(s')=[t^{d\_s'}_{\min},t^{d\_s'}_{\max}]$, respectively. In that case, we need to extend the class $\mathcal{C}_1$ defined in lemma 1.

Initially, all nodes except s, s', have an empty departure window. Then, when we calculate the sets $N^{\uparrow}(s)$ and $N^{\downarrow}(s')$, the departure time window of each reached node will be updated. More precisely, if a node v is settled by s in the set $N^{\uparrow}(s)$, its departure time window will be updated as $\mathcal{TW}_o(v)=[t^{o\_s}_{\min}+\tau_o(s\rightarrow v),t^{o\_s}_{\max}+\tau_o(s\rightarrow v)]$.

The nodes that are in both search spaces $N^{\uparrow}(s)$ and $N^{\downarrow}(s')$ will have two departure time windows, one from the offer o, and another from the demand d. Finally, each node v in $N^{\uparrow}(s)\cap N^{\downarrow}(s')$ (where $\mathcal{TW}_o(v)\cap\mathcal{TW}_d(v)=\emptyset$) is removed from the class $\mathcal{C}_1$, i.e. the rider and the driver can not meet each other at this location. Furthermore, our algorithm can potentially take advantage of this constraint which allows to reduce the number of edges inserted in the constructed graph.

5 Experiments

The proposed heuristics as well as the recurring ridesharing approach were computationally compared. The results were evaluated in terms of quality, number of matchings and running-time.

Environment. All experiments were carried out on an Intel Xeon E5620 2.4 Ghz processor, with 8 GB RAM memory. The tested algorithms were coded in C language. A binary heap was used as the priority queue data structure.

Road network. In our experiments, the USA road graphs derived from the available data of DIMACS[1] challenge website were used. Table 2 lists the 4 USA road networks that were used. Each graph includes the real distances, the transit times and the node coordinates.

Table 2: Instances

Name	Description	# of nodes	# of arcs
E	Eastern USA	3,598,623	8,778,114
FLO	Florida	1,070,376	2,712,798
COL	Colorado	435,666	1,057,066
NY	New York City	264,346	733,846

Each node can be either an intermediate starting or an ending location in which the driver and the rider can meet each other. In our experiments, the real distances are considered as the traveling costs.

Generated instances. For each road network, we randomly generate the offers and the demands. More precisely, we build two scenarios.

In the first scenario (S_1), we generate 10 offers $o_i=(s_i,t_i)$, and for each offer o_i, we generate 10 instances. For each instance, a demand $d_j=(s'_j,t'_j)$ is generated such that its starting location s'_j (ending location t'_j) is situated within a radius $R_1 = r_1\times w(\mu^*_i)$ of the starting s_i (ending location t_i) of the offer o_i, where μ^*_i is the shortest path from s_i

[1] http://www.dis.uniroma1.it/challenge9/download.shtml

to t_i. Thus, 100 instances are generated in scenario 1.

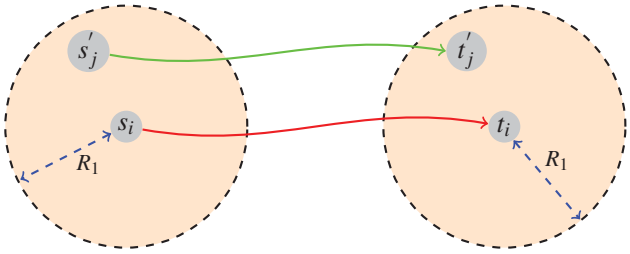


Figure 7: Generated instances according to scenario 1. The origin and destination of the riders are respectively generated around s_i and t_i. R_1 corresponds to the radius of the two circles.

In the second scenario (S_2), again 100 instances are generated such that, for each instance, both starting location s'_j and ending location t'_j of the demand d_j are generated within a radius $R_2 = r_2 \times w(\mu_i^*)$ of starting location s_i of offer o_i (the same reasoning can be applied around the ending location t_i).

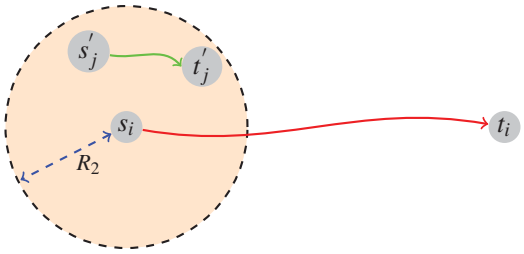


Figure 8: Generated instances according to scenario 2. The origin and destination of the riders are generated around the same location (either driver's origin or driver's destination), that is here, the origin of driver s_i. R_2 corresponds to the radius of the circle.

In all scenarios, the offers and the demands have departure time windows $\mathcal{TW}_o(s_i) =]-\infty, +\infty[$ and $\mathcal{TW}_d(s'_j) =]-\infty, +\infty[$, respectively. In other words, the driver and the rider are willing to travel at any time during the day.

For both scenarios, we derived (i) the number of driver-rider's matching detected by our heuristics and the recurring ridesharing approach compared to our enumerative approach, (ii) the gap with respect to the exact solution of the currently detected matchings and (iii) the time of calculation. We assume that the driver and the rider have the same cost, and the cost of the common path is shared equitably between them (i.e. $\varepsilon = 0.5$). After several test, we set $r_1 = 0.4$ and $r_2 = 0.3$.

Table 2 reports the results of the four tested algorithms (recurring ridesharing approach, $\mathcal{HM}_1$, $\mathcal{HM}_2$, enumerative approach). For each approach, the following results are reported:

- Gap: the deviation of algorithms with respect to the optimal solution.
- Time: the required CPU time in seconds.
- Match: the percentage of number matching found.

In columns Gap and Time, we consider the average values on the 100 generated instances for each network.

The results show the effectiveness of the proposed approaches:

- Our heuristics deliver very-high quality solutions. In scenario 1, all demands are matched with both metrics $\mathcal{M}_1$ and $\mathcal{M}_2$, whereas, at most 54% of demands are matched with recurring ridesharing approach. Moreover, heuristics $\mathcal{HM}_1$ and $\mathcal{HM}_2$ provide optimal solutions for networks E, COL and NY, and efficient solutions with a Gap $= 0,02\%$ for network FLO.
- Scenario 2 shows a sensitivity around the $\mathcal{HM}_1$ heuristic. Matchings are not well detected by $\mathcal{M}_1$ because in this metric, the partitioning of potential nodes is based only on their distance to (s,s') and (t,t'). Thus, $\mathcal{M}_1$ performance is deteriorated by detecting at most 86.95% of matchings with a Gap of 0.44%. Concerning the $\mathcal{HM}_2$ approach, the percentage of matching increases up to 95.65% with an interesting Gap which does not exceed the 0.1%, resulting in a better stability of metric $\mathcal{M}_2$.

  The recurring ridesharing approach detects at most 30% of matching with a significant Gap which reaches 17.67% (see Figure 9). The Gap of recurring ridesharing approach can be considered as the additional cost-savings generated by using approach with intermediate locations.
- Our heuristics provide high quality solutions within short CPU times. Heuristic using metric $\mathcal{M}_1$ is slightly less time consuming. This can

Table 3: Performance of algorithms

	Network	Classic ridesharing		$\mathcal{HM}_1$			$\mathcal{HM}_2$			Enumerative-M
		Match (%)	Gap (%)	Match (%)	Gap (%)	Time (s)	Match (%)	Gap (%)	Time (s)	Time (s)
S_1	E	54	8.39	100	0	7.32	100	0	7.54	6531.3
	FLO	38	11.01	100	0.02	1.77	100	0.02	1.84	1557.33
	COL	38	9.08	100	0	0.78	100	0	0.84	690
	NY	38	8.14	100	0	0.47	100	0	0.49	450
S_2	E	4.76	17.67	80.95	0.01	7.46	95.23	0.03	7.65	6676.23
	FLO	13.04	3.43	86.95	0.44	1.79	95.65	0.05	1.85	1642.5
	COL	30.55	6.18	72.22	0.34	0.8	86.11	0.10	0.86	657
	NY	9.67	2.36	74.19	0.83	0.47	93.54	0.07	0.49	421

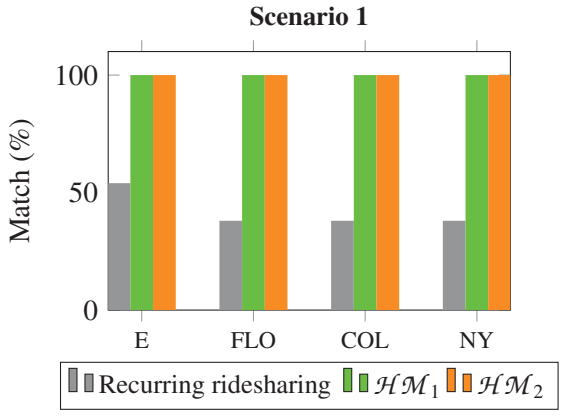
be explained by the fact that $\mathcal{HM}_2$ approach requires additional time due to pre-processing, but it remains in the order of microseconds.

Regarding results of scenario 2 on COL and E instances, $\mathcal{M}_1$ works as well as $\mathcal{M}_2$ in terms of Gap. This is due to the fact that the Gap is calculated over instances where matching is found, and $\mathcal{M}_2$ detects more matchings than $\mathcal{M}_1$. In order to have a better idea on the performance of each metric, we compare our heuristics on instances where a matching is found by both $\mathcal{M}_1$ and $\mathcal{M}_2$, hereafter denoted by I. Table 4 reports the results for the two heuristics $\mathcal{HM}_1$ and $\mathcal{HM}_2$. Column $B_{\mathcal{M}_1}$ gives the percentages in cases where a better solution is obtained only by $\mathcal{HM}_1$. Whereas column $B_{\mathcal{M}_2}$ shows the percentages in cases where a better solution is obtained only by $\mathcal{HM}_2$.

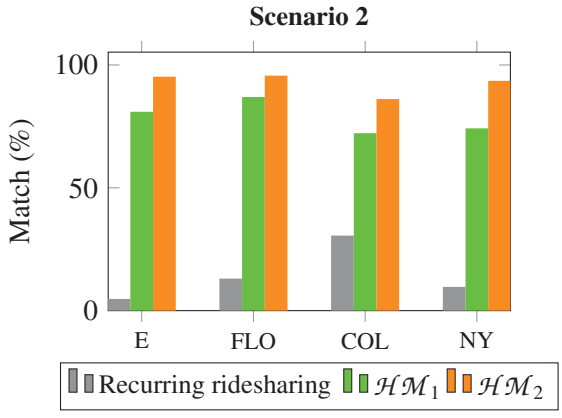
Table 4: Performance of $\mathcal{HM}_1$ and $\mathcal{HM}_2$ on instance I

	Network	$\mathcal{HM}_1$		$\mathcal{HM}_2$	
		$B_{\mathcal{M}_1}$ (%)	Gap (%)	$B_{\mathcal{M}_2}$ (%)	Gap (%)
S_1	E	0	0	0	0
	FLO	0	0,02	0	0,02
	COL	0	0	0	0
	NY	0	0	0	0
S_2	E	0	0,01	34,78	0
	FLO	0	0,44	25	0,03
	COL	0	0,02	15,38	0
	NY	0	0,84	5,55	0,02

From Table 4, we observe that $\mathcal{HM}_2$ outperforms $\mathcal{HM}_1$, in particular for the instances of scenario 2. Indeed, for FLO and NY instances, the Gap of $\mathcal{HM}_1$ is, respectively, 0.44% and 0.84%, but for $\mathcal{HM}_2$, the Gap is equal to 0.03% and 0.02%, respectively.



(a)



(b)

Figure 9: Match (%) visualization of two scenarios. Figure (a) concerns the scenario 1, while figure (b) concerns the scenario 2.

6 Conclusion

In this work, we have modeled a new case of dynamic ridesharing with intermediate locations and provided efficient approaches to solve it. The objective function is to minimize the sum of costs spent by the driver and the rider. The experimental results show the effectiveness and efficiency of our system compared to the recurring approach of ridesharing in terms of cost-savings. We used a simple objective function, which, although appropriate for certain types of users, might be redefined in several terms: the deviation from the origin-destination trip for the driver, the total travel time, etc. Our approach can integrate those features as a weighted sum in the objective function. The matching selection is assumed done based on geographical information between a driver and a rider. Nevertheless, our method can be included to improve the matching. While we only consider two agents, our approach may be extended to a few number of riders and one driver, either by using the same pick-up and drop-off points or by enumerating the set of pick-up and drop-off points. Finally, the introduction of acceleration approaches such as contraction hierarchies [19] in our algorithms may be considered.

References

[1] Agatz, N.A.H., Erera, A., Savelsbergh, M., Wang, X.: Optimization for dynamic ridesharing: A review. European Journal of Operational Research, 223, pp. 295-303 (2012)

[2] Furuhata, M., Dessouky, M., Ordonez, F., Brunet, M.E., Wang, X., Koenig, S.: Ridesharing: The state-of-the-art and future directions. Transportation Research Part B: Methodological, 57, pp. 28-46 (2013)

[3] Agatz, N.A.H., Erera, A., Savelsbergh, M.W.P., Wang, X.: Dynamic ridesharing: a simulation study in metro atlanta. Transportation Research Part B, Methodological, 45, pp. 1450-1464 (2011)

[4] Baldacci, R., Maniezzo, V., Mingozzi, A.: An exact method for the car pooling problem based on lagrangean column generation. Operations Research, 52, pp. 422-439 (2004)

[5] Herbawi, W., and Weber, M. 2012b. Modeling the multihop ridematching problem with time windows and solving it using genetic algorithms. In Proceedings of the 2011 IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI '12, IEEE Computer Society. (Athens, Greece, 2012)

[6] Ghoseiri, K., Haghani, A., Hamedi, M.: Real-time rideshare matching problem. University of Maryland, Department of Civil and Environmental Engineering, UMD-2009-05 (2011)

[7] Amey, A.: Proposed Methodology for Estimating Rideshare Viability Within an Organization: Application to the MIT Community, Transportation Research Board Annual Meeting 2011, pp. 11-2585 (2011)

[8] Winter, S., Nittel, S: Ad-hoc shared-ride trip planning by mobile geosensor networks. International Journal of Geographic Information Science, 00, pp. 1-21 (2006)

[9] Xing, X., Warden, T., Nicolai, T., Herzog, O.: Smize: a spontaneous ridesharing system for individual urban transit. In: Proceedings of the 7th German Conference on Multiagent System Technologies, MATES'09. Springer-Verlag, Berlin Heidelberg, pp. 165-176 (2009)

[10] Geisberger, R., Luxen, D., Neubauer, S., Sanders, P., Volker, L.: Fast Detour Computation for Ride Sharing. In: 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems - ATMOS'10. Editors: Th. Erlebach, M. Lubbecke, pp. 88-99 (2010)

[11] Calvo, R.W., de Luigi, F., Haastrup, P., Maniezzo, V.: A distributed geographic information system for the daily car pooling problem. Computers & Operations Research, 31, 2263-2278 (2004)

[12] Herbawi, W., Weber, M.: The ridematching problem with time windows in dynamic ridesharing: a model and a genetic algorithm In: Proceedings ACM Genetic and Evolutionary Computation Conference (GECCO), pp. 1-8 (2012)

[13] Herbawi, W., Weber, M.: Evolutionary multiobjective route planning in dynamic multi-hop ridesharing. In: EvoCOP'11, pp. 84-95 (2011)

[14] Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Efficient Models for Timetable Information in Public Transportation Systems. ACM Journal of Experimental Algorithmics, 12 (2.4) (2007)

[15] Drews, F., Luxen, D.: Multi-hop ride sharing. In: Proceedings of the Sixth Annual Symposium on Combinatorial Search, pp. 71-79 (2013)

[16] Bit-Monnot, A., Artigues, C., Huguet, M.-J., Killijian, M.-O.: Carpooling: the 2 Synchronization Points Shortest Paths Problem. In: 13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Vol. 33, pp. 150-163 (2013)

[17] Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1, pp. 269-271 (1959)

[18] Hart. P.E., Nilsson, N., Raphael, B.:A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Trans. on Sys. Sci. and Cyb. 4, pp. 100-107 (1968)

[19] Geisberger, R., Sanders, P., Schultes, D., Vetter, C.: Exact Routing in Large Road Networks Using Contraction Hierarchies. Transp. Sci. 46, pp. 388-404 (2012)

[20] Arz, J., Luxen, D., Sanders, P.: Transit Node Routing Reconsidered. In: International Symposium on Experimental Algorithms (SEA13). LNCS, volume 7933, pp. 55-66. Springer, Rome (2013)