A Real-Time Algorithm to Solve the Peer-to-Peer Ride-Matching Problem in a Flexible Ridesharing System

October 16, 2017

Transportation Research Part B

Neda Masoud
Assistant professor
Department of Civil and Environmental Engineering
University of Michigan Ann Arbor, Ann Arbor, MI, 48109
nmasoud@umich.edu
Corresponding author

R. Jayakrishnan
Professor
Department of Civil and Environmental Engineering
University of California Irvine, Irvine, CA, USA, 92697
rjayakri@uci.edu

Abstract

Real-time peer-to-peer ridesharing is a promising mode of transportation that has gained popularity during the recent years thanks to the wide-spread use of smart phones, mobile application development platforms, and online payment systems. An assignment of drivers to riders, known as the ride-matching problem, is a central component of a peer-to-peer ridesharing system. In this paper, we discuss the features of a flexible ridesharing system, and propose an algorithm to optimally solve the ride-matching problem in a flexible ridesharing system in real-time. We generate random instances of the problem, and perform sensitivity analysis over some of the important parameters in a ridesharing system. Furthermore, we discuss two novel approaches to increase the performance of a ridesharing system.

Keywords: On-demand transportation, ridesharing, ride-matching, multi-modal transportation

1 Introduction

Congestion in urban transportation networks is one of the common problems faced by many countries around the world. In addition to having a direct impact on travel time and fuel consumption, congestion imposes indirect costs by increasing travel time uncertainty, as well as emission levels which adversely affect human health and ecosystems. Managing congestion by expanding the infrastructure is costly and damaging to the environment. An alternative way is to make more efficient use of the existing transportation infrastructure. Public transportation is a conventional way of using the existing capacity on the roadway network more efficiently.

Transit systems in urban networks mainly include buses and rail services. They typically carry multiple passengers, and therefore can help reduce vehicle miles traveled (VMT) and ease congestion in urban networks. One drawback of transit services is that they operate on fixed routes and schedules, which limits their coverage of the network, both geographically and temporally. Urban transit systems are typically and necessarily designed to satisfy peak period demands. Due to significant peaking behavior of demand, the system capacity is often drastically under-utilized during off-peak periods, which causes significant cost inefficiencies. Government regulations on fares exacerbate the cost concerns. Thus transit services usually fail to act as financially independent entities, and are in need of subsidies.

Para-transit services were originally introduced to run as supplementary services alongside transit, and as a means to increase the flexibility of public transportation. Para-transit services include all shuttle-like services that serve customers such as airport travelers, employee or student commuters, or the elderly and disabled. Since the passage of the Americans with Disabilities Act of 1990, however, the term para-transit has been used more commonly to refer to the services provided to persons with disabilities, or to the elderly. In this paper, we reserve the term para-transit to refer to such services. These services are demand-responsive, and usually serve multiple passengers at a time, based on spatiotemporal proximity of the requests they receive. Routes and schedules are fairly flexible and demand-dependent.

Although para-transit services can be beneficiary to demographics they target, these demographics are fairly limited. In addition, most services that fall under this category are offered by non-profit organizations that are supported by federal or state funding. Due to the limit on the amount of subsidies, it is not possible to extend these services to the general public. These limitations alongside the desire for more comfortable (and possibly quicker) travel options have resulted in a much higher demand for private sector alternatives, such as taxis.

Taxis are a private form of demand-responsive transportation alternative. They provide door-to-door transportation, but at a higher cost that not everyone can afford. Shared taxi/limousine/shuttle services provide an opportunity for customers to share their trips, cutting the cost of the journey, and potentially reducing the total miles traveled in the network. Over the years, different shared-use services have been designed. Flexible route transit systems (Quadrifoglio et al. 2008; Li and Quadrifoglio 2010; Qiu et al. 2014),

and High-Coverage Point-to-Point Transit (HCPPT) (Cortés and Jayakrishnan, 2002) are a few examples. HCPPT is perhaps the first conceptual system that envisaged the option of private drivers offering rides to be shared, and entirely eliminated fixed route transit, though the somewhat misleading word "transit" was used in the name.

In all alternative modes of transportation above, with the exception of HCPPT, drivers work as system employees. Some of the discussed transportation alternatives are more flexible than others in terms of routing and scheduling, and some have the potential to benefit the environment and customers by allowing them to share (parts of) their trips. In the next set of transportation alternatives we discuss, drivers are not employed by the agencies.

A different family of transportation alternatives which has attracted considerable attention during the recent years is founded based on the principle of shared-use mobility (Shaheen and Chan, 2015). These alternatives try to reduce the cost of flexible transportation by reducing (or completely eliminating) the capital investments and operating costs.

Informal carpooling is one of the first and common forms of trip sharing in which a number of individuals share a vehicle for their trips, which typically involve the same origin and/or destination. Examples include parents who take turns in taking their children to school, colleagues who carpool to work, etc. In this form of carpooling drivers own the vehicles and have personal interest in the trips, regardless of whether additional passengers are present in the car. The incentives for informal carpooling may include saving time through use of carpool lanes and not having to drive one's own vehicle every single day, if participants take turns in driving. This form of carpooling is usually pre-arranged, and happens among individuals who share commonalities beyond the time and location of their trips.

Transportation Network Companies (TNC), such as Uber, Lyft, and Didi, are among the more recent faces of shared-use mobility alternatives. TNCs use private vehicle owners and their personal vehicles to provide flexible and on-demand transportation. These individuals collaborate with the company as independent contractors, and not employees. This substantially reduces the cost of capital and human resources for TNCs while generating revenue for both the company and the drivers. Essentially, the basic services provided by TNCs act as a lower-cost alternative to taxi services, and hence not only do they not address the problem of increasing travel demand and congestion, but they add to it as a result of the empty trips required to pick up passengers. In terms of sustainability, TNCs impose the same cost to the environment as taxi companies do. Recently, two of the more prominent TNCs, Uber and Lyft, have introduced sharing services Uberpool and Lyft Line, respectively. Such services can certainly reduce the cost and environmental impacts of the base services.

Peer-to-peer (P2P) ridesharing aims at capturing the benefits of TNCs while alleviating their adverse impact on the environment. Ridesharing systems are founded on the principle of sharing economy. Sharing economy, also known as collaborative consumption, is a fairly old concept that focuses on the benefits obtained from sharing resources (products or services) that would otherwise go unused. This economic model has gained more popularity in the recent years, giving birth to many P2P services in different fields (for examples, refer to Böckmann, 2013). The rapid global spread of the internet during the past decade has extended the domains of sharing economy to global populations, and has highlighted its benefits. Moreover, new computer platforms allow easy and quick development of companion mobile applications that facilitate sharing economy.

Similar to TNCs, drivers in a P2P ridesharing system use their personal vehicles to transport passengers, and do not work as agency employees. Contrary to the TNC operations, drivers in a P2P ridesharing system are making trips to perform activities of self interest (as in the case of informal carpooling), i.e., they do not roam around the city only to pick up and drop off passengers. This setting can lead to services that are more environmentally-friendly and cost-efficient compared to the sharing services offered by TNCs. Scoop is an example of P2P ridesharing company which is currently operating in various cities throughout the US.

The overwhelming success and high acceptance rates of TNCs by the public suggest a bright future

Table 1: Transportation alternatives differing in flexibility, cost, and environmental impacts

System components		Transit	Para-Transit (disabled)	Taxi	Shared Taxi/Van	Informal Carpooling	TNCs	P2P Ridesharing
Drivers	Employees	✓	✓	✓	✓			
	Peer to riders					✓	✓	✓
Arrangement	Pre-arranged	✓	✓	✓	✓	✓		✓
	On-demand		✓	✓	✓		✓	✓
Multiple passengers		✓	✓		✓	✓	✓	✓
Financially self-sufficient			✓	✓	✓	✓	✓	?

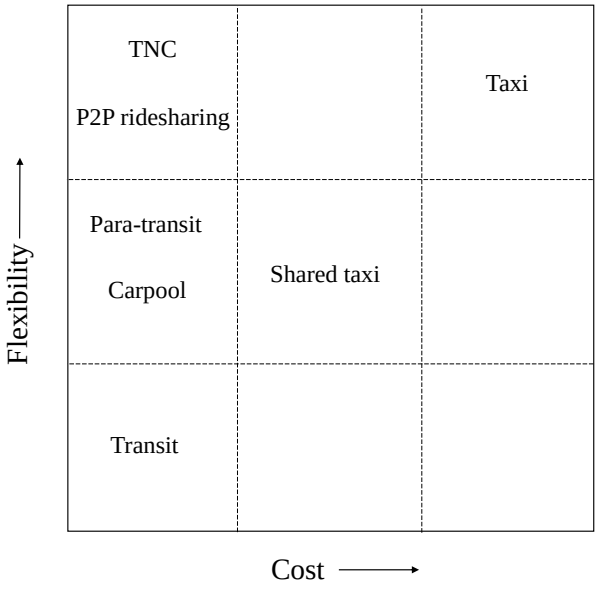


Figure 1: Comparison of shared-use mobility services in terms of cost and flexibility

for the more environmentally-friendly ridesharing systems. Since founded in 2009, Uber has managed to expand its operations in 75 countries and more than 450 cities worldwide. According to BUSINESS INSIDER, in 2014 Uber had more than 160,000 registered driver-partners in the US, and the number of new drivers subscribing to Uber each month increased exponentially, approaching 40,000 (Business Insider, 2015). The company reportedly earned $1.5 billion in revenue in 2015 (Business Insider, 2015).

Table 1 and Figure 1 summarize the characteristics of the transportation alternatives discussed in this section. They suggest that ridesharing systems have the potential to outperform other alternatives in terms of cost, flexibility, and impact on congestion and the environment, if they can manage to operate as financially independent systems. This paper introduces the characteristics of a flexible P2P ridesharing system, and proposes a methodology to match riders and drivers in such a system.

2 Literature Review

The ride-matching problem can be viewed as the engine of a ridesharing system. This problem can be considered as a generalization of the classic Dial-a-Ride problem (DARP) in the transportation literature.

Dial-a-ride problem optimizes the pick-up and delivery of passengers in special settings that typically involve door-to-door transportation and is commonly used in para-transit systems or shuttle-like services. In such systems, passengers in need of rides contact service providers regarding their transportation needs, and the service providers assign vehicles to their requests. In the basic form of DARP, vehicles and riders are assumed to be homogeneous. All vehicles start from and return to the same depot (Savelsbergh and Sol, 1995; Cordeau and Laporte, 2007; Cordeau and Laporte, 2007). Over the years, researchers have studied more practical forms of DARP, by assuming time windows for requests (Jaw et al., 1986; Psaraftis, 1983), multiple depots (Cordeau and Laporte, 2007), heterogeneity among vehicles and passengers (Carnes et al., 2013; Braekers et al., 2014), and the possibility of transfers for passengers, i.e., multi-hop property (Masson et al., 2014; Stein, 1978; Liaw et al., 1996).

The ride-matching problem is very similar to the more advanced versions of DARP in definition and therefore in mathematical modeling. The main characteristic of a P2P ride-matching problem that differentiates it from DARP is the fact that drivers in a ridesharing system are also considered as customers. Additionally, the set of vehicles (and their drivers) is not deterministic, i.e., their availability is not generally known in advance. Once they register their trips, drivers are available only in narrow sprites in space and time. These characteristics play an important role when it comes to developing efficient algorithms to find optimal matchings. Since riders and drivers have very limited spatiotemporal proximity compared to DARP where drivers are much more flexible, the standard formulation of DARP is not the most efficient way to model a P2P ridesharing system. In a ridesharing system, it is possible to take advantage of the narrow time windows within which riders and drivers are available to perform value reduction on the domains of decision variables, rather than adding time window constraints to the optimization problem, hence reducing the complexity of the problem (Masoud and Jayakrishnan, 2017).

If we were to base the mathematical modeling of a P2P ridesharing problem on the formulation of DARP, we would have to add multiple sets of constraints (and decision variables depending on the properties of the defined ridesharing system) based on the characteristics of drivers' trips (e.g., time windows and pre-determined origins and destinations of trips) and their preferences (e.g., non-smoking passengers or passengers within a certain age group). These additional constraint sets increase the complexity of the corresponding optimization problems. In this paper rather than introducing new constraint sets, we use these additional sets of constraints to reduce the solution space of the problem, and develop an efficient solution algorithm for the multi-hop ride-matching problem.

In practice, ridesharing systems usually match a rider with a driver if they share the same origin and destination. From a computational point of view, this task can be easily accomplished by making a series of comparisons between origins and destinations of riders and drivers in polynomial time. More sophisticated ridesharing systems allow drivers to specify their routes, and assign a driver to a riders whose origin and destination lie along the driver's pre-specified route. Computationally, this is a simple form of matching and is sufficient if the goal of the system is to match each driver with a single rider. Matching in this one-to-one setting can be easily accomplished by modeling the system as a bipartite graph, with riders and drivers forming two mutually exclusive sets of vertices. Although this type of matching can be done in real-time, it does not meet the requirements of a fully flexible system. Schaub et al. (2014) show that even a slight deviation from this simple model can render the matching problem computationally intractable. For example, they show that in a system where a vehicle can carry multiple passengers, relaxing the vehicle's path (as opposed to assuming a pre-specified and fixed path for the vehicle) can make the problem NP-complete, under specific objective functions.

The first feature of a fully flexible ridesharing system is its ability to propose multi-hop itineraries to riders. This feature allows riders to transfer between vehicles, if necessary. In addition, incorporating this feature in the model enables a ridesharing system to provide multi-modal itineraries by extending the concept of vehicles in a ridesharing system to include different modes of transportation.

The general belief among transportation planners that transfers in transport systems are intolerable while

understandable, is not entirely accurate. The current unfavorable perception of transfers is mainly due to evaluating transfers in the context of transit systems. Since transit operations are typically sparse in time and space, making transfers from one transit line to another typically involves long wait times, as well as (sometimes considerable) deviations from one's shortest path. The idea of having passengers transfer between vehicles/modes of transportation, however, can considerably improve connectivity in transportation networks-the principle that motivated incorporating them in transit systems in the first place.

It is possible to take advantage of the benefits transfers have to offer while minimizing their potential inconveniences by properly planning them in time and space. Transfers are in fact an inseparable part of a high-quality multi-modal transport system (Vukan, 1981). Designing multi-hop ridesharing systems can facilitate integrating these systems with public transportation, providing a seamless transport platform where ridesharing systems can serve as feeders to public transit. Masoud and Jayakrishnan (2017) show that considering transfers can cosiderably increase the number of matched riders. Agatz et al. (2009), Herbawi and Weber (2011), and Ghoseiri (2013) are some other works in the literature that incorporate the concept of transfers in their models.

Another desirable feature in a ridesharing system is its ability to optimally route drivers to place them in spatiotemporal proximity of riders. Next to multi-hop routes, this feature plays an important role in increasing the number of served riders (Masoud and Jayakrishnan, 2017). Multiple works in the literature incorporate this feature in their models of ridesharing systems (Masoud and Jayakrishnan 2017; Agatz et al. 2009, 2011; Teodorovic and DellOrco 2005; Baldacci et al. 2004; Di Febbraro et al. 2013; Herbawi and Weber 2012.)

A flexible ridesharing system should be equipped with a ride-matching problem that is capable to providing optimal (or high-quality, near-optimal) solutions in a short period of time. There are several studies in the literature that present a mathematical model of the ride-matching problem, but do not discuss a solution methodology or use the general branch-and-bound method devised for mixed integer-linear programs in their case studies (Agatz et al. 2009; Di Febbraro et al. 2013; Cangialosi et al. 2016). Others propose heuristic solutions without proving/showcasing the accuracy of their proposed algorithms. Masoud and Jayakrishnan (2017) present a solution methodology to solve the matching problem more efficiently (i.e., using less computational resources and faster, compared to direct us of commercial optimization software), and to optimality. The final solution times, however, are not appropriate for real-time implementations. Similarly, Regue et al. (2016) propose an aggregation/disaggregation algorithm that is capable of providing high quality solutions for recurrent commuter trips with a set of shared vehicles, under the assumption that trips are registered in advance. Their solution methodology, however, is not scalable to large systems.

Finally, a flexible ridesharing system should be equipped with a ride-matching algorithm that can find an optimal match for a set of riders and drivers simultaneously. Ridesharing systems that ignore this aspect of matching typically serve riders in a first-come, first-served (FCFS) order; that is, once a rider is matched, the routes of the drivers involved in the rider's itinerary are fixed. These drivers can still be considered in the ride-matching problems solved for future riders, but with (partially) fixed routes. Fixing drivers' routes comes at the opportunity cost of fewer options for serving the riders that join the system in the future. This is a missing component from most of the existing studies in the literature.

Table 2 lists the features of a flexible ridesharing system as introduced in this paper, and indicates the extent to which current studies in the literature cover these features. This table indicates that there are a few studies that include all aspects of a fully-flexible ridesharing system; however, the solution times reported in these studies are too high for large-scale real-time implementations. In this paper, we propose a methodology that can find the optimal solution to a semi-flexible ride-matching problem in which riders are served on a FCFS basis and drivers are routed optimally by the system, in a matter of seconds. Next, we introduce a mechanism that can help shift the solution of this semi-flexible system towards that of a fully-flexible one, where the FCFS order of serving passengers is disturbed in order to improve system throughput.

Table 2: Literature on P2P ridesharing

Authors	Flexible Paths	Multi-Hop	Multiple Riders	Formulation/ Solution Algorithm	Optimal Solution
Wolfler Calvo et al. (2004)	✓		✓	Greedy heuristic	
Baldacci et al. (2004)	✓		✓	Optimization	✓
Teodorovic and DellOrco (2005)	✓		✓	Bee colony optimization (meta-heuristic)	
Agatz et al. (2009)	✓	✓		Optimization	✓
Agatz et al. (2011)	✓			Optimization	✓
Herbawi and Weber (2011)		✓		Genetic/Evolutionary algorithms	
Herbawi and Weber (2012)	✓		✓	Exact formulation + Heuristic solution	
Di Febbraro et al. (2013)	✓		✓	Optimization	✓
Ghoseiri (2013)		✓	✓	Exact formulation + Heuristic solution	✓
Schaub et al. (2014)			✓	Bipartite matching	✓
Regue et al. (2016)	✓	✓	✓	Aggregation/disaggregation algorithm	
Cangialosi et al. (2016)	✓	✓	✓	Optimization	✓
Masoud and Jayakrishnan (2017)	✓	✓	✓	Decomposition algorithm	✓
Current study	✓	✓	?	Dynamic Programming algorithm	✓

3 Ridesharing System Architecture

Envision an on-demand, real-time ridesharing system where individuals request transportation services when they need them. Let P denote the set of participants in such a system. We divide the set of participants into a set of riders, denoted by R, and a set of drivers, denoted by D. A rider $r \in R$ joins the system hoping to find a set of drivers who can collectively serve his/her ride request (multi-hop property). A driver, on the other hand, is planning to travel to perform an activity of self-interest, and is willing to share the unused capacity of his/her vehicle with those who are in need of rides. We start the modeling process assuming that sets R and D are mutually exclusive, although we study the impact of dropping this assumption in later sections.

Let us identify a set of stations in the network, denoted by S, where participants can start and end their trips, and riders can transfer between drivers/transportation modes. Stations should be located at areas with high demand levels for ridesharing, such as high density residential/recreational areas, business districts, and central transit stations. Furthermore, let us discretize the study time horizon (e.g., a morning/evening peak hour) into a set of short time intervals of length δt, and index the i^{th} time interval by t_i. In this study, the length of each time interval is set to one minute. Discretizing the study time horizon into short time intervals allows for using time-dependent travel time matrices, and alongside discretizing the space dimension through introducing stations provides the basis for the proposed ride-matching algorithm in this paper.

In order to participate in the ridesharing system, riders and drivers need to register their trips. During the registration process, a participant $p \in P$ is required to provide information on their origin and destination stations, denoted by O_p and D_p, respectively, their max ride time (i.e., the max time they are willing to spend on the trip), denoted by T_p^{TB}, and a travel time window for their trip, denoted by $[T_p^{ED}, T_p^{LA}]$, where T_p^{ED} is the earliest time interval participant p is willing to start their trip, and T_P^{LA} is the latest time interval by which they need to have arrived at their destination. In addition to max ride time and travel time window, in an attempt to provide a pleasant ridesharing experience for the system users, riders are asked to inform the system on the maximum number of transfers they are willing to make (denoted by V_r, $\forall r \in R$), and drivers are asked to set a limit on the number of passengers they are willing to have on board at any moment in time (denoted by C_d, $\forall d \in D$). In addition, all participants can indicate their individual preferences, such as age, gender, etc., on people with whom they travel.

Although drivers use their personal vehicles, they are encouraged to leave the routing of their vehicles

to the system. The system attempts to find itineraries for riders using the empty seats in the drivers' vehicles (and possibly using the available transit system), taking into consideration the travel time windows and max ride time constraints of riders and rivers, max number of transfers specified by riders, vehicle capacities, and the participants' individual preferences.

An on-demand ridesharing system has to serve riders on a FCFS basis, i.e., the system should search for an itinerary for a rider as soon as the rider puts in a trip request. Therefore, the appropriate matching algorithm for such a system is a many-to-one algorithm, where a single rider can be routed through transferring between multiple vehicles. If a match is found for a rider, the itineraries of the involved drivers will be (partially) fixed and announced to them. These drivers can still contribute to the itineraries of future riders by being routed through the portions of their paths that do not concern the current rider. Furthermore, these drivers can pick up more passengers in the fixed portions of their paths, if their capacity constraints allow. The objective of the system is to find the best itinerary for a rider given the spatiotemporal and quality of service constraints on both riders and drivers.

4 The Ellipsoid Spatiotemporal Accessibility Method (ESTAM)

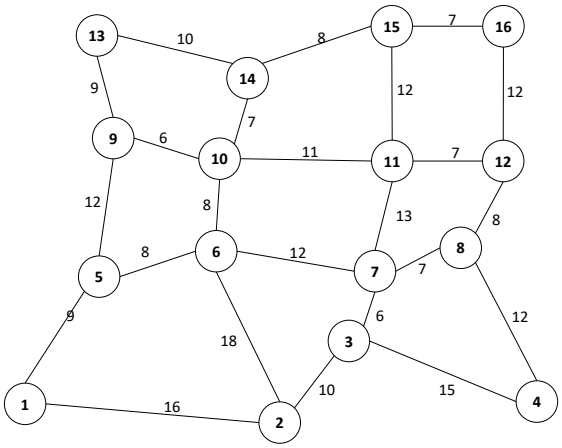
In this section we introduce what we call the "Ellipsoid Spatiotemporal Accessibility Method (ESTAM)". ESTAM generates for each rider a "time-expanded feasible network" in which the rider's optimal itinerary could be searched. This network is constructed from the links on the original network that are reachable by drivers. ESTAM first uses a simple geometric tool to identify the set of stations spatially reachable by each participant (driver or rider). Next, ESTAM finds the time intervals during which each station can be reached, forming the set of feasible links for each participant. Finally, the rider's time-expanded feasible network is generated using the set of feasible links reachable by drivers who are within spatiotemporal proximity of the rider. The three steps of ESTAM are discussed in more detail in the following.

Throughout this paper, we use the following problem instance to illustrate the concepts and demonstrate different steps of the proposed methodology. This example contains a network with 16 stations, one rider, and four drivers. The network and the characteristics of the participants' trips are displayed in Figure 2. Each participant starts and ends his/her trip from one of the stations. For simplicity, It is assumed that participants have no restrictions on the people with whom they travel.

4.1 Approximating the Set of Reachable Stations

Spatial proximity of participants can be determined based on their origin and destination stations and max ride times. Masoud and Jayakrishnan (2017) show that the region in the network accessible to participant $p \in P$ is restricted to inside and on the circumference of an ellipse. The focal points of this ellipse are the participant's origin and destination stations, and its transverse diameter is an upper-bound on the distance the participant can travel given their max ride time. Such an ellipse is displayed in Figure 3.

For an ellipse with focal points f_1 and f_2, the total straight traveling distance between any point M on the circumference of the ellipse and the foci (i.e., $d(f_1,M)+d(M,f_2)$, where the function d measures the Euclidean distance between its two arguments) is equal to the transverse diameter of the ellipse. The transverse diameter corresponds to the max ride time of the participant, assuming an underestimate of the traveling speed in the region. Consequently, any station located outside of the ellipse cannot be reached without violating the max ride time constraint of the participant. Still, note that not all stations located inside a participant's ellipse are necessarily reachable by them due to the fact that actual travel times on the network are typically higher than straight-line distance travel times on the map. Masoud and Jayakrishnan (2017) show that this procedure does not cut off the optimal solution, but reduces the feasible region based on the spatiotemporal characteristics of the trip.



Participant	O_p	D_p	T_p^{ED}	T_p^{LA}	T_p^{TB}
r_1	5	12	2	39	37
d_1	7	12	23	40	17
d_2	5	14	3	28	25
d_3	6	7	11	25	14
d_4	1	2	10	29	19

Figure 2: Problem instance used throughout the paper to illustrate the concepts

This method can substantially reduce the number of stations a participant can visit, simply by evaluating the coordinates of a set of candidate (and not all) stations in the equation of an ellipse. Storing stations in an appropriate data structure based on their geographical location makes it easy to find the set of candidate stations . We call the part of the network confined within and on the circumference of participant p's ellipse the "reduced network" the participant, and denote it by G_p.

Figure 4 displays the reduced networks of the participants in our example. Rider 1 is planning to travel from station 5 to station 12. This rider's reduced network, G_r, is highlighted in Figure 4 as the section of the network confined within the blue ellipse. The reduced networks for the four drivers in our example are also demonstrated in this figure, using green ellipses. The shaded area shows the intersection of the rider's reduced network with those of the drivers. We refer to the set of stations within the shaded area as the "feasible network" for rider r, denoted by G_r^F. These stations are highlighted in Figure 4. The feasible network is defined for a rider, and contains only the section of the network that the rider can reach due to the space proximity provided by drivers in set D.

Defining reduced networks for participants serves three purposes. First, by studying the reduced networks of all participants, it is possible to make an early diagnosis whether a rider can be served. For a rider to be served, he/she needs to have both time and space proximity with at least one driver along a path that connects the rider's origin and destination stations. Therefore, the minimum requirement for a rider to be served is to have space-proximity with at least one driver at both his/her origin and destination stations. In Figure 4, for example, the rider has spatial proximity with driver 2 (traveling from station 5 to station 14) at his/her origin station and with driver 1 (traveling from station 7 to 12) at his/her destination station. In the absence of any of these two drivers, the rider could not be served.

In addition, studying the reduced networks makes it possible to shrink the driver set for each rider. For a given rider, a driver who does not contribute to the rider's feasible network (i.e., the driver's ellipse does not overlap with the rider's ellipse) can be filtered out from the ride-matching problem. As an example, driver 4 in Figure 4 can be filtered out from the problem, reducing set $D = \{1,2,3,4\}$ to $D = \{1,2,3\}$. Driver 3 does not have proximity to the rider's origin or destination, but can contribute to portions of the trip in case

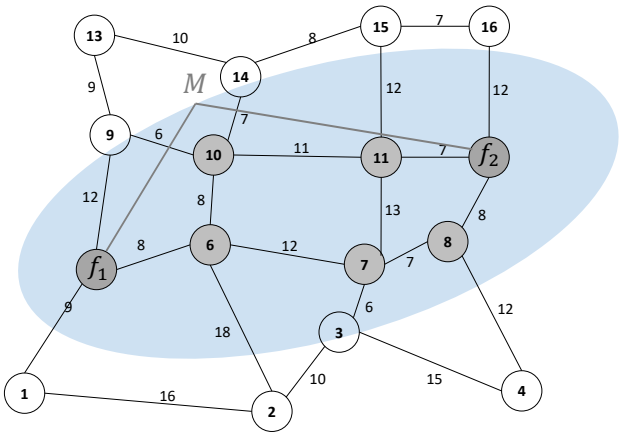


Figure 3: The reduced network G_r on the original network G. Stations inside the reduced network are highlighted.

of a transfer. In fact, in this particular example, the only way the rider can travel from station 5 to station 12 is by being picked up by driver 2, transferring to driver 3, and finally transferring to driver 1. The exact transfer locations and whether this route is feasible at all under the participants' temporal constraints will be examined later. For instance, it might be the case that by the time the rider arrives at station 6 or 10, driver 2 needs to have traveled beyond those stations to get to his/her destination on time (thereby, a spatiotemporally feasible path for the rider would not exist).

The third benefit of forming reduced networks is that the network-dependent computations can be done much faster on G_p than G. Different types of geometric containers have been proposed in the literature for more efficient computation of shortest paths in large scale networks (Wagner and Willhalm, 2003; Wagner et al., 2005). One example of the computational savings obtained by reducing G to G_p manifests itself in the second step of ESTAM, when time-proximity of stations is examined. To determine time-proximity, we need to search for multiple shortest travel time paths between the origin and destination stations of participants. Shortest travel time matrices can be computed more efficiently on G_p. To demonstrate the computational savings that can be obtained by replacing the original network with a set of reduced networks, we find the shortest travel time paths between the origin and destination stations of participants for problem instances of different sizes (i.e., various number of participants, and various network sizes), using Dijkstra's algorithm, both on G and G_p. Results are displayed in Figure 5. Dijkstra's algorithm is used here because it is the shortest path algorithm we have used for the experiments in this paper; however, similar computational savings are expected for any other shortest path algorithm. Figure 5 showcases the computational savings that can be obtained by replacing the original network with the reduced networks.

4.2 Constructing Time-Expanded Reduced Networks

In the previous step, we constructed the participants' reduced networks, and used them to form the feasible reduced network for the rider. However, not all stations in the rider's feasible reduced network are necessarily reachable by the rider, since we have yet to study the time proximity of stations. To study time-proximity, we have to find the set of time intervals during which station $i \in G_p$ can be reached by participant p.

To study time-proximity of participant p at each station in G_p, we identify all paths in G_p with travel times less than or equal to T_p^{TB}. To avoid enumerating the paths, we find the first k-shortest paths in G_p such that the travel time on the k^{th} shortest path is less than or equal to T_p^{TB}, and the travel time on the

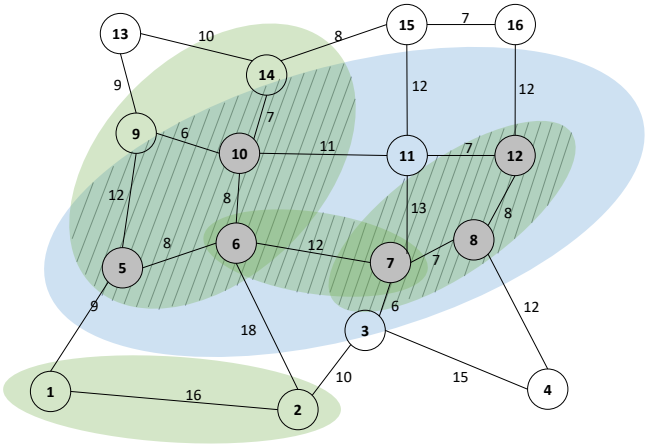


Figure 4: Reduced networks of all the participant in the example problem instance. The shaded area is the rider's feasible network, G_r^F. The stations inside G_r^F are highlighted.

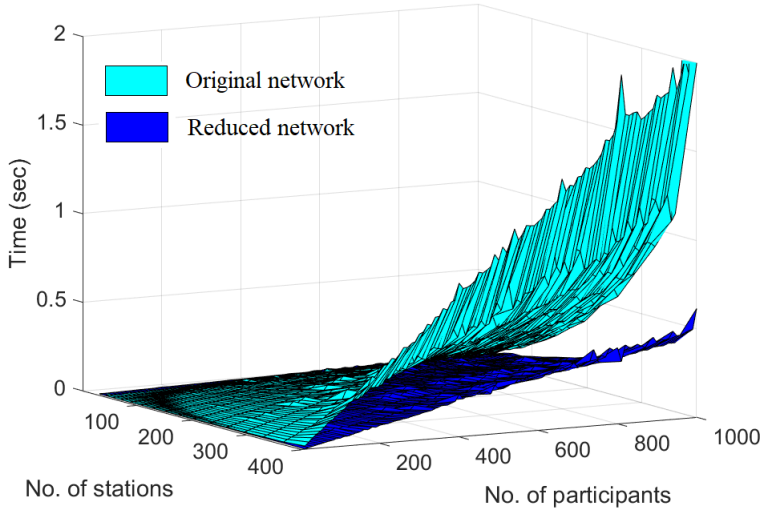


Figure 5: The surface on the top/bottom shows Dijkstra's running time on G/G_p. Results are averaged over 10 randomly generated networks for each problem

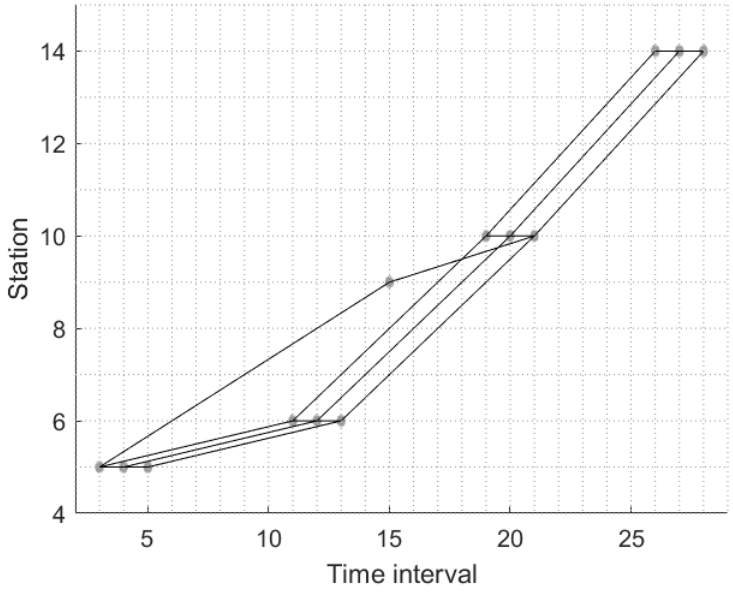


Figure 6: Time-expanded reduced network for driver 2 in our example

$(k+1)^{th}$ shortest path is higher than T_p^{TB}. Recall that computing the shortest paths on reduced networks is far more computationally efficient than the original network. After finding the j^{th} shortest path $(j \leq k)$ on G_p, we follow in order the stations on each path and use time-dependent travel time matrices to find the time intervals during which each station $s \in G_p$ can be visited, denoted by T_s^p. We can use this information to construct the "time-expanded reduced network", $G_p(t)$, for each participant $p \in P$. Each node in $G_p(t)$ takes the form of a tuple (t_i, s_i), representing a visit to station s_i during time interval t_i. Consequently, a link in the form of (t_i, s_i, t_j, s_j) represents a trip that starts at station s_i during time interval t_i and ends at station s_j at time interval t_j. Links in the form of (t_i, s_i, t_i+1, s_i) represent "waiting" links, where an individual is waiting at station s_i for one time interval, starting at time interval t_i. Figure 6 displays the time-expanded reduced network for driver 2 in our example. The time-expanded reduced networks for the rest of the participants in our example are presented in Appendix A.

Note that it is possible to have more than one origin and/or destination node on a time-expanded reduced network. Nodes in the form of $(t, O_p), \forall t \in T_{O_p}^p$ are all origin nodes, and nodes in form of $(t, D_p), \forall t \in T_{D_p}^p$ are all destination nodes. We keep these nodes in origin and destination sets, denoted as $OS(p)$ and $DS(p)$, respectively. In Figure 6, $OS(d_2) = \{(3,5),(4,5),(5,5)\}$, and $DS(d_2) = \{(26,14),(27,14),(28,14)\}$.

4.3 Constructing a Rider's Time-Expanded Feasible Network

In section 4.1, we generated the feasible network for rider r, G_r^F, by scanning through the stations in the rider's reduced network, G_r, and checking whether these stations exist in any of the drivers' reduced networks. We can repeat a similar procedure on the links in the time-expanded reduced network of the rider, $G_r(t)$. The result is the rider's "time-expanded feasible network", $G_r^F(t)$. More specifically, in this step we scan through the drivers' time-expanded feasible networks, looking for links that they have in common with the rider. For each link in the riders' time-expanded reduced network, the drivers that also have that link in their networks can potentially carry the rider on that link. Similarly, if a link in the rider's time-expanded reduced network does not appear in any of the drivers' networks, that link is in fact not feasible for the rider, since there are no drivers to transport the rider on that link. In other words, links in set $G_r^F(t)$ are links that

can be traversed by the rider, and at least one driver. At this stage, we can filter out the driver set D even further by eliminating drivers who do not contribute to the rider's time-expanded feasible network. Note that in the example used in this paper, no additional drivers are filtered out at this stage, since all remaining drivers contribute to the rider's time-expanded feasible network (see Figure 7(b)).

5 Searching for the Optimal Itineraries

The ESTAM algorithm constructs the time-expanded feasible network for the rider in our problem, on which the rider's optimal itinerary can be searched. Even though this network is substantially smaller than the original network, the number of feasible routes for the rider can still be very large. Different combinations of links on which to travel and drivers with whom to ride can make enumerating the paths computationally prohibitive. In this section, we introduce a dynamic programming algorithm to search for a rider's optimal path on their time-expanded feasible network.

5.1 Preparing the Graph

Figure 7(a) shows the time-expanded feasible network for the rider in our example. Let us define rider r's origin/destination node as the node with the smallest/largest time index in the rider's origin/destination set. Our goal is to find an optimal itinerary to take the rider from his/her origin node to his/her destination node. In our example, the origin node is $(2,5)$, and the destination node is $(39,12)$. We define the optimal itinerary as one that minimizes a linear combination of in-vehicle travel time, wait time, and number of transfers.

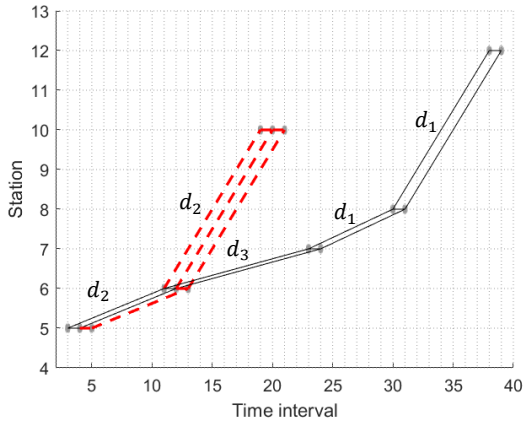
Before searching for the optimal itinerary, two preliminary operations need to be performed on the graph. First, nodes that are not descendants of the origin node or predecessors of the destination node should be removed from the graph, since they cannot be part of a path that connects the origin node to the destination node. An example of this operation is demonstrated in Figure 7(a). This figure displays the time-expanded feasible network for the rider in our example (who is traveling from station 5 to station 12). The sub-graph $\{(5,5),(12,6),(19,10),(20,10),(21,10)\}$ (marked by red in Figure 7(a)) has to be eliminated from the graph, since the nodes in this subgraph are not predecessors of the destination node $(39,12)$. This task can be easily accomplished by scanning the adjacency matrix of the graph.

The second operation is to topologically sort the graph. A topological ordering for such a graph always exists, since the graph is directed and acyclic. (The direction of the links can be determined based on the time dimension. The graph is acyclic since nodes have a time component, and the link generation procedure does not generate links that represent traveling back in time.) Figure 7(b) displays the resultant graph after performing these two preliminary operations. The numbers next to nodes in this figure show their topological orders. Links on this graph are annotated by drivers who can carry the rider on them.

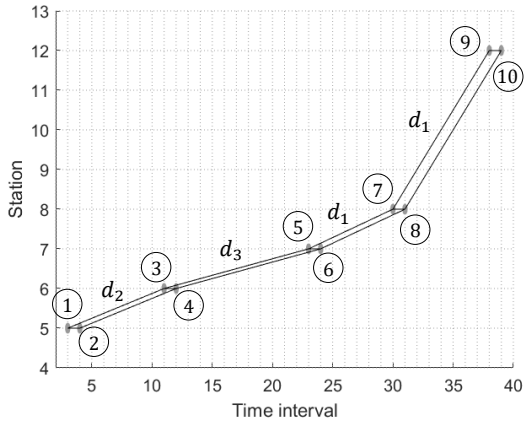
Conducting depth first search (DFS) on the "revised time-expanded feasible network" in Figure 7(b) could determine whether an itinerary for the rider exists at all. Starting from the first node in the topologically ordered graph, if DFS does not conquer a node that contains the rider's destination station, then the rider cannot be served in the system. Otherwise, if DFS finds a path (establishing that a matching is spatially and temporally feasible), we will use the proposed dynamic programming (DP) algorithm in the following section to find the best itinerary for the rider.

5.2 The Dynamic Programing (DP) Algorithm

Let us start by defining sets D_j^{IN} and D_j^{OUT} as the set of drivers who enter and exit node j, respectively, and set DN_j to include tuples (i,d) such that there is a link for driver d from node i to node j in the revised time-expanded feasible network. Furthermore, we denote by $V(j,d)$ the minimum cost of the optimal path



(a) Time-expanded feasible network. The dashed lines marked in red will not be part of the optimal solution and can be deleted.



(b) Revised time-expanded feasible network (after performing the preliminary operations). Numbers next to nodes show their order in the topologically-ordered graph. Links are annotated with drivers who can carry the rider on them.

Figure 7: Time-expanded feasible network for the rider in our example before and after revision

from the start node (node 1 in the topologically ordered graph) to node j, such that the last link on the path is traversed by driver d. The goal is to find $V_r^* = \min_{j \in DS(r), d \in D_j^{IN}} \{V(j,d)\}$.

We initialize the $N^r \times D^r$ matrix $V(\cdot)$ to infinity, where N^r is the number of nodes in rider r's revised time-expanded network, and D^r is the number of drivers that contribute to rider r's revised time-expanded feasible network. To keep track of the optimal solution, we introduce two additional matrices of the same size, n_{pred} and d_{pred}, and initialize them to zero. These matrices keep the predecessor node and the driver that takes the rider to the predecessor node on the optimal path, respectively.

We compute the cost of a path as a linear combination of travel time, wait time, and penalty for transfer. Let us define the function $C(i,j)$ to return the cost of traveling on link $(i,j) = (t_i, s_i, t_j, s_j)$. This cost can be calculated using equation (1). The cost of travel as defined in equation (1) is the travel time on the link in case a trip takes place (i.e., $s_i \neq s_j$), or a penalty W if the rider waits at a station $(i.e., s_i = s_j)$ for one time interval. Note that if the rider stays in the same station for multiple time intervals, the algorithm will automatically accumulate W the appropriate number of times. This is due to the fact that the same station at different time intervals corresponds to separate nodes on the time-expanded feasible network. Furthermore, note that $C(i,j)$ in equation (1) can be easily extended to $C(i,j,d)$, in case a more customized driver-dependent cost is desired. Additionally, we denote by C_T the penalty (cost) of a transfer, and use the expression $C_T\,(1_{d \neq d'})$ to capture the total penalty of transfers, where the indicator function $(1_{d \neq d'})$ returns the value 1 if $d \neq d'$ (i.e., if the rider makes a transfer), and the value 0 otherwise.

$$C(i,j) = \begin{cases} (t_j - t_i + 1)dt & s_i \neq s_j \\ W & s_i = s_j \end{cases} \tag{1}$$

We set the initial condition as $V(1,d) = 0$, for all $d \in D_1^{OUT}$. In the revised time-expanded feasible network in Figure 7(b), the initial condition is $V(1,d_1) = 0$. After setting the initial condition, we traverse the nodes in the (topologically ordered) graph in ascending order. For each node j, and each driver d in the set D_j^{IN}, $V(j,d)$ is computed as in equation (2). The set $ED(i,d')$ in equation (2) includes all drivers on the optimal path to node i, excluding the driver on the last link (d'). We will discuss the necessity of introducing this set further down this section.

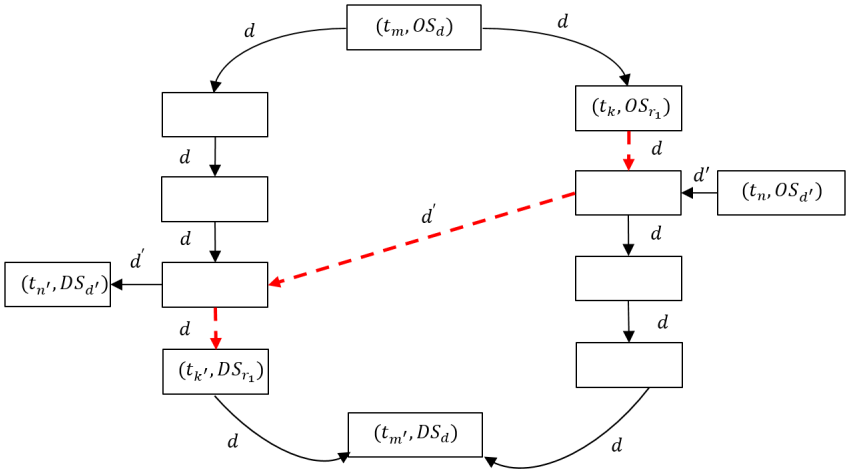


Figure 8: Example of an infeasible solution

$$V(j,d) = \min_{i:(i,d)\in DN_j} \Big\{ \min_{d'\in D_i^{IN}\setminus ED(i,d')} \{V(i,d') + C_T 1_{\{d'\neq d\}}\} + C(i,j) \Big\} \tag{2}$$

After computing $V(j,d)$ using the recursive function in equation (2), we use sets $n_{pred}(j,d)$ and $d_{pred}(j,d)$ to keep track of the predecessor node and driver in the optimal solution, respectively:

$$n_{pred}(j,d) = \text{argmin}_{i:(i,d)\in DN_j} \Big\{ \min_{d'\in D_i^{IN}\setminus ED(i,d')} \{V(i,d') + C_T 1_{\{d\neq d'\}}\} + C(i,j) \Big\}$$

$$d_{pred}(j,d) = \text{argmin}_{d'\in D_i^{IN}\setminus ED(i,d')} \{V(i,d') + C_T 1_{\{d\neq d'\}}\}$$

Once the min-cost itinerary for the rider is found, these matrices can be used to retrieve the optimal itinerary (i,e., the order of following the nodes, and the drivers to carry the rider between the nodes.)

Recall that set $ED(i,d')$ includes all drivers on the optimal path to node i, excluding the driver on the last link (d'). Drivers in this set are excluded from set D_i^{IN} in equation (2) to ensure the feasibility of the solution. Infeasibility can arise in a scenario where a driver covers multiple links on the rider's itinerary, but these links belong to different feasible but conflicting paths of the driver (i.e., paths that cannot co-exist). In such a scenario, the resulting itinerary would not be feasible, since it would require a driver to be at two different locations at the same time. To clarify, an example is shown in Figure 8. In this example, driver d has two potential paths to make his/her trip, while driver d' has a single path. There is only one itinerary for the rider, marked by dashed red arrows. This itinerary consists of three links. The first link is covered by driver d via this driver's first potential path, the second link is covered by d', and the third link is covered by d via this driver's second potential path. Clearly the resulting itinerary is not feasible, since d cannot carry the rider on both links. Excluding drivers in set ED from the list of potential drivers in equation (2) ensures that such an itinerary is never generated.

It is easy to incorporate individuals' personal preferences into this framework. When forming the list of drivers for a rider, drivers who do not meet the list of requirements by the rider can be easily excluded from this list. Similarly, a driver can ask to be considered as a potential match only for riders with specific characteristics. Using the proposed DP algorithm, not only the optimal itinerary but all the feasible itineraries for a rider are generated. Therefore, if the solution that is deemed optimal based on the objective function defined by the system needs to be re-considered for any reason (e.g., in case a rider cancels a ride, or a driver

gets stuck in traffic and cannot make it to the pick up point on time), we can easily retrieve the next best solution.

Once a rider's itinerary is fixed, we can use a straight forward procedure to find the optimal itineraries of drivers. If a driver is part of a rider's itinerary, then the portion of the driver's path where he/she is scheduled to carry the rider becomes fixed. The problem of finding the driver's optimal itinerary becomes reduced to finding the shortest travel time paths between the fixed potions. We can use Dijkstra's algorithm on the driver's time-expanded reduced network to efficiently search for the shortest paths from the driver's origin to the origin of his/her first fixed portion, between the fixed portions, and from the destination of the last fixed portion to the driver's ultimate destination. The union of all these shortest paths and the fixed portions constitutes the drivers optimal itinerary.

6 Numerical Experiments

In this section, we generate multiple random instances of the ridesharing problem in a network with 49 stations. All problem instances contain 1000 participants. We vary the ratio of drivers and riders in each problem instance, and assess the performance of the proposed methodology as the ratio of riders in the problem changes.

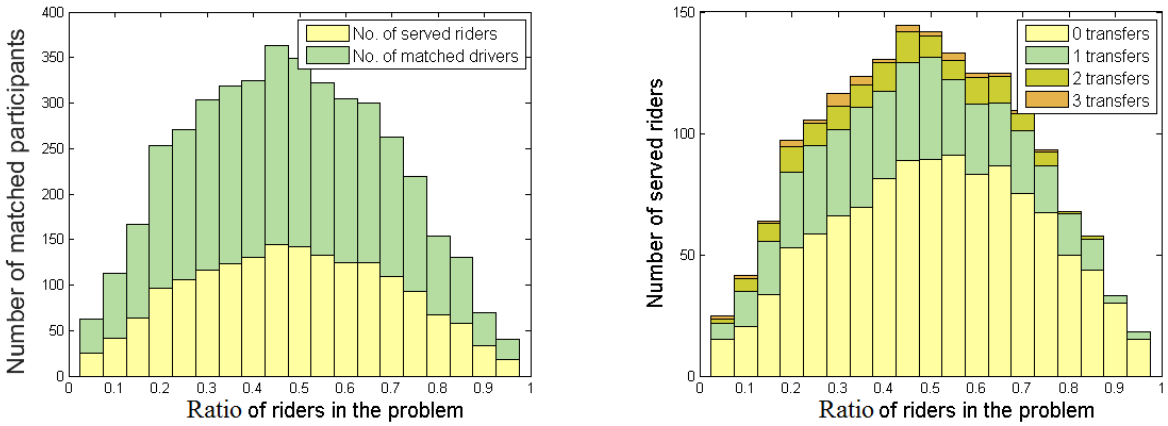
We use one-min time intervals, and consider a randomly generated maximum ride time within the interval $[T_p^{ST}, \kappa T_p^{ST}]$ for participant $p \in P$, where T_p^{ST} is the shortest path travel time between the participant's origin and destination stations, and κ is called the "max ride time factor", and takes a value greater than or equal to 1. The participants' earliest arrival times are generated uniformly within a one hour time period. The latest arrival time for a participant is computed as the sum of their earliest departure time and max ride time. All drivers are assumed to have four empty seats, and riders are assumed to accept up to 3 transfers. In the interest of simplicity, we assume that participants have no personal requirements on the people with whom they travel. In addition to running experiments with the default parameter values reported above, we perform sensitivity analysis over the number of participants in the system and their max ride time factors, in order to gain insight on the performance of the proposed methodology under different scenarios.

In the next two sections, we use the proposed algorithm to find the optimal solutions for the randomly generated problems through simulations. In section 6.1, the origin and destination locations of participants will be selected based on a uniform distribution. This type of selection serves as the default setting in our simulations. In section 6.2, we study a more practical scenario, where trip ends are determined randomly, but based on a clustering of the network.

6.1 Uniform Random Selection of Trip End Locations

For the problem instances generated in this section, the origin and destination stations of participants are selected from 49 stations based on a uniform random distribution. The results we report in this section are averaged over 10 simulation runs for each problem instance. As mentioned before, riders are considered in a FCFS basis in simulations. Once a rider registers in the system, a ride-matching problem is solved for that rider. For each problem instance in this section, the highest solution time among riders, averaged over 10 simulation runs, is less than 0.5 sec.

Figure 9(a) displays the number of matched riders and drivers. This figure shows that the number of served riders is maximized in problem instances where the number of riders and drivers is about the same. This figure suggests that the number of matched participants is maximized in the same range where the number of served riders is maximized. In fact, when riders constitute about 45% of the total number of participants, the number of matched riders and drivers are both maximized. Notice that while it is expected in a one-to-one system (where each driver is matched with exactly one rider) for the number of matched



(a) Distribution of number of matched participants

(b) Distribution of number of transfers

Figure 9: Algorithm performance for the randomly generated problem instances, averaged over 10 runs

riders and drivers to be equal, this is not an obvious conclusion in a multi-hop system where each driver can carry multiple riders and each rider can transfer between multiple drivers. Furthermore, notice that these results are only valid for the case where trip origins and destinations are randomly selected. We show in the next section that in more practical settings where demand is not uniformly distributed (and therefore trip ends have a higher degree of spatial proximity), the number of matched participants can increase drastically.

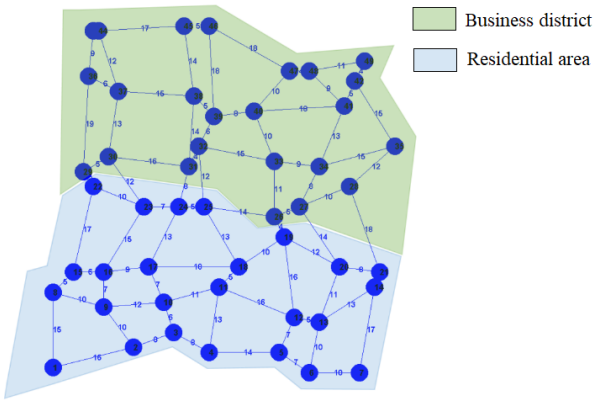
Figure 9(b) shows the distribution of number of rider transfers for each problem instance. This figure suggests that the majority of served riders do not have to make any transfers. This figure also suggests that the number of additional riders that can be served as a result of allowing transfers in the system is not trivial.

6.2 Targeted Random Selection of Trip Ends in Clusters

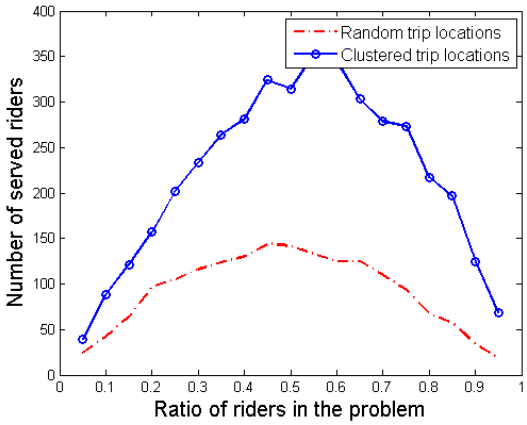
In the previous section, we made the assumption that trip origins and destinations are completely random. In reality, commercial and business districts are usually denser within certain geographical regions, and are distinct from residential areas. In this section, we identify two geographically distinct regions on the network to represent residential and business districts, and generate trips with origins in one region and destinations in the other. Similar to the previous section, we generate different problem instances, varying the percentage of riders in the problems. For each problem instance, we conduct 10 simulation runs, and report the average results. Results are summarized in Figure 10.

Figure 10(a) shows the clustering of a sample network. The area on the top is considered to be the business district, and the area on the bottom is assumed to be the residential area. Each problem instance represents the simulation of a morning peak period, in which participants travel from a randomly selected station in the residential area to one in the business district.

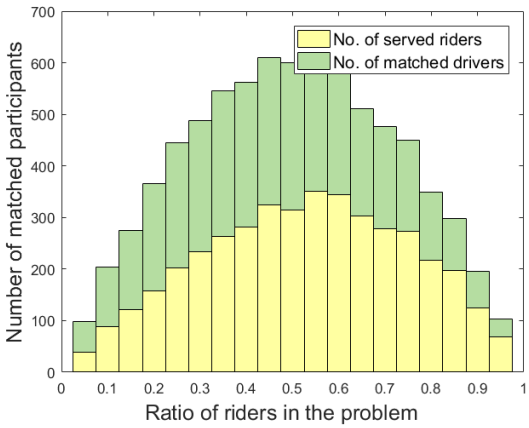
Figure 10(b) shows that the number of served riders under this more realistic scenario follows the same general trend as the one observed in the scenario with randomly generated trip ends; however, in this case the number of served riders has more than doubled. Figure 10(c) suggests that in this more practical setting, when the system has the optimum ratio of drivers and riders, more than 60% of the participants will be matched. Finally, Figure 10(d) shows the distribution of number of transfers for riders. Similar to Figure 9(b), most trips can be completed with 0-1 number of transfers. This consistency between results from simulations with very low (random) and very high (clustered) spatial proximity among trips suggests that a multi-hop system is not likely to produce solutions with large numbers of transfers. This observation is



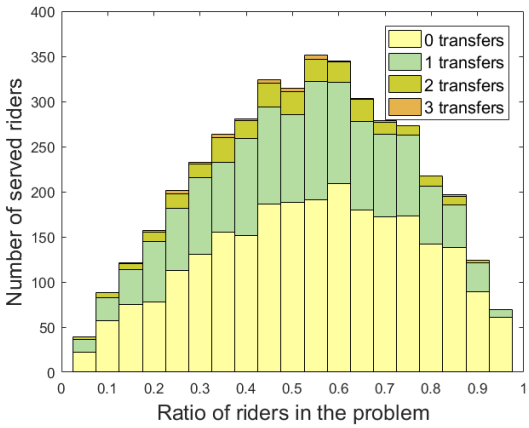
(a) A sample clustering of the network



(b) Comparison of number of served riders between the completely random scenario and the clustering scenario



(c) Distribution of number of matched participants



(d) Distribution of number of transfers

Figure 10: Algorithm performance for the scenario with distinct business and residential areas, averaged over 10 runs.

important due to the perceived discomfort associated with high numbers of transfers, and the consequent concerns over the acceptance of multi-hop systems that follows. Note that even though the number of transfers is not high, they allow for higher coverage of the network by the same set of vehicles and therefore lead to higher matching rates, as depicted in Figures 9(b) and 10(d).

The random demand distributions in section 6.1 and the two-clustered demand distribution in section 6.2 are two extreme cases. In reality, the demand distribution, and hence the performance of the system, falls somewhere between these two extremes, such that there are a multitude of residential areas and business districts, as well as regions that host both types of activities. A real-world study using the proposed methodology was conducted in the Los Angeles County, California. In this case study the proposed methodology was used to connect travelers to transit using P2P ridesharing, serving as a solution to the first/last mile problem (Masoud, Nam, et al., 2017). This study showcases the scalability of the proposed methodology in larger, realistic networks with higher levels of demand.

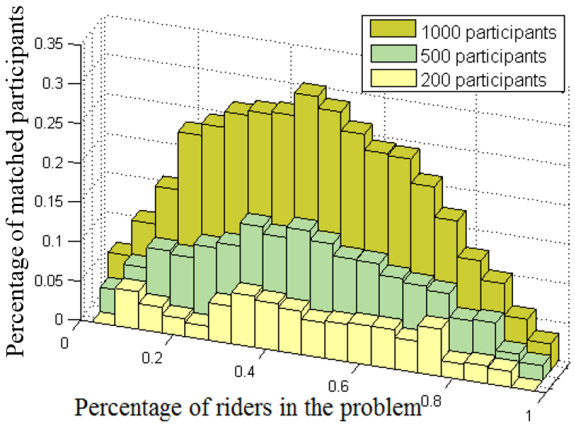


Figure 11: Sensitivity analysis over the number of participants in the system

6.3 The Critical Mass

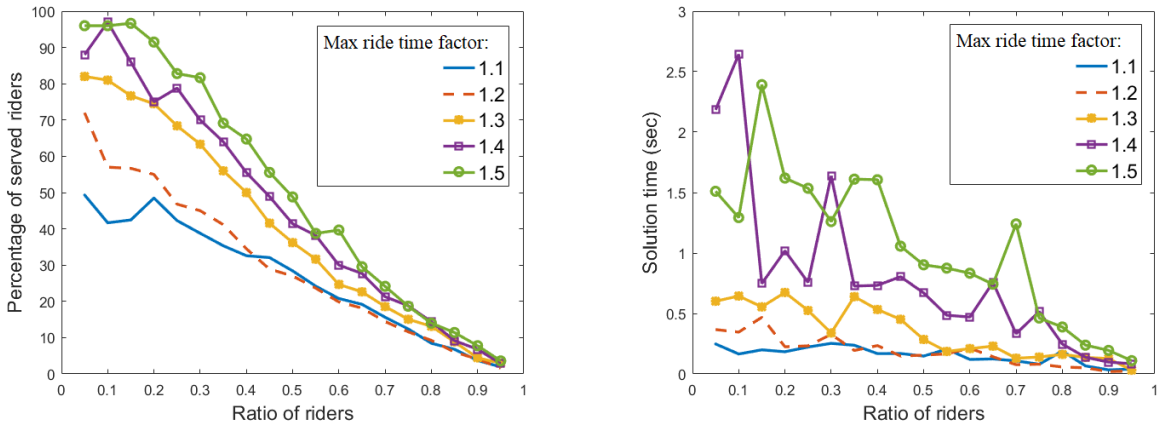
For a ridesharing system to be able to work independently, a critical mass of participants is required. In the previous sections, we demonstrated the performance of a ridesharing system with 1000 participants. We showed that in a more practical scenario with distinct business and residential areas, more than 60%, and in the completely random case about 35% of participants can be successfully matched. It is intuitive to expect the average number of successful matches to increase with the number of participants. However, for a system to survive, a minimum number of participants should be able to use the system successfully. This success will encourage users to continue participating in the system, and generates positive word of mouth.

In this section, we conduct sensitivity analysis over the number of participants in the system. This analysis can shed light on the relationship between system performance and the number and ratio of participants. Figure 11 shows the percentage of matched participants as the total number of participants changes from 200 to 1000. Similar to the previous sections, for a given number of participants, we generate multiple random problem instances by changing the ratio of riders. The trip ends are selected at random with uniform probabilities. Figure 11 suggests that increasing the number of participants leads to higher system performance levels in terms of the matching rate. The peak performance of the system in all cases occurs within the same range, where the ratio of riders in the problem is around 0.4. This analysis can help system operators trying to reach a critical mass make strategic decisions on system expansion and targeted marketing by (*i*) identifying rider to driver ratios that lead to higher matching rates, and (*ii*), establishing a relationship between the matching rate and number of participants under any rider to driver ratio.

6.4 Max ride time factor

In this section, we look into the scalability of the proposed methodology by expanding the size of the time-expanded network on which the search for the optimal solution is conducted. There are multiple parameters that can affect the size of the rider's time-expanded feasible network. For instance, increasing the size of number of drivers, the spatio-temporal proximity among trips, number of stations, and max ride times of participants can increase the size of the network. In this section, we study the impact of increasing κ, the max ride time factor, on solution times and matching rates. Higher κ values correspond to more flexibility in participants' departure times and detours from their shortest paths. Consequently, participants can stay longer, and reach farther in the network, leading to larger link sets.

Figure 12(a) shows the change in the percentage of served riders as κ increases. This figure suggests that more flexibility leads to higher matching rates, as expected. It is interesting to note that the impact



(a) Percentage of served riders

(b) Solution times

Figure 12: Algorithm performance under different max ride time factor (κ) values

of this higher degree of flexibility is more substantial when the ratio riders is low, and diminishes as the ratio of riders increases. When the rider to driver ratio is low, there are many opportunities for riders to be served. With increased flexibility introduced by increasing the participants' max ride times, network coverage increases substantially to the point where at $\kappa = 1.5$, almost all riders can be served. As the rider to driver ratio increases, the impact of higher degree of flexibility diminishes as the number of drivers is decreasing and the number of riders is increasing. Eventually, at very high rider to driver ratios, the higher flexibility does not have a substantial impact on the percentage of served riders since the resources (i.e., the number of drivers) is very limited to begin with.

Figure 12(b) shows the changes in solution time as κ, and hence the size of the rider's time-expanded feasible network, increases. For the same value of κ, this figure shows that the solution time decreases as the ratio of riders increases. The reason for this observation is that the number of participants is fixed, and hence a higher ratio of riders translates into a lower ratio of drivers, and therefore a smaller time-expanded feasible network for every rider. On the other hand, for a given ratio of riders, the size of the time-expanded feasible network for each rider increases with κ, leading to higher solution times. Figure 12(b) shows that even with higher values of κ, the solution times remain acceptable for a dynamic system.

6.5 P2P Ride Exchange

Ridesharing systems are in general spatiotemporally sparse. The temporal sparsity stems from the rather tight time windows of participants, and the spatial sparsity is due to the participants' fixed origin and destination locations. This spatiotemporal sparsity limits the number of satisfied requests. In fact, one of the contributors to the initial failure of P2P ridesharing systems in the US in the 1990's was the very small number of served rider requests.

Today, advancements in the communication technology and prevalence of smartphones has led to a larger pool of riders and drivers interested in ridesharing. However, as discussed in the previous sections, a critical mass of participants is still an important requirement for ridesharing systems to enable their independent and budget-balanced operations. Therefore, a ridesharing system needs to make the best use of its limited supply resources, i.e., the drivers.

A ridesharing system that looks to match riders one at a time is in fact wasting its very limited and valuable resources by fixing the itineraries of matched drivers. Figure 13 shows an example of how fixing

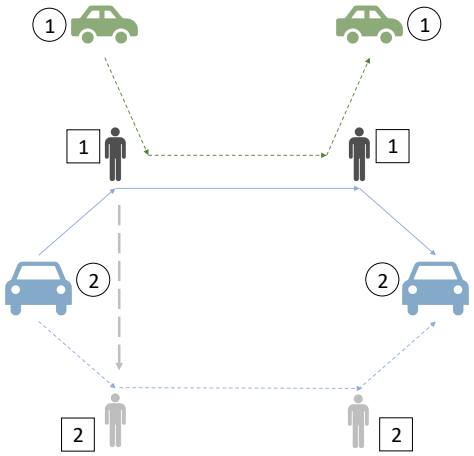


Figure 13: P2P ride exchange

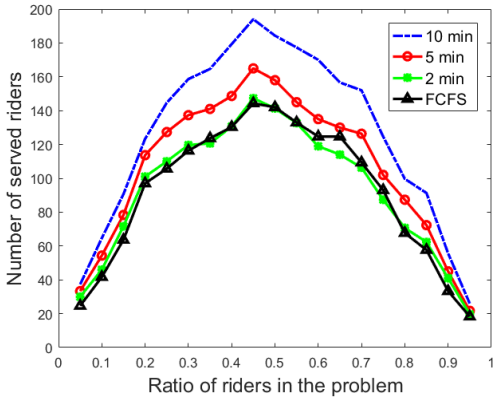
drivers' itineraries can deteriorate the performance of the system. This example includes two riders and two drivers. Assume rider 1 registers in the system first, and hence the system starts by finding a match for rider 1. Vehicles 1 and 2 are both candidates to be matched with this rider. Eventually the system matches rider 1 with driver 2, since driver 1 has to take a larger detour to carry the rider. Driver 2's itinerary, marked in solid blue, is then fixed. Next, rider 2 registers their trip. This rider could have been matched with driver 2 through the route marked with the blue dashed line, if driver 2's itinerary had not been fixed. However, under the current state of the system, rider 2 cannot be matched. The ridesharing system, therefore, can only match one rider with one driver.

In the example in Figure 13, had we had the information on both trips in advance, we could have increased the system performance by solving a many-to-many ride-matching problem that included both riders, as opposed to considering them one at a time. The solution to such a problem would match rider 1 with driver 1, and rider 2 with driver 2, serving both riders, and involving all 4 participants.

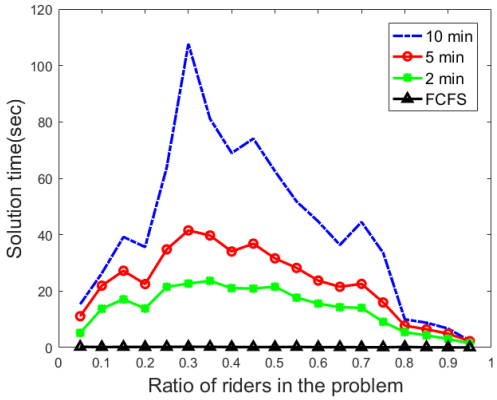
There are two issues that may prevent us from solving a many-to-many matching problem. First, if rider 2 has not joined the system yet at the time when we have to notify rider 1 of the results, then formulating the many-to-many problem would not be a possibility. Second, solving the many-to-many problem could be too time-consuming and not appropriate for real-time implementations.

To demonstrate the difference between a many-to-many ride-matching problem (which includes multiple riders and multiple drivers) and a many-to-one problem (which includes one rider and multiple drivers) in terms of solution time and matching rate, we solve the problem instances in section 6.1 using both approaches. For the many-to-many problem, we use the decomposition algorithm proposed by Masoud and Jayakrishnan (2017) on a rolling time-horizon basis with 2, 5, and 10 min re-optimization periods. In this approach, instead of matching one rider at a time, we solve a problem that includes all the riders who register their trips within the mentioned re-optimization period. Figure14 depicts the results of this comparison.

Figure 14 suggests that the solution time for a many-to-many ride-matching problem is too high for real-time implementations. On the other hand, the number of matched riders improves when a many-to-many problem is solved. In other words, we can use the proposed methodology in this paper to solve a many-to-one ride-matching problem to optimality; however, the modeling of the system as a many-to-one problem itself is not optimal, and is constrained by the unwritten FCFS rule. This modeling approach is considered due to the real-time and dynamic nature of requests, and the lower computational complexity of the many-to-one problem. To address this issue, in this section we introduce what we call "P2P ride exchange" as a mechanism to shift the solution of a many-to-one ride-matching problem towards that of a many-to-many problem, while still serving riders in real-time. P2P ride exchange can help us move from the sub-optimal



(a) Number of served riders



(b) Solution times

Figure 14: Number of served riders and solution times for randomly generated problem instances under different re-optimization periods (i.e., 2 min, 5 min, and 10 min) as well as in a FCFS system where a matching problem is solved as soon as a rider requests a trip.

solution obtained by matching one rider as a time, towards the optimal solution that can be obtained by including all riders in the matching problem, without the unattractive side-effect of the increased solution time.

In this approach, we still solve the matching problem for one rider at a time, and based on the FCFS rule. However, we do not fix the itineraries of matched drivers. If a rider can be served only using a driver that has been previously matched and is unable to take a detour as a result of his/her commitment to prior riders, then the two riders (the one whose trip is currently at stake, and the rider who was previously matched with the driver) can start a negotiation to make a ride exchange. Note that incorporating P2P ride-exchange does not increase the computational burden on the system due to the properties of the proposed algorithm in this paper. It is easy for the system to propose alternative itineraries to riders involved in the exchange, since all the feasible itineraries for riders are generated using the DP algorithm.

In the example in Figure 13, if we solve the ride-matching problem for rider 2 without fixing the itinerary of driver 2, the solution will have driver 2 routed on the dashed path and matched with rider 2. However, driver 2 was previously routed differently, and assigned to rider 1. These two riders can now engage in a negotiation. Since rider 1 can also be matched with driver 1, there is a good chance that rider 1 will accept the proposed ride exchange by rider 2, in exchange for money or credit toward future trips. In case the negotiation is successful, the final matching would be equal to the matching obtained by solving a many-to-many ride-matching problem.

In addition to the benefits that P2P ride exchange can offer in terms of system performance, it can also make riders more engaged with the system. Riders can earn money or credit by selling their itineraries and settling for less efficient itineraries suggested by the system. However, for such a system to work properly, a detailed mechanism needs to be designed to ensure incentive-compatibility, individual rationality, and budget balancedness of the proposed trade. A simple form of such a trade is described in Masoud et al. (2017).

6.6 Overlapping Sets of Drivers and Riders

Initially, we made the assumption that riders and drivers form two mutually exclusive sets. In this section, we study the scenario in which participants who register in the system as riders are doing so because they prefer traveling as riders, and not because they do not own or have access to a personal vehicle. If the system

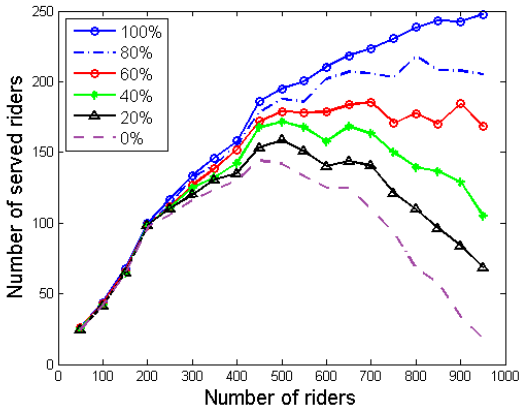


Figure 15: Number of served riders under different percentage of riders (who can switch and become drivers)

is not able to serve these participants as riders, then they will drive their own vehicles and join the system as drivers. An extreme case would be to study the scenario where all riders have access to vehicles and are willing to join the system as drivers. This scenario demonstrates the highest benefits a ridesharing system can offer.

In this section, we study the impact of this assumption on the problem instances in section 6.1. Figure 15 shows the maximum number of served ride requests under different percentage of riders who are willing and able to act as drivers if not successfully served as riders. The number of served riders reported in this figure is the number of participants whose last status is a rider. For example, if a participant who is originally registered as a rider switches to being a driver, they are considered as a driver and not a served rider. In the case of 0%, the problem becomes equivalent to the problem of a system with two mutually exclusive sets of riders and drivers. As this percentage increases, the shape of the curve starts to change. The best performance of the system can be obtained when 100% of riders have access to vehicles, and are willing to join the system as drivers. It is interesting to note that in this best case scenario, the number of served riders has an increasing trend as the percentage of riders in the problem instances increases (and the percentage of drivers decreases).

In the basic scenario with two mutually exclusive sets of riders and drivers, the number of served riders reaches its peak when riders constitute around 45% of the participants. When working with mutually exclusive sets of rider and drivers, the number of served riders decreases as the ratio of riders in the problem increases, because there are fewer drivers available. If riders who cannot be matched join the system as drivers, this practically increases the number of drivers in the ranges where previously there was a shortage of drivers, and this leads to higher number of riders being served. Figure 15 suggests that encouraging this flexible form of participation in a ridesharing system can drastically improve its performance.

7 Conclusion

In this paper, we define a flexible ridesharing system as a dynamic, real-time, and multi-hop system with the ability to find itineraries for riders by means of optimally routing drivers. Real-time ride-matching is a central component of a flexible P2P ridesharing system. It ensures that users who look for rides not long before their desired departure times have a chance to be served in the system. We propose an optimal and real-time ride-matching algorithm that maximizes the number of served riders in the system, while making the trips as comfortable as possible by taking into consideration users' preferences on whom to ride with, and by minimizing the number of transfers and waiting times for riders. Numerical experiments suggest that

the proposed algorithm can solve matching problems in large-scale ridesharing systems in a fraction of a second. Furthermore, results suggest that allowing transfers can have a considerable impact on the number of served riders.

We start developing the framework under the realistic assumption that riders in a real-time system need to be served on a first-come, first-served (FCFS) basis. We later introduce the concept of P2P ride exchange to reverse the negative impacts of the FCFS rule on system performance, without increasing the complexity of the problem. Furthermore, we investigate the impact of considering intersecting sets of riders and drivers on system performance, and demonstrate using numerical experiments that under the scenario where a portion of unmatched riders are willing to take the driver role, the matching rate improves significantly.

References

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang (2009). Sustainable passenger transportation: Dynamic ride-sharing. *ERIM Report Series Reference No. ERS-2010-010-LIS*.

Agatz, N. A., A. L. Erera, M. W. Savelsbergh, and X. Wang (2011). Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological 45*(9), 1450 – 1464.

Baldacci, R., V. Maniezzo, and A. Mingozzi (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research 52*(3), pp. 422–439.

Böckmann, M. (2013). The shared economy: It is time to start caring about sharing; value creating factors in the shared economy. *University of Twente, Faculty of Management and Governance*.

Braekers, K., A. Caris, and G. K. Janssens (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological 67*(0), 166 – 186.

Business Insider (2015). Uber Just Released Its First Report On Its Drivers Here Are The Numbers. http://www.businessinsider.com/uber-driver-data-report-2015-1. Accessed: 2015-06-20.

Cangialosi, E., A. Di Febbraro, and N. Sacco (2016). Designing a multimodal generalised ride sharing system. *IET Intelligent Transport Systems 10*(4), 227–236.

Carnes, T. A., S. G. Henderson, D. B. Shmoys, M. Ahghari, and R. D. MacDonald (2013). Mathematical programming guides air-ambulance routing at ornge. *Interfaces 43*(3), 232–239.

Cordeau, J.-F. and G. Laporte (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153*(1), 29–46.

Cortés, C. and R. Jayakrishnan (2002). Design and operational concepts of high-coverage point-to-point transit system. *Transportation Research Record: Journal of the Transportation Research Board* (1783), 178–187.

Di Febbraro, A., E. Gattorna, and N. Sacco (2013). Optimization of dynamic ridesharing systems. *Transportation Research Record: Journal of the Transportation Research Board* (2359), 44–50.

Ghoseiri, K. (2013). Dynamic rideshare optimized matching problem. *PhD Dissertation at University of Maryland*.

Herbawi, W. and M. Weber (2011, Nov). Ant colony vs. genetic multiobjective route planning in dynamic multi-hop ridesharing. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pp. 282–288.

Herbawi, W. M. and M. Weber (2012). A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, New York, NY, USA, pp. 385–392. ACM.

Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, and N. H. Wilson (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological 20*(3), 243 – 257.

Li, X. and L. Quadrifoglio (2010). Feeder transit services: choosing between fixed and demand responsive policy. *Transportation Research Part C: Emerging Technologies 18*(5), 770–780.

Liaw, C.-F., C. C. White, and J. Bander (1996, Sep). A decision support system for the bimodal dial-a-ride problem. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 26*(5), 552–565.

Masoud, N. and R. Jayakrishnan (2017). A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. *Transportation Research Part B: Methodological 99*, 1 – 29.

Masoud, N., R. Lloret-Batlle, and R. Jayakrishnan (2017). Using bilateral trading to increase ridership and user permanence in ridesharing systems. *Transportation Research Part E: Logistics and Transportation Review 102*, 60–77.

Masson, R., F. Lehuédé, and O. Péton (2014). The dial-a-ride problem with transfers. *Computers & Operations Research 41*(0), 12 – 23.

Psaraftis, H. N. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science 17*(3), 351–357.

Qiu, F., W. Li, and J. Zhang (2014). A dynamic station strategy to improve the performance of flex-route transit services. *Transportation Research Part C: Emerging Technologies 48*, 229–240.

Quadrifoglio, L., M. M. Dessouky, and F. Ordóñez (2008). Mobility allowance shuttle transit (mast) services: Mip formulation and strengthening with logic constraints. *European Journal of Operational Research 185*(2), 481–494.

Regue, R., N. Masoud, and W. Recker (2016). Car2work: A shared mobility concept to connect commuters with workplaces. *Transportation Research Record: Journal of the Transportation Research Board 2542*, 102–110.

Savelsbergh, M. W. P. and M. Sol (1995). The general pickup and delivery problem. *Transportation Science 29*(1), 17–29.

Schaub, T., G. Friedrich, and B. O'Sullivan (2014). *ECAI 2014: 21st European Conference on Artificial Intelligence*, Volume 263. IOS Press.

Shaheen, S. and N. Chan (2015). Mobility and the sharing economy: Impacts Synopsis. Technical report, TRANSPORTATION SUSTAINABILITY RESEARCH CENTER, University of California Berkeley.

Stein, D. M. (1978). Scheduling dial-a-ride transportation systems. *Transportation Science 12*(3), 232–249.

Teodorovic, D. and M. DellOrco (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pp. 51–60.

Vukan, V. (1981). *Urban Public Transportation: Systems and Technology*. Prentice-Hall, Englewood Cliffs, NJ, EU.

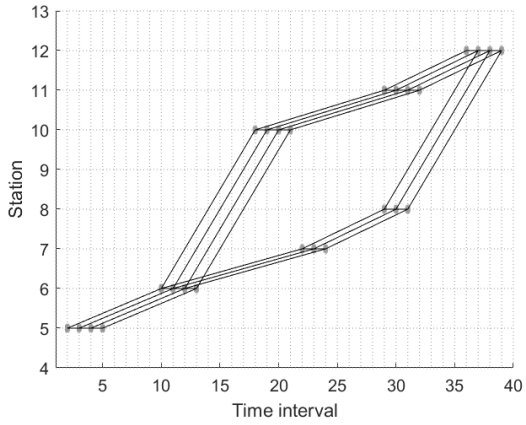
Wagner, D. and T. Willhalm (2003). Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Esa*, Volume 3, pp. 776–787. Springer.

Wagner, D., T. Willhalm, and C. Zaroliagis (2005). Geometric containers for efficient shortest-path computation. *Journal of Experimental Algorithmics (JEA) 10*, 1–3.

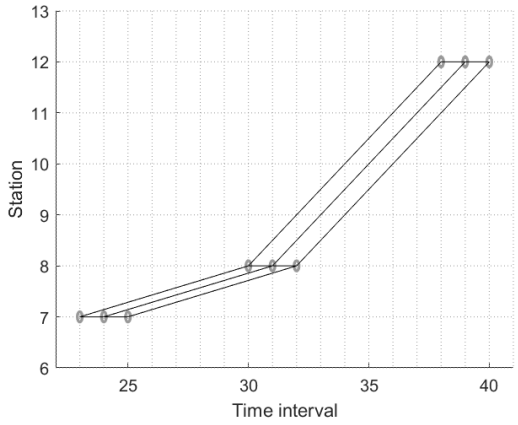
Wolfler Calvo, R., F. de Luigi, P. Haastrup, and V. Maniezzo (2004). A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research 31*(13), 2263 – 2278.

A Appendix

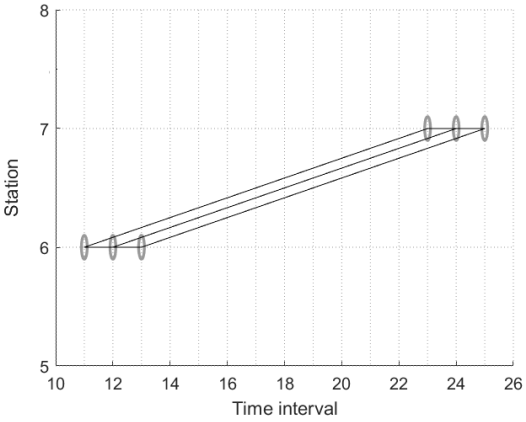
Figure 16 shows the time-expanded reduced networks of the rider, and drivers 1, 3, and 4.



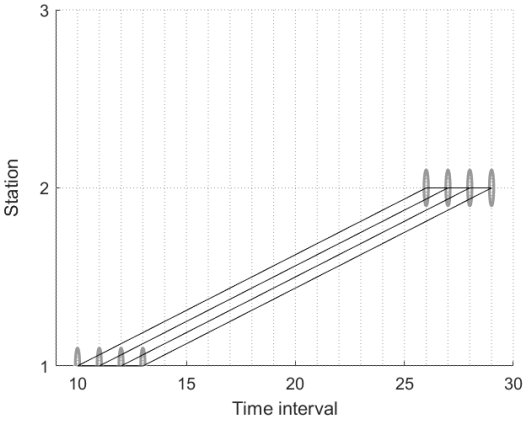
(a) Rider



(b) Driver 1



(c) Driver 3



(d) Driver 4

Figure 16: Time expanded networks of participants in the example presented in Figure 2