Published in Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS 2015), 2015, 8 p. which should be cited to refer to this work

Multi-modal Transportation with Public Transport and Ride-sharing
Multi-modal Transportation Using a Path-based Method

Sacha Varone[1], Kamel Aissat[2]
[1]*University of Applied Sciences and Arts Western Switzerland (HES-SO), HEG Genève, Switzerland*
[2]*LORIA, University of Lorraine, France*
sacha.varone@hesge.ch, kamel.aissat@loria.fr

Keywords: Multi-modal, Public transportation, Ride-sharing, Real-time, Geographical maps.

Abstract: This article describes a multi-modal routing problem, which occurs each time a user wants to travel from a point A to a point B, using either ride-sharing or public transportation. The main idea is to start from an itinerary using public transportation, and then substitute part of this itinerary by ride-sharing. We first define a closeness estimation between the user's itinerary and available drivers. This allows to select a subset of potential drivers. We then compute sets of driving quickest paths, and design a substitution process. Finally, among all admissible solutions, we select the best one based on the earliest arrival time. We provide numerical results using benchmarks based on geographical maps, public transportation timetabling and simulated requests and driving paths. Our numerical experiment shows a running time of a few seconds, suitable for a new real-time transportation application.

1 INTRODUCTION

Mobility within a region can be done using different transportation modes, either public or private, which depends on the preferences or the possibility of the commuter. In this paper, we deal with a combination of transportation modes, to travel from a point A to a point B. The originality of our research resides in the mix between public transportation and ride-sharing, which results in a new real-time transportation business opportunity.

We start from using public transportation, included walking sub-paths and then, considering (quasi) real-time ride-sharing opportunities, try to transfer sub-paths from public modality to ride-sharing modality. For that purpose, we consider riders, who provide requests to go from one location to another one, and drivers who offer ride-sharing. Riders' goal is to find a quickest trip in a dynamic context that effectively combines the use of several modes of transportation.

The network is represented as a directed graph model $G(V,E)$, in which V represents a set of nodes and E the set of arcs. Nodes model intersections and arcs depict street segments. A non negative weight is associated to each arc: its cost. A path that minimizes the sum of its cost is called a quickest/shortest path. We define a *stop* to be the locations for which a transit or road node exists. Stops correspond to bus stops, subway stations, parking, etc. In this multi-modal context, G is modeled as the union of multiple networks, representing different travel modes (bike, car, public transport, ...). Our model uses public transport network and road network. In this paper we propose an algorithmic approach to the real-time multi-modal earliest-arrival problem (EAP) in urban transit network, using public transportation and ride-sharing, so that the trip duration is minimized. We transfer part of the trip from public transport to ride-sharing sub-paths between two stops.

We organize the paper as follows: Section 2 refers to the motivation and a brief literature review is presented. Section 3 explains the algorithmic process and its complexity, each step being illustrated with an example. Finally, Section 4 presents insight about its efficiency and gives concluding remarks.

2 BACKGROUND

The search for a quickest or shortest path has been well studied in the literature, and has now very efficient algorithms to solve it, in the order of a millisecond for a continental trip: The authors in (Bast et al., 2014) survey recent advances in algorithms for route planning, updating the survey of the work addressed

in (Delling et al., 2009).

Although the computation of shortest paths are key components in route planning, they are not sufficient to deal with multi-modal problems, since the latter involve multi-criteria paths, transition or waiting times, etc. which is usually not taken into account in shortest paths on pure road networks. For example, the authors in (Ambrosino and Sciomachen, 2014) include several features in their objective function, so that they focus on the modal change node, and propose a two-step algorithm for computing multi-modal routes. An exact algorithm considering arrival and departure time-windows has been proposed in (Liu et al., 2014). Multi-criteria search by computing the Pareto set is done in (Delling et al., 2013).

Public transportation problems have received a lot of attention, solving earliest-arrival problems (EAP) knowing the departure and arrival station, departure time and timetable information. A review of this topic can be found in (Müller-Hannemann et al., 2007; Pyrga et al., 2008). The two main multi-modal networks are described, namely the time-expanded graph and the time-dependent graph. Our approach uses the time-dependent graph, since does not explode the number of nodes. Timetabling information and EAP solving is nowadays often available on-line, either via a web browser or via requests to a restful server. In some cases, timetables information might not be accurate and approximations based on probability distribution might be applied, as done in (Murueta et al., 2014). Our approach considers that such a service is available.

Real-time journey using ride-sharing opportunities faces the commuting point problem: it has to be decided where the pick-up and drop-off locations have to be located. Bit-Monnot *et al.*(Bit-Monnot et al., 2013) define this problem as the 2 synchronization points shortest path problem (2SPSPP). More precisely, for a given driver and a rider, 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. Their objective function minimizes the cumulated travel time of driver and rider. But the time complexity of this method prevents its use in real-time ride-sharing, and their model does not take into account the driver's detour time constraint, i.e., the total time of the detour should be less than a given threshold specified for the driver.
In our approach, we first find a shortest path using public transportation and consider its different transit stops. We then only allow pick-up and drop-off around those stops to reduce the search space. We also consider the *best offer selection problem*, i.e. for a given rider, we select the best driver that improves the rider's itinerary by setting the different transit stops as potential pick-up and drop-off locations, under driver's detour time and driver's waiting time constraints.

3 APPROACH: SUB-PATH SUBSTITUTION

The problem to be solved is presented as follows: a user wishes to go from an origin point O_u to a destination point D_u. He might use either public transportation, ride-sharing or a combination of both. All public transportation timetabling are assumed to be known or at least accessible easily. In Switzerland, one might use the Swiss public transport API (Application Programming Interface)[1]; in France, a similar service is available[2]. The origin-destination (OD) couple of potential ride-sharing drivers are also detected and located in real-time.

Throughout this section, together with the described algorithmic process, we present an illustrative example in order to better understand the different steps that constitute our approach. The example represents the following situation: a user u requests to travel from an origin O_u to a destination D_u, starting at time t_u = 9:00. Public transportation allows him to go from a point x_2 to another point x_{nbs}, closed to respectively points O_u and D_u, via points $x_2, x_3, \ldots, x_{nbs-1}$. In order to simplify the understanding, our illustrative example supposes that the origin O_u is already a bus stop, hence $O_u = x_1$. For simplicity again, only one driver k is considered (see Figure 3-9).

The public transportation path is noted as $P = O_u, x_2, \ldots, x_{nbs}, D_u$, its successive points are called "stops". Note that "nbs" stands for number of stops.
Let's call OD the set of origin-destination couple of users. An element $O_k D_k \in OD$ is characterized by its origin O_k, its destination D_k and its starting time t_0^k for user k.

Notation :

[1] http://transport.opendata.ch
[2] http://www.navitia.io

$s \rightsquigarrow e$	driving quickest path between s and e.
$\delta(s,e)$	duration of a quickest path between s and e.
$d(s,e)$	distance as the crow flies between s and e.
$\hat{\delta}(s,e)$	estimated smallest duration from s to e $\hat{\delta}(s,e) = \frac{d(s,e)}{v_{\max}}$, where $v_{\max}$ is the maximal speed.
λ_k	detour coefficient, $\lambda_k \geq 1$.
$\tau(x)_{ab}$	time required for moving from modality a to b at x.
$t_a(x,m)$	arrival time at x using the m transport modality starting at time t_0.
$t_d(x,m)$	departure time at x using the m transport modality starting at time t_0.
$w^k_{\max}$	maximum waiting time for driver k at pick-up point.
$LP^{\downarrow}_k$	forward search space from O_k.
$LP^{\uparrow}_k$	backward search space from D_k.

We will further note as p the public transportation modality, and as c the car modality.

3.1 Initial Request Processing

As a ride request O_uD_u arrives in the system at time t_0, a shortest path using public transportation is processed, as well as possible driving substitution sub-paths along the public transportation path. This is the purpose of Algorithm 1.

Algorithm 1 Initial request processing

Require: Demand O_uD_u, t_0
Ensure: Path P, driving quickest subpaths, *PDrive*
1: Find path $P = O_u, x_1, \ldots, x_{nbs}, D_u$ using public transportation API, starting in O_u at time t_0, with its associated arrival times $t_a(x,p)$ and departure times $t_d(x,p)$, $x \in P$.
2: Compute driving quickest paths along P, with its associated arrival times $t_a(x,c)$, $x \in P$.
3: Set $PDrive := \emptyset$
4: **for all** $x, y \in P$, x before y **do**
5: **if** $t_a(x,p) + \tau(x)_{pc} + \delta(x,y) \leq t_a(y,p)$ **then**
6: $PDrive := PDrive \cup \{(x,y)\}$
7: **end if**
8: **end for**
9: $LDriver = \emptyset$

Step 1 gives the path P using public transportation, with its arrival time and departure time on each of its commute node between position O_u and destination D_u. Step 2 computes possible driving substitution paths along P. Only those whose arrival time at the drop-off stop is less than the arrival time using public transportation are kept. The time at the potential pick-up stop plus the transshipment time plus the ride-sharing time until stop y, is compare to the arrival time at y without ride-sharing. If the gain in time for the user is not positive, then the considered ride-sharing (x,y) is not admissible. The admissible ride-sharing set is defined by *PDrive*. Step 9 initializes a list *LDriver* of potential drivers associated with the current rider u.

Figure 1 illustrates an instance where a rider u travels at starting time t_u = 9:00 from his origin O_u, which is also the first stop x_1, to his destination D_u. Figure 1 represents the situation after step 1 of Algorithm 1. For each stop $x_i \in P$, we associate a time window $[t_a(x_i,p), t_d(x_i,p)]$ that represents the arrival time at the stop x_i and the departure times from stop x_i, respectively. The path found by the API is composed of three different modes. The rider waits at his origin 3 minutes before boarding the bus at 9:03, then he is dropped off at stop x_2, walks from stop x_2 to stop x_3 during 5 minutes, and finally waits 5 more minutes before taking the train to reach his final destination D_u (see Figure 1).

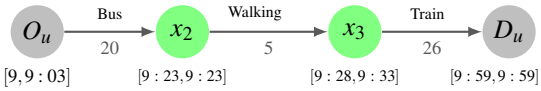

Figure 1: Path P using public transportation.

Figure 2 represents step 2 of Algorithm 1, where all potential driving substitution sub-paths are computed.

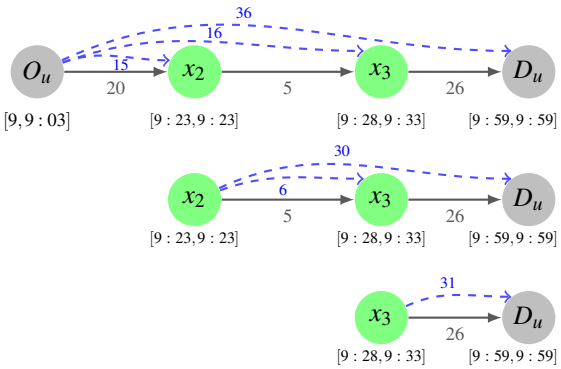

Figure 2: Quickest driving paths for every pair (x_i, x_j) such that stop x_j is situated after stop x_i are shown as dashed blue line.

In Figure 3 are shown the admissible potential substitution sub-paths, returned by Algorithm 1. The ride-sharing path (x_2, x_3) would results in an arrival time at x_3 later than that one if public transportation is use, since it requires 6 minutes from x_2 to

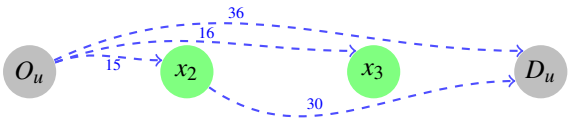

Figure 3: Admissible driving substitution arcs.

x_3, rather then 5 minutes. A similar situation occurs for the ride-sharing path (x_3, D_u): 31 minutes of ride-sharing compared to 26 minutes by public transportation. Therefore both sub-paths (x_2, x_3) and (x_3, D_u) are canceled.

3.2 Closeness Estimation

We define an estimate on how close is a substitution driving path OD to a public transportation sub-path of P. This estimated distance is defined as the minimal sum of the estimated distance from a vertex in OD to a vertex in P, and backward from P to OD. Four different points are used, so that only non-trivial substitution are allowed (i.e. no substitution of a single vertex). For that purpose, we estimate the distance between two points given by their latitude/longitude with the Haversine formula.
This formula uses a spherical model to estimate the distance between two points $x = (\lambda_1, \theta_1)$ and $y = (\lambda_2, \theta_2)$ on the earth surface.

$$a = sin(\frac{\theta_2 - \theta_1}{2})^2 + cos(\theta_1) * cos(\theta_2) * sin(\frac{\lambda_2 - \lambda_1}{2})^2$$

$$d(x,y) = R * 2 * atan2(\sqrt{a}, \sqrt{1-a})$$

where $\lambda_i, i = 1,2$ are the latitudes, $\theta_i, i = 1,2$ are the longitudes, $R \approx 6371$ [km] is the earth's radius. Thus, the estimated smallest duration from x to y is noted by $\hat{\delta}(x,y) = \frac{\hat{d}(x,y)}{v_{max}}$, such that v_{max} is the maximal speed of a car in concerned area. This is a solution to the so called "great-circle distance between two points problem (sometimes also called "orthodromic distance" problem).

Algorithm 2 restricts the list of potential drivers to those whose OD is close enough to P, and whose detour time is less than a threshold value. Step 5 checks if the driver's time from its origin to destination including the detour via x and y is less than its maximal detour bound.

Algorithm 2 Closeness substitution sub-path for driver k

Require: path P, λ_k, O_kD_k of a driver, $PDrive$, $LDriver$
Ensure: Potential pick-up locations $LP_k^{\downarrow}$
Potential drop-off locations $LP_k^{\uparrow}$
Update $LDriver$.

1: $LP_k^{\downarrow} := \emptyset$, $LP_k^{\uparrow} := \emptyset$
2: **for** each $(x,y) \in PDrive$ **do**
3: Estimate $\hat{\delta}(O_k, x)$ from O_k to x
4: Estimate $\hat{\delta}(y, D_k)$ from y to D_k
5: **if** $\hat{\delta}(O_k,x) + \delta(x,y) + \hat{\delta}(y,D_k) \leq \lambda_k \delta(O_k, D_k)$ **then**
6: Update $LP_k^{\downarrow} := LP_k^{\downarrow} \cup \{x\}$
7: Update $LP_k^{\uparrow} := LP_k^{\uparrow} \cup \{y\}$
8: Update $LDriver := LDriver \cup \{k\}$
9: **end if**
10: **end for**

Figure 4 shows the situation at the beginning of Algorithm 2. In the next steps describing Algorithm 2, driver's detour should not exceed more than 20% of his initial trip duration (i.e. $\lambda_k = 1.2$).

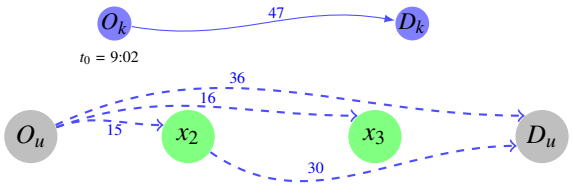

Figure 4: Driver k drives from O_k to D_k, starting at time 9:02. Nodes in path P with potential ride-sharing sub-paths are shown in dashed blue lines.

Figure 5 shows the results of the estimation based on the Haversine formula.

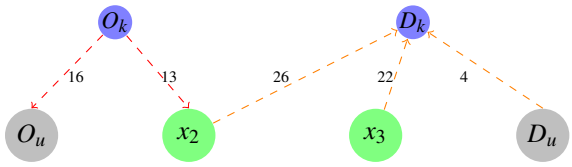

Figure 5: Estimation of the duration to each potential pick-up nodes, and from each potential drop-off nodes, for driver k, using the Haversine formula. Each geographical position are supposed to be known.

Figure 6 shows each potential substitution sub-path, starting from the driver's origin and ending to the driver's destination. The numbers on the segments represent the estimated duration, whereas the numbers along the blue arcs represent the true duration, as computed in Algorithm 1.

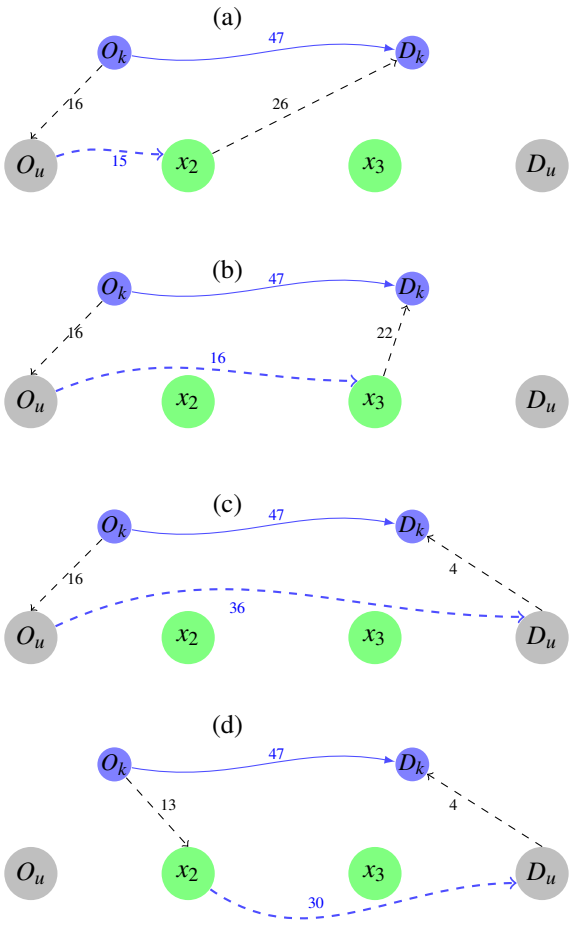

Figure 6: All potential substitution sub-paths are evaluated against the detour constraint.

Figure 7 shows the result of Algorithm 2, where arc (O_u, x_2) has been removed since ride-sharing on this path would implies to drive during $16+15+26=57$ minutes, exceeding the detour limit of $1.2 \times 47 = 56.4$ minutes. Then, we remove the potential driving substitution arc (O_u, x_2) because the lower bound of driver's detour duration is not satisfied. This allows to restrict the list of potential pick-up and drop-off locations

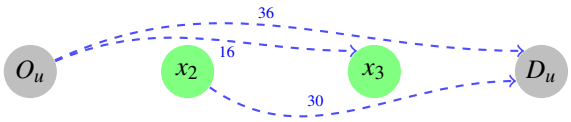

Figure 7: Only potential substitution sub-paths that do not violate the detour constraint are kept.

3.3 Shortest Paths Computation

Once a driver k is classified as a potential driver in *LDriver*, we compute with Algorithm 3 a set of shortest paths in order to check the exact constraints of *time detour* and *waiting time*.

Algorithm 3 Paths computation to/from P for driver k

Require: demand O_uD_u, O_kD_k of a driver, public transportation database (API), potential pick-up locations $LP_k^{\downarrow}$, potential drop-off locations $LP_k^{\uparrow}$.
Ensure: Set of quickest paths to pick-up locations LO_k
Set of quickest paths from drop-off locations LD_k

1: Compute driving quickest paths to $LP_k^{\downarrow}$
$LO_k = O_k \rightsquigarrow \left\{x \in LP_k^{\downarrow} \backslash \{D_u\}\right\}$
2: Compute driving quickest paths from $LP_k^{\uparrow}$
$LD_k = \left\{y \in LP_k^{\uparrow} \backslash \{O_u\}\right\} \rightsquigarrow D_k$

Algorithm 3 provides a set of computed shortest paths. It serves as a basis to substitute part of the P public transportation path with a ride-sharing modality. Step 1 constructs shortest driving paths from the position of the driver to each of the nodes in $LP_k^{\downarrow}$, except the last one D_u. The goal is to find a driving way to possible commutation points. Step 2 computes driving paths from nodes in $LP_k^{\uparrow}$, except the first one O_u, to the driver's destination D_k. Such a path will be used once a driver has dropped off a rider, and then have to find its route back to its destination D_k. (see Figure 8)

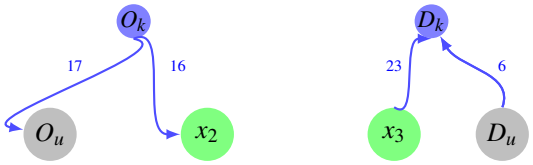

Figure 8: Exact driving paths computation. Only quickest paths from the origin to admissible pick-up and from admissible drop-off to destination are computed.

3.4 Substitution Process According to Time-savings

We only consider feasible substitution sub-paths that do not increase the arrival time of the rider to his destination. For that purpose we introduce the notion of *reasonable substitution sub-path* with the following definition:

Definition 3.1. (reasonable substitution sub-path) *We say that an arc (x,y) form a reasonable substitution sub-path for the driver k and the rider u if and only if the maximal waiting time $w^k_{\max}$ constraint for the driver k at a pick-up location (1), the maximum detour constraint (2) for the driver k and the latest arrival time constraints (3) for the rider u are satisfied, i.e,*

$$t_a(x,p)+\tau(x)_{pc}-t^k_a(x,c)\leq w^k_{\max} \qquad \textit{waiting} \qquad (1)$$

$$\delta(O_k,x)+\max\left\{t_a(x,p)+\tau(x)_{pc}-t^k_a(x,c),0\right\}+ \delta(x,y)+\delta(y,D_k)\leq \lambda_k\delta(O_k,D_k) \qquad \textit{detour} \qquad (2)$$

$$t^k_a(x,c)+\max\left\{t^k_a(x,c)-(t_a(x,p)+\tau(x)_{pc}),0\right\}+ \tau(x)_{pc}+\delta(x,y)\leq t_a(y,p) \qquad \textit{arrival} \qquad (3)$$

Constraints (1), (2) and (3) take into account time required for moving from public transportation modality to ride-sharing modality.

The objective function that defines the best substitution sub-path is based on time-savings: among all *reasonable substitution sub-path*, we select the *best substitution sub-path* (x,y) that generates for the rider the most positive time-savings, if he uses ride-sharing with driver k from the pick-up stop x to the drop-off stop y rather than public transportation (see Figure 9).

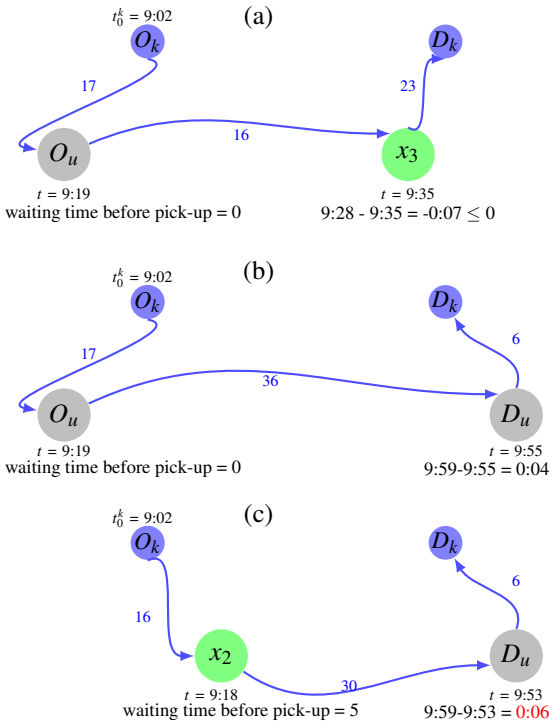

Figure 9: Best substitution sub-path. Ride-sharing (O_u,x_3) increases the earliest arrival time at x_3; therefore it is canceled. Between ride-sharing (O_u,D_u) and (x_2,D_u), the last one better improves the arrival time at D_u; it is therefore the best substitution sub-path.

Algorithm 4 defines this process.

Algorithm 4 Best substitution sub-path with driver k

Require: demand O_uD_u, O_kD_k of a driver, A_1, A_2, A_3
Ensure: best substitution sub-path $(x^\star,y^\star)$ or failed insertion

$\sigma^\star \leftarrow 0$
for each (x,y) in the substitution sub-path set **do**
  if (x,y) form a *reasonable substitution sub-path* for driver k and rider u **then**
    $\sigma = t_a(y,p) - \left(t^k_a(x,c) + \max\{t^k_a(x,c) - (t_a(x,p)+\tau(x)_{pc}),0\}+\tau(x)_{pc}+\delta(x,y)\right)$
    if $\sigma^\star \leq \sigma$ **then**
      $\sigma^\star \leftarrow \sigma$
      Update the best substitution sub-path $(x^\star,y^\star)$
    end if
  end if
end for

Note that from a drop-off location, the rider's route have to be recomputed to take into account his new arrival time. In this example, the drop-off location corresponds to the destination of rider. The new

itinerary of the rider is represented in Figure 10.

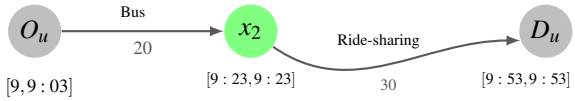

Figure 10: New itinerary of the rider.

Finally, in order to find the best substitution sub-path among all drivers, it remains to scan each driver k in the driver's list *LDrivers* and select the driver $k^\star$ that generates the best substitution sub-path $(x^\star, y^\star)$. This is done by Algorithm 5. It is to be noted that from $y^\star$, the rider's route have to be recomputed in order to take into account his new arrival time at $y^\star$. Therefore, the path P is updated, using $y^\star$ as the new origin (Step 13). The new starting time t'_0 is the sum of its arrival time at $y^\star$ and the modality change duration (Step 12). The driver's route is then $O_k \rightsquigarrow x^\star \rightsquigarrow y^\star \rightsquigarrow D_k$ (Step 15). In our example, the new itinerary of the driver k will become $O_k \rightsquigarrow x_2 \rightsquigarrow D_u \rightsquigarrow D_k$.

Algorithm 5 Best substitution sub-path among all drivers

Require: demand $O_u D_u, t_0$
Ensure: Best substitution sub-path among all drivers or failed insertion
1: Initial request processing using **Algorithm 1**
2: **for** each driver $k \in LDriver$ **do**
3: Compute the closeness substitution sub-path for driver k, using **Algorithm 2**
4: **if** $k \in LDriver$ **then**
5: Compute the paths to/from $LP_k^{\downarrow}/LP_k^{\uparrow}$ for driver k, using **Algorithm 3**
6: Find the best substitution sub-path with driver k, using **Algorithm 4**
7: Update the best substitution sub-path $(x^\star, y^\star)$ with associated driver $k^\star$
8: **end if**
9: **end for**
10: Substitution of x to y in P with $x^\star \rightsquigarrow y^\star$
11: **if** $y^\star \neq D_u$ **then**
12: $t'_0 = t_a^k(y^\star, c) + \tau(y^\star)_{cp}$
13: Find the path $P' = y^\star \ldots D_u$ using public transportation API, starting at time t'_0
14: **end if**
15: Update the driver's route to $O_{k^\star} \rightsquigarrow x^\star \rightsquigarrow y^\star \rightsquigarrow D_{k^\star}$.

3.5 Complexity

We describe the complexity of the whole process in a worst case analysis. Algorithm 1 initializes the request processing in $O(\mathcal{D}_{road} \cdot (|P|-1) + \mathcal{D}_{API})$-time, where $\mathcal{D}_{road}$ (resp. $\mathcal{D}_{API}$) is the complexity of the Dijkstra algorithm in a road (resp. multi-modal) network. Algorithm 2, which allows to determine the driver's admissibility to join the rider's trip, runs in $O(|PDrive|)$-time. Then, in order to check the exact detour time constraint of the driver, Algorithm 3 allows to compute two sets of shortest paths in the road network, one from the driver's origin to stops (LO_k) and another from the stops toward the driver's destination (LD_k). Thus, its complexity is in $O(2 \cdot \mathcal{D}_{road})$. Having the two sets LO_k and LD_k as input, Algorithm 4 finds the best substitution sub-path with driver k that satisfy the maximum driver's detour time and runs in $O(|PDrive|)$-time. Finally, Algorithm 5 uses as subroutine the previously described algorithms. We note that Algorithms 2, 3 and 4 have to be run for each driver k in the set *LDriver*. Therefore, the complexity of the whole process described by Algorithm 5 is in $O(\mathcal{D}_{road} \cdot (|P|-1) + \mathcal{D}_{API} + 2 \cdot |LDriver| \cdot (|PDrive| + \mathcal{D}_{road}))$.

4 NUMERICAL RESULTS AND CONCLUSION

The methodology to test our approach is based on simulations, since we are not aware of benchmarks that fit the problem we solve. We first chose k clusters corresponding to $k = 4$ cities in Switzerland (Fribourg, Lausanne, Bern, Neuchâtel). We then create $L_r = 10$ requests between for each pair of clusters, starting randomly between 7:00 and 8:00 AM on a weekday. For each request, we retrieve public transportation from available API. We then create $L_d = 5$ driving trips closed enough from the stops given by public transportation segments, using geographical maps from Open Street Map[3], as well as routing process available through the open source tool Osmsharp[4]. We then apply our algorithms to measure the gain in terms of arrival time, as well as the running time of our algorithms.

The results of our simulation give some insight about the efficiency of our approach. First, this shows that a true application running in real-time is possible since the whole process running on a personal computer requires only a few seconds and slightly

[3] http://www.openstreetmap.org
[4] http://www.osmsharp.com

depends on geographic density of drivers around an itinerary of a rider. The reason for a so small running time is mainly due to the filtered nature of Algorithm 2, which considerably reduces the list of potential drivers.

Second, there is a significant gain for the rider in terms of arrival time. Nevertheless the objective function, although appropriate for certain types of users, might be redefined in several terms: the total monetary cost for the rider, the deviation from the origin-destination trip for the driver, the number of transshipment stops, etc. Our approach can integrate those features as a weighted sum in the objective function. To conclude, the problem described in this paper is a first step in designing new multi-modal transportation process. Multi modalities usually includes pedestrian, cycling, private car or public bus or train transportations. Our approach allows to also include ride-sharing.

The particular interest of our work is in making the service of ride-sharing and public transportation more flexible and efficient. Specifically, in contrast to the traditional ride-sharing service, our approach allows to reduce the driver's detour by using intermediate pick-up and drop-off locations for the rider, and to increase the savings for both drivers and riders, compared to the traditional ride-sharing service. The ride-sharing service can be considered as a complement to transit for public transportation, i.e. the ride-sharing will improve transportation service in rural area, difficult to serve by public transportation only.

This is the main reason that will incite the public transport agencies to use ride-sharing to complement their services. Despite the relative importance of integrating ride-sharing into public transport services, no previous work exists that allows to deal with this problem in dynamic and real-time context. One of the reasons could be the difficulty to define and combine in real-time the two services: ride-sharing and public transportation.

ACKNOWLEDGEMENTS

This research was partially funded thanks to the swiss CTI grant 15229-1 PFES-ES received by the first author.

REFERENCES

Ambrosino, D. and Sciomachen, A. (2014). An algorithmic framework for computing shortest routes in urban multimodal networks with different criteria. *Procedia - Social and Behavioral Sciences*, 108(0):139 – 152. Operational Research for Development, Sustainability and Local Economies.

Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., and Werneck, R. (2014). Route planning in transportation networks. MSR-TR-2014-4 8, Microsoft Research.

Bit-Monnot, A., Artigues, C., Huguet, M.-J., and Killijian, M.-O. (2013). Carpooling : the 2 synchronization points shortest paths problem. In *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*, volume 13328, page 12, Sophia Antipolis, France.

Delling, D., Dibbelt, J., Pajor, T., Wagner, D., and Werneck, R. (2013). Computing multimodal journeys in practice. In Bonifaci, V., Demetrescu, C., and Marchetti-Spaccamela, A., editors, *Experimental Algorithms*, volume 7933 of *Lecture Notes in Computer Science*, pages 260–271. Springer Berlin Heidelberg.

Delling, D., Sanders, P., Schultes, D., and Wagner, D. (2009). Engineering route planning algorithms. In *Algorithmics of large and complex networks*, Lecture notes in computer science. Springer.

Liu, L., Yang, J., Mu, H., Li, X., and Wu, F. (2014). Exact algorithms for multi-criteria multi-modal shortest path with transfer delaying and arriving time-window in urban transit network. *Applied Mathematical Modelling*, 38(9-10):2613–2629.

Müller-Hannemann, M., Schulz, F., Wagner, D., and Zaroliagis, C. (2007). Timetable information: Models and algorithms. In Geraets, F., Kroon, L., Schoebel, A., Wagner, D., and Zaroliagis, C., editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 67–90. Springer Berlin Heidelberg.

Murueta, P. O. P., García, E., and de los Angeles Junco Rey, M. (2014). Finding in multimodal networks without timetables. In *VEHICULAR 2014 : The Third International Conference on Advances in Vehicular Systems, Technologies and Applications*.

Pyrga, E., Schulz, F., Wagner, D., and Zaroliagis, C. (2008). Efficient models for timetable information in public transportation systems. *J. Exp. Algorithmics*, 12:2.4:1–2.4:39.