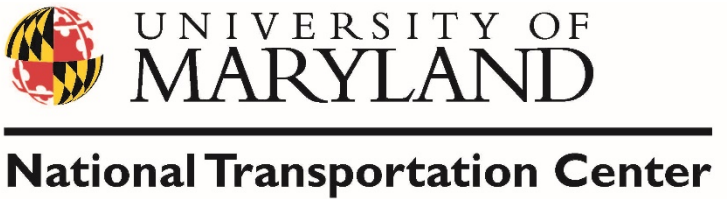


Project ID: NTC2014-SU-R-03

Congestion Mitigation Potential of Autonomous (Driverless) Vehicles: A Scenario-based Approach

Final Report

by

Xuesong Zhou
Associate Professor, School of Sustainable Engineering and the Built Environment
Arizona State University, Tempe, AZ, xzhou74@asu.edu

Monirehalsadat Mahmoudi
Graduate Student, School of Sustainable Engineering and the Built Environment
Arizona State University, Tempe, AZ, mmahmoudi@asu.edu

Ram Pendyala
Professor, School of Civil and Environmental Engineering
Georgia Institute of Technology, Atlanta, GA, ram.pendyala@ce.gatech.edu

Hossein Jalali
Graduate Student, School of Sustainable Engineering and the Built Environment
Arizona State University, Tempe, AZ, Hossein.Jalali@asu.edu

for

National Transportation Center at Maryland (NTC@Maryland)
1124 Glenn Martin Hall
University of Maryland
College Park, MD 20742

July, 2015

ACKNOWLEDGEMENTS

This project was funded by the National Transportation Center @ Maryland (NTC@Maryland), one of the five National Centers that were selected in this nationwide competition, by the Office of the Assistant Secretary for Research and Technology (OST-R), U.S. Department of Transportation (US DOT).

DISCLAIMER

TABLE OF CONTENT

Exclusive Summary

Optimization of on-demand transportation systems and ride-sharing services involves solving a class of complex vehicle routing problems with pickup and delivery with time windows (VRPPDTW). This research first proposes a new time-discretized multi-commodity network flow model for the VRPPDTW based on the integration of vehicles' carrying states within space-time transportation networks, so as to allow a joint optimization of passenger-to-vehicle assignment and turn-by-turn routing in congested transportation networks. Our three-dimensional state-space-time network construct is able to comprehensively enumerate possible transportation states at any given time along vehicle space-time paths, and further allow a forward dynamic programming solution algorithm to solve the single vehicle VRPPDTW problem. By utilizing a Lagrangian relaxation approach, the primal multi-vehicle routing problem is decomposed to a sequence of single vehicle routing sub-problems, with Lagrangian multipliers for individual passengers' requests being updated by sub-gradient-based algorithms. We further discuss a number of search space reduction strategies and test our algorithms, implemented through a specialized program in C++, on medium-scale and large-scale transportation networks, namely the Chicago sketch and Phoenix regional networks.

1. Introduction

The advent of new vehicular technologies has raised considerable debate about the potential impacts of such disruptive technologies on traveler behavior, demand for transportation services and infrastructure, and transportation network performance. There are a number of disruptive technologies that are being considered with various levels of automation, control, and communication protocols. The US Department of Transportation has ongoing initiatives related to the deployment of connected vehicle systems, and the development of analysis, modeling, and simulation tools that would facilitate the analysis of the impacts and potential congestion benefits that such connected vehicle infrastructure systems may provide. The challenge facing the profession is that there is very little information, analysis, modeling, or behavioral studies that provide a rigorous prediction of the potential impacts of these technologies on human activity-travel behavior, freight systems, public transit and taxi systems, and household and firm location choices (land use). The overall goal of this project is to develop a rigorous framework that is founded on sound behavioral constructs and analytical methods that would allow the accurate estimation of the impacts of autonomous, driverless, connected, and other advanced vehicular technologies under a variety of scenarios.

The recent emerging trend of self-driving cars (SDC), made available by private technology vendors, is likely to create a revolutionary paradigm shift in the coming years for real-time traffic system automation and control. The use of large-scale scheduling algorithms for autonomous agents represents a fundamentally new approach that will include real-time transportation system optimization, ubiquitous communication, and diverse data synthesis. When modeling different stages of SDC deployments, we will consider two types of SDC use modes: (1) a car used solely for essentially each person/household, and (2) one car shared among travelers through a transportation network company (TNC) such as Lyft and Uber. In this research, we focus on the second type of SDC use mode, shared autonomous vehicles (SAV), which can offer an economically efficient approach to meet increasing transportation demand, considering this fact that most personal cars are currently used by single drivers only 1-2 hours during a day. This SAV approach could lead to a long list of benefits, to name a few, reducing driver stress and driving costs, improving mobility for non-drivers, increasing road capacity, reducing operating costs, increasing fuel efficiency, and reducing pollution.

Through the SAV, each user, instead of using his own car, call a car just a few minutes before leaving from his origin, or pre-schedule a car in advance. The SAV system has been designed intelligently in which no one wait long for a vehicle even if he resides in a high-demand area. As illustrated in Fig. 1, the main players of operating the transportation network in a city with fully coordinated vehicle sharing system may include centralized or decentralized cloud computing (CC) centers, public-sector traffic management centers, private SAV providers, as well as a network of SAVs equipped with two-way communication capabilities. Each vehicle communicates with the traffic information providers to receive up-to-date network traffic conditions, as well as to share the traffic data where the vehicle is traversing. The scheduling

algorithm can assign multiple trip requests to a SAV, e.g. pick up three passengers from their homes and transport them to their final destinations. Passengers using their own cars to go from their home to their office would require three cars for three trips, while our proposed system could use just one SDC to satisfy all demands.

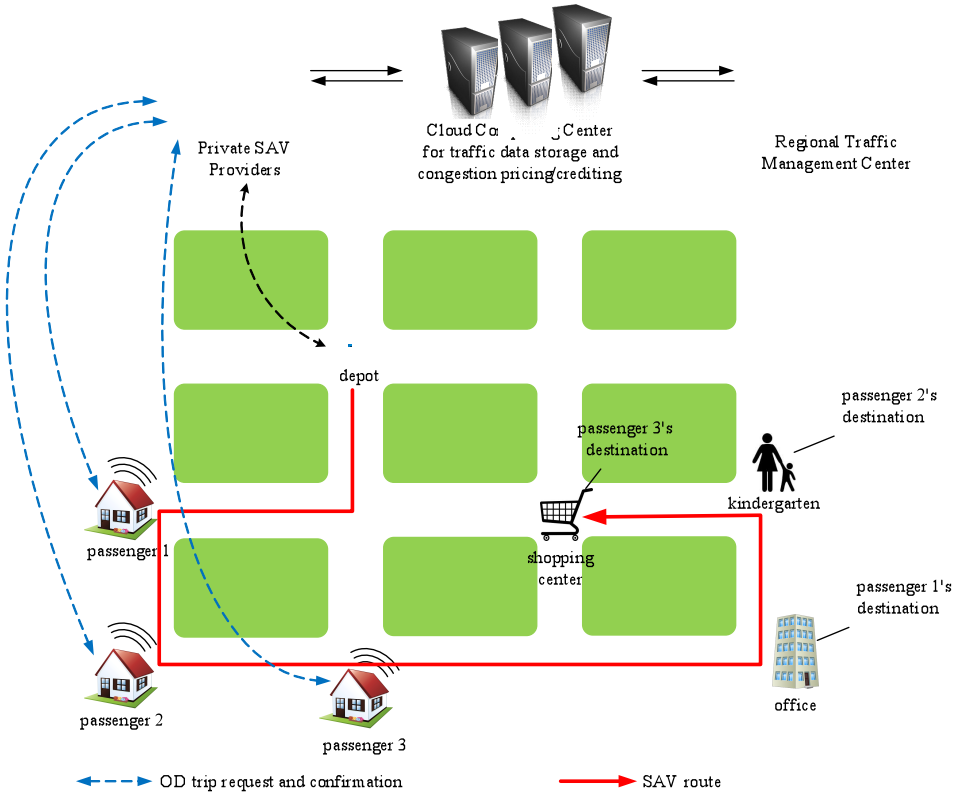


Fig. 1. The infrastructure of a city with fully coordinated vehicle sharing system

In addition to the passengers' convenience and safety factors in the second type of SDC use mode, suppose the imposed charge of this mode of transportation be considerably less than the transportation cost of the mode in which the passenger uses his own car. All these facts may arise this question that does owning a personal car remain economical in the near future?

It is not far that one day all personal cars be replaced by the SAVs. Therefore, if one is planning to establish an infrastructure for a city with fully coordinated vehicle sharing system, answering the following key questions in advance is basically required: How many cars a city should use to support the overall transportation activity demand/desires, at different levels of coordination and pre-trip scheduling? How many parking lots and road infrastructure are required? We plan to develop a holistic optimization approach for synchronizing travel activity schedules, transportation services, and infrastructure on urban networks. For answering these questions, it is required to examine the ride-sharing problem.

The ride-sharing problem can be mathematically modeled by one of the well-known optimization problem which is the vehicle routing problem with pickup and delivery (VRPPD). In this research, in order to improve the solution quality and computational efficiency of on-

demand transportation systems and dynamic ride-sharing services, especially for large-scale real-world transportation networks, we propose a new mathematical programming model for the vehicle routing problem with pickup and delivery with time windows (VRPPDTW) that can fully recognize time-dependent link travel time caused by traffic congestion at different times of day. Based on the Lagrangian relaxation solution framework, we further present a holistic optimization approach for matching passengers' requests to transportation service providers, synchronizing transportation vehicle routing, and determining request pricing (e.g. through Lagrangian multipliers) for balancing transportation demand satisfaction and resource needs on urban networks.

2. Literature Review

The vehicle routing problem with pickup and delivery with time windows (VRPPDTW) or simply, pickup and delivery problem with time windows (PDPTW), is a generalized version of the vehicle routing problem with time windows (VRPTW), in which each transportation request is a combination of pickup at the origin node and drop-off at the destination node (Desaulniers et al. 2002). The PDPTW under consideration in this paper contains all constraints in the VRPTW plus an added constraint in which either pickup or delivery has given time windows, and each request must be served by the same vehicle. The PDPTW may be observed as the dial-a-ride problem in the literature as well. Since the VRPTW is an NP-hard problem, the PDPTW is also NP-hard (Baldacci et al. 2011).

Several applications of the VRPPDTW have been reported in road, maritime, and air transportation environments, to name a few, Fisher et al. (1982), Bell et al. (1983), Savelsbergh and Sol (1998), Wang and Regan (2002), and Zachariadis et al. (2015) in road cargo routing and scheduling; Psaraftis et al. (1985), Fisher and Rosenwein (1989), and Christiansen (1999) in sea cargo routing and scheduling; and Solanki and Southworth (1991), Solomon et al. (1992), Rappoport et al. (1992), and Rappoport et al. (1994) in air cargo routing and scheduling. Further applications of the VRPPDTW can be found in transportation of elderly or handicapped people (Jaw et al. 1986; Alfa, 1986; Ioachim et al. 1995; and Toth and Vigo, 1997), school bus routing and scheduling (Swersey and Ballard, 1983; and Bramel and Simchi-Levi, 1995), and ride-sharing (Hosni et al. 2014; and Wang et al. 2015). Recently, Furuhata et al. (2013) offers an excellent review and provides a systematic classification of emerging ridesharing systems.

Although clustering algorithms (Cullen et al. 1981; Bodin and Sexton, 1986; Dumas et al. 1989; Desrosiers et al. 1991; and Ioachim et al. 1995), meta-heuristics (Gendreau et al. 1998; Toth and Vigo, 1997; and Paquette et al. 2013), neural networks (Shen et al. 1995), and some heuristics such as double-horizon based heuristics (Mitrovic-Minic et al. 2004) and regret insertion heuristics (Diana and Dessouky, 2004) have been shown to be efficient in solving a particular size of PDPTW, in general, finding the exact solution via optimization approaches has still remained theoretically and computationally challenging. Focusing on the PDPTW for a single vehicle, Psaraftis (1980) presented an exact backward dynamic programming (DP) solution algorithm to minimize a weighted combination of the total service time and the total

waiting time for all customers with $O(n23n)$ complexity. Psaraftis (1983) further modified the algorithm to a forward DP approach. Sexton and Bodin (1985a, b) decomposed the single vehicle PDPTW to a routing problem and a scheduling sub-problem, and then they applied Benders' decomposition for both master problem and sub-problem, independently. Based on a static network flow formulation, Desrosiers et al. (1986) proposed a forward DP algorithm for the single-vehicle PDPTW with the objective function of minimizing the total traveled distance to serve all customers. After presenting our proposed model in the later section, we will conduct a more systematical comparison between our proposed state-space-time DP framework and the classical work by Psaraftis (1983) and Desrosiers et al. (1986).

There are a number of studies focusing on the multi-vehicle pickup and delivery problem with time windows. Dumas et al. (1991) proposed an exact algorithm to the multiple vehicle PDPTW with multiple depots, where the objective is to minimize the total travel cost with capacity, time window, precedence and coupling constraints. They applied a column generation scheme with a shortest path sub-problem to solve the PDPTW, with tight vehicle capacity constraints, and a small size of requests per route. Ruland (1995) and Ruland and Rodin (1997) proposed a polyhedral approach for the vehicle routing problem with pickup and delivery. Savelsbergh and Sol (1998) proposed an algorithm for the multiple vehicle PDPTW with multiple depots to minimize the number of vehicles needed to serve all transportation requests as the primary objective function, and minimizing the total distance traveled as the secondary objective function. Their algorithm moves toward the optimal solution after solving the pricing sub-problem using heuristics. They applied their algorithm for a set of randomly generated instances. In a two-index formulation proposed by Lu and Dessouky (2004), a branch-and-cut algorithm was able to solve problem instances. Cordeau (2006) proposed a branch-and-cut algorithm based on a three-index formulation. Ropke et al. (2007) presented a branch-and-cut algorithm to minimize the total routing cost, based on a two-index formulation. Ropke and Cordeau (2009) presented a new branch-and-cut-and-price algorithm in which the lower bounds are computed by the column generation algorithm and improved by introducing different valid inequalities to the problem. Based on a set-partitioning formulation improved by additional cuts, Baldacci et al. (2011) proposed a new exact algorithm for the PDPTW with two different objective functions: the primary is minimizing the route costs, whereas the secondary is to minimize the total vehicle fixed costs first, and then minimize the total route costs.

3. Research Motivations

Previous research has made a number of important contributions to this challenging problem along different formulation or solution approaches. On the other hand, there are a number of modeling and algorithmic challenges for a large-scale deployment of a vehicle routing and scheduling algorithm, especially for regional networks with various road capacity and traffic delay constraints on freeway bottlenecks and signal timing on urban streets. A majority of previous research does not directly consider the underlying transportation network (with time of day traffic congestion) and has defined the PDPTW on a directed graph containing customers'

origin and destination locations connected by some links which are representative of the shortest distance or least travel time routes between origin-destination pairs. That is, with each link, there are associated routing cost and travel time between the two service nodes. Unlike the existing offline network for the PDPTW in which each link has a fixed routing cost (travel time), our research particularly examines the PDPTW on real-world transportation networks containing a transportation node-link structure in which routing cost (travel time) along each link may vary over the time.

In order to consider many relevant practical aspects, such as waiting costs at different locations, we adopt and develop Yang and Zhou's (2014) space-time scheme to formulate the PDPTW on state-space-time transportation networks. The constructed networks are able to conveniently represent the complex pickup and delivery time windows without adding the extra constraints typically needed for the classical PDPTW formulation (e.g. Cordeau, 2006). The introduced state-space-time networks also enable us to embed computationally efficient dynamic programming algorithms for solving the PDPTW without relying on off-the-shelf optimization solvers. Even though the solution space created by our formulation has multiple dimensions and accordingly large in its sizes, the readily available large amount of computer memory in modern workstations can easily accommodate the multi-dimensional solution vectors utilized in our application. Our fully customized solution algorithms, implemented in an advanced programming language such as C++, hold the promise of tackling large-sized regional transportation network instances. To address the multi-vehicle assignment requirement, we relax the transportation request satisfaction constraints into the objective function and utilize the related Lagrangian relaxation (LR) solution framework to decompose the primal problem to a sequence of time-dependent least-cost-path sub-problems.

In our proposed solution approach, we aim to incorporate several lines of pioneering efforts in different directions. Specifically, we (1) reformulate the VRPPDTW as a time-discretized, multi-dimensional, multi-commodity flow model with linear objective function and constraints, (2) extend the static DP formulation to a fully time-dependent DP framework for single-vehicle VRPPDTW problems, and (3) develop a LR solution procedure to decompose the multi-vehicle scheduling problem to a sequence of single-vehicle problems and further nicely integrate the demand satisfaction multipliers within the proposed state-space-time network.

Based on the Lagrangian relaxation solution framework, we further present a holistic optimization approach for matching passengers' requests to transportation service providers, synchronizing transportation vehicle routing, and determining request pricing (e.g. through Lagrangian multipliers) for balancing transportation demand satisfaction and resource needs on urban networks.

The rest of the research is organized as follows. Section 3 contains a precise mathematical description of the PDPTW in the state-space-time networks. In section 4, we present our new integer programming model for the PDPTW followed by a comprehensive comparison between Cordeau's model and our model. Then, we will show how the main problem is decomposed to an easy-to-solve problem by the Lagrangian relaxation algorithm in section 5. Section 6 provides

computational results of the six-node transportation network, followed by the Chicago sketch and Phoenix regional networks to demonstrate the computational efficiency and solution optimality of our developed algorithm coded by C++. After large-scale network experiments, we conclude the research in section 7 with discussions on possible extensions.

4. Problem Statement Based on State-space-time Network Representation

In this section, we first introduce our new mathematical model for the PDPTW. This is followed by a comprehensive comparison between our proposed model and the three-index formulation of Cordeau (2006) for the PDPTW, presented in Appendix A, for the demand node-oriented network.

4.1.Description of the PDPTW in State-space-time Networks

We formulate the PDPTW on a transportation network, represented by a directed graph and denoted as $G(N, A)$, where N is the set of nodes (e.g. intersections or freeway merge points) and A is the set of links with different link types such as freeway segments, arterial streets and ramps. As shown in Table 1, each directed link (i,j) has time-dependent travel time $TT(i,j,t)$ from node i to j starting at time t. Every passenger p has a preferred time window for departure from his origin, $[ap,bp]$, and a desired time window for arrival at his destination, $[ap',bp']$, where ap, bp, ap', and bp' are passenger p's earliest preferred departure time from his origin, latest preferred departure time from his origin, earliest preferred arrival time at his destination, and latest preferred arrival time at his destination, respectively. Each vehicle v also has the earliest departure time from its starting depot, ev, and the latest arrival time at its ending depot, lv. In the PDPTW, passengers may share their trip with each other; in other words, every vehicle v, considering its capacity and the total routing cost, may serve as many passengers as possible provided that passenger p is picked up and dropped-off in his preferred time windows, ap, bp and ap',bp', respectively.

Each transportation node has the potential to be the spot for picking up or dropping off a passenger. Likewise, a vehicle's depot might be located at any node in the transportation network. To distinguish regular transportation nodes from passengers' and vehicles' origin and destination, we add a single dummy node ov for vehicle v's origin depot and a single dummy node dv for vehicle v's destination depot. Similarly, we can also add dummy nodes op and dp for passenger p. Each added dummy node is only connected to its corresponding physical transportation node by a link. The travel time on this link can be interpreted as the service time if the added dummy node is related to a passenger's origin or destination, and as preparation time if it is related to a vehicle's starting or ending depot. Table 1 lists the notations for the key sets, indices and parameters in the PDPTW.

Table 1. Sets, indices and parameters in the PDPTW.

Symbol	Definition
V	Set of physical vehicles
$V*$	Set of virtual vehicles
P	Set of passengers
N	Set of physical transportation nodes in the physical traffic network based on geographical location
W	Set of possible passenger carrying states
v	Vehicle index
$vp*$	Index of virtual vehicle exclusively dedicated for passenger p
p	Passenger index
w	Passenger carrying state index
(i,j)	Index of physical link between adjacent nodes i and j
TTi,j,t	Link travel time from node i to node j starting at time t
	Maximum capacity of vehicle
ap	Earliest departure time from passenger p's origin
bp	Latest departure time from passenger p's origin
ap'	Earliest arrival time to passenger p's destination
bp'	Latest arrival time to passenger p's destination
ap, bp	Departure time window for passenger p's origin
ap',bp'	Arrival time window for passenger p's destination
ov'	Dummy node for vehicle v's origin
dv'	Dummy node for vehicle v's destination
ev	Vehicle v's earliest departure time from the origin depot
lv	Vehicle v's latest arrival time to the destination depot
op	Dummy node for passenger p's origin (pickup node for passenger p)
dp	Dummy node for passenger p's destination (delivery node for passenger p)

We now use an illustrative example to demonstrate key modeling features of constructed networks. Consider a physical transportation network consisting of six nodes presented in Fig. 2. Each link in this network is associated with time-dependent travel time $TT(i,j,t)$. Without loss of

generality, the number written on each link denotes the time-invariant travel time *TT(i,j)* in terms of minutes. Suppose two requests with two origin-destination pairs should be served. For simplicity, it is assumed that both passengers have the same origin (node 2) and the same drop-off node (node 3). There is only one vehicle available for serving. Moreover, it is assumed that the vehicle starts its route from node 4 and ends it at node 1. Passenger 1 should be picked up from dummy node *o1* in time window *[4,7]* and dropped off at dummy node *d1* in time window *11,14*, while Passenger 2 should be picked up from dummy node *o2* in time window *[8,10]* and dropped off at dummy node *d2* in time window *13,16*. Vehicle 1 also has the earliest departure time from its starting depot, *t=1*, and the latest arrival time at its ending depot, *t=20*.

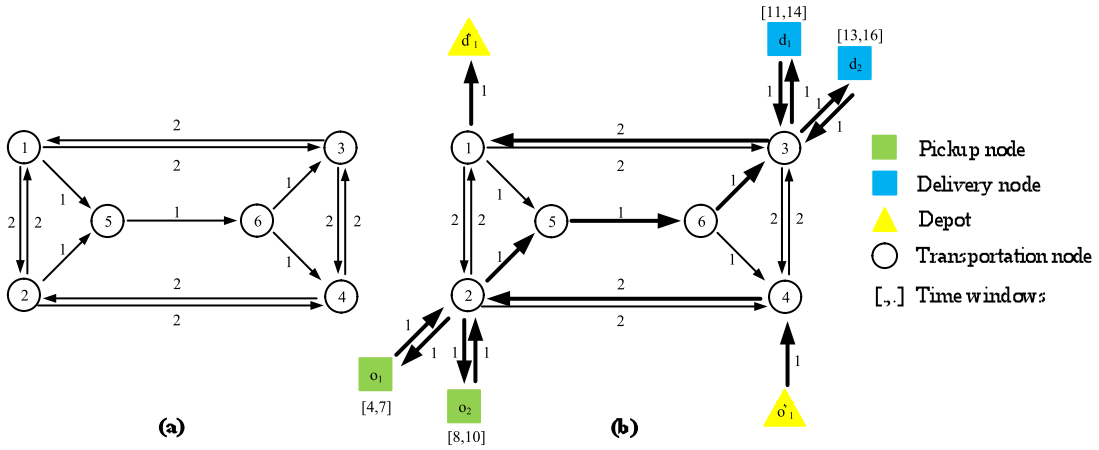


Fig. 2. (a) Six-node transportation network; (b) transportation network with the corresponding dummy nodes.

Note that the shortest path with node sequence *(o1',4, 2, o1, 2,o2,2, 5, 6, 3, d1,3, d2,3, 1, d1')* from vehicle 1's origin to its ending depot is shown by bold arrows when it serves both passenger 1 and 2. To construct a state-space-time network, the time horizon is discretized into a series of time intervals with the same time length. Without loss of generality, we assume that a unit of time has one minute length. To avoid more complexity in the vehicle's space-time network illustrated in Fig. 3, only those arcs constituting the shortest paths from vehicle 1's origin to its destination are demonstrated.

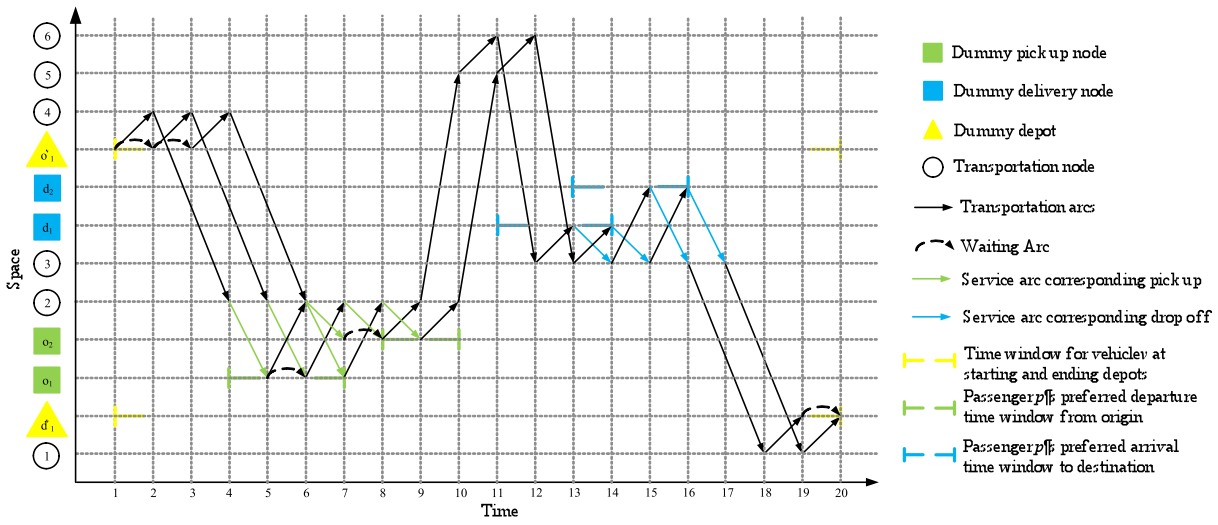


Fig. 3. Shortest paths with node sequence *(o1',4, 2, o1, 2,o2,2, 5, 6, 3, d1,3, d2,3, 1, d1')* in vehicle 1's space-time network.

Our formulation has a set of precise rules to allow or restrict the vehicle waiting behavior in the constructed space-time network, depending on the type of nodes and the associated time window. First, vehicle v may wait at its own origin or destination depot or at any other physical transportation nodes. If a vehicle arrives at passenger p's origin node before time ap, it must wait at the related physical node until the service is allowed to begin. Moreover, we assume that a vehicle is not allowed to stop at passenger p's dummy origin node after time bp. Similarly, if a vehicle arrives at passenger p's destination node before time ap', it must wait until it is allowed to drop-off passenger p, and vehicle v is not allowed to stop at passenger p's dummy destination node after time bp'.

In the problem under consideration, we assume all passengers' desired departure and arrival time windows are feasible. However, it is quite possible that some passenger transporting requests could not be satisfied at all since the total number of physically available vehicles in the ride-sharing company or organization is not enough to satisfy all the demands. To avoid infeasibility for the constructed optimization problem, we define a virtual vehicle for each passenger exclusively. We assume that both starting and ending depots of virtual vehicle $vp*$ are located exactly where passenger p is going to be picked up. By doing so, there is no cost incurred if the virtual vehicle is not needed to carry the related passenger, and in this case the virtual vehicle simply waits at its own depot. On the other hand, if the virtual vehicle is needed to perform the service and ensure there is a feasible solution, then virtual vehicle $vp*$ starts its route from its starting depot, picks up passenger p, delivers him to his destination, and then comes

back to its ending depot. Fig. 4 shows the shortest paths with node sequence *(o1∗',2, o1, 2, 5, 6, 3, d1,3, 1, 2, d1∗')* in vehicle *v1∗*'s space-time network.

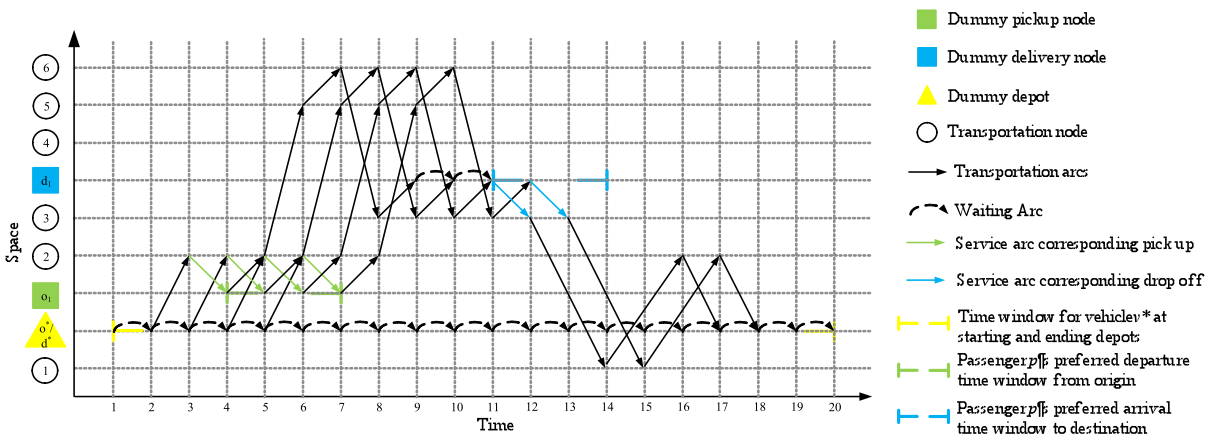


Fig. 4. Shortest paths with node sequence *(o1∗',2, o1, 2, 5, 6, 3, d1,3, 1, 2, d1∗')* in vehicle *v1∗*'s space-time network.

4.2. Representing the State of System and Calculating the Number of States

In the context of dynamic programming, we need to decompose the complex VRP structure into a sequence of overlapping stage-by-stage sub-problems in a recursive manner. For each stage of the optimization problem, we need to define the state of the process so that the state of the system with stages to go can fully summarize all relevant information of the system for future decision-making purposes no matter how the process has reached the current stage . In our pickup and delivery problem, in each vehicle's network, the given time index acts as the stage, and the state of the system is jointly defined by two indexes: node index and the passenger carrying state index . The latter passenger carrying state can be also represented as a vector with number of elements , where equals 1 or 0 and denotes the status of passenger whether he is riding the vehicle or not. To facilitate the descriptions of the state transition, we introduce the following equivalent notation system for passenger carrying states: if a vehicle carries passenger , the th element of the state is filled with passenger 's id; otherwise, it is filled with a dash sign, as illustrated in Table 2.

Table 2. Binary representation and equivalent character-based representation for passenger carrying states.

Binary representation	Equivalent character-based representation
	[_ _ _]

Without loss of generality, for a typical off-line vehicle routing problem, the initial and ending states of the vehicles are assumed to be empty, corresponding to the state *[_ _ _]*. For an on-line dynamic vehicle dispatching application, one can define the starting passenger carrying state to indicate the existing passengers riding the vehicle, for example, if passenger 1 is being served currently. We use an illustrative example to demonstrate the concept of a passenger's carrying state clearly. Suppose three requests with three different origin-destination pairs should be served. There is only one vehicle available for serving and let's assume that the vehicle can carry up to two passengers at the same time. We can enumerate all different carrying states for the vehicle. The first state is the state in which the vehicle does not carry any passenger*[_ _ _]*. There are number of possible carrying states in which the vehicle only carries one passenger at time :, , and . Similarly, there are number of possible carrying states in which the vehicle carries two passengers at time which are , , and . Since the vehicle can carry up to two passengers at the same time, the state of is infeasible. Fig. 5(a) and Fig. 5(b) show shared ride state and single-passenger-serving state .

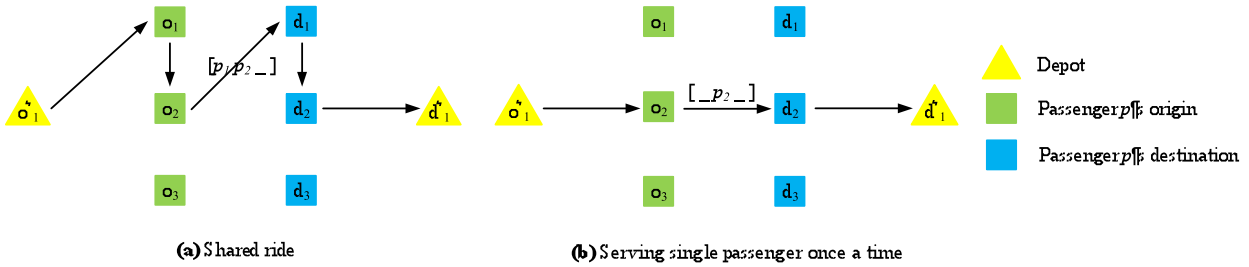


Fig. 5. State transition path (a) Passenger carrying state ; (b) Passenger carrying state .

We are further interested in the number of feasible states, which critically determines the computational efforts of the DP-based solution algorithm. First, there is a unique state in which vehicle does not carry any passenger, which is a combinatory of for selecting passengers from the collection of passengers. Similarly, there are number of possible carrying states in which vehicle only carries one passenger at a time. Likewise, there are number of possible carrying states in which vehicle carries passengers at a time. Note that . Therefore, the total number of possible passenger carrying states is equal to . It should be remarked that, according to the earliest departure time from the origin and the latest arrival time to the destination of different passengers, some of the possible carrying states, say , might be infeasible as there is insufficient transportation time to pick up those two passengers together while satisfying their time window constraints.

Consider the following example, where passenger 1 should be picked up in time window [4,7] and delivered in time window [9,12], whereas passenger 3's preferred time windows for being picked up and delivered are [20,24] and [25,29], respectively. So, it is obvious that passenger 1 and 3 cannot share their ride with each other and be transported at the same time by the same vehicle. Therefore, carrying state is definitely infeasible in this example. We will

further explain how to reduce the search region by defining some rational rules and simple heuristics in section 5.3.

4.3. State Transition Associated with Pickup and Delivery Links

Each vehicle starts its trip from the empty state in which the vehicle does not carry any passengers. We call this state as the initial state (). Each vertex in the constructed state-space-time network is recognized by a triplet of three different indexes: node index i, time interval index t, and passenger carrying state index w. In the space-time transportation network construct, we can identify a traveling arc starting from node i at time t arriving at node j at time s. Accordingly, in the state-space-time network, each vertex is connected to vertex through arc . To find all feasible combination of passenger carrying state transition on an arc, it is sufficient to follow these rules:

Rule 1. On a pick-up link (with the passenger origin dummy node as the downstream node), vehicle picks up passenger , so is changed from 0 to 1, or equivalently, theth element of the corresponding states should be changed from a dash sign to passenger id.

Rule 2. On a drop-off link (with the passenger destination dummy node as the upstream node), vehicle drops off passenger , so is changed from 1 to 0, and the th element of the corresponding states should be changed from passenger id to a dash sign.

Rule 3. On a transportation link or links connected to vehicle dummy nodes, vehicle neither picks up nor drops off any passenger, and all elements of and should be the same.

To find all feasible passengers state transition , we need to examine all possible combinations of and . Consider a three-passenger case, in which Table 3 identifies all possible combinations of these state transitions. Note that the vehicle can carry up to two passengers at the same time. The empty cells indicate impossible state transitions in the constructed space-time network with dedicated dummy nodes. The corresponding possible passenger carrying state transitions (pickup or drop-off) are illustrated in one graph in Fig. 6. Fig. 7 represents the projection on state-space network for the example presented in section 3.1.

Table 3. All possible combinations of passenger carrying states.

	[_ _ _]						
[_ _ _]	no change	pickup	pickup	pickup			
	drop-off	no change			pickup	pickup	
	drop-off		no change		pickup		pickup
	drop-off			no change		pickup	pickup
		drop-off	drop-off		no change		
		drop-off		drop-off		no change	
			drop-off	drop-off			no change

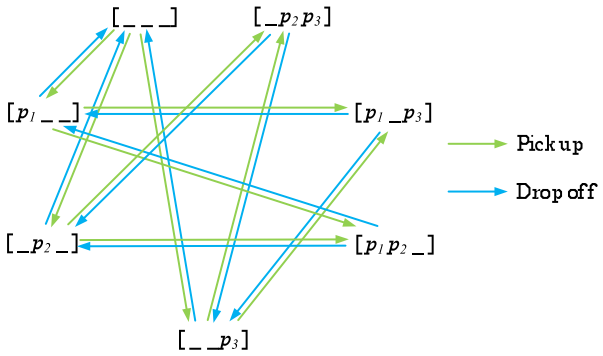


Fig. 6. Finite states graph showing all possible passenger carrying state transition (pickup or drop-off).

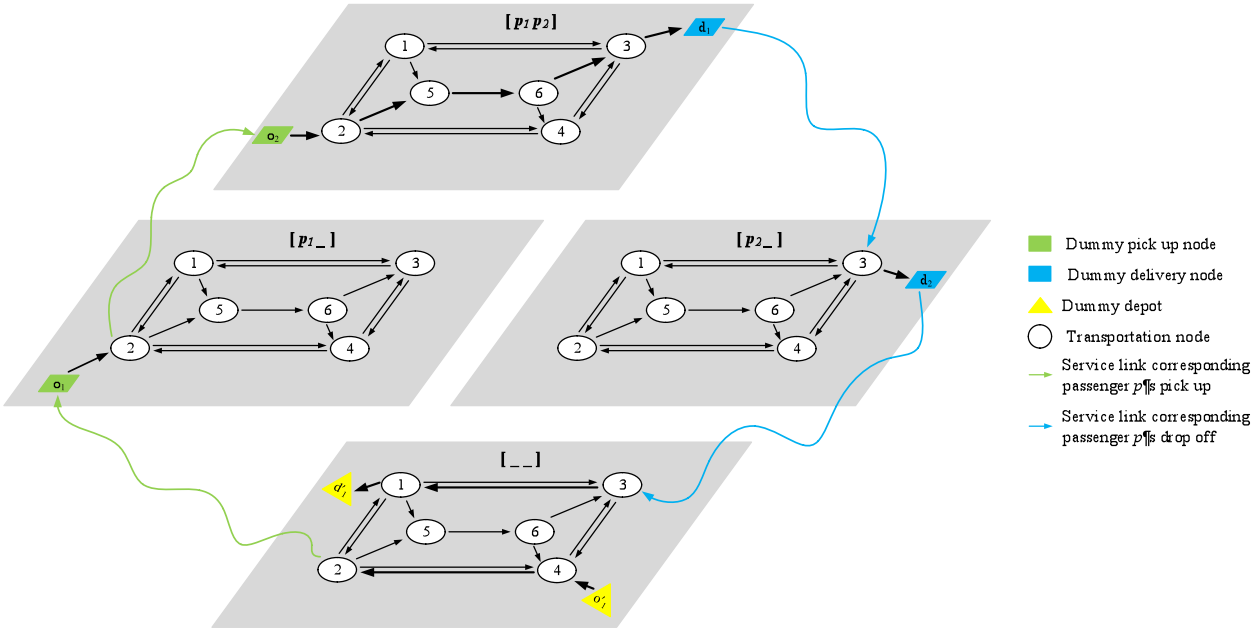


Fig. 7. Projection on state-space network representation for ride-sharing path (pick up passenger *p1* and then *p2*).

5.Time-discretized Multi-commodity Network Flow Programming Model

Based on the constructed state-space-time networks that can capture essential pickup and delivery time window constraints, we now start constructing a multi-commodity network flow programing model for the VRPPDTW. In this multi-dimensional network, the challenge is to systematically describe the related flow balance constraints for vehicles and request satisfaction constraints for passengers. As shown in Table 4, we use i,t,w to represent the indices of state-space-time vertices, and the corresponding arc index which is i,j,t,s,w,w'. Let Bv denote the set

of state-space-time arcs in vehicle v's network, which has three different types of arcs, namely, service arcs, transportation arcs and waiting arcs.

i. All passenger carrying state transitions (i.e., pickup or drop-off) occurs only on service arcs. In other words, all incoming arcs to passengers' origin nodes (pickup arcs shown by green lines in Figures 6 and 7) and all outgoing arcs to their destination nodes (drop-off arcs shown by blue lines in Figures 6 and 7) are considered service arcs.
ii. A link with both ends as physical nodes or vehicle dummy nodes are considered transportation arcs.
iii. Vehicles (both physical and virtual) may wait at any node i of the state-space-time network through waiting arcs $(i,i,t,t+1,w,w)$ from time t to time $t+1$ with the same passenger carrying state w.

Table 4. Indexes and variables used in the time-discretized network flow model.

Symbol	Definition
(i,t,w), (j,s,w')	Indexes of state-space-time vertexes
(i,j,t,s,w,w')	Index of a space-time-state arc indicating that one can travel from node i at time t with passenger carrying state w to the node j at time s with passenger carrying state w'
	Set of state-space-time arcs in vehicle v's network
	Routing cost of arc i,j,t,s, w,w' traveled by vehicle v
	Travel time of arc i,j,t,s, w,w' traveled by vehicle v
	Set of pickup service arcs of passenger p in vehicle v's networks
	Set of drop-off service arcs of passenger p in vehicle v's networks
	$=1$ if arc i,j,t,s,w,w' is used by vehicle v; $=0$ otherwise

In general, the travel time TTv,i,j,t,s, w,w' is the travel time of traversing from node i at time t with passenger carrying state w to node j at time s with passenger carrying state w' by vehicle v. As we mentioned before, travel time for service arcs can be interpreted as the service time needed to pick up or drop-off a passenger, and as the preparation time if the arc is related to a vehicle's starting or ending depot. In addition, the travel time of the waiting arcs is assumed to be a unit of time.

The routing cost cv,i,j,t,s, w,w' for an arc can be defined as follows. The routing cost of a transportation arc is defined as a ratio of its travel time. For the physical vehicle, this ratio is basically the total transportation cost per hour when the vehicle is traveling, which may include the fuel, maintenance, depreciation, insurance costs, and more importantly, the cost of hiring a

full-time or part-time driver. Let's assume that, in total, the transportation by a physical vehicle costs x dollars per hour. Since passengers should be served by physical vehicles by default and virtual vehicles serve passengers only if there is no available physical vehicle to satisfy their demand, we impose a quite expensive transportation cost per hour for virtual vehicles, let's say $2x$ dollars per hour. The routing cost of the service arcs are defined similarly to the routing cost of the transportation arcs. The routing cost of a waiting arc is also defined as a ratio of its travel time. However, this ratio is basically the total transportation cost of the physical vehicle v per hour when the driver has turned off the vehicle and is waiting at a node, which may only include the cost of hiring a full-time or part-time driver. Let's assume that, in total, waiting at a node by a physical vehicle costs y dollars per hour, with a typical relationship of waiting cost < transportation cost per hour, i.e., $y<x$. We assume that waiting at a destination depot for physical vehicles has no cost in order to encourage a vehicle to reduce the total transportation time, if possible. Moreover, for virtual vehicles, the waiting cost is always equal to zero to allow a virtual vehicle be totally idle at its own depot.

The model uses binary variables yv,i,j,t,s,w,w' equal to 1 if and only if state-space-time arc i,j,t,s,w,w' is used by vehicle v. Without loss of generality, we assume that a vehicle does not carry any passenger when it departs from its origin depot or arrives to its destination depot, which correspond to the passenger carrying state at node $i=ov', t=ev)$ and $(j=dv',s=lv)$ as an empty state denoted by $w0$. Note that, since passenger carrying state transitions only occur through service arcs, $w=w'=w0$ for yv,ov',j,ev,s, w,w' and yv,i,dv',t,lv,w,w'. After constructing the state-space-time transportation network for each vehicle, the PDPTW can be formulated as follows:

$$Min\ Z=v\in(V\cup V*)i,j,t,s,w,w'\in Bvcv,i,j,t,s,\ w,w'yv,i,j,t,s,w,w' \quad (1)$$

s.t.

Flow balance constraints at vehicle v's origin vertex

$$i,j,t,s,\ w,w'\in Bvyv,i,j,t,s,\ w,w'=1 \qquad i=ov',\ t=ev,\ w=w'=w0,\ \forall v\in(V\cup V*) \quad (2)$$

Flow balance constraint at vehicle v's destination vertex

$$i,j,t,s,\ w,w'\in Bvyv,i,j,t,s,w,w'=1 \qquad j=dv',s=lv,\ w=w'=w0,\ \forall v\in(V\cup V*) \quad (3)$$

Flow balance constraint at intermediate vertex

$$j,s,w''yv,i,j,t,s,\ w,w''-\ j',s',w'yv,j',i,s',t,\ w',w=0 \quad i,t,wov',ev,\ w0dv',lv,\ w0,\ \forall v\in(V\cup V*) \quad (4)$$

Passenger p's pick-up request constraint

$$v\in(V\cup V*)i,j,t,s,w,w'\in\Psi p,vyv,i,j,t,s,w,w'=1 \qquad \forall p\in P \quad (5)$$

Passenger p's drop-off request constraint

$$\sum_{v\in(V\cup V_*)}\sum_{i,j,t,s,w,w'\in\Phi p,v} y_{v,i,j,t,s,w,w'}=1 \qquad \forall p\in P \quad (6)$$

Binary definitional constraint

$$y_{v,i,j,t,s,w,w'}\in\{0,1\} \qquad \forall i,j,t,s,w,w'\in B_v,\ \forall v\in(V\cup V_*) \quad (7)$$

The objective function (1) minimizes the total routing cost. Constraints (2) to (4) ensure flow balance on every vertex in vehicle v's state-space-time transportation network. Constraints (5) and (6) express that each passenger is picked up and dropped-off exactly once by a vehicle (either physical or virtual). Constraint (7) defines that the decision variables are binary.

The three-index formulation of Cordeau (2006) for the PDPTW in the origin-destination network is presented in Appendix A. Table 5 shows that our proposed model encompasses all constraints used in Cordeau's model.

Table 5. An analogy between Cordeau's model and our model for the PDPTW.

Cordeau (2006)	Our Model
three-index variables $\boldsymbol{x_{ijv}}$ for vehicle v on link (i,j)	Seven-index variable $y_{v,i,j,t,s,w,w'}$ for vehicle v on arc i,j,t,s,w,w'.
(A.1) minimizes the total routing cost.	(1) minimizes the total routing cost.
(A.2) guarantees that each passenger is picked up.	(5) guarantees that each passenger is picked up.
(A.2) and (A.3) ensure that each passenger's origin and destination are visited exactly once by the same vehicle.	(2) to (6) ensure that the same vehicle v transports passenger p from his origin to his destination.
(A.4) expresses that each vehicle $\boldsymbol{v}$ starts its route from the origin depot.	(2) expresses that each vehicle v starts its route from the origin depot.
(A.5) ensures the flow balance on each node.	(2) to (4) ensure flow balance on every vertex in vehicle v's network.
(A.6) expresses that each vehicle $\boldsymbol{v}$ ends its route to the destination depot.	(3) expresses that each vehicle v ends its route to the destination depot.
(A.7) ensures the validity of the time variables.	The essence of state-space-time networks ensures the time variables are calculated correctly through arc i,j,t,s,w,w', where arrival time $s = t+TT(i,j,t)$.
(A.8) ensures the validity of the load variables.	The structure of state-space-time networks ensures that each vehicle transports a number

	of passengers up to its capacity at a time, in terms of feasible states (w,w').
(A.9) defines each passenger's ride time.	Employing state-space-time networks defines each passenger's ride time.
(A.10) imposes the maximal duration of each route.	Vehicle v's network is constructed subject to time window $[ev, lv]$.
(A.11) imposes time windows constraints.	Passenger p's network is constructed subject to time window $[ap, bp']$.
(A.12) imposes ride time of each passenger constraints.	Passenger p's network is constructed subject to time window $[ap, bp']$.
(A.13) imposes capacity constraints.	The structure of state-space-time networks ensures that each vehicle transports a number of passengers up to its capacity at a time.
(A.14) defines that the decision variables are binary.	(7) defines binary decision variables.

6. Lagrangian Relaxation-based Solution Approach

Constraints (5) and (6) express that each passenger is picked up and dropped off exactly once by a vehicle (either physical or virtual). On the other hand, the constraints (3) force the vehicle to end its route at the destination depot with the empty passenger carrying state. Therefore, if vehicle v picks up passenger p from his origin, to maintain the flow balance constraints (3), the vehicle must drop-off the passenger at his destination node so that the vehicle comes back to its ending depot with the empty passenger carrying state. As a result, constraint (6) is redundant and it does not need to enter into the following discussion for Lagrangian relaxation-based algorithms.

Defining multi-dimensional decision variables yv,i,j,t,s,w,w' leads to computational challenges for the large-scale real-world data sets, which should be addressed properly by specialized programs and an innovative solution framework. We reformulate the problem by relaxing the complicating constraints (5) into the objective function and introducing Lagrangian multipliers, λp, to construct the dualized Lagrangian function (8).

$$L= \sum_{v\in(V\cup V_*)i,j,t,s,w,w'\in B} vcv,i,j,t,s,w,w'yv,i,j,t,s,w,w'+\sum_{p\in P}\lambda p\left(\sum_{v\in(V\cup V_*)i,j,t,s,w,w'\in\Psi p,v} yv,i,j,t,s,w,w'-1\right) \quad (8)$$

Therefore, the new relaxed problem can be written as follows:

$Min\ L$

(9)

s.t.

$i,j,t,s,\ w,w' \in B_v y_{v,i,j,t,s,\ w,w'} = 1 \qquad i = o_{v'},\ t = e_v,\ w = w' = w0,\ \forall v \in (V \cup V_*)$ (10)

$i,j,t,s,\ w,w' \in B_v y_{v,i,j,t,s,w,w'} = 1 \qquad j = d_{v'}, s = l_v,\ w = w' = w0,\ \forall v \in (V \cup V_*)$ (11)

$j,s,w'' y_{v,i,j,t,s,\ w,w''} - j',s',w' y_{v,j',i,s',t,\ w',w} = 0 \quad i,t,w o_{v'},e_v,\ w0 d_{v'},l_v,\ w0,\ \forall v \in (V \cup V_*)$

(12)

$y_{v,i,j,t,s,w,w'} \in \{0, 1\} \qquad \forall i,j,t,s,w,w' \in B_v,\ \forall v \in (V \cup V_*)$ (13)

If we further simplify function L, the problem will become a time-dependent least-cost path problem in the constructed state-space-time network. The simplified Lagrangian function L can be written in the following form:

$L = v \in (V \cup V_*) i,j,t,s,w,w' \in B_v \xi_{v,i,j,t,s,\ w,w'} y_{v,i,j,t,s,w,w'} - p \in P \lambda_p$ (14)

Where the generalized arc cost $\xi_{v,i,j,t,s,\ w,w'}$ equals $c_{v,i,j,t,s,\ w,w'} + \lambda_p$ for each arc $i,j,t,s,w,w' \in \Psi_{p,v}$, and equals $c_{v,i,j,t,s,\ w,w'}$, otherwise.

6.1.Time-dependent Forward Dynamic Programming and Computational Complexity

In this section, we use a time-dependent dynamic programming (DP) algorithm to solve the least-cost path problem obtained in section 4. The structure of the state-space-time network ensures that time always advances on the arcs of the networks. In this research, let us consider the unit of time as one minute. Let $\mathcal{N}$ denote the set of nodes including both physical transportation and dummy nodes, $\mathcal{A}$ denote the set of links, $\mathcal{T}$ denote the set of time stamps covering all vehicles' time horizons, $\mathcal{W}$ denote the set of all feasible passenger carrying states, and $L_{i,t,w}$ denote the label of vertex i,t,w and term "pred" stands for the predecessor. The algorithm described below uses forward dynamic programming:

// time-dependent forward dynamic programming algorithm

for each vehicle $v \in (V \cup V_*)$ **do**

begin

// initialization

$L_{.,.,.} := +\infty$;

node pred of vertex $_{.,.,.} := -1$;

time pred of vertex $_{.,.,.} := -1$;

state pred of vertex $_{.,.,.} := -1$;

// vehicle v starts its route from the empty state at its origin at the earliest departure time

$L_{o_{v'},\ e_v,w0} := 0$;

```
for each time t∈ev,lv do
begin
    for each link (i,j) do
    begin
        for each state w do
        begin
        derive downstream state w′ based on the possible state transition on link (i,j);
        derive arrival time s=t+TT(i,j,t);
        if (L(i,t,w) +ξv,i,j,t,s, w,w′ <L(j,s,w′)) then
            begin
            L(j,s, w′) := L(i,t,w) + ξv,i,j,t,s, w,w′ ; //label update
            node pred of vertex j,s,w′ :=i;
            time pred of vertex j,s,w′ :=t;
            state pred of vertex j,s,w′ :=w;
            end;
        end;
    end;
end;
```

Let's define $|\mathcal{T}|$, $\mathcal{A}$, $\mathcal{W}$ as the number of time stamps, links, and passenger carrying states, respectively. Therefore, the worst-case complexity of the DP algorithm is $\mathcal{V}|\mathcal{T}|\mathcal{A}\mathcal{W}$, which can be interpreted as the maximum number of steps to be performed in this algorithm in this four-loop structure, corresponding to the sequential loops over vehicle, time, link, and starting carrying state dimensions. It should be remarked that the ending state w' is uniquely determined by the starting state w and the related link (i,j) depending on its service type: pickup, delivery, or pure transportation. In a transportation network, the size of links is much smaller than the counterpart in a complete graph, that is, $\mathcal{A} \ll \mathcal{N}\mathcal{N}$; in fact, the typical out-degree of a node in transportation networks is about 2-4.

Table 6 shows detailed comparisons between the existing DP-based approach (Psaraftis, 1983 and Desrosiers et al. 1986) and our proposed method. We guarantee the completeness of state representation. The state representation of Psaraftis (1983), $(L,k1, k2, ..., kn)$, consists of L, the location currently being visited, and ki, the status of passenger i. In this representation, $L=0$, $L=i$, and $L=i+n$ denote starting location, passenger i's origin, and passenger i's destination, respectively. In addition, the status of passenger i is chosen from the set $\{1,2,3\}$, where 3 means

passenger *i* is still waiting to be picked up, 2 means passenger *i* has been picked up but the service has not been completed, and 1 means passenger *i* has been successfully delivered. Desrosiers et al. (1986) use state representation *(S,i)*, where *S* is the set of passengers' origin, *{1, ..., n}*, and destination, *{n+1, ...2n}*. State *(S,i)* is defined if and only if there exists a feasible path that passes through all nodes in *S* and ends at node *i*. In fact, our time-dependent state *(w,i, t)*, which is jointly defined by three indexes: the status of customers, the current node being visited, and *(iii)* the current time, is more focused on the time-dependent current state at exact time stamp *t*, while *(L,k1, k2, ..., kn)* and *(S,i)* representations use a time-lagged time-period-based state representation to cover complete or mutually exclusive states from time *0* to time *t*.

Table 6. Comparison between existing DP based approach and the method proposed in this research.

Features	Existing DP based approach		DP proposed in this research
	Psaraftis (1983)	Desrosiers et al. (1986)	
Type of problem	Single vehicle, Many-to-many, Single depot	Single vehicle, Many-to-many, Single depot	Multiple vehicle, Many-to-many, Multiple depot
Network	Consists of passengers' origin and destination nodes and the vehicle depot	Consists of passengers' origin and destination nodes and the vehicle depot	Consists of transportation nodes, passengers' origin and destination, and vehicles' depots
Time-dependent link travel time	No	No	Yes
Objective function	Minimize route duration	Minimize total distance traveled	Minimize total routing cost consisting of transportation and waiting costs
State	state-space *(L,k1, k2, ..., kn)*	state-space *(S,i)*	state-space-time *(w,i,t)*
Stage	Node index	Node index	Time index
States reduction due to the vehicle capacity and time windows	Yes	Yes	Yes

We come back to the illustrative example presented in section 3.1. Let's assume the routing cost of a transportation or service arc traversed by a physical vehicle is $22/hr, while the routing cost of a transportation or service arc traversed by a virtual vehicle is $50/hr. Moreover, assume that the waiting cost of a physical vehicle is $15/hr, while the waiting cost of a virtual vehicle is assumed to be $0/hr. Table 7 shows how the label of each vertex is calculated by the DP solution algorithm presented above. Note that *w0*, *w1*, *w2*, and *w3* are passenger carrying states *[_ _]*, *[p1 _]*, *[p1 p2]*, and *[_ p2]*, respectively. For instance, according to Fig. 2, traveling from node 4 to node 2 takes 2 minutes. Since the number written on each link denotes the time-invariant travel time *TT(i,j)*, we can conclude that travel time for link starting at time stamp is also 2 minutes. To update the label corresponding to node 2, it is sufficient to calculate the routing cost of the stated arc in terms of dollars which can be obtained by $hr) and add it to the current label of node 4 which is 0.37. Therefore, the updated label for node 2 will be 1.1. Similarly, we can calculate the routing cost of a waiting link *(o2,o2)* starting at time stamp *t=7* by *160×15*$hr.

Table 7. State-space-time trajectory for ride-sharing service trip with node sequence *(o1',4, 2, o1, 2,o2,2, 5, 6, 3, d1,3, d2,3, 1, d1')*.

Time index	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16	18	19	20
Node index		4	2		2			2	5	6	3		3		3	1		
State index	*w0*	*w0*	*w0*	*w1*	*w1*	*w2*	*w2*	*w2*	*w2*	*w2*	*w2*	*w2*	*w3*	*w3*	*w0*	*w0*	*w0*	*w0*
Cost	0.0	.37	.73	.37	.37	.37	.25	.37	.37	.37	.37	.37	.37	.37	.37	.73	.37	0.0
Cumulative cost	0.0	.37	1.1	1.47	1.84	2.21	2.46	2.83	3.2	3.57	3.94	4.31	4.68	5.05	5.42	6.15	6.52	6.52

6.2.Lagrangian Relaxation-based solution procedure

In this section, we describe the Lagrangian relaxation (LR) solution approach implemented to solve the time-dependent least cost path problem presented in section 5. According to Eq. (14), $\xi_{v,i,j,t,s,\ w,w'}$ is only updated for $\forall v,i,j,t,s,w,w' \in \Psi_{p,v}$. Table 8 lists the notations for the sets, indices and parameters required for the Lagrangian relaxation algorithm.

Table 8. Notations used in LR algorithm.

Symbol	Definition
$\lambda_{k(p)}$	Lagrangian relaxation multiplier corresponding to the passenger *p*'s pick-up request constraint at iteration
$\xi_{v,i,j,t,s,\ w,w'}$	Modified routing cost of arc $i,j,t,s,\ w,w'$ after introducing Lagrangian multipliers
k	Iteration number
Y	Set of vectors $y_{v,i,j,t,s,\ w,w'}$

LBk	Global lower bound for the object function value at iteration
UBk	Global upper bound for the object function value at iteration
$YLBk$	Set of vectors Y in the lower bound solution at LR iteration k
$YUBk$	Set of vectors Y in the upper bound solution at LR iteration k
θk	Step size at iteration
$LB*$	Best global lower bound of the objective function value
$UB*$	Best global upper bound of the objective function value
$Y*$	Best solution derived from best lower bound
VOT	The amount of money (in terms of dollars) passenger p offers to be served

The Lagrangian relaxation algorithm can be described as follows:

// Lagrangian relaxation algorithm

// step 0. initialization

- set iteration $k=0$;
- initialize $YLB0$, $YLB0$, $Y*$, and $\lambda 0(p)$ to zero; $\theta 0p$ to VOT; $LB*$ to $-\infty$; and $UB*$ to $+\infty$;
- define a termination condition such as if k becomes greater than a predetermined maximum iteration number, or if the relative gap percentage between $LB*$ and $UB*$ becomes less than a predefined gap;

while termination condition is false, for each LR iteration k **do**

begin

- reset the visit count for each arc $v,i,j,t,s,w,w' \in \Psi p,v$ to zero; // $v \in (V \cup V*)$;
- initialize LBk and UBk to 0;

// step 1. generating LBk

// step 1.1. least cost path calculation for each vehicle sub-problem

for each vehicle $v \in (V \cup V*)$ **do**

begin

// input: $\xi v,i,j,t,s,w,w'$

- compute time dependent least cost state-space-time path for vehicle v by calling time-dependent DP;
- update the visit count for each arc $v,i,j,t,s,w,w' \in \Psi p,v$;

// output: $YLBk$

end;

// step 1.2. update $LB*$

- update LBk by the new value of dualized Lagrangian function L obtained from $YLBk$;
- update $LB*$ by $max(LBk,current\ LB*)$ and $Y*$ by its corresponding solution;

// step 1.3. sub-gradient calculation

- calculate the total number of visits of passenger p's origin by expression (15);

$$v,i,j,t,s,w,w' \in \Psi p,v\ y v,i,j,t,s,w,w' \quad (15)$$

- compute sub-gradients by Eq. (16);

$$\nabla L\lambda k(p) = v \in (V \cup V*) v,i,j,t,s,w,w' \in \Psi p,v\ y v,i,j,t,s,w,w' - 1 \quad (16)$$

- update arc multipliers by Eq. (17);

$$\lambda k+1(p) = \lambda k(p) + \theta kp \nabla L\lambda k(p) \text{ for } \forall p \quad (17)$$

- update arc cost $\xi v,i,j,t,s,\ w,w'$ for each arc $v,i,j,t,s,w,w' \in \Psi p,v$ by Eq. (18);

$$\xi v,i,j,t,s,\ w,w' = cv,i,j,t,s,\ w,w' + \lambda k+1(p) \quad (18)$$

- update step size by Eq. (19);

$$\theta k+1p = \theta 0p\ k+1 \quad (19)$$

// Step 2. generating UBk

// step 2.1. adopt the solution $YLBk$ from the LBk: $YUBk = YLBk$

for each passenger $p \in P$ **do**

begin

// route virtual vehicle $vp*$ to serve unserved passenger p

if passenger p is not served by any vehicle (physical or virtual) **then**

begin

for arc $vp*,i,j,t,s,w,w' \in \Psi p,vp*$ **do**

begin

- set $cvp*,i,j,t,s,\ w,w'$ temporarily to $-M$; // M is chosen a very large positive constant number in order to route virtual vehicle $vp*$ to certainly serve passenger p;
- compute time dependent least cost path for vehicle $vp*$ by calling time-dependent DP;
- add the virtual vehicle in to solution $YUBk$

end;

end;

end;

// step 2.2. update UBk

- update UBk by the new value of primal objective function Z obtained from $YUBk$;

  // the original value of $cv,i,j,t,s,\ w,w'$ should be considered in updating UBk

// step 2.3. update $UB*$

- $UB* = minUBk, current\ UB*$;
- find the relative gap percentage between $LB*$ and $UB*$ by $UB*-LB*UB*\times 100$;
- $k=k+1$;

end;

Regarding relative gap properties, after adopting the solution $YLBk$ from the LBk, we can find three different following cases.

(i) All passengers are assigned to the physical vehicles perfectly. This case is the ideal case which shows all demands have been satisfied by the physical vehicles since the total number of available vehicles has been enough to serve all requests. In this case, since each passenger has been matched to a vehicle (Eq. (5) has been met perfectly), we expect the relative gap value to be zero.

(ii) Each passenger is assigned to a vehicle; however, there are some passengers who have been assigned to the virtual vehicles. Similar to case *(i)*, since each passenger has been matched to a vehicle (Eq. (5) has been met perfectly), we expect the relative gap value to be zero.

(iii) There might be some passengers who are not assigned to any vehicle at all (neither a physical nor a virtual vehicle), or there might be some passengers who are served by more than one vehicle. In this case, we set $cvp*,i,j,t,s,\ w,w'$ temporarily to $-M$. M is chosen a very large positive constant number in order to route virtual vehicle $vp*$ to certainly serve passenger p. In this case, the virtual vehicle is dispatched to serve the corresponding passenger; however, when the virtual vehicle drops off the passenger, it should perform a deadheading trip with significantly high cost from the passenger's destination to its depot (the passenger's origin) which causes the quite large gap between the corresponding lower bound and upper bound.

6.3.Search Region Reduction

In this section, we describe how to reduce the search region by the aid of some simple heuristics in which some rational rules are applied.

Let EDT, LDT, EAT, and LAT denote the earliest departure time from origin, latest departure time from origin, earliest arrival time to destination, and latest arrival time to destination, respectively. In addition, let $TTSPx \rightarrow y$ denote the travel time corresponding to the shortest path from node x to node y.

Rule 1. No overlapping time windows: The first rational rule is that if $LATp1<EDT(p2)$, then passenger $p1$ and $p2$'s ride-sharing is impossible, or in other words, passenger $p1$ and $p2$ cannot be in the same vehicle at the same time. Therefore, all possible passenger carrying states in which both $p1$ and $p2$ are present in the same vehicle at the same time can be eliminated. Fig. 8 illustrates an example of two passengers whose ride-sharing is impossible due to no overlapping time windows.

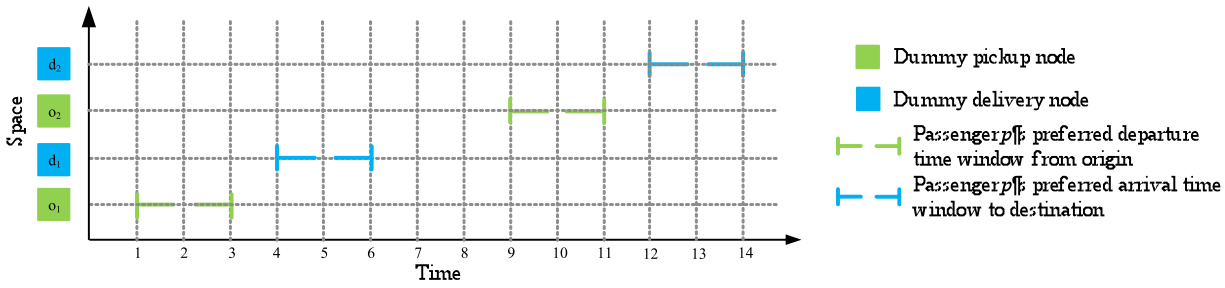


Fig. 8. Illustration of the first rational rule for search region reduction.

Rule 2. Travel time is insufficient: The second rational rule can be stated as follows: if $LDTp2-EDTp1< TTSPop1 \rightarrow op2$ & $LDTp1-EDTp2<TTSPop2 \rightarrow op1$, then passenger $p1$ and $p2$ cannot be in the same vehicle at a time. It means that if the maximum time a vehicle can have to go from passenger $p1$'s origin to $p2$'s origin, $LDTp2-EDTp1$, is less than the total travel time corresponding to the shortest path from $op1$ to $op2$, and also if the maximum time a vehicle can have to go from passenger $p2$'s origin to $p1$'s origin, $LDTp1-EDTp2$, is less than the total travel time corresponding to the shortest path from $op2$ to $op1$, then passenger $p1$ and $p2$'s ride-sharing is impossible. Similarly, if $LATp2-EATp1< TTSPdp1 \rightarrow dp2$ & $LATp1-EAT(p2)<TTSPdp2 \rightarrow dp1$, then passenger $p1$ and $p2$'s ride-sharing is impossible. The total number of passenger carrying states is dramatically decreased via this rule. Fig. 9 illustrates the second rule by an example. Suppose two requests with two origin-destination pairs should be served by a vehicle. Fig. 8(a) illustrates transportation network with the corresponding dummy nodes and time windows. According to the Fig. 8(a), $TTSPop1 \rightarrow op2$ and $TTSPop2 \rightarrow op1$are 5

and 6, respectively. Since *6−4<5 & 5−4<6*, then passenger *p1* and *p2*'s ride-sharing is impossible.

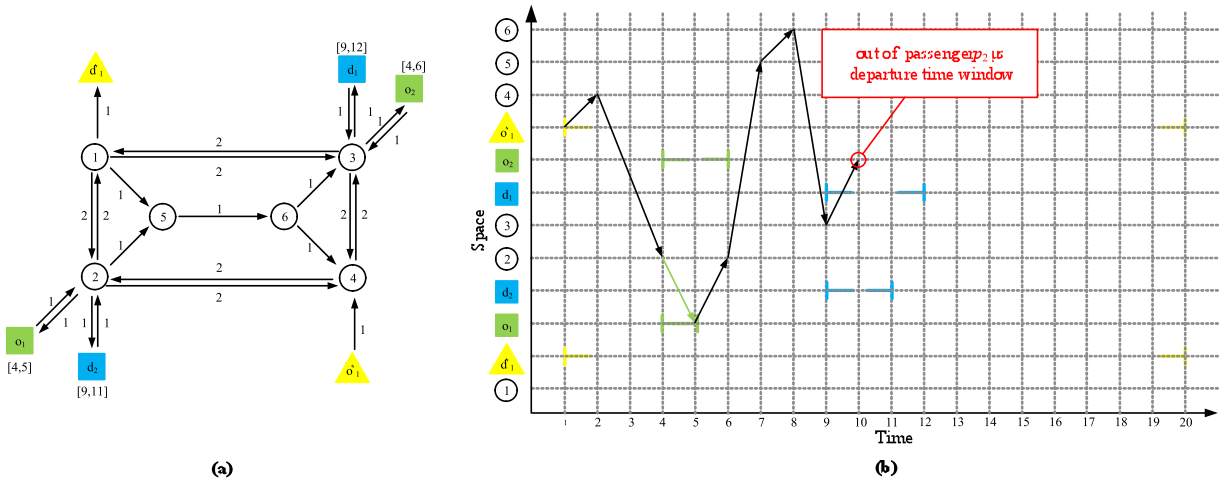


Fig. 9. Illustration of the second rational rule for search region reduction; (a) transportation network with the corresponding dummy nodes and time windows; (b) vehicle 1's space-time network.

Rule 3. A node is too far away from the vehicle starting or ending depot: The third rational rule is stated as follows: if *(TTSPov→x +TTSPx→dv)>(LAT(v)−EDT(v))*, then vehicle v does not have enough time to visit node x in its time horizon; therefore, node x is not accessible for vehicle v and should not be considered in vehicle v's search region. Note that node x can be any physical or dummy node. Fig. 10 illustrates the third rule by an example. Suppose a passenger with an origin-destination pair should be served by a vehicle. Fig. 10(a) illustrates transportation network with the corresponding dummy nodes and time windows. Fig. 10(b) shows that passenger *p1*'s origin, *o1*, is not accessible for the vehicle.

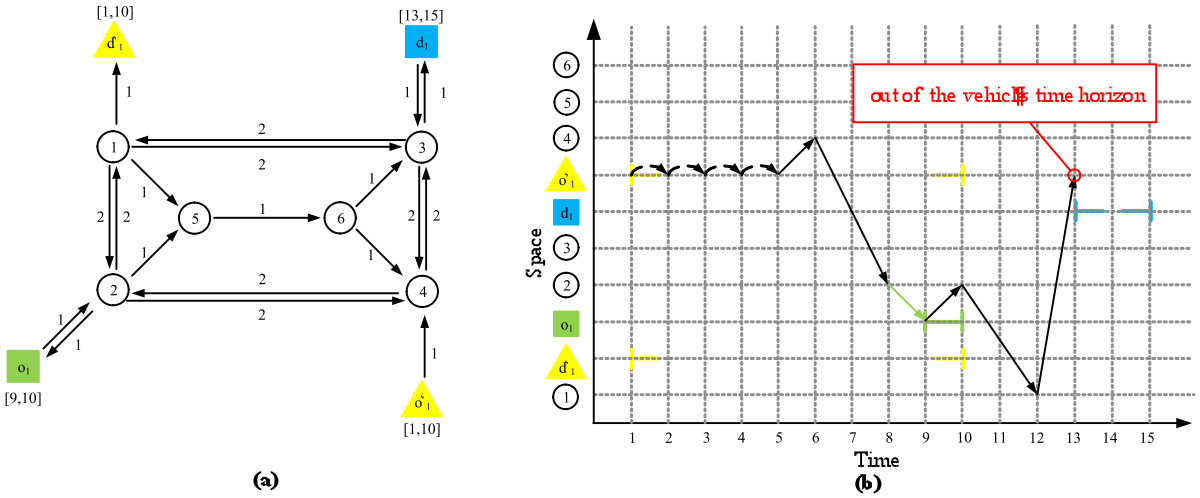


Fig. 10. Illustration of the third rational rule for search region reduction; (a) transportation

network with the corresponding dummy nodes and time windows; (b) vehicle 1's space-time network.

The first three rules are hard rules at which we are able to eliminate some vertices in the state-space-time networks. The forth heuristic is the way of estimating the search region reduction ratio. Let path α be the longest possible path in vehicle v's state-space-time networks with total travel time $\tau\alpha$. Let denote the middle point of passenger 's departure time window. Therefore, . Let's assume that , the middle point of a passenger's departure time window, is a random variable uniformly distributed in vehicle v's time horizon with $LAT(v)-EDT(v)$ length. It may be reasonable to assume that if $>\tau\alpha$, then passenger $p1$ and $p2$ cannot be in the same vehicle at a time. We use an example to show that this rule can reduce the search region considerably. Assume vehicle v's time window is [0, 240], and is a random variable uniformly distributed in vehicle v's time horizon $[0,240]$. Let's assume $\tau\alpha=60$ minutes. The probability of having two passengers who share their ride with each other can be calculated by finding the $Prob\leq 60\ minutes$, where and are randomly generated from [0, 240]. This probability equals to . This can be shown with the following derivation. The shaded area in in Fig. 11 shows $Prob\leq 60$.

$$Prob\leq 60 = Prob\leq 60$$

$$Prob\leq 60 = 1-Prob<-60 + Prob>60$$

$$= 1-180\times 180 2240\times 240 + 180\times 180 2240\times 240 =$$

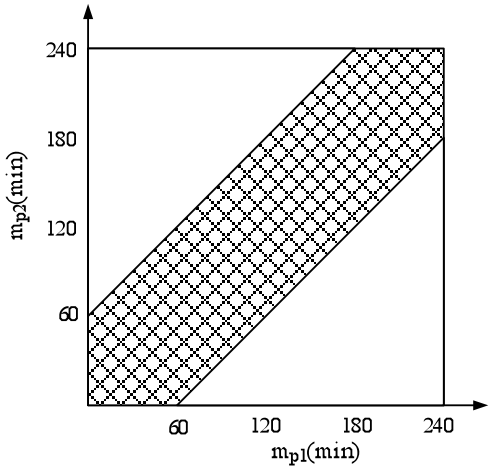


Fig. 11. The probability of having two passengers who share their ride with each other where and are uniformly distributed in [0, 240]. Note that $\tau\alpha=60$ min.

Therefore, by considering this practical rule in this example, we can reduce the total number of passenger carrying states in which two passengers share their ride with each other by more than half. By considering this rational rule, calculating the probability of having more than two passengers at the same time in vehicle v is more complicated, but at least we know that the

probability of having k number of passengers ($k>2$) who may share their ride with each other is certainly less than .

7. Computational Results

The algorithms described in this research were coded in C++ platforms. The experiments were performed on an Intel Workstation running two Xeon E5-2680 processors clocked at 2.80 GHz with 20 cores and 192GB RAM running Windows Server 2008 x64 Edition. In addition, parallel computing and OpenMP technique are implemented for step 2.1 in the Lagrangian relaxation algorithm. In this section, we initially examine our proposed model on a six-node transportation network followed by the medium-scale and large-scale transportation networks, Chicago and Phoenix, to demonstrate the computational efficiency and solution optimality of our developed algorithm. The scenarios and test cases are randomly generated in those transportation networks.

As we mentioned in section 5.1, it is assumed that the routing cost of a transportation or service arc traversed by a physical vehicle is $22/hr, while the routing cost of a transportation or service arc traversed by a virtual vehicle is $50/hr. Moreover, the waiting cost of a physical vehicle is $15/hr, while the waiting cost of a virtual vehicle is assumed to be $0/hr. The value of VOT is also assumed to be $10 for all passengers.

7.1.Six-node Transportation Network

Initially, we test our algorithm on the six-node transportation network illustrated in Fig. 2(a) for six scenarios. Table 9 shows these scenarios with various number of passengers and vehicles, origin-destination pairs, and passengers' departure and arrival time windows. Then, we will examine the results corresponding to each scenario individually. Terms "TW" and "TH" stands for time window and time horizon, respectively.

Scenario I. Two passengers are served by one vehicle, where passengers have different origin-destination pairs with overlapping time windows. In this case, the vehicle serves both passengers in their preferred time windows through ride-sharing mode.

Scenario II. Two passengers with different origin-destination pairs are served by one vehicle; however, unlike in scenario I, passengers could not share their ride with each other due to their time windows. In this case, the vehicle may wait at any node to finally serve both passengers.

Scenario III. Two passengers with different origin-destination pairs and one vehicle are present in the system; however, due to the passengers' overlapping time windows, serving both passengers by one vehicle is impossible. Therefore, the driver would prefer to transport a passenger incurring the least cost. In this case, passenger $p1$ is selected to be served.

Scenario IV. Two passengers with different origin-destination pairs and two vehicles are present in the system and, due to the passengers' and vehicles' time windows, $p1$ is assigned to $v1$ and $p2$ is assigned to $v2$.

Scenario V. Three passengers are served by one vehicle, where passengers have different origin-destination pairs with overlapping time windows. In this case, the vehicle serves all passengers in their preferred time windows through ride-sharing mode.

Scenario VI. One passenger and two vehicles are present in the system. In this case, two vehicles compete for serving the passenger. Ultimately, the vehicle whose routing is less costly wins the competition and serves the passenger.

Table 9. Six scenarios with various number of passengers and vehicles, origin-destination pairs, and passengers' departure and arrival time windows.

Scenario	I	II	III	IV	V	VI
Number of passengers	2	2	2	2	3	1
Number of vehicles	1	1	1	2	1	2
o1	Node 2	Node 2	Node 2	Node 2	Node 2	Node 2
d1	Node 6	Node 6	Node 1	Node 1	Node 3	Node 6
o2	Node 5	Node 5	Node 3	Node 3	Node 5	-
d2	Node 3	Node 3	Node 6	Node 6	Node 3	-
o*3*	-	-	-	-	Node 6	-
d*3*	-	-	-	-	Node 1	-
o1′	Node 4	Node 4	Node 4	Node 2	Node 4	Node 4
d1′	Node 1	Node 1	Node 1	Node 1	Node 1	Node 1
o2′	-	-	-	Node 3	-	Node 6
d2′	-	-	-	Node 6	-	Node 1
TWo1	[5, 7]	[5, 7]	[4, 5]	[4, 5]	[4, 7]	[4, 7]
TWd1	[9, 12]	[9, 12]	[8, 10]	[8, 10]	[13, 16]	[9, 12]
TWo2	[8, 10]	[16, 19]	[3, 5]	[4, 6]	[7, 10]	-
TWd2	[11, 14]	[21, 24]	[11, 14]	[11, 14]	[14, 18]	-
TWo*3*	-	-	-	-	[10, 13]	-
TWd3	-	-	-	-	[19, 23]	-
THv1	[1, 30]	[1, 30]	[1, 30]	[1, 30]	[1, 30]	[1, 30]
THv2	-	-	-	[1, 30]	-	[1, 30]

Table 10 shows the results corresponding each scenario. Fig. 12 also presents the vehicle routing corresponding each scenario.

Table 10. Results obtained from testing our algorithm on the six-node transportation network for six scenarios.

iteration k	$LB*$	$UB*$	gap%	vehicles assigned to $p1$, $p2$, and $p3$	$\lambda k(p1)$	$\lambda k(p2)$	$\lambda k(p3)$
Scenario I. Two passengers are served by one vehicle through ride-sharing mode.							
1	1.47	5.75	74.5%	$v1$, $v1$, -	10	10	-
2	1.47	5.75	74.5%	$v1$, $v1$, -	5	5	-
3	5.75	5.75	0.0%	$v1$, $v1$, -	5	5	-
Scenario II. Two passengers are served by one vehicle (not through ride-sharing mode).							

1	1.47	7.22	79.68%	$v1$, $v1$, -	10	10	-
2	5.55	7.22	23.10%	$v1$, $v1$, -	5	5	-
3	7.22	7.22	0.0%	$v1$, $v1$, -	5	5	-
Scenario III. Two passengers and one vehicle; one passenger remains unserved.							
1	1.47	10.43	85.94%	$v1$, $v2_*$, -	10	10	-
2	7.1	10.43	31.95%	$v1$, $v2_*$, -	5	10	-
3	10.43	10.43	0.0%	$v1$, $v2_*$, -	5	10	-
Scenario IV. Two passengers and two vehicles; each vehicle is assigned to a passenger							
1	2.2	6.13	64.13%	$v1$, $v2$, -	10	10	-
2	2.2	6.13	64.13%	$v1$, $v2$, -	5	5	-
3	6.13	6.13	0.0%	$v1$, $v2$, -	5	5	-
Scenario V. Three passengers are served by one vehicle through ride-sharing mode							
1	1.47	6.97	78.95%	$v1$, $v1$, $v1$	10	10	10
2	1.47	6.97	78.95%	$v1$, $v1$, $v1$	5	5	5
3	6.97	6.97	0.0%	$v1$, $v1$, $v1$	5	5	5
Scenario VI. Two vehicles compete for serving a passenger							
1	2.57	5.13	50.0%	$v1$, -, -	10	-	-
2	2.63	5.13	48.70%	$v1$, -, -	10	-	-
3	5.13	5.13	0.0%	$v1$, -, -	10	-	-

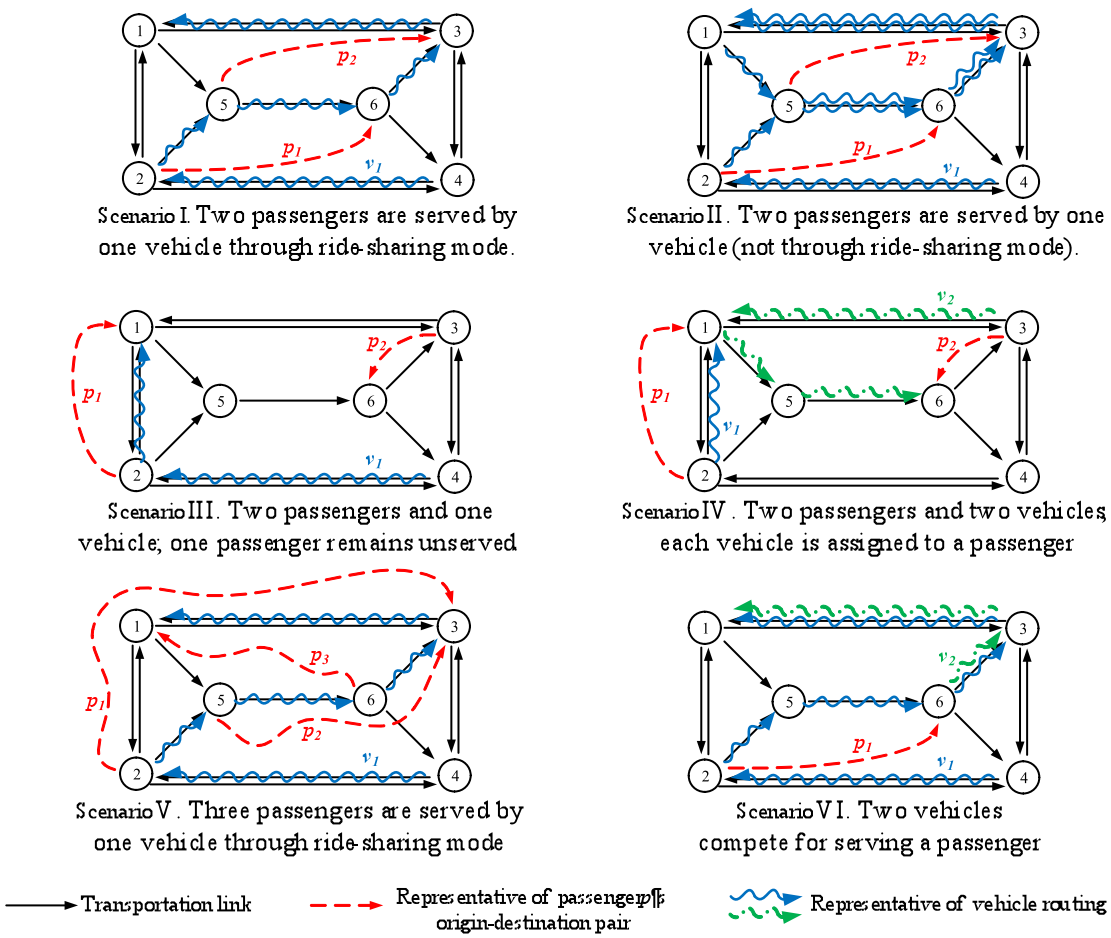


Fig. 12. The vehicle routing corresponding each scenario.

We increase the number of passengers and vehicles to show the computational efficiency and solution optimality of our developed algorithm. Table 11 shows the results for the six-node transportation network when the number of passengers and vehicles have been increased.

Table 11. Results for the six-node transportation network.

Test case number	Number of iterations	Number of passengers	Number of vehicles	$LB*$	$UB*$	Gap (%)	Number of passengers not served	CPU running time (sec)
1	30	6	1	15.83	15.83	0.00%	0	5.94
2	30	12	2	33.17	33.17	0.00%	0	12.02
3	30	24	4	61.67	65.33	5.61%	0	30.97

We explain the pricing mechanism in this algorithm via test case 1 with 6 passengers and 1 vehicle. Fig. 13 shows $\lambda k(pi)$, $i=1,2,..,6$, along 30 iterations. It is clear that each passenger's Lagrangian multiplier ultimately converges to a specific value. This value can be literally

interpreted as the passenger *p*'s service price. Through the pricing mechanism of this algorithm, the provider would be able to offer a reasonable bid to its customers to be served.

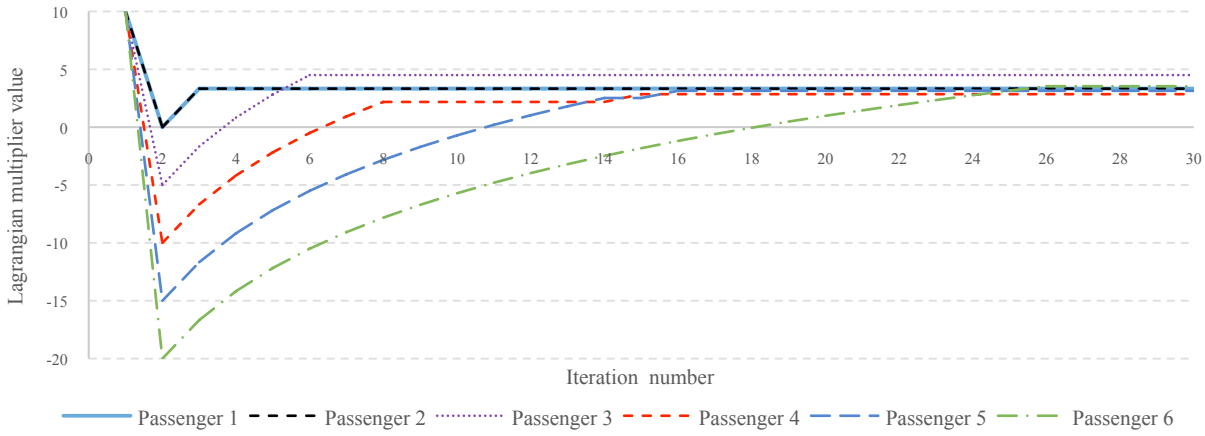
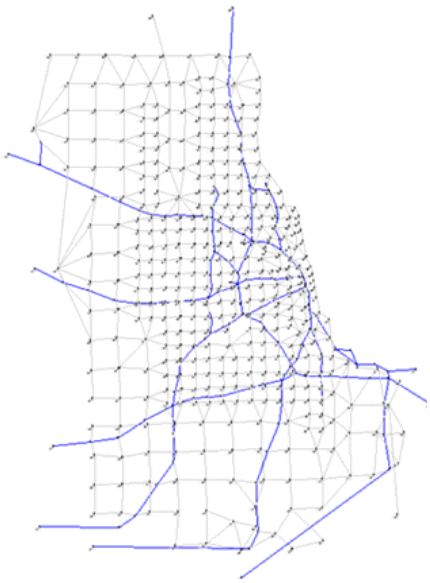


Fig. 13. Lagrangian multipliers along 30 iterations in test case 1 for the six-node transportation network.

7.2.Medium-scale and Large-scale Networks

In our computational experiments for the medium-scale and large-scale networks, for simplicity, we assume that each passenger has a fixed departure time (the earliest and latest departure time are the same). In addition, we assume that no passenger has a preferred time window for arrival to his destination. Tables 12 and 13 show the results for the Chicago transportation network, shown as Fig. 14(a) with 933 nodes and 2,967 links, and the Phoenix transportation network, as shown in Fig. 14(b) with 13,777 nodes and 33,879 links, respectively.



(a) Chicago sketch network

(b) Phoenix metropolitan regional network

Fig. 14. Medium and large-scale transportation networks for computational performance testing.

Note that we generally run the algorithm for a fixed number of iterations; however, the algorithm may converge in less number of iterations. Fig. 15 shows the gap percentage along 20 iterations corresponding each test case.

Table 12. Results for the Chicago network with 933 transportation nodes and 2,967 links.

Test case number	Number of iterations	Number of passengers	Number of vehicles	LB_*	UB_*	Gap (%)	Number of passengers not served	CPU running time (sec)
1	20	2	2	108.43	108.43	0.00%	0	17.43
2	20	11	3	352.97	352.97	0.00%	0	91.87
3	20	20	5	616.66	626.18	1.52%	1	327.51
4	20	46	15	1586.81	1664.07	4.64%	2	4681.52
5	20	60	15	1849.98	1878.55	1.52%	3	7096.50

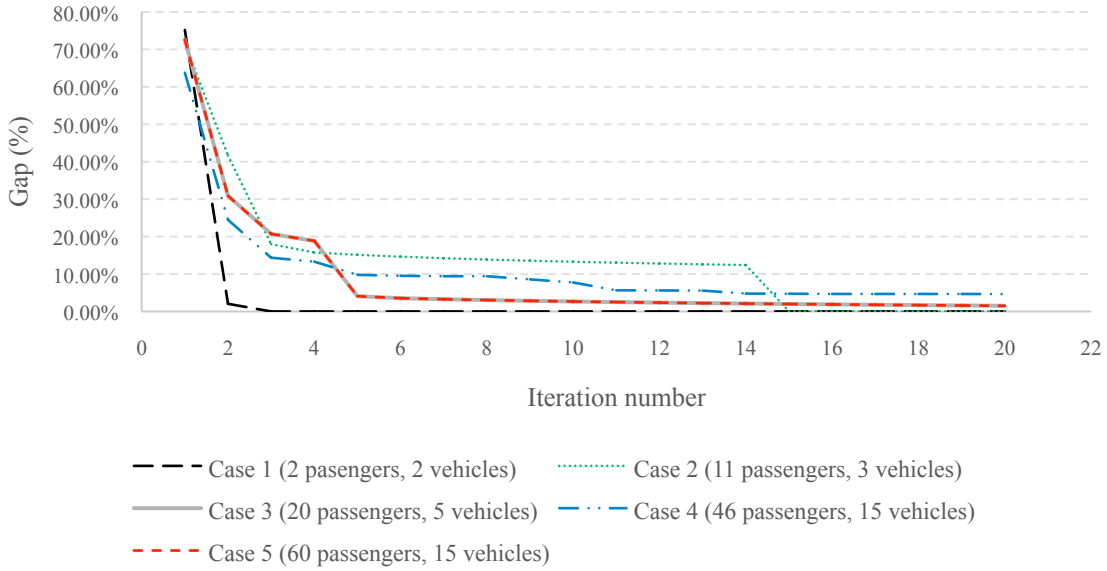


Fig. 15. Gap percentage along 20 iterations corresponding each test case in Chicago network.

As you can see in Fig. 15, after 10-15 iterations, the sub-gradient algorithm is typically able to converge to a small gap (about 5%) for the Chicago Network.

Table 13. Results for the Phoenix network with 13,777 transportation nodes and 33,879 links.

Test case number	Number of iterations	Number of passengers	Number of vehicles	LB_*	UB_*	Gap (%)	Number of passengers not served	CPU running time (sec)
1	6	4	2	70.95	70.95	0.00%	0	110.39
2	6	10	5	191.55	207.05	7.49%	1	398.37
3	6	20	6	310.37	310.37	0.00%	0	1323.18

4	6	40	12	622.23	622.23	0.00%	0	3756.505
5	6	50	15	784.07	784.07	0.00%	0	6983.189

7.3.Optimum Number of Self-driving Cars

Now it is time to discuss about the key question we initially asked: How many self-driving cars a city needs to support the overall transportation activity demand, at different levels of coordination and pre-trip scheduling? To answer this question, it is sufficient to assume that there are large number of vehicles available in the depots. In other words, the number of vehicles available in the depots should be much more than enough. Then, after finding the optimal solution by considering this assumption, we may find a number of vehicles which are not assigned to any transportation request. It is obvious that according to the results, these vehicles seems to be redundant. Therefore, to find how many self-driving cars a city needs to support the overall transportation activity demand, it is sufficient to subtract the number of vehicles which are not assigned to any transportation request from the total number of vehicles available at the depots. Table 14 shows the number of self-driving cars needed in the six-node transportation network.

Table 14. Results for the six-node transportation network.

Test case number	Number of iterations	Number of passengers	Number of vehicles	LB_*	UB_*	Gap (%)	Number of passengers not served	Number of SAVs needed	CPU running time (sec)
1	30	6	3	16.49	20.85	20.92%	1	1	1.07
2	30	12	4	33.01	42.63	22.58%	0	3	12.02

8. Conclusions

A new generation of transportation network companies uses mobile-phone-based platforms to seamlessly connect drivers to passengers from different origins to different destinations with specific, preferred departure or arrival times. Many relevant practical aspects need to be carefully formulated for real-world planning/dispatching system deployment, such as time-dependent link travel times on large-scale regional transportation networks, and tight vehicle capacity and passenger service time window constraints.

By reformulating the PDPTW through space time networks to consider time window requirements, our proposed approach can not only solve the vehicle routing and scheduling problem directly in large-scale transportation networks with time-dependent congestion, but also avoid the complex procedure to eliminate any sub-tour possibly existing in the optimal solution for many existing formulations. By further introducing virtual vehicle constructs, the proposed approach can fully incorporate the full set of interacting factors between passenger demand and limited vehicle capacity in this model to derive feasible solutions and practically important system-wide cost-benefit estimates for each request through a sub-gradient-based pricing method. This joint optimization and pricing procedure can assist transportation network service providers to quantify the operating costs of spatially and temporally distributed trip requests.

Future work will concentrate on the development of the model for the following cases: *(i)* Passengers may desire different ride-sharing capacities (i.e. a passenger may desire to share his ride with up to only one passenger, whereas the other passenger may have no restriction about the number of passengers which share their ride with him). *(ii)* A passenger may desire to be or not to be served by a particular vehicle. *(iii)* A transportation request could contain a group of passengers who have the same origin, while they may or may not have the same destination. Alternatively, a transportation request could contain a group of passengers who have the same destination, while they may or may not have the same origin. In this case, we are interested in adding dummy nodes corresponding to passengers' origins and destinations more wisely and efficiently.

9. Appendices

Appendix A: Description of the PDPTW in the Origin-Destination Network

Cordeau (2006) formulated the PDPTW on a network that is built based on demand request nodes and the links are defined as direct connections between pickup and delivery nodes (without explicitly considering transportation links or paths). For a systematic comparison, the following notation is adapted from Cordeau (2006).

Table A.1. Sets, indices and parameters used in Cordeau (2006) for the PDPTW.

Symbol	Definition
n	Number of passengers
P	Set of passengers' pickup nodes. $P=1, ..., n$
D	Set of passengers' delivery nodes. $D=\{n+1, ..., 2n\}$
0	Node representative of origin depot
$2n+1$	Node representative of destination depot
N	Set of passengers' pickup and drop-off nodes and vehicles' depots. $N=\{P, D, 0, 2n+1\}$
A	Set of arcs
G	Directed graph $G=N, A$
i	Passenger i's pickup node
$n+i$	Passenger i's delivery node
qi	Load at node i, ($i \in N$)
di	Service duration at node i, ($i \in N$)
ei	Earliest time at which service is allowed to start at node i, ($i \in N$)
li	Latest time at which service is allowed to start at node i, ($i \in N$)

(i,j)	Index of arc between adjacent nodes i and j
cij	Routing cost of arc (i,j)
tij	Travel time of arc (i,j)
V	Set of vehicles
v	Vehicle index
Qv	Capacity of vehicle v
Tv	Maximal duration of vehicle v's route
L	Maximum ride time of a passenger

Note that $q0=q2n+1=0$, $qi\geq 0$ for $(i=1, ..., n)$, and $qi=-qi-n$ $(i=n+1,..., 2n)$, and service duration $di\geq 0$ and $d0=d2n+1=0$. Time window ei,li is also specified either for the pickup node or for the drop-off node of a request, but not for both. The arc set is also defined as $A=i,j:i=0,j\in P$ or $i\in P\cup D, j\in P\cup D, i\neq j, i\neq n+j$ or $(i\in D,\ j=2n+1)$. The model uses three-index variables $xijv$ being equal to 1 if and only if vehicle v travels from node i to node j. Let Biv be the time at which vehicle v begins servicing node i and Qiv be the load of vehicle v upon departing from node i. Finally, for each passenger i, let Liv be the ride time of passenger i on vehicle v. The PDPTW can be formulated as follows:

$$Min \sum_{v\in V}\sum_{i\in N}\sum_{j\in N} cijv\,xijv \quad \text{(A.1)}$$

s.t.

$$\sum_{v\in V}\sum_{j\in N} xijv=1 \qquad \forall i\in P \quad \text{(A.2)}$$

$$\sum_{j\in N} xijv-\sum_{j\in N} xn+i,jv=0 \qquad \forall i\in P,\ v\in V \quad \text{(A.3)}$$

$$\sum_{j\in N} x0jv=1 \qquad \forall v\in V \quad \text{(A.4)}$$

$$\sum_{j\in N} xjiv-\sum_{j\in N} xijv=0 \qquad \forall i\in P\cup D,\ v\in V \quad \text{(A.5)}$$

$$\sum_{i\in N} xi,\,2n+1v=1 \qquad \forall v\in V \quad \text{(A.6)}$$

$$xijv\,Biv+di+tij\leq Bjv \qquad \forall i\in N,\ j\in N,\ v\in V \quad \text{(A.7)}$$

$$xijv\,Qiv+qj\leq Qjv \qquad \forall i\in N,\ j\in N,\ v\in V \quad \text{(A.8)}$$

$$Liv=Bn+iv-Biv+di \qquad \forall i\in P,\ v\in V \quad \text{(A.9)}$$

$$B2n+1v-B0v\leq Tv \qquad \forall v\in V \quad \text{(A.10)}$$

$$ei\leq Biv\leq li \qquad \forall i\in N,\ v\in V \quad \text{(A.11)}$$

$$ti,n+i\leq Liv\leq L \qquad \forall i\in P,\ v\in V \quad \text{(A.12)}$$

$$max0,qi\leq Qiv\leq minQv,\ Qv+qi \qquad \forall i\in N,\ v\in V \quad \text{(A.13)}$$

$$xijv\in 0,1 \qquad \forall i\in N,\ j\in N,\ v\in V \quad \text{(A.14)}$$

The objective function (A.1) minimizes the total routing cost. (A.2) guarantees that each passenger is definitely picked up. (A.2) and (A.3) ensure that each passenger's origin and destination are visited exactly once by the same vehicle. (A.4) expresses that each vehicle v starts its route from the origin depot. (A.5) ensures the flow balance on each node. (A.6) expresses that each vehicle v ends its route at the destination depot. (A.7) and (A.8) ensure the validity of the time and load variables. (A.9) defines each passenger's ride time. (A.10) to (A.13) impose maximal duration of each route, time windows, the ride time of each passenger, and capacity constraints, respectively. Since the non-negativity of the ride time of each passenger guarantees that node i is visited before node $n+i$, (A.12) also functions as precedence constraints.

Apprendix B: Learning Documents

To help students understand the impact of traffic propagation under different scenarios, we have prepared a learning document about using simulation based traffic impact analysis as training material and user guides for undergraduate and graduate students interested in this subject. The document is Lesson 6.1 Understand Traffic Congestion Propagation, available at www.learningtransportation.org. The 37 pages of learning documents can be found at https://docs.google.com/document/d/1b0lss-F1fSyz5T4L7LpOMOvD29PHcC8JyCpjKopK2MQ/edit#

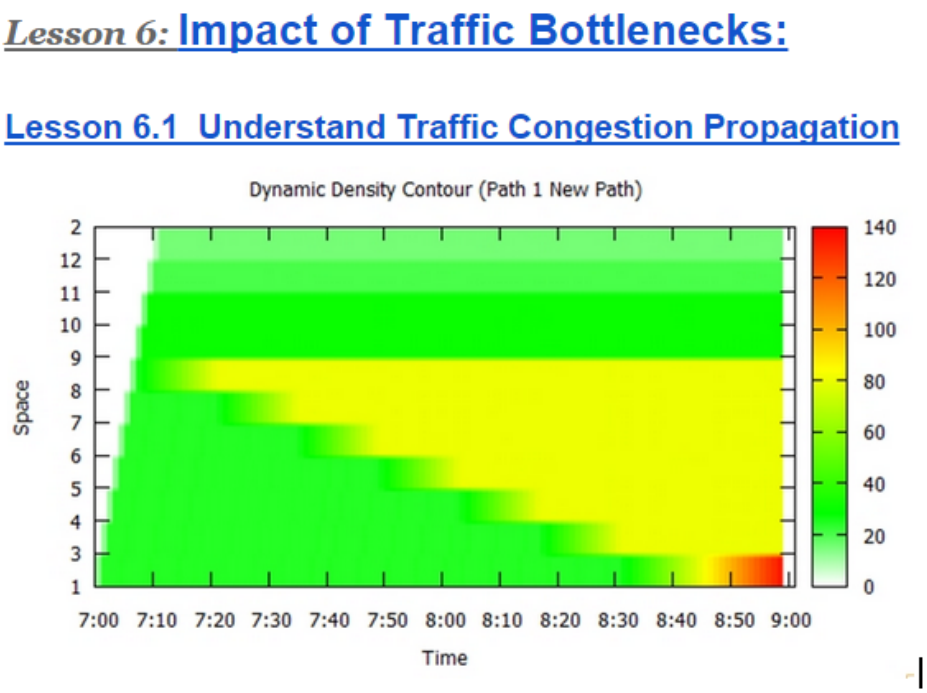


Fig. B.1. Screenshots of open learning documents for students to understand traffic congestion propagation.

Contents

Fig. B.2. Table of content for students to understand traffic congestion propagation.

9. References

Alfa, A. S. (1986). Scheduling of vehicles for transportation of elderly. *Transportation Planning and Technology*, 11, 203–212.

Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2) 414-426.

Bell, W., Dalberto, L., Fisher, M. L., Greenfield, A., Jaikumar, R., Kedia, P., Mack, R., and Prutzman, P. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13, 4–23.

Bodin, L.D. and Sexton, T. (1986). The multi-vehicle subscriber dial-a-ride problem. *TIMS Studies in the Management Sciences*, 22, 73–86.

Bramel, J., and Simchi-Levi, D. (1995). A location based heuristic for general routing problems. *Operations Research*, 43, 649–660.

Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J. (2000). Parallel Programming in OpenMP. Morgan Kaufmann.

Christiansen, M. (1999). Decomposition of a combined inventory routing and time constrained ship routing problem. *Transportation Science*, 33, 3–16.

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

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3) 573-586.

Cullen, F., Jarvis, J., and Ratliff, D. (1981). Set partitioning based heuristics for interactive routing. *Networks*, 11, 125–144.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis. F. (2002). The VRP with pickup and delivery. In Toth, P. and Vigo, D. editors, The Vehicle Routing Problem, *SIAM Monographs on Discrete Mathematics and Applications*, Chapter 9, SIAM, Philadelphia, 225–242.

Desrosiers, J., Dumas, Y., and Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6, 301-325.

Diana, M. and Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 38, 539–557.

Dumas, Y., Desrosiers, J., and Soumis. F. (1989). Large scale multi-vehicle dial-a-ride problems. Technical Report Cahiers du GERAD G–89–30, École des Hautes Études Commerciales, Montréal, Canada.

Dumas, Y., Desrosiers, J., and Soumis. F. (1991). The pickup and delivery problem with time windows. *European Journal of Operations Research*, 54, 7–22.

Fisher, M. L., Greenfield, A., Jaikumar, R., and Lester, J. (1982). A computerized vehicle routing application. *Interfaces*, 12, 42–52.

Fisher, M. L., and Rosenwein, M. B. (1989). An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistic Quarterly*, 35, 27–42.

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

Gendreau, M., Guertin, F., Potvin, J.-Y., and Séguin, R. (1998). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Technical Report CRT–98–10, Centre de recherche sur les transports, Université de Montréal, Canada.

Hosni, H., Naoum-Sawaya, J., Artail, H., (2014). The shared-taxi problem: formulation and solution methods. *Transportation Research Part B: Methodological*, 70, 303–318.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M. M., and Villeneuve, D. (1995). A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science*, 29, 63–78.

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

Lu, Q. and Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4) 503–514.

Mitrovic-Minic′, S., Krishnamurti, R., and Laporte G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38, 669–685.

Paquette, J., Cordeau, J.-F., Laporte, G., and Pascoal, M. M. B. (2013). Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52, 1–16.

Psaraftis, H. N. (1980). A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2) 130–154.

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.

Psaraftis, H. N., Orlin, J. B., Bienstock, D., and Thompson, P. M. (1985). Analysis and solution algorithms of sealift routing and scheduling problems: Final report. Technical Report 1700-85, MIT, Sloan School of Management, Cambridge, MA.

Rappoport, H. K., Levy, L. S., Golden, B. L., and Toussaint, K. (1992). A planning heuristic for military airlift. *Interfaces*, 22:73–87, 1992.

Rappoport, H. K., Levy, L. S., Toussaint, K., and Golden, B. L. (1994). A transportation problem formulation for the MAC airlift planning problem. *Annals of Operations Research*, 50, 505–523.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4) 258–272.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3) 267–286.

Ruland, K. S. (1995). Polyhedral solution to the pickup and delivery problem. PhD thesis, Sever Institute of Technology, Washington University, St. Louis, MO.

Ruland, K. S. and Rodin, E. Y. (1997). The pickup and delivery problem: Faces and branch and-cut algorithm. *Computers and Mathematics with Applications*, 33, 1–13.

Savelsbergh, M. and Sol, M. (1998). Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4) 474-490.

Sexton, T. R. and Bodin, L. D. (1985a). Optimizing single vehicle many-to-many operation with desired delivery times: I. Scheduling. *Transportation Science*, 19(4) 378-410.

Sexton, T. R. and Bodin, L. D. (1985b). Optimizing single vehicle many-to-many operation with desired delivery times: II. Routing. *Transportation Science*, 19(4) 411-435.

Shen, Y., Potvin, J.-Y., Rousseau, J.-M., and Roy, S. (1995). A computer assistant for vehicle dispatching with learning capabilities. *Annals of Operations Research*, 61, 189–211.

Solanki, R. S., and Southworth, F. (1991). An execution planning algorithm for military airlift. *Interfaces*, 21, 121–131.

Solomon, M. M., Chalifour, A., Desrosiers, J., and Boisvert, J. (1992). An application of vehicle routing methodology to large-scale larvicide control programs. *Interfaces*, 22, 88–99.

Swersey, A. and Ballard, W. (1983). Scheduling school buses. *Management Science*, 30, 844–853.

Toth, P. and Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31, 60–71.

Wang, X., Dessouky, M., and Ordonez, F. (2015). A Pickup and Delivery Problem for Ridesharing Considering Congestion (working paper).

Wang, X. and Regan, A. C. (2002). Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36, 97–112.

Yang, L. and Zhou X. (2014). Constraint reformulation and a Lagrangian relaxation-based solution algorithm for a least expected time path problem. *Transportation Research Part B: Methodological*, 59, 22–44.

Zachariadis, E., Tarantilis, C., and Kiranoudis, C. (2015). The load-dependent vehicle routing problem and its pick-up and delivery extension. *Transportation Research Part B: Methodological*, 71, 158–181.