A Posteriori Approach of Real-time Ridesharing Problem with Intermediate Locations

Kamel Aissat[1] and Ammar Oulmara[2]
[1]*University of Lorraine - LORIA, Nancy, France*
[2]*University of Lorraine, Ile de Saulcy, Metz, France*

Keywords: Operations Research, Transport, Ridesharing, Shortest Path Problem, Geographical Maps.

Abstract: Ridesharing is a travel mode that provides several benefits and solutions, such as the reduction of travel cost, the reduction of the traffic congestion and the provision of travel options. In the classical ridesharing approach, the driver makes a detour to the rider's origin in order to pick-up the rider, then drives him to his destination and finally the driver goes to his own destination. This implies that the driver endures the whole detour and may not accept such matching if the detour is too long. However, the matching could be accepted if the rider meets the driver at an intermediate location. In this paper, we present a general ridesharing approach in which a driver and a rider accept to meet each other at an intermediate pick-up location and to separate at an intermediate drop-off location not necessarily their origins and destinations locations, respectively.
Thus, for a given rider, we propose an exact and heuristic methods to determine the best driver and the best meeting locations that minimize a total travel cost. Finally, we perform a numerical study using a real road network and a real dataset. Our experimental analysis shows that our heuristics provide efficient performance within short CPU times and improve participants cost-savings and matching rate compared to the classical ridesharing.

1 INTRODUCTION

Several worldwide urban cities face mobility problems among which the insufficiency and the saturation of the current offer of public transport, traffic congestion and carbon emissions. Those problems impact the environment and the quality of life of citizens. The transportation activity is responsible for a great part of carbon emissions. At least half of total carbon emissions is due to private vehicles (Metz et al., 2005). Nowadays, governments begins to be conscious of the urgency to conserve the environment by investing in more environmentally friendly and safe modes of transportation like Electric vehicles (Sassi and Oulamara, 2014) and ridesharing systems. Ridesharing helps reducing the number of vehicles since most private vehicle trips are made with only one passenger occupancy. The total cost of single passenger trips is quantified in (Hartwig and Buchmann, 2006). This study estimates that the cost of "empty seats" in the world would be about 500 billion dollars per year. Thus, a significant scope for reduction of cost and the number of cars on the road requires intelligent usage of private vehicles. In that case, a vehicle is considered as a means of mobility rather than an ownership. In order to avoid empty seats during trips, recent initiatives such as ridesharing services, have been deployed with success.

A ridesharing service brings together users with similar itineraries and time schedules. More precisely, a driver with his vehicle and a rider, traveling from their individual starting locations to their destinations, share a common part of their itinerary using the driver's vehicle as well as a part of vehicle-related expenses.

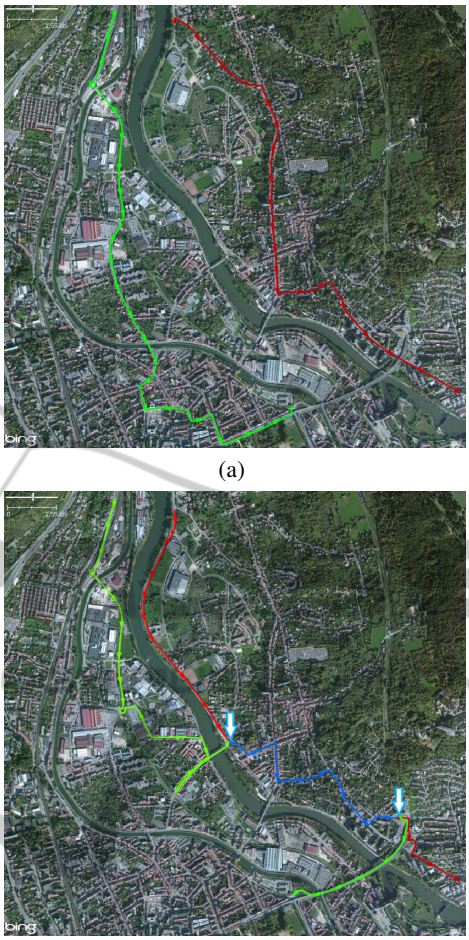
The considerable progress in embedded and mobile computing technologies such as smartphones, along with the pervasive diffusion of online social networking tools and geolocation devices (Global Positioning System - GPS), facilitate the use of ridesharing services. Thus, a new and innovative solution of ridesharing service has emerged. It consists in an automatic and instant matching of riders through a network service by using smartphones as both geolocation and communication devices. This service is called real-time ridesharing.

Several methods have been proposed in the literature to solve the simplest form of ridesharing, also

Aissat K. and Oulmara A..
A Posteriori Approach of Real-time Ridesharing Problem with Intermediate Locations.
DOI: 10.5220/0005256100630074
In *Proceedings of the International Conference on Operations Research and Enterprise Systems* (ICORES-2015), pages 63-74
ISBN: 978-989-758-075-8

called classical ridesharing, see (Agatz et al., 2012). In this simplest form, the driver picks up the rider at his starting location, drops him off at his ending location and then continues to his target location. This method is the most implemented in ridesharing services. Thus, users (drivers and riders) send their requests (demands and offers of ridesharing) and the system proposes the best matchings of offers and demands regarding the itinerary of each user. However, the matching opportunities of drivers and riders in the classical ridesharing decrease when itineraries of riders and drivers are slightly different. In order to overcome this lack of flexibilty, instead of allowing the driver to go to a rider's origin to pick him up and drop him off at his destination, in this paper, we allow the driver and the rider to meet and to separate at intermediate locations, not necessarily at the rider's origin and rider's destination (see Figure 1).

Specifically, the rider travels on his own to the first intermediate location, where he is picked up by the driver. He is then dropped off at the second intermediate location, where he will continue to his ending location. Solving the problem of ridesharing with intermediate locations requires the development of efficient algorithms that determine the optimal intermediate locations regarding the itineraries of riders and drivers.



(a)

(b)

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

2 LITERATURE REVIEW

Depending on the form of the ridesharing in practice, we distinguish the following optimization problems: *Slugging* problem, *Taxi-ridesharing* problem and *Ridesharing* problem.

Slugging Problem. High Occupancy Vehicle (HOV) lanes are traffic lanes set aside for vehicles with a minimum number of occupants. In order to meet the occupancy requirements of HOV lanes, users have developed a new way of travel, known as *slugging*, that offers the benefits of traveling on an HOV lane without forming classical ridesharing. In *slugging*, the driver maintains his original route and does not make any detour to pick-up or drop-off the rider. Thus, the rider walks to the driver's origin, boards at the driver's departure time, alights at the driver's destination, then walks from there to his own destination. Authors in (Ma and Wolfson, 2013) formalize and study this *slugging* problem. Variants of the *slugging* problem that also consider the vehicle capacity constraint and travel time delay, has been proven to be the NP-complete and they propose a quadratic algorithm to solve it optimally. They also develop heuristics that have been evaluated on real data. Another work on slugging problem is considered in (Mote and Whitestone, 2011).

Taxi-ridesharing Problem. The problem of *Taxi-ridesharing* consists in assigning taxi vehicles to riders requests which are spread over different locations in a given zone. Several constraints can be considered such as the vehicle capacity and time windows of riders. Thus, this problem concerns the assignment of a time to each pickup and delivery event, within these time windows.

In contrast to *Slugging* problem, the route of driver trip (i.e. taxi driver) change to accommodate riders, and the pick-up and drop-off locations for the riders correspond to their origins and destinations, respec-

tively. Authors in (Ma et al., 2013) propose a practical taxi ridesharing service in which an organization operates a dynamic taxi ridesharing service. In their system, the taxi drivers can independently determine when they join and leave the service. Thus, riders submit ride queries in real time via a mobile device, e.g., a smartphone. Each query indicates the origin and destination locations of the trip, as well as the time windows during which the rider wants to be picked up and dropped off. Once a new query is received, the system determines an appropriate taxi which is able to satisfy both the new query and the trips of existing riders who are already assigned to that taxi. The updated schedules and routes will then be given to the corresponding taxi driver and riders. Another work on taxi-ridesharing problem is considered in (Varone and Janilionis, 2014).

Classical Ridesharing Problem. The principle is rather similar to *Taxi-ridesharing*. The main differences are, (i) the classical ridesharing is based on private cars in which the rider shares his trip with a simple driver, while in taxi ridesharing the presence of the taxi driver is obligatory, (ii) the taxi ridesharing usually needs appropriate pricing mechanisms, generally more expensive than a classical ridesharing, to incite taxi drivers. Several research has been reported recently in the fields of ridesharing, see (Agatz et al., 2012) and (Furuhata et al., 2013). In ridesharing system, when a driver's offer or rider's request arrives in the system, some options should be specified. For instance, when the driver offers a ride, he may specify if he is willing to take a single rider or multiple riders. Similarly, when the rider looks for a ride, he specifies if he wants to ride with a single driver or may accept to ride with multiple drivers and will be transferred from one to another to reach his final destination. Thus, we distinguish four variants, namely, single-driver and single-rider (Geisberger et al., 2010), (Amey, 2011), single-driver and multiple-riders (Baldacci et al., 2004), (Herbawi and Weber, 2012), multiple-drivers and single-rider (Drews and Luxen, 2013), and finally multiple-drivers and multiple-riders (Herbawi and Weber, 2012). In each variant, the matching between riders and drivers depends on one or more objective functions, such as the maximization of the number of matchings, the maximization of the cost saving, the maximization of distance saving, etc.

The *ridesharing problem with intermediate locations* can be seen as an extension of the *slugging problem* and the classical *ridesharing problem*. Firstly, the driver's route can change to accommodate riders compared to *slugging problem*. Secondly, in contrast to the *classical ridesharing problem*, the pick-up and drop-off locations are not necessarily the origins and the destinations of rider, respectively.

The *problem of ridesharing with intermediate locations* is addressed in (Aissat and Oulamara, 2014b), (Bit-Monnot et al., 2013) and (Aissat and Oulamara, 2014a). In (Aissat and Oulamara, 2014b), the authors consider the round trip ridesharing problem with an intermediate meeting location. The rider drives to the intermediate meeting location using his private car and parks it there, so in the return trip, he must be dropped off at that location to get his car back. Thus, for a given demand, the optimization system determines the best meeting location, the best driver in outgoing trip and the best driver in return trip passing via the intermediate meeting location where the rider's car was left. Authors develop an efficient approach where the objective is to minimize the total cost in round trip. Their approach was validated by experiments based on real data of ridesharing. The main advantage of their approach is increasing the opportunity of matching between riders-drivers and then a significant reduction of the total travel cost compared to the classical approach of round trip ridesharing.

In (Bit-Monnot et al., 2013), the authors consider the problem of ridesharing with intermediate locations, where the rider can use the public transportation either in order to reach the meeting location from his starting location, or to reach his ending location coming from the separate location. The authors propose an optimal method to find the pick-up and drop-off locations in $O(m \cdot n^2)$, where n is the number of nodes and m is the number of edges in the graph, in which the objective is to minimize the cumulated travel time for both the driver and the rider. However, the time complexity of this method prevents its use in real-time ridesharing, and their model does not take into account the detour time constraint (i.e., the total time of the detour should be less than a given value fixed by the driver (rider)) and detour cost constraint (i.e., the incurred cost of the driver (rider) using ridesharing mode is more attractive than the incurred cost when they travel alone).

In this paper, we consider the problem of ridesharing with intermediate locations. The objective function is to minimize the total travel cost in scenario involving transportation modes with time-independent arc costs, while ensuring that their detour costs and times remain reasonable. Thus, for a given rider, we determine the best driver, the best meeting and separate locations that minimize the total travel cost under constraints of time and cost detour of rider and driver. We suggest some heuristic methods that reduce the number of shortest paths computations, based on the

exact pruning algorithm.

The remainder of the paper is structured as follows. Section 3 details the model of ridesharing with two intermediate locations. Section 4 proposes two solving approaches. Section 5 introduces an exact method and an offer selection heuristic. Section 6 generalizes our approach by introducing some parameters. Section 7 analyses the performance of our algorithms. Finally, concluding remarks and future research are included in Section 8.

3 PROBLEM DESCRIPTION

A road network is represented by a weighted graph $G = (V,E)$, where V is the set of nodes, E the set of edges. Nodes model intersections and edges depict street segments. With each edge $(i,j) \in E$ two weights are associated. $c_k(i,j)$ depicts the traveling cost and $\tau_k(i,j)$ depicts 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 cost $c(\mu)$ of path μ is given by the sum of costs of all edges in μ. A shortest-path between a source node u and a target node v is a path with minimal cost among all paths from u to v. In the following, a shortest-path between node u and node v will be represented by $u \to v$.

We consider an offer and a demand of ridesharing represented by $o = (s,t,[t_o^{\min},t_o^{\max}],\Delta_o)$ and $d = (s',t',[t_d^{\min},t_d^{\max}],\Delta_d)$, respectively, where s and s' (t and t') are the starting (ending) locations of the driver and the rider, respectively, $[t_o^{\min},t_o^{\max}]$ ($[t_d^{\min},t_d^{\max}]$) is the departure time window of the driver (rider), Δ_o (Δ_d) is the maximal detour time that can be accepted by the driver (rider), i.e. the extra-time that the driver (rider) can accept additionally to the travel time corresponding to his shortest path.

An edge (i,j) of G has a nonnegative traveling cost $c_k(i,j)$ depending on the fact that the edge is used by a driver ($k = o$) or by a rider ($k = d$).

A basic way to implement the ridesharing (also called classical ridesharing) is the case where the driver picks up the rider at his starting location, drops him off at his ending location and the driver continues to his target location. Thus, 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 $(c_o(\gamma_{od}) - c_o(s \to t))$, remains reasonable, i.e.,

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

where ε in (1) is a *detour factor* (Geisberger et al., 2010). The term $\varepsilon \cdot c_o(s' \to t')$ is the reward that rider provides to driver. Specifically, it represents the maximal detour cost that still guarantee an incentive to the driver to pick-up the rider at his starting location s' and to drop him off at his ending location t'. Clearly, if $c_o(\gamma_{od}) - c_o(s \to t) > \varepsilon \cdot c_o(s' \to t')$, the driver does not accept the demand d. Authors in (Geisberger et al., 2010) show that the reasonable choice of ε is at most 0.5.

However, in order to get more opportunities of ridesharing, a rider can accept two different intermediate locations, one for pick-up and another for the drop-off. Especially, the rider travels by his own to the first intermediate location r_1 with a cost $c_d(s' \to r_1)$, where he will be picked up by the driver and dropped off at the second intermediate location r_2, then continue his journey from r_2 to t' on his own. Thus, the driver and the rider will share the common travel cost $c_o(r_1 \to r_2)$. If the rider rewards the driver with the amount $\varepsilon \cdot c_o(r_1 \to r_2)$, then the cost of rider is $c_d(s' \to r_1) + \varepsilon \cdot c_o(r_1 \to r_2) + c_d(r_2 \to t')$.

Depending on how the value of ε is set, we distinguish two cases, (i) ***A priori approach*** in which the user (driver and/or rider) fixe the *detour factor* ε, for instance, the rider wants to share equitably the common path cost with the driver, i.e. $\varepsilon = 0.5$, (ii) ***A posteriori approach*** in which the user (driver/rider) does not set the *detour factor* ε in advance, but the system calculates the most attractive value of ε that guarantees the best matching between the rider and the driver.

The ***a priori approach*** is adressed in (Aissat and Oulamara, 2014a). Given a fixed value of ε, the authors developed efficient heuristics that determine the best matching between the driver and the rider, while ensuring that the detour cost of the rider and the driver remains reasonable. Their objective function is to minimize the total traveling cost.

In this paper, we address the ***a posteriori approach*** of a ridesharing with intermediate locations, while considering additional constraints, namely, the time windows, detour time and the desired minimum cost-saving of users. More formally, 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 pick-up location r_1 and drops him off at the intermediate drop-off location r_2. The rider travels on his own from s' to r_1 and from r_2 to t' (see Figure 2).

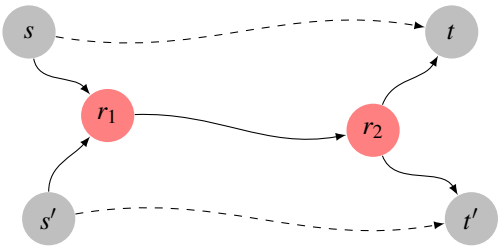


Figure 2: The solid lines symbolize the path of the driver and the rider in order to form a joint trip, whereas, the dashed ones stand for the shortest paths of the driver and the rider.

The objective is to minimize the total travel cost, while ensuring the cost and time detour constraint for the driver and the rider.

3.1 Matching Constraints

In our problem, a matching between a driver and a rider can be established only under timing constraint of the ride (Definition 1), and the travel cost constraint (Definition 2).

Definition. 1. (Time Synchronization) *We say that an offer o and a demand d form a time synchronization at location v if and only if $\beta \geq 0$, where*

$$\beta = \min \begin{cases} t_o^{\max} + \tau_o(s \to v) - (t_d^{\min} + \tau_d(s' \to v)) & \text{(2a)} \\ t_d^{\max} + \tau_d(s' \to v) - (t_o^{\min} + \tau_o(s \to v)) & \text{(2b)} \end{cases}$$

Equation (2a) guarantees that if the rider leaves his origin location s at $t_d^{\min}$ to reach the intermediate pick-up location v, he must arrive no later than the latest arrival time of the driver at v. The same argument is applied to the driver in equation (2b). Thus, if $\beta \geq 0$, the propagation of the departure time windows of the driver and the rider at the intermediate pick-up location v will coincide. So, they can meet each other at the earliest at this location at time $\max\{t_d^{\min} + \tau_d(s' \to v), t_o^{\min} + \tau_o(s \to v)\}$.

Definition. 2. (Reasonable Fit) *We say that an offer $o = (s, t, [t_o^{\min}, t_o^{\max}], \Delta_o)$ and a demand $d = (s', t', [t_d^{\min}, t_d^{\max}], \Delta_d)$ 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, o and d form a time synchronization at location r_1 and*

$$c_o(s \to t) + c_d(s' \to t') - (c_o(s \to r_1) + c_d(s' \to r_1) + c_o(r_1 \to r_2) + c_o(r_2 \to t) + c_d(r_2 \to t')) \geq 0 \quad (3)$$

$$\tau_o(s \to r_1) + \tau_o(r_1 \to r_2) + \tau_o(r_2 \to t) \leq \tau_o(s \to t) + \Delta_o \quad (4)$$

$$\tau_d(s' \to r_1) + \tau_o(r_1 \to r_2) + \tau_d(r_2 \to t') \leq \tau_d(s' \to t') + \Delta_d \quad (5)$$

The inequality (3) represents the cost-saving of matching the offer o with the demand d. The cost-saving is defined as the difference between the travel cost of the driver and the rider when each of them travels alone without ridesharing and the total travel cost of the driver and the rider with a shared path between r_1 and r_2.
The term $(\tau_o(s \to t) + \Delta_o)$ in (4) (resp. $\tau_d(s' \to t') + \Delta_d$ in (5)) allows to limit the amount of time that the driver (resp. rider) passes in traveling. We ignore any extra-time during pick-up or drop-off of riders.

In the following lemma, we show that if the generated cost-saving is positive, then it is always possible to allocate the cost-saving among the driver and the rider so that each of them has an individual benefit.

Lemma. 1. *If an offer $o = (s, t, [t_o^{\min}, t_o^{\max}], \Delta_o)$ and a demand $d = (s', t', [t_d^{\min}, t_d^{\max}], \Delta_d)$ form a reasonable fit, then there exists a detour factor $\varepsilon \in [0, 1]$ such that the gains of the rider and the driver are positive.*

Proof. Assume that an offer o and a demand d form a reasonable fit. From definition (2), there exist two intermediate locations r_1 and r_2, such that o and d form a time synchronization at location r_1 and

$$c_o(s \to t) + c_d(s' \to t') - (c_o(s \to r_1) + c_d(s' \to r_1) + c_o(r_1 \to r_2) + c_o(r_2 \to t) + c_d(r_2 \to t')) \geq 0$$

Assume that there exists a detour factor $\varepsilon \in [0, 1]$ in which the common trip cost of the driver and the rider is shared according to ε, i.e., the rider rewards the driver with amount $\varepsilon \cdot c_o(r_1 \to r_2)$.
On the one hand, the driver can accept to share a ride with the rider, only if his total cost using ridesharing modality is less than his cost if he travels alone, i.e.,

$$c_o(s \to r_1) + (1 - \varepsilon) \cdot c_o(r_1 \to r_2) + c_o(r_2 \to t) \leq c_o(s \to t) \quad (6)$$

From (6) we obtain, $\varepsilon \geq \varepsilon_1$, where

$$\varepsilon_1 = \frac{c_o(s \to r_1) + c_o(r_1 \to r_2) + c_o(r_2 \to t) - c_o(s \to t)}{c_o(r_1 \to r_2)}$$

On the other hand, the rider accepts to be picked up and dropped off by the driver in intermediate locations, only if his total cost using ridesharing modality is less than his cost if he travels alone, i.e.,

$$c_d(s' \to r_1) + \varepsilon \cdot c_o(r_1 \to r_2) + c_d(r_2 \to t') \leq c_d(s' \to t') \quad (7)$$

From (7) we obtain, $\varepsilon \leq \varepsilon_2$, where

$$\varepsilon_2 = \frac{c_d(s' \to t') - c_d(s' \to r_1) - c_d(r_2 \to t')}{c_o(r_1 \to r_2)}$$

Furthermore, it is easy to see that the existence of the solution is constrained by $\varepsilon_1 \leq \varepsilon \leq \varepsilon_2$.
In order to satisfy the constraint (3), the ε must be in the interval $[\varepsilon_1, \varepsilon_2]$. Thus, any value of ε in the interval $[\varepsilon_1, \varepsilon_2]$ satisfies the constraints of matching between the rider and the driver. A reasonable value of ε might be the average value of the interval $[\varepsilon_1, \varepsilon_2]$, i.e. $\varepsilon = \frac{\varepsilon_1+\varepsilon_2}{2}$. □

3.2 Objective of Ridesharing System

As in (Aissat and Oulamara, 2014a), we use the term *global-path* (s, s', r_1, r_2, t, t') 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. $(s, s', r_1, r_2, t, t') = (s \to r_1 \oplus s' \to r_1 \oplus r_1 \to r_2 \oplus r_2 \to t \oplus r_2 \to t')$. A shortest *global-path* between source nodes s, s' and target nodes t, t' is a *global-path* with minimal cost $c(s, s', r_1, r_2, t, t')$ among any *global-path* from s, s' to t, t', where

$$\begin{aligned} c(s,s',r_1,r_2,t,t') &= c_o(s \to r_1) + c_d(s' \to r_1) \\ &\quad + c_o(r_1 \to r_2) + c_o(r_2 \to t) \\ &\quad + c_d(r_2 \to t') \end{aligned} \tag{8}$$

Thus, the objective of the ridesharing system is to determine the best intermediate locations r_1 and r_2 that minimize the shortest *global-path* such that the offer o and a demand d form a reasonable fit.

4 PROPOSED APPROACH

The proposed methods are based on computing several shortest paths. Shortest path algorithms have been extensively studied, since they are used in almost all real cases of transportation problems. Efficient algorithms are especially needed when geographical information systems (GIS) are involved. The well-known algorithm to find the shortest path is the Dijkstra algorithm (Dijkstra, 1959). In (Sanders and Schultes, 2007), the authors note that the Dijkstra algorithm can in theory be used to find the shortest path on a road network, but for large networks it would be far too slow. Different speed-up techniques, such as bidirectional search, goal direction, etc., can be used to improve performance. An overview of such techniques is available in (Sanders and Schultes, 2007).

4.1 Search Space of Potential Intermediate Locations

The matching constraints defined in section 3.1 allows us to limit the search space of intermediate locations. In the following, we provide characteristics of potential intermediate locations.

Definition. 3. (Potential Intermediate Nodes). *Let $N^{\uparrow}(s)$, $N^{\uparrow}(s')$, $N^{\downarrow}(t)$ and $N^{\downarrow}(t')$ be defined as*

$$\begin{aligned} N^{\uparrow}(s) = \{v \in V \mid\ & c_o(s \to v) \leq c_o(s \to t) \\ & \wedge\ \tau_o(s \to v) + \tau_o(v \to t) \leq \tau_o(s \to t) + \Delta_o\} \end{aligned}$$

$$\begin{aligned} N^{\downarrow}(t) = \{v \in V \mid\ & c_o(v \to t) \leq c_o(s \to t) \\ & \wedge\ \tau_o(s \to v) + \tau_o(v \to t) \leq \tau_o(s \to t) + \Delta_o\} \end{aligned}$$

$$\begin{aligned} N^{\uparrow}(s') = \{v \in V \mid\ & c_d(s' \to v) \leq c_d(s' \to t') \\ & \wedge\ \tau_d(s' \to v) + \hat{\tau}(v \to t') \leq \tau_d(s' \to t') + \Delta_d\} \end{aligned}$$

$$\begin{aligned} N^{\downarrow}(t') = \{v \in V \mid\ & c_d(v \to t') \leq c_d(s' \to t') \\ & \wedge\ \hat{\tau}(s' \to v) + \tau_d(v \to t') \leq \tau_d(s' \to t') + \Delta_d\} \end{aligned}$$

$N^{\uparrow}(s)$ represents the set of potential pick-up locations of the driver. Indeed, if v is the pick-up location, the best situation for the driver is to share the cost $c_o(v \to t)$ with the rider as long as v satisfies the traveling time constraint $\tau_o(s \to v) + \tau_o(v \to t) \leq \tau_o(s \to t) + \Delta_o$. Thus, if a node v does not respect the cost and the traveling time constraints, then v cannot be an pick-up location. Set $N^{\downarrow}(t)$ represents the potential drop-off locations of the driver. The same reasoning as for the set $N^{\uparrow}(s)$ is applied to $N^{\downarrow}(t)$.

On the other hand, the sets $N^{\uparrow}(s')$ and $N^{\downarrow}(t')$ represent the potential pick-up and drop-off locations of the rider, respectively.

Remark that in the traveling time constraint of set $N^{\uparrow}(s')$ (resp. $N^{\downarrow}(t')$), we use $\hat{\tau}(v \to t')$ (resp. $\hat{\tau}(s' \to v)$), i.e., the traveling time corresponding to the estimated distance between two locations v and t' (resp. s' and v) using Haversine formula, instead of the shortest traveling time $\tau_d(v \to t')$ (resp. $\tau_d(s' \to v)$).
In fact, for a given node v, if $\tau_d(s' \to v) + \tau_d(v \to t') > \tau_d(s' \to t') + \Delta_d$, we can't deduce that v cannot be a pick-up location. This is due to the fact that there may be another node v', where v' is a potential drop-off location and $t_o(v \to v') + t_d(v' \to t') < \tau_d(v \to t')$.

The Haversine formula estimates the distance between two locations based on their latitude/longitude. This formula uses a spherical model to estimate the distance between two points on the earth surface. Specifically, given two locations $x = (\theta_1, \lambda_1)$ and $y = (\theta_2, \lambda_2)$ where θ_i, $i = 1,2$, is the latitude and λ_i, $i = 1,2$, is the longitude. The distance between x and y is given by $\hat{d}(x,y) = 2 \times R \times arcsin(\sqrt{a})$ where $R \approx 6371(km)$ is the earth's radius in kilometers, and $a = sin^2(\frac{\theta_2-\theta_1}{2}) + cos(\theta_1) \times cos(\theta_2) \times sin^2(\frac{\lambda_2-\lambda_1}{2})$. Thus, the estimated smallest duration from x to y is noted by $\hat{\tau}(x \to y) = \frac{\hat{d}(x,y)}{v_{max}}$, such that v_{max} is the maximal speed among all used modality of the driver and the rider.

The following lemma simultaneously characterizes the set of potential intermediate locations for both driver and rider.

Lemma. 2. *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 $\mathcal{C}_2 = N^{\downarrow}(t) \cap N^{\downarrow}(t')$.*

4.2 Solving Methods

In this section, we propose two solving methods for our model. The first method is based on bidirectional search, and the second method is based on one-to-all shortest path. Both methods are based on the modification of the graph G.

Once the potential intermediate locations class $\mathcal{C}$ has been determined, we add to the graph G two dummy nodes $S^\star$ and $T^\star$. The node $S^\star$ is connected to each node v, $v \in \mathcal{C}_1$ with an arc $(S^\star, v)$ and the cost $c(S^\star, v) = c_o(s \to v) + c_d(s' \to v)$. The node $T^\star$ is connected to each node v, $v \in \mathcal{C}_2$ with an arc $(v, T^\star)$ and the cost $c(v, T^\star) = c_o(v \to t) + c_d(v \to t')$. The arc $(S^\star, v)$ represents the driver and the rider moves out from their starting locations to the pick-up node v. The arc $(v, T^\star)$ represents the driver and the rider moves out from the drop-off node v to their ending locations. In the following, we describe our methods.

Bidirectional Search Algorithm - BSA. The BSA Algorithm works with two simultaneous searches of the shortest path, one from the source node $S^\star$ and the second from the target node $T^\star$ at the same time, until "the two search frontiers meet". More precisely, BSA Algorithm maintains two priority queues, one for the search from the source $S^\star$ denoted by Q_1, as in simple Dijkstra Algorithm, and one for the backward search from the target $T^\star$ denoted by Q_2, which is a forward search in the reversed graph G^{-1}, in which each arc (u, v) of G is replaced by (v, u) in G^{-1}.
In each iteration, we settle the node with the smallest overall tentative cost, either from the source $S^\star$ or to the target $T^\star$. In order to do this, a simple comparison of the minima of the two priority queues Q_1 and Q_2 suffices. Once we settle a node v in one queue that is already settled in the other queue, we get the first tentative cost of a shortest path from $S^\star$ to $T^\star$. Its cost is the cost of the path found by the forward search from $S^\star$ to v, plus the cost of the path found by the backward search from v to $T^\star$.
Even when the two parts of the tentative solution of the forward and the backward meet each other, the concatenated solution path is not necessarily optimal. To guarantee optimality, we must continue until the tentative cost of the current minima of the queues is above the current tentative shortest path cost (which then corresponds to the cost of the shortest path). Thus, at each iteration when the tentative cost of a shortest path from $S^\star$ to $T^\star$ is updated, we check the feasibility of constraints (3), (4) and (5), and finally we keep the admissible path that minimizes the *global-path*.
In the BSA Algorithm, when a node v is settled, then for each outgoing arc (v, u) from the node v, we check whether via this arc the node u can be reached with a cost less than or equal to the current cost of u. If yes, then the cost of u is updated to the new lower cost. This procedure is called *relaxing an arc*. The difference with the traditional *relaxing an arc* procedure of the literature is simply on updating the cost of the node even if the new cost is equal to the current cost of the node. Indeed, this modification allows us to ensure that if the shortest path from $S^\star$ to $T^\star$ contains only one intermediate node i.e., $S^\star \overset{arc}{\rightsquigarrow} r_1 \overset{arc}{\rightsquigarrow} T^\star$, then the driver and rider will never be able to form a reasonable fit. Specifically, the constraint (3) will never be satisfied.
Finally, in the found shortest path $S^\star \to T^\star$ (i.e. $S^\star \overset{arc}{\rightsquigarrow} r_1^\star \to r_2^\star \overset{arc}{\rightsquigarrow} T^\star$), we recover the best pick-up location $r_1^\star$ and the best drop-off location $r_2^\star$ that correspond to the successor of $S^\star$ and the predecessor of $T^\star$, respectively.

Shortest Path One-to-All - SPOA. In this method, instead of computing one shortest path from $S^\star$ to $T^\star$, we compute for each node v in $\mathcal{C}_2$ the shortest path $S^\star \to v$. This approach allows us to find all paths that satisfy constraint (3). Once shortest paths satisfying the constraint of *cost-saving* have been enumerated, we select the *global-path* with minimal cost that satisfies constraints (4) and (5).

The main advantage of the first method (**BSA**) lies in the fact that its running-time is less important than the second method (**SPOA**), which requires to compute all shortest paths to each node in class $\mathcal{C}_2$. On the other hand, the second method has a greater control over the constraints of *time detour* for the driver and the rider. We note that the *relaxing arc* procedure applied in this method is the same as that described above.

5 THE BEST MATCHING SELECTION

The approach developed above and in (Aissat and Oulamara, 2014a) correspond to the case where an offer and a demand are already fixed. However, in ridesharing system, several demands and offers arrive in the system. Thus, in this section, we consider the

problem of the best driver selection. Especially, given a set of offers already in the system, when a demand request d arrives in the system, the objective is to select the offer candidate which is able to fulfill the ride request.

Given a demand request d, we start by determining the two sets $N^{\uparrow}(s')$ and $N^{\downarrow}(t')$ of potential intermediate pick-up and drop-off locations, respectively. (see Definition 3).

The selection rule of the best offer requires the storage of some informations for each offer: when an offer o_i is added to the system, we store all possible intermediate locations for this offer. Thus, we determine the costs $c_o(s_i \to v)$ and $c_o(v \to t_i)$. The procedure of storage is detailed below.

Informations Storage: We denote by $M^{\uparrow}(s_i)$ and $M^{\downarrow}(t_i)$ the *forward search space* from a source s_i and the *backward search space* from target t_i, respectively. A forward search space $M^{\uparrow}(s_i)$ is a set of triplets $(v, c^{\uparrow}_{s_i}, \tau^{\uparrow}_{s_i})$, where $(v, c^{\uparrow}_{s_i}, \tau^{\uparrow}_{s_i})$ means that the shortest path from s_i to v has a cost $c^{\uparrow}_{s_i}$ and a travel time $\tau^{\uparrow}_{s_i}$. A backward search space $M^{\downarrow}(t_i)$ is a set of triplets $(v, c^{\downarrow}_{t_i}, \tau^{\downarrow}_{t_i})$ where $(v, c^{\downarrow}_{t_i}, \tau^{\downarrow}_{t_i})$ means that the shortest path from v to t_i has a cost $c^{\downarrow}_{t_i}$ and a travel time $\tau^{\downarrow}_{t_i}$.

Using *detour time constraint* (4) of the driver, we limit the search space. Then the triplets *(node, cost, time)* are stored in $M^{\uparrow}(s_i)$ and $M^{\downarrow}(t_i)$ without considering any demand.

In order to determine the sets $M^{\downarrow}(t_i)$ and $M^{\uparrow}(s_i)$, we use a modified $A^{\star}$ algorithm (Hart et al., 1968). Firstly, we determine the set $M^{\downarrow}(t_i)$ using reverse $A^{\star}$ algorithm from the destination t_i toward the origin location s_i. Then, we determine the set $M^{\uparrow}(s_i)$ using $A^{\star}$ algorithm from the origin s_i toward the destination t_i. Recall that in $A^{\star}$ algorithm, the nodes are retrieved from the priority queue by the sum of their tentative cost from the origin to the nodes and the value of the heuristic function from the nodes to the destination. Thus, for a given node v, the cost function used in the first $A^{\star}$ Algorithm to determine $M^{\downarrow}(t_i)$ is $\hat{\tau}(s_i \to v) + \tau_o(v \to t_i)$, whereas in the second $A^{\star}$ Algorithm that determines $M^{\uparrow}(s_i)$, we use the cost function $\tau_o(s_i \to v) + \tau_o(v \to t_i)$.

Remark that in the second $A^{\star}$ algorithm, we use $\tau_o(v \to t_i)$ instead of the estimation duration $\hat{\tau}(v \to t_i)$, since it is already calculated in the first $A^{\star}$ algorithm. We note that the traditional $A^{\star}$ algorithm stops after the destination has been reached. However, our $A^{\star}$ algorithm used to determine $M^{\downarrow}(t_i)$ (resp. $M^{\uparrow}(s_i)$) continues even when the destination s_i (resp. t_i) is reached. The algorithm stops once all nodes with cost function less than $\tau_o(s_i \to t_i) + \Delta_i$ are labeled.

Furthermore, we define the bucket $B_o(v)$ as the set which stores all drivers' trips which can pass via this location without violating the constraint of *time detour*. Then, in the second algorithm $A^{\star}$, when a node v is labeled, we add the entries in

$$B_o(v) := B_o(v) \cup \{(i, c^{\uparrow}_{s_i}, \tau^{\uparrow}_{s_i}, c^{\downarrow}_{t_i}, \tau^{\downarrow}_{t_i})\}. \quad (9)$$

5.1 Exact Offer Selection

Let $\sigma^{\star}$ be the cost of the optimal global-path, initially $\sigma^{\star} = +\infty$. After constructing the sets $N^{\uparrow}(s')$ and $N^{\downarrow}(t')$, we scan the nodes of $N^{\uparrow}(s')$. For each potential intermediate location v_1 in the set $N^{\uparrow}(s')$, we calculate $c_o(v_1 \to v_2)$ using the forward one-to-all Dijkstra algorithm for all $v_2 \in N^{\downarrow}(t')$. For each offer o_i such that $(i, d^{\uparrow}_{s_i}, \tau^{\uparrow}_{s_i}, d^{\downarrow}_{t_i}, \tau^{\downarrow}_{t_i}) \in B_o(v_1)$, we scan each node v_2 in the set $N^{\downarrow}(t')$. Then, if i is in the set $B_o(v_2)$ such that o_i and d form a reasonable fit with v_1 (v_2) as an intermediate pick-up (drop-off) location, and the cost of the global path $c(s, s', v_1, v_2, t, t')$ is less than $\sigma^{\star}$, then we update the optimal global-path $\sigma^{\star}$ to $c(s, s', v_1, v_2, t, t')$. The detail of the procedure is given in Algorithm 1.

Algorithm 1: Exact offer selection.

Require: Demand d, sets $N^{\uparrow}(s')$, $N^{\downarrow}(t')$, and $B_o(v), \forall v \in G$.
Ensure: The best driver $i^{\star}$, global-path $\sigma^{\star}$.

```
1: Initialization, i* ← −1, σ* ← +∞.
2: for all v1 in N↑(s') do
3:    Using the forward one-to-all Dijkstra Algorithm,
      compute the cost c_o(v1 → v2), ∀ v2 ∈ N↓(t').
4:    for all i in B_o(v1) do
5:       if i and d form a time synchronization at location
         v1 then
6:          for all v2 in N↓(t') do
7:             if i ∈ B_o(v2) and driver i and rider d form a
               reasonable fit with v1 as intermediate pick-
               up location and v2 as intermediate drop-off
               location then
8:                if σ* > c(s_i, s', v1, v2, t_i, t') then
9:                   i* ← i
10:                  σ* ← c(s_i, s', v1, v2, t_i, t')
11:               end if
12:            end if
13:         end for
14:      end if
15:   end for
16: end for
```

Note that the runtime of Dijkstra's Algorithm using Fibonacci Heaps is bounded by $O(|V|\log|V| + |E|)$. So, the two sets $N^{\uparrow}(s')$, $N^{\downarrow}(t')$ can be computed in $O(2 \cdot (|V|\log|V| + |E|))$. Furthermore, Algorithm 1 runs in $O(|N^{\uparrow}(s')|(|V|\log|V| + |E|) + |N^{\downarrow}(t')| \cdot$

$\sum_{v_1 \in N^{\uparrow}(s')} |B_o(v_1)|)$-time. Hence, the worst-case complexity of the *Exact offer selection* is $O((|N^{\uparrow}(s')| + 2)(|V|\log|V| + |E|) + |N^{\downarrow}(t')| \cdot \sum_{v_1 \in N^{\uparrow}(s')} |B_o(v_1)|)$-time.

5.2 Offer Selection Heuristic

Even if the exact offer selection method provides a solution in polynomial time, the computation time can be too long for large instances with real road network. In particular, when the proposed method is used in real time ridesharing system, in which a solution should be provided in a few seconds. In this section, we propose an heuristic method with a lower time complexity, that allows us to select the best offer that will be matched with the demand d.

Algorithm 2: Scan the set $N^{\uparrow}(s')$.

Require: Demand d, set $N^{\uparrow}(s')$, set $N^{\downarrow}(t')$ and $B_o(v), \forall v \in G$.
Ensure: The best driver $i^{\star}$, global-path $\sigma^{\star}$.

1: Initialization, $i^{\star} \leftarrow -1$, $\sigma^{\star} \leftarrow \infty$.
2: Using backward one-to-all Dijkstra Algorithm, compute the cost $c_o(v_1 \rightarrow t'), \forall\, v_1 \in N^{\uparrow}(s')$.
3: **for all** v_1 in $N^{\uparrow}(s')$ **do**
4: **for all** i in $B_o(v_1)$ **do**
5: **if** i and d form a time synchronization at location v_1 **then**
6: **if** $i \in B_o(t')$ **and** $\tau_d(s' \rightarrow v_1) + \tau_o(v_1 \rightarrow t') \leq (\tau_d(s' \rightarrow t') + \Delta_d)$ **and** $\tau_o(s_i \rightarrow v_1) + \tau_o(v_1 \rightarrow t') + \tau_o(t' \rightarrow t_i) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_{o_i})$ **and** $c_o(s_i \rightarrow t_i) + c_d(s' \rightarrow t') - c(s_i, s', v_1, t', t_i, t') \geq 0$ **then**
7: **if** $\sigma^{\star} > c(s_i, s', v_1, t', t_i, t')$ **then**
8: $i^{\star} \leftarrow i$
9: $\sigma^{\star} \leftarrow c(s_i, s', v_1, t', t_i, t')$
10: **end if**
11: **end if**
12: **if** $\tau_d(s' \rightarrow v_1) + \tau_o(v_1 \rightarrow t_i) + \tau_d(t_i \rightarrow t') \leq (\tau_d(s' \rightarrow t') + \Delta_d)$ **and** $\tau_o(s_i \rightarrow v_1) + \tau_o(v_1 \rightarrow t_i) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_{o_i})$ **and** $c_o(s_i \rightarrow t_i) + c_d(s' \rightarrow t') - c(s_i, s', v_1, t_i, t_i, t') \geq 0$ **then**
13: **if** $\sigma^{\star} > c(s_i, s', v_1, t_i, t_i, t')$ **then**
14: $i^{\star} \leftarrow i$
15: $\sigma^{\star} \leftarrow c(s_i, s', v_1, t_i, t_i, t')$
16: **end if**
17: **end if**
18: **end if**
19: **end for**
20: **end for**

The *offer selection heuristic* is provided by Algorithm 4 composed of Algorithms 2 and 3.
Algorithm 2 allows us to determine an offer that minimizes the *global-path*, by considering the nodes of the set $N^{\uparrow}(s')$ as potential pick-up locations, while the drop-off location is fixed at the rider (and driver) destination. Steps 6 and 12 of Algorithm 2 check the admissibility of the found path based on the maximum detour time allowed by the driver and the rider, as well as the generated cost-saving.

Algorithm 3: Scan the set $N^{\downarrow}(t')$.

Require: Demand d, set $N^{\uparrow}(s')$, set $N^{\downarrow}(t')$ and $B_o(v), \forall v \in G$.
Ensure: The best driver $i^{\star}$, global-path $\sigma^{\star}$.

1: Initialization, $i^{\star} \leftarrow -1$, $\sigma^{\star} \leftarrow \infty$.
2: Using forward one-to-all Dijkstra algorithm, compute the cost $c_o(s' \rightarrow v_2), \forall\, v_2 \in N^{\downarrow}(t')$.
3: **for all** v_2 in $N^{\downarrow}(t')$ **do**
4: **for all** i in $B_o(v_2)$ **do**
5: **if** i and d form a time synchronization at location s' **then**
6: **if** $i \in B_o(s')$ **and** $\tau_o(s' \rightarrow v_2) + \tau_d(v_2 \rightarrow t') \leq (\tau_d(s' \rightarrow t') + \Delta_d)$ **and** $\tau_o(s_i \rightarrow s') + \tau_o(s' \rightarrow v_2) + \tau_o(v_2 \rightarrow t_i) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_{o_i})$ **and** $c_o(s_i \rightarrow t_i) + c_d(s' \rightarrow t') - c(s_i, s', s', v_2, t_i, t) \geq 0$ **then**
7: **if** $\sigma^{\star} > c(s_i, s', s', v_2, t_i, t')$ **then**
8: $i^{\star} \leftarrow i$
9: $\sigma^{\star} \leftarrow c(s_i, s', s', v_2, t_i, t')$
10: **end if**
11: **end if**
12: **end if**
13: **if** i and d form a time synchronization at location s_i **then**
14: **if** $\tau_d(s' \rightarrow s_i) + \tau_o(s_i \rightarrow v_2) + (\tau_d(v_2 \rightarrow t') \leq (\tau_d(s' \rightarrow t') + \Delta_d)$ **and** $\tau_o(s_i \rightarrow v_2) + \tau_o(v_2 \rightarrow t_i) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_{o_i})$ **and** $c_o(s_i \rightarrow t_i) + c_d(s' \rightarrow t') - c(s_i, s', s_i, v_2, t_i, t') \geq 0$ **then**
15: **if** $\sigma^{\star} > c(s_i, s', s_i, v_2, t_i, t')$ **then**
16: $i^{\star} \leftarrow i$
17: $\sigma^{\star} \leftarrow c(s_i, s', s_i, v_2, t_i, t')$
18: **end if**
19: **end if**
20: **end if**
21: **end for**
22: **end for**

On the other hand, Algorithm 3 determines the best offer that minimizes the *global-path*, by considering the nodes of the set $N^{\downarrow}(t')$ as potential intermediate drop-off locations, while the pick-up location of the ride is fixed to the origin location of the driver (and rider). Finally, the best offer i^* is given by the best result of Algorithms 2 and 3. In order to improve the cost of the global-path of matching (i^*, d), we use the method described in section 4.2 to find the best intermediate locations. The detail of the procedure is given in Algorithm 4.

Algorithm 2 and Algorithm 3 run in respectively, $O(|V|\log|V| + |E| + \sum_{v_1 \in N^{\uparrow}(s')} |B_o(v_1)|)$-time and $O(|V|\log|V| + |E| + \sum_{v_2 \in N^{\downarrow}(t')} |B_o(v_2)|)$-time, respectively. Then, having the sets $N^{\uparrow}(s')$, $N^{\downarrow}(t')$ and the best offer $i^{\star}$, the step 4 in the Algorithm 4 can be

Algorithm 4: Offer selection heuristic.

Require: Demand d, set $N^{\uparrow}(s')$, set $N^{\downarrow}(t')$ and $B_o(v), \forall v \in G$.
Ensure: The best driver $i^{\star}$, global-path $\sigma^{\star}$.

1: Find the best driver $i_1^{\star}$ in $N^{\uparrow}(s')$ using **Algorithm 2**.
2: Find the best driver $i_2^{\star}$ in $N^{\downarrow}(t')$ using **Algorithm 3**.
3: Select between the two drivers $i_1^{\star}$ and $i_2^{\star}$ the driver $i^{\star}$ with shortest global-path $\sigma^{\star} = \min(\sigma_1^{\star}, \sigma_2^{\star})$.
4: Find the best intermediate locations r_1 and r_2 between driver $i^{\star}$ and demand d using method described in section 4.2.

done in $O((|V|\log|V| + |E|) + |N^{\uparrow}(s')| + |N^{\downarrow}(t')|)$-time. Finally, the *offer selection heuristic* method allows us to reduce the complexity to $O(5 \cdot (|V|\log|V| + |E|) + |N^{\uparrow}(s')| + |N^{\downarrow}(t')| + \sum_{v_1 \in N^{\uparrow}(s')} |B_o(v_1)| + \sum_{v_2 \in N^{\downarrow}(t')} |B_o(v_2)|)$.

6 REWARD CONSIDERATION

In our a posteriori approach, the user (driver and/or rider) does not set the detour factor ε in advance, and the system calculates the most attractive value of ε that guarantee the best matching between the rider and the driver. However in some cases, either the driver or the rider may ask for a minimum rate of the saved cost in his trip. For instance, the driver requires at least 10% of the saved cost relative to his initial travel cost. In that case, we extend model to take into account that requirement. More precisely, an offer and a demand of ridesharing will be represented by $o = (s, t, [t_o^{\min}, t_o^{\max}], \Delta_o, \sigma_o)$ and $d = (s', t', [t_d^{\min}, t_d^{\max}], \Delta_d, \sigma_d)$, respectively, where σ_o (σ_d) is the minimum percentage of cost-saving fixed by the driver (rider) relative to his shortest path. Constraint 3 of the definition 2 is modified as follow.

$$c_o(s \to t) + c_d(s' \to t') - (c_o(s \to r_1) + c_d(s' \to r_1) + c_o(r_1 \to r_2) + c_o(r_2 \to t) + c_d(r_2 \to t')) \geq \sigma_o \cdot c_o(s \to t) + \sigma_d \cdot c_d(s' \to t') \quad (10)$$

The values of ε_1 and ε_2 of lemma 2 are modified in order to consider a constraint (10).

7 COMPUTATIONAL RESULTS

In this section, we provide experimental results using the proposed methods, i.e, the two heuristics **BSA** and **SPOA**, the exact method (**EM**) and the classical ridesharing (**CR**).

Environment. The proposed methods were implemented in C# Visual Studio 2010. The experiments were done on an Intel Xeon E5620 2.4 Ghz processor, with 8 GB RAM memory.

Offers-demands Data. In our experiments, we use real data provided by Covivo company[1]. These data concern employees of Lorraine region traveling between their homes and their work places. The real data instance is composed of 1513 participants. Among them, 756 are willing to participate in ridesharing service only as drivers. The rest of participants (i.e, 757) are willing to use other modes in order to reach the intermediate pick-up location from their origins, as well as from the intermediate drop-off location to their destinations. However, given the lack of available data of others transportation modes in the French region Lorraine, we assume that the driver and the rider have the same cost (i.e, each rider owns a private car that he might use during a part of his trip, either in order to reach the intermediate pick-up location coming from his origin, or/and reaching his destination coming from the intermediate drop-off location). Thus, the data instance is composed of 756 offers and 757 demands. The smallest and the greatest trip's distances are 2 *km* and 130 *km*, respectively.

The set of offers and demands are filtered such that we can never find a driver's offer and a rider's demand which have either the same starting or ending locations. This way will allow us to test the flexibility of our approach compared to the classical ridesharing approach. The time window for each trip is fixed as follows. The early departure time and the latest departure time are fixed at 7:30 a.m. and at 8:00 a.m, respectively. The time detour of the driver (rider) is fixed to at most 20% of his initial trip duration.

Road Network. Several geographical maps are available for geolocalised applications. The most famous ones are Google Maps, Bing Maps, Nokia Maps and OpenStreetMap[2] (OSM). Our road network of the French region Lorraine was derived from the publicly available data of OpenStreetMap (OSM) and was provided by GeoFabrik [3]. It consists of 797830 nodes and 2 394 002 directed edges. Each node in the road network can be an intermediate pick-up (resp. drop-off) location in which the driver and the rider can meet (resp. separate). OSM is used in several projects and it facilitates the map integration or exploitation. For instance, OsmSharp is an open-source mapping tool designed to work with OpenStreetMap-data. In our experiments, we use OsmSharp's routing and OSM data processing library to test our shortest path computation on real data set.

[1] http://www.covivo.fr
[2] http://www.openstreetmap.org
[3] http://www.geofabrik.de/

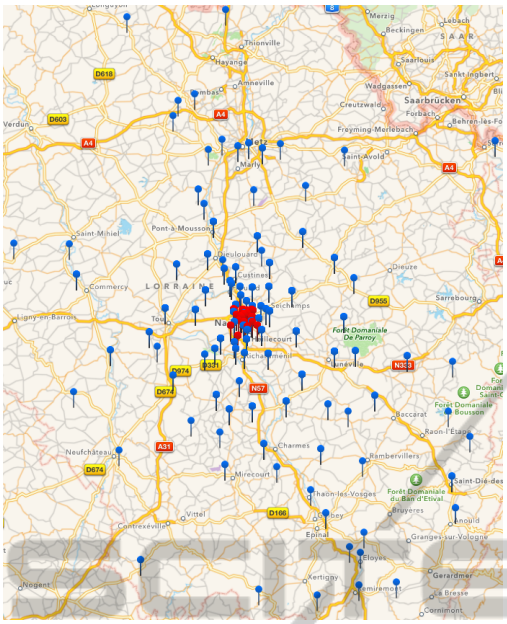


Figure 3: Map visualization of offers of ridesharing. Red and blue points are the geocoded home and work locations, respectively.

Computational Results. Two computational experiments were conducted. In the first experiment, a demand and an offer of ridesharing were fixed, and we evaluate the performance of four methods (**BSA**, **SPOA**, **EM** and **CR**) in terms of cost-saving, number of matchings and running-time (see Table 1). In this first experiment, two scenarios were tested. In the first scenario, the driver's detour time was limited to 20%, and in the second scenario, the time detour is limited to 10%. In each scenario, several instances were generated as follow: we scan all demands and for each demand we randomly select 10 offers, then 7570 instances (757 (demands) $\times$ 10 (offers)) were tested. For each method, the following results are reported: Gap: the deviation of algorithms with respect to the optimal solution, Time: the required CPU time in seconds, Match($\mathcal{M}$): the percentage of number matching found. In Columns *Gap* and *Time*, the average values on the 7570 instances are reported. In the row ($|\mathcal{C}|$), the average number of nodes contained in potential intermediate locations (class $\mathcal{C}$) over the 7570 instances is reported.

As we can see, the proposed heuristics (**BSA** and **SPOA**) detect all matchings and provide exact results for all instances when the detour time is less than 20%. Whereas, when the detour time is less than 10%, the Match($\mathcal{M}$) of **BSA** method decreases to 94% and **SPOA** method decreases to 97%. On the other hand, the **CR** approach detects at most 55.7% when the detour time is less than 20% and decreases more sharply to 15% with detour time less than 10%. The Gap of **CR** can be considered as the additional cost-saving generated by using methods with intermediate locations. The running time in **BSA** method is less than in **SPOA** method, this is due to the one-to-all Dijkstra algorithm used in **SPOA** compared to **BSA**.

Table 1: Performance of our algorithm with fixed offer.

	Detour time ≤ 0.2			Detour time ≤ 0.1		
	Gap (%)	$\mathcal{M}$ (%)	Time (s)	Gap (%)	$\mathcal{M}$ (%)	Time (s)
EM	-	-	844	-	-	620
BSA	0	100	1.78	0.4	94	1.22
SPOA	0	100	2.10	0.2	97	1.58
CR	27	55.7	0.92	34	15	0.83
$\lvert\mathcal{C}\rvert$	450			742		

In the second experiment, we evaluate the heuristic selection of offers. This allows us to know in practice, if the selected offer found using one intermediate location (either pick-up or drop-off location) is generally the same selected offer found using two intermediate locations (see Table 2). The column "Time" represents the running-time of the whole algorithm, whereas the row "Same offer (SA)" represents the success rate in which the selected offer in the step 3 (**Algorithm 4**) is the same offer selected by the exact method (**EM**).

Table 2: Performance of the whole process.

	Detour time ≤ 0.2			Detour time ≤ 0.1		
	Gap (%)	SA (%)	Time (s)	Gap (%)	SA (%)	Time (s)
EM	-	-	9409	-	-	7803
BSA	0.7	91	3.79	2.1	83	2.64
SPOA	0.3	91	5.22	1.4	83	3.87
$\sum_{v\in\mathcal{C}}\lvert B(v)\rvert$	113836			84742		

From Table 2, we can see that 83% (resp. 91% when detour time ≤ 0.2) of selected offers found using one intermediate location are the same with selected offers found using the two intermediate locations. Furthermore, when detour time ≤ 0.2, in **BSA** (resp. **SPOA**) method, the solution is found in about 3 seconds (resp. 5 seconds) compared to 9409 seconds for the exact method (**EM**). The Gap between the two heuristics does not exceed 2.1% in both cases of detour time values.

8 CONCLUSION

In this paper, we considered the problem of ridesharing with intermediate locations. We proposed solution methods that allow us to determine the interme-

diate pick-up and drop-off locations that minimize the total travel cost of rider and driver. The particular interest of our work is in making the service of ridesharing more flexible and efficient. Our approach is simple to use and allows to reduce the driver's detour by using intermediate locations, and to increase the savings for both drivers and riders compared to the classical ridesharing.

REFERENCES

Agatz, N., Erera, A., Savelsbergh, M., and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223:295–303.

Aissat, K. and Oulamara, A. (2014a). Dynamic ridesharing with intermediate locations. In *proceedings of SSCI 2014*.

Aissat, K. and Oulamara, A. (2014b). Round trip ridesharing with an intermediate meeting location. In *proceedings of MOSIM 2014*.

Amey, A. (2011). Proposed methodology for estimating rideshare viability within an organization: Application to the mit community. In *Transportation Research Board Annual Meeting*, pages pp. 11–25.

Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52:422–439.

Bit-Monnot, A., Artigues, C., Huguet, M.-J., and Killijian, M.-O. (2013). Carpooling: the 2 synchronization points shortest paths problem. In *OASIcs-OpenAccess Series in Informatics*, volume 33. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Drews, F. and Luxen, D. (2013). Multi-hop ride sharing. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search*, pages 71–79,.

Furuhata, M., Dessouky, M., Brunet, F. O. M., Wang, X., and Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46.

Geisberger, R., Luxen, D., Neubauer, S., Sanders, P., and Volker, L. (2010). Fast detour computation for ride sharing. In *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems - ATMOS'10*, pages pp. 88–99. Th. Erlebach, M. Lubbecke.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Sys. Sci. and Cyb*, 4:100–107.

Hartwig, S. and Buchmann, M. (2006). Empty seats traveling: next-generation ridesharing and its potential to mitigate traffic-and emission problems in the 21st century. *NOKIA Research Center*.

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

Ma, S. and Wolfson, O. (2013). Analysis and evaluation of the slugging form of ridesharing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 64–73. ACM.

Ma, S., Zheng, Y., and Wolfson, O. (2013). T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE.

Metz, B., Davidson, O., De Coninck, H., Loos, M., and Meyer, L. (2005). Ipcc special report on carbon dioxide capture and storage. prepared by working group iii of the intergovernmental panel on climate change. *IPCC, Cambridge University Press: Cambridge, United Kingdom and New York, USA*, 4.

Mote, J. E. and Whitestone, Y. (2011). The social context of informal commuting: Slugs, strangers and structuration. *Transportation Research Part A: Policy and Practice*, 45(4):258–268.

Sanders, P. and Schultes, D. (2007). Engineering fast route planning algorithms. In *Experimental Algorithms*, pages 23–36. Springer.

Sassi, O. and Oulamara, A. (2014). Joint scheduling and optimal charging of electric vehicles problem. In *Computational Science and Its Applications–ICCSA 2014*, pages 76–91. Springer.

Varone, S. and Janilionis, V. (2014). Insertion heuristic for a dynamic dial-a-ride problem using geographical maps. In *proceedings of MOSIM 2014*.