

# The Round-Trip Ridesharing Problem with Relay Stations

Kamel Aissat<sup>1,2(✉)</sup> and Ammar Oulamara<sup>1,2</sup>

<sup>1</sup> University of Lorraine - LORIA, Nancy, France  
{kamel.aissat, ammar.oulamara}@loria.fr

<sup>2</sup> University of Lorraine, Ile de Saulcy, Metz, France

**Abstract.** In this work, we investigate the potential benefits of introducing relay stations in the round-trip ridesharing problem. In the classical round-trip ridesharing system, the pick-up and drop-off locations for the rider don't differ from his origin and destination, respectively, for both outgoing and return trips. This system is straightforward but inflexible and unbalanced as it puts the whole detour effort on the driver's shoulders. In this paper, we propose to consider a meeting location as flexible and to determine its optimal position minimizing total travel cost for the round-trip. The meeting locations correspond to relay stations in which riders find ridesharing vehicles. The introduction of relay stations in the round-trip ridesharing problem creates dependency between the outgoing and return trips, in the sense that the relay stations must be chosen in such a way as to minimize the combined travel cost of the outgoing and return trips. In this setting, the rider is supposed to drive to the relay station with his private car and to park it there, so the return trip has to drop him there to get his car back. We present efficient algorithms to solve this problem and, finally, we perform a comparative evaluation using a real road network and real dataset provided by a local company. Our numerical results show the effectiveness of our system, which improves participants' cost-savings and matching rate compared to the classical round-trip ridesharing system.

## 1 Introduction

The growth of the nation combined with the need to meet mobility, environmental, and energy objectives require other alternative transportation systems. In fact, public transportation systems are insufficient to address the needs of commuters in terms of flexibility and availability. One potential solution to meet these requirements without expanding service area or increasing service frequency of public transportation, is to use ridesharing services. Ridesharing service is based on better use of vehicle by connecting drivers and riders so they can share all or part of their commute, cutting transportation costs for the participants while reducing traffic congestion and pollution. Both drivers and riders benefit from this service; the drivers save money by sharing the trips' cost, and the riders obtain their travels with attractive costs. 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 and drivers through a network service by using a smartphone both as a geolocation and a communication device. Several research has been reported recently in the fields of ridesharing, as in [1] and [6]. The authors in [3] consider the one-way ridesharing problem with intermediate meeting locations. Their system is defined as follows: given a set of drivers' offers already in the system, and a new rider's request, they determine a best driver, a best pick-up and drop-off locations, and a sharing cost rate between rider and driver for their common path. In [9], an approach that allows a dynamic scheduling of ridesharing requests (offers and demands), in a context which also involves standard public transportation modes is proposed. This approach is based upon dynamic labelling of the nodes of some transit network together with a filtered search for existing potential riders and drivers.

Although ridesharing provides many advantages, some users are reluctant to participate as rider in such a service. This concerns especially users who are not ready to drop their private cars for a ride with others drivers if their return trips are not ensured by the ridesharing service.

Very few researches are focused on the round-trip ridesharing problem. In [2], the authors study the round-trip ridesharing 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. When a user is assigned as a rider, their system ensures another matching for the return trip. In [8], the authors propose an approach that synchronizes an outgoing path and a return path in a location, while minimizing the global cost of the two paths. This problem which is defined as the 2-Way Multi Modal Shortest Path problem doesn't take into consideration ridesharing as a mode of transport. Furthermore, in existing round-trip ridesharing systems, a rider's origin can't differ from his pick-up location. However, in some situations, the rider accepts to travel with his private car to a relay station which can be considered as a new pick-up location, more or less close to his initial starting location, where he will be picked up by the driver and dropped off at his ending location. In this case the ridesharing system should guarantee the existence of a new matching for the rider's return trip passing through the relay station, i.e, a matching with another driver that accepts to share his car with the rider and drop him off where his car was left.

In this study, we pursue a two-fold goal.

- (i) The primary purpose is to increase the opportunity to obtain a matching between drivers and riders. Indeed, if the rider accepts to travel on his own car to a relay station close to driver's origin, the driver will make less detour to pick-up the rider. Additionally, this allows to reduce the total travel cost.
- (ii) The second goal is to ensure for the rider the return trip via the relay station where he left his private car. In fact, the rider will not accept to leave his car in a relay station to participate in a ridesharing service, if the return trip passing through this relay station is not ensured by the system.

To the best of our knowledge, our work is the first to consider the round-trip ridesharing problem with relay stations. This problem consists in minimizing the total cost of the round-trip. In this study, we consider a practical setting by exploiting a real road network of the French Lorraine region with a validation of the proposed solutions on real data.

The remainder of the paper is structured as follows. Section 2 describes our model of ridesharing. Section 3 explains the algorithmic details. Section 4 presents detailed experimental analysis of our algorithms. Finally, concluding remarks and future research are included in Section 5.

## 2 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. In our model, for an edge  $(i, j) \in E$  we associate two weights  $c(i, j)$  and  $\tau(i, j)$ , where  $c(i, j)$  represents the traveling cost and  $\tau(i, j)$  the traveling time between  $i$  and  $j$ , respectively. A path in a graph  $G$  is represented by a vector  $\mu = (u, \dots, v)$  of nodes in which two successive nodes are connected by an edge of  $E$ . The cost  $c(\mu)$  of path  $\mu$  is the sum of costs of all edges in  $\mu$ . A shortest-path between a source node  $u$  and a target node  $v$  is the 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  $u \rightarrow v$ .

An offer  $i$  of ridesharing is represented by  $o_i = (s_i, e_i, [t_i^{\min}, t_i^{\max}], \Delta_i)$  where  $s_i$  is the starting location,  $e_i$  the ending location,  $[t_i^{\min}, t_i^{\max}]$  the departure time window and  $\Delta_i$  is the detour time. A detour time is the maximal time that the driver accepts as overtime of his shortest-path from  $s_i$  to  $e_i$ . When a driver travels in round-trip, we generate two offers separately, one for outgoing trip and another for return trip.

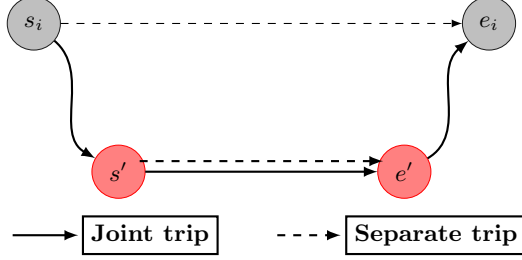
A demand of ridesharing is represented by  $d = (s', e', [t_{\min}^{\text{out}}, t_{\max}^{\text{out}}], [t_{\min}^{\text{ret}}, t_{\max}^{\text{ret}}], \Delta_d^{\text{out}}, \Delta_d^{\text{ret}})$  where  $s'$  is the starting location,  $e'$  is the ending location,  $[t_{\min}^{\text{out}}, t_{\max}^{\text{out}}]$  and  $[t_{\min}^{\text{ret}}, t_{\max}^{\text{ret}}]$  are the outgoing departure time window and return departure time window, respectively, and  $\Delta_d^{\text{out}}$  and  $\Delta_d^{\text{ret}}$  are the detour time of the outgoing and return trips, respectively.

In our road graph  $G$ , traveling costs of drivers and riders are distinguished, more precisely, an edge  $(i, j)$  has a nonnegative traveling cost  $c_k(i, j)$  depending on the fact that the edge is used by driver, i.e.,  $k = o$ , or by rider, i.e.,  $k = d$ .

In [2], authors consider a constraint on saved cost of traveling  $cs(o_i, d)$  as a necessary condition of a feasible matching between a rider's request and a driver's offer. The saved cost of traveling is defined as the difference between the travel cost of the driver including serving the rider and the travel costs of the driver and the rider if each of them travels alone, i.e

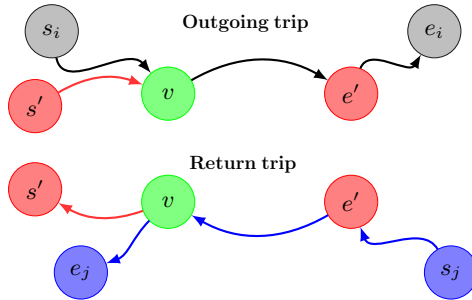
$$\begin{aligned} cs(o_i, d) &= c_o(s_i \rightarrow e_i) + c_d(s' \rightarrow e') \\ &\quad - (c_o(s_i \rightarrow s') + c_o(s' \rightarrow e') + c_o(e' \rightarrow e_i)) \end{aligned} \quad (1)$$

Thus, a feasible matching is accepted only if the cost of a *Joint trip* is less than the cost of the separate trips, i.e.,  $cs(o_i, d) > 0$ . If the cost of *Joint trip* is more expensive than the cumulated cost of the individual trips of the driver and the rider, then the matching will not be considered (see Figure 1).



**Fig. 1.** Example of single rider, single driver ride-share matching.

In order to increase the opportunity of matching, the rider may accept to be picked-up in relay station  $v$ . More precisely, the rider travels with his private car from his starting location  $s'$  to relay station  $v$ . He parks his car at relay station  $v$ , and he shares a ride with driver  $i$ , from relay station  $v$  till his end location  $e'$ . Whereas, a driver  $i$  travels from his starting location  $s_i$  to relay station  $v$ , he picks up the rider traveling together till end location  $e'$ , where the rider is dropped off, and finally the driver continues to his ending location  $e_i$ . The system should guarantee to the rider the existence of a second feasible matching with another driver  $j$  to reach the relay station  $v$  from  $e'$  in the return trip. The travel paths of the rider  $d$ , the driver  $i$  and the driver  $j$  are depicted in Figure 2.



**Fig. 2.** Round-trip ridesharing with  $v$  as a relay station.

## 2.1 Matching Constraints

In ridesharing systems a matching between drivers and riders can be established only when constraints of matching are satisfied. In our approach, we consider

two constraints of matching, namely, the timing constraint, and the travel cost constraint.

**Definition 1. (time synchronization)**

We say that a demand  $d$  and an offer  $o_i$  form a time synchronization at relay station  $v$  in outgoing trip if and only if there exists  $\beta \geq 0$ , where,

$$\beta = \min \begin{cases} t_{o_i}^{\max} + \tau_{o_i}(s_i \rightarrow v) - (t_{\min}^{\text{out}} + \tau_d(s' \rightarrow v)) & (2a) \\ t_{\max}^{\text{out}} + \tau_d(s' \rightarrow v) - (t_{o_i}^{\min} + \tau_{o_i}(s_i \rightarrow v)) & (2b) \end{cases}$$

Equation (2a) means that when a rider leaves his starting location  $s'$  at  $t_{\min}^{\text{out}}$  to reach the relay station  $v$ , he must arrive no later than the latest arrival time of the driver at the relay station  $v$ . The same reasoning is applied for the driver in equation (2b). Thus, when  $\beta \geq 0$ , the arrival time window for the driver and the rider at the relay station  $v$  will coincide. So, they can meet each other in this station at time  $\max \{t_{\min}^{\text{out}} + \tau_d(s' \rightarrow v), t_{o_i}^{\min} + \tau_{o_i}(s_i \rightarrow v)\}$ .

**Definition 2. (reasonable fit in outgoing trip)**

We say that a demand  $d$  and an offer  $o_i$  form a reasonable fit in outgoing trip with  $v$  as a relay station if and only if  $d$  and  $o_i$  form a time synchronization at relay station  $v$  in outgoing trip, and

$$c_o(s_i \rightarrow e_i) + c_d(s' \rightarrow e') - (c_o(s_i \rightarrow v) + c_d(s' \rightarrow v) + c_o(v \rightarrow e') + c_o(e' \rightarrow e_i)) \geq 0 \quad (3)$$

$$\tau_o(s_i \rightarrow v) + \tau_o(v \rightarrow e') + \tau_o(e' \rightarrow e_i) \leq (\tau_o(s_i \rightarrow e_i) + \Delta_i) \quad (4)$$

$$\tau_d(s' \rightarrow v) + \tau_o(v \rightarrow e') \leq (\tau_d(s' \rightarrow e') + \Delta_d^{\text{out}}) \quad (5)$$

The constraint (3) ensures that the incurred cost in the ridesharing is more attractive than the cost when the driver and the rider travel alone. The constraints (4) and (5) concern the detour time in outgoing trip. The term  $\tau_o(s_i \rightarrow e_i) + \Delta_i$  in (4) (resp.  $\tau_d(s' \rightarrow e') + \Delta_d^{\text{out}}$  in (5)) allows to limit the amount of time that the driver (resp. rider) passes in traveling in outgoing trip. We consider that the service times to pick-up and drop-off of riders are not significant and can be considered as instantly done.

**Definition 3. (reasonable fit in return trip)**

We say that a demand  $d$  and an offer  $o_j$  form a reasonable fit in return trip with  $v$  as a relay station if and only if  $d$  and  $o_j$  form a time synchronization at location  $e'$  in return trip, such that :

$$c_o(s_j \rightarrow e_j) + c_d(e' \rightarrow s') - (c_o(s_j \rightarrow e') + c_o(e' \rightarrow v) + c_d(v \rightarrow s') + c_o(v \rightarrow e_j)) \geq 0 \quad (6)$$

$$\tau_o(s_j \rightarrow e') + \tau_o(e' \rightarrow v) + \tau_o(v \rightarrow e_j) \leq (\tau_o(s_j \rightarrow e_j) + \Delta_j) \quad (7)$$

$$\tau_o(e' \rightarrow v) + \tau_d(v \rightarrow s') \leq (\tau_d(e' \rightarrow s') + \Delta_d^{\text{ret}}) \quad (8)$$

The constraint (6) considers the travel cost-savings by both rider  $d$  and driver  $j$  in return trip. Whereas, the constraints (7) and (8) consider the detour time for the driver and the rider, respectively.

**Lemma 1.** *A demand  $d$  and two offers  $o_i$  and  $o_j$  form a reasonable fit in round-trip, if and only if, there exists a relay station  $v$ , where  $d$  and  $o_i$  form a reasonable fit in outgoing trip with  $v$  as a relay station in outgoing trip, and,  $d$  and  $o_j$  form a reasonable fit in return trip with  $v$  as a relay station.*

## 2.2 Objective of Ridesharing System

In the following we use the term *round-trip-path*  $\langle d, i, j, v \rangle$  to describe the round-trip of rider's demand  $d$  with the driver  $i$  in outgoing trip and the driver  $j$  in return trip, passing through the relay station  $v$ .

A shortest *round-trip-path*  $\langle d, i, j, v \rangle$  is the *round-trip-path* with minimal cost, denoted by  $c(\langle d, i, j, v \rangle)$ , such that

$$c(\langle d, i, j, v \rangle) = c(\langle d, i, v \rangle)_{out} + c(\langle d, j, v \rangle)_{ret} \quad (9)$$

where

$$c(\langle d, i, v \rangle)_{out} = c_o(s_i \rightarrow v) + c_d(s' \rightarrow v) + c_o(v \rightarrow e') + c_o(e' \rightarrow e_i) \quad (10)$$

and

$$c(\langle d, j, v \rangle)_{ret} = c_o(s_j \rightarrow e') + c_o(e' \rightarrow v) + c_o(v \rightarrow e_j) + c_d(v \rightarrow s') \quad (11)$$

In a nutshell, for a demand  $d$ , our objective is to determine, an offer  $o_i$  in the outgoing trip, a relay station  $v$ , and an offer  $o_j$  in return trip such that the cost of *round-trip-path* is minimized, and the demand  $d$  form a reasonable fit in round-trip with the offer  $o_i$  and the offer  $o_j$  having  $v$  as a relay station.

## 3 Algorithmic Details

In this section, we explain the algorithmic details of our solving approaches. Note that the system is launched when a rider request enters the system, but when a driver proposes an ridesharing offer, some informations are stored.

### 3.1 Adding Offer

Each new offer entering into the system generates new ridesharing opportunities. For this purpose, when an offer  $o_i$  is added to the system, we store all possible relay stations for this offer, then we determine the costs  $c_o(s_i \rightarrow v)$  and  $c_o(v \rightarrow t_i)$ . More precisely, we denote by  $N^\uparrow(s_i)$ ,  $N^\downarrow(t_i)$  the *forward search space* from a source  $s_i$  and the *backward search space* from target  $t_i$ , respectively. A forward search space  $N^\uparrow(s_i)$  is a set of triplets: node, cost and time  $(v, d_{s_i}^\uparrow, \tau_{s_i}^\uparrow)$  such that there is a path from  $s_i$  to  $v$  with cost  $d_{s_i}^\uparrow$  and travel time  $\tau_{s_i}^\uparrow$  to reach node  $v$ . A backward search space  $N^\downarrow(t_i)$  is a set of triplet: node, cost, and time  $(v, d_{t_i}^\downarrow, \tau_{t_i}^\downarrow)$

such that there is a path from  $v$  to  $t_i$  with cost  $d_{t_i}^\downarrow$  and travel time  $\tau_{t_i}^\downarrow$  to reach  $t_i$  from  $v$ .

Based on the constraints of *detour time*, we can limit the search spaces when we compute these costs without considering any demand. In our proposed approach, we started by computing the set  $N^\downarrow(t_i)$  using reverse Dijkstra Algorithm, and for each settled node  $v$ , we check the constraint  $\tau_o(v \rightarrow t_i) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_i)$ . The search procedure is stopped at the first settled node  $v$  which violates this constraint. In the second step, we compute the forward search space  $N^\uparrow(s_i)$ , using Dijkstra Algorithm [4]. The search procedure is stopped when the first node  $v$  that violates the time constraint  $\tau_o(s_i \rightarrow v) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_i)$  is settled. Finally, we keep a node  $v$  in  $N^\uparrow(s_i)$  only if  $(\tau_o(s_i \rightarrow v) + \tau_o(v \rightarrow t_i)) \leq (\tau_o(s_i \rightarrow t_i) + \Delta_i)$ .

At each iteration, when a node  $v$  which corresponds to relay station is selected in the set  $N^\uparrow(s_i)$ , we add the entries in the *bucket*  $B(v)$ , i.e.,

$$B(v) := B(v) \cup \{(i, d_{s_i}^\uparrow, \tau_{s_i}^\uparrow)\}. \quad (12)$$

The bucket serves to store for a relay station  $v$ , all trips which can pass via this location without violating the lower bound on constraint of *detour time*.

To take into consideration the drivers having already begun their trips, we must update their locations at each time unit. For simplicity, we consider only drivers who have not yet started their trips when the rider enters the system.

### 3.2 Adding a Demand

For given demand  $d$ , the objective of the matching procedure is to select the *best fit in round-trip* by scanning the potential buckets in round-trip. More precisely, we determine a class  $\mathcal{C}$  of potential relay stations, where the *detour time* constraint of the demand  $d$  is respected simultaneously in the *outgoing* and *return trips*. Algorithm 1 allows to recover the class  $\mathcal{C}$  of potential relay stations.

---

#### Algorithm 1. Potential relay stations in round-trip

---

**Require:** Graph  $G(V, E)$ , demand  $d$ .

**Ensure:** Class of potential relay stations in round-trip  $\mathcal{C}$ .

---

- 1: Set  $\mathcal{C}_{out} = \emptyset$  and  $\mathcal{C}_{ret} = \emptyset$ .
  - 2: Compute  $N^\uparrow(s')$ , using a forward one-to-all Dijkstra algorithm from  $s'$  with rider cost bounded by  $(\tau_d(s' \rightarrow t') + \Delta_d^{out})$ .
  - 3: Compute  $N^\downarrow(t')$ , using a backward one-to-all Dijkstra algorithm from  $t'$  with driver cost bounded by  $(\tau_d(s' \rightarrow t') + \Delta_d^{out})$ , for each settled relay station  $v$  that satisfied the equation (5),  $\mathcal{C}_{out} = \mathcal{C}_{out} \cup \{v\}$ .
  - 4: Compute  $N^\downarrow(s')$ , using a backward one-to-all Dijkstra algorithm from  $s'$  with rider cost bounded by  $(\tau_d(t' \rightarrow s') + \Delta_d^{ret})$ .
  - 5: Compute  $N^\uparrow(t')$ , using a forward one-to-all Dijkstra algorithm from  $t'$  with driver cost bounded by  $(\tau_d(t' \rightarrow s') + \Delta_d^{out})$ , for each settled relay station  $v$  that satisfied the equation (8),  $\mathcal{C}_{ret} = \mathcal{C}_{ret} \cup \{v\}$ .
  - 6:  $\mathcal{C} = \mathcal{C}_{out} \cap \mathcal{C}_{ret}$ .
-

Steps 4 and 5 allow to store all nodes  $v$  that satisfy the constraint of detour time for the rider in outgoing trip (5). The same reasoning is applied in steps 6 and 7 on the constraint of detour time in the return trip (8). Finally in the step 8, we keep in a class  $\mathcal{C}$  only nodes that satisfy both constraints (5) and (8).

Once the class of potential relay stations  $\mathcal{C}$  is determined, it remains to scan each bucket of this class and select the relay station  $v$  with the minimum cost of the *round-trip-path*. The scanning method of buckets is described in the Algorithm 2.

Thus, the *best fit in round-trip* for a given demand  $d$  is computed by Algorithm 3.

**Complexity.** The runtime of Dijkstra's Algorithm using Fibonacci Heaps is bounded by  $O(|V| \log |V| + |E|)$ . Hence, the worst-case complexity of the Algorithm 3 is  $O(4(|V| \log |V| + |E|) + \sum_{v \in \mathcal{C}} |B(v)|)$  where  $|B(v)|$  is the number of offers in the bucket  $v$ .

---

**Algorithm 2.** Scan bucket  $B(v)$ 


---

**Require:**  $B(v)$ , demand  $d$ .

**Ensure:** *round-trip-path*  $c(d, i^*, j^*, v)$ .

```

1: Initialization: Cost_out  $\leftarrow \infty$ , Cost_ret  $\leftarrow \infty$ ,  $i^* \leftarrow -1$ ,  $j^* \leftarrow -1$ .
2: for all  $i$  in  $B(v)$  do
3:   if  $o_i$  and  $d$  form a reasonable fit in outgoing trip with  $v$  as a relay station then
4:     Compute  $c(\langle d, i, v \rangle)_{out}$  as described in (10)
5:     if  $c(\langle d, i, v \rangle)_{out} < \text{Cost\_out}$  then
6:       Cost_out  $\leftarrow c(\langle d, i, v \rangle)_{out}$ 
7:        $i^* \leftarrow i$ 
8:     end if
9:   end if
10:  if  $o_i$  and  $d$  form a reasonable fit in return trip with  $v$  as a relay station then
11:    Compute  $c(\langle d, i, v \rangle)_{ret}$  as described in (11)
12:    if  $c(\langle d, i, v \rangle)_{ret} < \text{Cost\_ret}$  then
13:      Cost_ret  $\leftarrow c(\langle d, i, v \rangle)_{ret}$ 
14:       $j^* \leftarrow i$ 
15:    end if
16:  end if
17: end for
18:  $c(\langle d, i^*, j^*, v \rangle) \leftarrow \text{Cost\_out} + \text{Cost\_ret}$ 

```

---



---

**Algorithm 3.** Adding a demand

---

**Require:** Graph  $G(V, E)$ , demand  $d$ .

**Ensure:** driver  $i^*$ , driver  $j^*$ , best relay station  $v^*$  and  $c(\langle d, i^*, j^*, v^* \rangle)$ .

```

1: Compute  $\mathcal{C}$  using Algorithm 1
2: for all  $v$  in  $\mathcal{C}$  do
3:   Scan a Bucket  $B(v)$  using Algorithm 2
4:   Store the minimum cost  $c(\langle d, i^*, j^*, v^* \rangle)$ 
5: end for

```

---



### 3.3 Total Gain in Round-trip

The *Matching procedure* described above, ensures for the rider a gain simultaneously in the *outgoing* and *return* trips (i.e, constraints (3) and (6)). In this way, the sharing of the *cost-savings* between the ride-share partners in the *outgoing* trip is independent of ride-share partners in the *return* trip. But in practice, the gain must be ensured in *round-trip*, not necessarily in both *outgoing* trip and *return* trip. For example, we can have a negative *cost-savings* in outgoing trip equal to -3 between the rider  $d$  and the driver  $i$ . In the return trip, the rider  $d$  can be matched with the driver  $j$  with *cost-savings* equal to 8. Thus, the rider can recover the cost that he lost in outgoing trip (i.e, -3). The main difficulty here lies in the way of sharing the saved cost, which depends both on the rider-share partners in outgoing and return trips. Note that the sharing procedure of saved cost is not considered in this study.

To take into account the total gain in round-trip instead of separate gain in outgoing and return trips, we just need to modify the lines 5, 12 and 20 in Algorithm 2. More precisely, in the lines 5 and 12, we only check the constraints of *detour time* in outgoing and return trips, respectively. Finally, in line 20, we must ensure the constraint of the *cost-savings* between the rider  $d$ , the driver  $i^*$  and the driver  $j^*$ , defined as follows:

$$c_d(s' \rightarrow t') + c_o(s_{i^*} \rightarrow e_{i^*}) + c_d(t' \rightarrow s') + c_o(s_{j^*} \rightarrow e_{j^*}) - c(\langle d, i^*, j^*, v \rangle) \geq 0 \quad (13)$$

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  $o_i = (s_i, e_i, [t_i^{\min}, t_i^{\max}], \Delta_i, \sigma_{o_i})$  and  $d = (s', e', [t_{\min}^{\text{out}}, t_{\max}^{\text{out}}], [t_{\min}^{\text{ret}}, t_{\max}^{\text{ret}}], \Delta_d^{\text{out}}, \Delta_d^{\text{ret}}, \sigma_d)$ , respectively, where  $\sigma_{o_i}$  ( $\sigma_d$ ) is the minimum percentage of cost-saving fixed by the driver (rider) relative to his shortest path. In such case, the constraint of the *cost-savings* between the rider  $d$ , the driver  $i^*$  and the driver  $j^*$ , defined as follows:

$$c_d(s' \rightarrow t') + c_o(s_{i^*} \rightarrow e_{i^*}) + c_d(t' \rightarrow s') + c_o(s_{j^*} \rightarrow e_{j^*}) - c(\langle d, i^*, j^*, v \rangle) \geq \sigma_{o_i} \cdot c_o(s_{i^*} \rightarrow t_{i^*}) + \sigma_{o_j} \cdot c_o(s_{j^*} \rightarrow t_{j^*}) + \sigma_d \cdot c_d(s' \rightarrow t') \quad (14)$$

### 3.4 Removing Offers

To remove an offer  $o_i$ , we need to remove its entries from the buckets. In order to accelerate the time calculation of the system, we classify buckets according to calendar date. Furthermore, for each calendar date, we have not defined an order in which the entries of a bucket are stored. This makes adding operation of an offer very fast, but removing it, requires scanning the buckets. Thus, scanning all buckets is prohibitive as there are too many entries. Instead, it is faster to compute  $N^\uparrow(s_i)$  and  $N^\downarrow(t_i)$  in order to obtain the set of buckets which contains an entry of this offer  $o_i$ . Then, we only need to scan those buckets and remove the entries of offer  $o_i$ .

## 4 Experiments

Computational experiments have been conducted to compare the performance of the proposed algorithms. In this section, the classical approach of round-trip ridesharing was denoted by CRT, the round-trip ridesharing with relay stations while ensuring for the rider a gain in the *outgoing* and *return trips* was denoted by RTR, and the round-trip ridesharing with relay stations while ensuring for the rider a total gain in round-trip was denoted by RTRG. The results of experiments were evaluated in terms of quality, number of matchings and running-time.

*Environment.* The algorithms were coded in C# and run using an Intel(R) Core(TM) i7-3520M CPU 2.90 Ghz, with 8 GB RAM memory. A binary heap was used as the priority queue data structure.

*Offers-demands Data.* In our experiments, we use a real data provided by Covivo company<sup>1</sup>. These data concern employees of Lorraine region traveling between their homes and their work places. The real data instance is composed of 756 offers and 757 demands in *round-trip*. In our experiments, for each offer in round-trip, we generate two offers separately, one for outgoing trip and another for return trip. Thus, the number of generated offers reach 1512. Concerning the demands, each rider owns a private car that he might use during a part of his trip. The set of offers and demands are filtered in the way that we can never find an offer and a demand which have both the same starting and ending locations. The time window for each trip is fixed as follows. For the outgoing trip from home to work, the early departure time and the latest departure time are fixed at 7:30 a.m. and at 8:00 a.m, respectively. For the return trip, the early departure time and the latest departure time are fixed at 18:00 p.m. and at 18:15 p.m, respectively. The detour time of the driver (rider) is fixed to at most 20% of his initial trip duration.

*Road Networks.* Our road network of the French region Lorraine was derived from the publicly available data of OpenStreetMap<sup>2</sup> (OSM) and was provided by GeoFabrik<sup>3</sup>. It consists of 7 978 30 nodes and 2 394 002 directed edges. Each node in the road network can be a relay station in which the driver and the rider can meet each other. 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 paths computations on real data sets.

*Computational Results.* To facilitate our analysis, we divide the set of offers and demands into six main classes, and each class represents a geographical city (see Table 1). For each class, we compute the following parameters that allow us to compare the efficiency of different solution approaches.

<sup>1</sup> <http://www.covivo.fr>

<sup>2</sup> <http://www.openstreetmap.org/>

<sup>3</sup> <http://www.geofabrik.de/>

- Average success rate of matching ( $\mathcal{M}$ ): the number of matched demands divided by the total number of demands within each class.
- Average cost-savings rate in round-trip ( $\mathcal{C}$ ) : the sum of cost-savings rate in round-trip generated by the matched demands divided by the total number of matched demands within each class.

**Table 1.** Performance of three approaches

Classes (Cities)	# demands	CRT	RTR		RTRG	
		$\mathcal{M}$ (%)	$\mathcal{M}$ (%)	$\mathcal{C}$ (%)	$\mathcal{M}$ (%)	$\mathcal{C}$ (%)
Nancy	311	94.2	97.1	35.2	97.1	35.2
V.L.N	199	96.4	98.4	32.7	98.4	32.7
Metz	38	68.4	94.7	22.7	97.3	21.8
Luneville	107	95.3	99	25.7	99	25.7
Toul	74	91.8	97.2	27.8	97.2	27.8
Epinal	28	60.7	82.1	21.2	89.2	20.2

Results of Table 1 show that RTR and RTRG outperform CRT in terms of successful matching ( $\mathcal{M}$ ) over the all six classes.

The gap between CRT and RTR (RTRG) of successful matching decreases with the offers' density in the concerned classes. For instance, the gap between CRT and RTR (RTRG) reaches 26.3% (28.9%) in Metz class and 21.4% (28.5%) in Epinal class. In fact, when the density of offers is low and the offers are distant from demands, this implies that drivers must make long detours to pick-up the riders, however, the detour time and the detour cost limit the successful matching. Nevertheless, when the rider accepts to travel with his own car to a relay station (RTR and RTRG approaches), this allows to reduce the detour of the drivers, therefore more successful matchings are found. Note also that in RTR and RTRG, the rate of successful matching are the same in all classes, except in Metz and Epinal classes where RTRG outperforms RTR with a slight ratio that does not exceed the 7.1%.

Regarding results in Epinal and Metz classes, RTRG is less efficient than RTR in terms of cost-savings ( $\mathcal{C}$ ). This is due to the fact that the average cost-savings ( $\mathcal{C}$ ) is calculated over instances where successful matching is found, and RTRG detects more matchings than RTR. In order to have a better idea on the performance of CRT, RTR and RTRG in terms of cost-savings, we evaluate the additional cost-savings that our approaches provide compared to the classical ridesharing CRT on instances in which successful matching is found by three approaches (CRT, RTR and RTRG), hereafter denoted by  $\mathcal{I}$ . Then, we compare the approaches RTR and RTRG on instances where a matching is found only by both RTR and RTRG, hereafter denoted by  $\mathcal{H}$ . The results are reported in Table 2. In column  $\mathcal{C}_{\mathcal{I}}$ , the average cost-savings in round-trip on set of instances  $\mathcal{I}$  are shown. In column  $\mathcal{C}_{\mathcal{H}}$ , the average cost-savings in round-trip on set of instances  $\mathcal{H}$  are shown.

**Table 2.** Additional cost-savings

Classes (Cities)	$\mathcal{C}_{\mathcal{I}}$			$\mathcal{C}_{\mathcal{H}}$	
	CRT (%)	RTR (%)	RTRG (%)	RTR (%)	RTRG (%)
Nancy	31	36.2	36.2	35.2	35.2
V.L.N	29.7	33.3	33.3	32.7	32.7
Metz	25	32.3	32.3	22.7	22.7
Luneville	25.9	26.8	26.8	25.7	25.7
Toul	26.4	29.4	29.4	27.8	27.8
Epinal	25	28.7	28.7	21.2	21.2

From the Table 2, we can see that the generated cost-saving is significant in both approaches RTR and RTRG compared to CRT for all classes. Indeed, the average cost-savings per round-trip reaches 36.2% with our approaches. Furthermore, the two approaches RTR and RTRG have the same cost-savings for the detected matchings. In the case where a matching is detected only by RTRG, the cost-savings of round-trip can never exceed 25%. Indeed, in outgoing trip (resp. return trip), we can never save more than 25% of the round-trip. Since RTR approach does not detect a matching, then either outgoing trip or return trip generates a negative cost-savings. So the cost-savings that RTRG saves in case RTR found no matching does not exceed 25%. Thus, the major advantage of the RTRG approach compared to RTR was in increasing the matching rate while conserving attractive cost-savings.

To illustrate the effectiveness of the whole system, we evaluate the time to add/remove a demand and an offer depending on the existing numbers of offers and buckets in the system. Tables 3 and 4 summarize the time needed to add/remove a demand and an offer depending on the numbers of offers and buckets. Each entry in the Table 3 is an average value over instances in each class.

From Table 3, the average time of adding a demand increases linearly with the number of buckets and the number of offers in buckets. In CRT, we only need to scan two buckets corresponding to vertices of the starting and ending locations of the demand. However, in RTR and RTRG, we need to scan all

**Table 3.** Adding demand

Classes (Cities)	#		# offers			Runtime [s]		
	buckets		in	buckets	[-10 <sup>3</sup> ]	CRT	RTR	RTRG
Nancy	2	285	0.064	48	48	0.19	0.44	0.44
V.L.N	2	374	0.090	47	47	0.24	0.55	0.56
Metz	2	2442	0.010	39	39	2.77	3.39	3.52
Luneville	2	1700	0.043	44	44	0.55	1.73	2.03
Toul	2	832	0.041	48	48	0.49	1.45	1.51
Epinal	2	3740	0.006	31	31	3.5	5.29	5.45

buckets in *the set of potential relay stations*  $\mathcal{C}$ . The slight difference in running-time between RTR and RTRG ( $< 0.2$  sec) is due to the way in which we scan a bucket. Contrary to RTR approach, in RTRG, we check the *constraint of cost-savings* only after having verified all offers contained in the bucket that satisfy *the constraint of detour time*. Thus, the number of filtered offers in bucket is less important compared to the RTR approach.

In order to evaluate the time needed to add and to remove an offer, we have fixed three offers  $o_1$ ,  $o_2$  and  $o_3$  composed of 99, 239 and 549 buckets, respectively. These offers are added/removed from the system according to the number of offers already contained in these buckets.

From Table 4, we observe that the time to add an offer  $o_i$  is independent of the number of offers already in the system. The main time is spent in computing  $N^\uparrow(s_i)$  and  $N^\downarrow(t_i)$ . However, removing the offer requires scanning these buckets. Thus, the run-time to add and remove offers remains at microseconds scale.

Figure 3 gets a better visualization of the running time of adding and removing offer.

**Table 4.** Adding and removing offer

Offer	# buckets	# offers in buckets	Add [s]	Remove [s]
$o_1$	99	100	0.03	0.032
		250	0.03	0.041
		550	0.03	0.053
$o_2$	239	100	0.05	0.052
		250	0.05	0.062
		550	0.05	0.074
$o_3$	549	100	0.08	0.083
		250	0.08	0.093
		550	0.08	0.1



**Fig. 3.** Adding and removing offers depending on the number of buckets and offers in the system.

Summarizing these results, the two approaches RTI and RTIG provide efficient results in terms of successful matching, cost-savings and running-time. They can also solve the problem of ridesharing with a high number of offers and relay stations within a few seconds of computation time.

## 5 Conclusion

This paper sets out methods for round-trip ridesharing that allow the rider to use his private car in order to reach a relay station, more or less close to his initial starting location. The decision making process on the choice of the relay station relates to the density of offers passing via this location. This approach was validated by experiments based on real data of ridesharing in the French Lorraine region. The main advantages of this approach are increasing the opportunity of matching between riders and drivers and then a significant reduction of the total travel cost compared to the classical approach of round-trip ridesharing. As perspectives, several extensions can be considered. For example, the case where some ride-share participants announce trips in which they are flexible to serve as drivers or riders. Ride-share matching optimization in this case must not only determine the best relay station, but also assign a role to each of the participants. For future work, it may be interesting to consider the approach described in [5] by allowing riders to switch between several drivers while guaranteeing their return trips. Secondly, some procedures for sharing saved costs between the outgoing and return trip of one round-trip will be investigated. Finally, a natural avenue for future research is accelerating our approach using Contraction Hierarchies [7].

## References

1. Agatz, N., Erera, A., Savelsbergh, M., Wang, X.: Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* **223**, 295–303 (2012)
2. Agatz, N., Erera, A., Savelsbergh, M., Wang, X.: Dynamic ride-sharing: a simulation study in metro atlanta. *Transportation Research Part B, Methodological* **45**, 1450–1464 (2011)
3. Aissat, K., Oulamara, A.: A posteriori approach of real-time ride-sharing problem with intermediate locations. In: *Proceedings of the 4rd International Conference on Operations Research and Enterprise Systems, ICORES 2015, Lisbon, Portugal, January 10–12, 2015* (2015)
4. Dijkstra, E.: A note on two problems in connexion with graphs. *Numerische Mathematik* **1**, 269–271 (1959)
5. Drews, F., Luxen, D.: Multi-hop ride sharing. In: *Proceedings of the Sixth Annual Symposium on Combinatorial Search*, pp. 71–79 (2013)
6. Furuhashi, M., Dessouky, M., Brunet, F.O.M., Wang, X., Koenig, S.: Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological* **57**, 28–46 (2013)
7. Geisberger, R., Sanders, P., Schultes, D., Vetter, C.: Exact routing in large road networks using contraction hierarchies. *Transportation Science* **46**, 388–404 (2012)

8. Huguet, M.J., Kirchler, D., Parent, P., Calvo, R.W.: Efficient algorithms for the 2-way multi modal shortest path problem. *Electronic Notes in Discrete Mathematics* **41**, 431–437 (2013)
9. Varone, S., Aissat, K.: Muti-modal transportation with public transport and ride-sharing mutli-modal transportation using a path-based method. In: *Proceedings of the 17th International Conference on Enterprise Information Systems, ICEIS 2015*, vol. 1, Barcelona, Spain, April 27–30, 2015 (2015)

Computational Logistics

6th International Conference, ICCL 2015, Delft, The Netherlands, September 23-25, 2015, Proceedings

Corman, F.; Voß, S.; Negenborn, R.R. (Eds.)

2015, XV, 752 p. 215 illus. in color., Softcover

ISBN: 978-3-319-24263-7