Transportation Letters
The International Journal of Transportation Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ytrl20

A column generation approach for a dynamic ridesharing problem

Minsu Kim, Hyunwoo Kim & Ilkyeong Moon

To cite this article: Minsu Kim, Hyunwoo Kim & Ilkyeong Moon (2023) A column generation approach for a dynamic ridesharing problem, Transportation Letters, 15:9, 1114-1125, DOI: 10.1080/19427867.2022.2133364

To link to this article: https://doi.org/10.1080/19427867.2022.2133364

TRANSPORTATION LETTERS
2023, VOL. 15, NO. 9, 1114–1125
https://doi.org/10.1080/19427867.2022.2133364

Taylor & Francis
Taylor & Francis Group

A column generation approach for a dynamic ridesharing problem

Minsu Kim[a], Hyunwoo Kim[a] and Ilkyeong Moon[b]

[a]Entrue Consulting, LG CNS, Seoul, South Korea; [b]Department of Industrial Engineering and Institute for Industrial Systems Innovation, Seoul National University, Seoul, South Korea

ABSTRACT

Recent advances in GPS and telecommunications technologies have made ridesharing a widespread practice around the world. By matching drivers and passengers with near-distance destinations and similar time schedules, ridesharing saves individuals travel costs by enabling them to share vehicles. It is also an effective way to reduce traffic congestion and greenhouse gas emissions. A ridesharing problem can be modeled as a dial-a-ride problem with time windows. This study looks at a solution to the dynamic ridesharing problem in which passengers share travel costs at the same ratio. We formulate the problem as a mixed-integer programming model and suggest a column generation approach. As the system's status is updated in real time, riders and drivers are matched, and new paths are created via column generation. Computational experiments show that our approach is superior to an existing algorithm when it is tested on instances of various sizes.

KEYWORDS

Ridesharing; dynamic dial-a-ride problems; mixed-integer programming; column generation

Introduction

Dynamic ridesharing refers to a system that supports an automatic ride-matching process between participants on very short notice, or even en-route (Agatz et al. 2012). According to a recent survey, the average occupancy rate for private vehicles in South Korea was 1.22 per vehicle, down 7.9% from 1.32 in 2010, while the number of private vehicles per household increased by 14.1% from 0.75 to 0.86 over the same time period (Korean Transport Database 2016). Similar tendencies were discovered in the United States (Sivak and Schoettle 2012). Such an increase in private vehicle usage inevitably leads to traffic jams and increases fuel consumption and greenhouse gas emissions. Rising oil prices, expensive traffic costs, and growing environmental concerns have increased the interest of policymakers and passengers in ridesharing. At the same time, increased use of smartphones and the development of GPS systems have taken ride sharing to a new level. Nowadays, travelers can schedule trips anytime and anywhere using smartphone apps. When the user enters a location and destination, the app will search for nearby drivers, and automatically calculate the route and estimated fare. With such advancements, transportation network companies (TNCs) such as Uber, Lyft, and Zipcar have appeared on the market and have succeeded. Uberpool, the ridesharing platform of Uber, reported that by April 2016, more than 100 million rides had been taken since it launched the service in August 2014. According to the statistics, about 20% of rides made globally were on the Uberpool service (Techcrunch 2016). In South Korea, a carsharing agency, TADA, recorded 600,000 members in seven months of service launch (JOINS 2019).

The expected economic impact of the introduction of ridesharing services is tremendous and includes reduced greenhouse gas emissions (Fagnant and Kockelman (2014), Bruck et al. (2017)), reduced energy consumption (Chan and Shaheen 2012), mitigation of traffic congestion (Wang, Dessouky, and Ordonez 2016), and cost savings (Delhomme and Gheorghiu 2016). The most significant advantage of ridesharing is that it provides additional flexibility in transportation. According to recent research by the South Korean Ministry of Land, Infrastructure, and Transport, 21.7% of taxis were oversupplied in South Korea (Ministry of Land, Infrastructure and Transport 2017). However, especially in big cities, such as Seoul, people are still suffering from lack of transportation when demand is high, such as during rush hours. According to a recent report by Kakao Mobility, a South Korean TNC, the number of taxi calls at commuting hours was 205,000, while the number of taxis supplied was 37,000, less than one-fifth of the demand. In addition, the number of calls at midnight was 130,000, while the number of taxis in operation was only one-third of this demand, at 41,000 (Kakao Mobility 2019). Given such a mismatch between supply and demand for transportation, ridesharing can be an excellent remedy by meeting excess demands that are not served by existing taxi service.

Ridesharing problems are modeled as dial-a-ride problems (DARPs), a field of vehicle routing problems primarily related to human transportation. DARP focuses on designing vehicle routes and schedules for users who specify pickup and delivery requests between origins and destinations. Therefore, the DARP generalizes the vehicle routing problem with time windows (VRPTW) and the pickup and delivery vehicle routing problem (PDVRP) at the same time (Cordeau and Laporte 2007). Depending on the matching type, DARPs can be characterized as either one-to-one (one rider with one driver; e.g., call taxis), many-to-one (many riders with one driver; e.g., carpooling), and one-to-many (one rider with many drivers; e.g., multi-hop ridesharing). General dynamic ridesharing problems can be seen as many-to-many DARPs, which involve heterogeneous drivers, heterogeneous riders, and multiple depots. DARPs can also be categorized into two categories, static and dynamic. In the case of static DARPs, all information about the drivers and riders (e.g., arrival times and locations) is known in advance, whereas only partial information is known in the dynamic cases, and the rest is gradually revealed in real-time.

In this study, we consider a dynamic peer-to-peer (P2P) ridesharing problem in which individuals schedule one-time shared

CONTACT Ilkyeong Moon ikmoon@snu.ac.kr Department of Industrial Engineering and Institute for Industrial Systems Innovation, Seoul National University, Seoul, South Korea

rides to save travel costs. In our model, a driver can offer a ride to multiple passengers simultaneously, but we do not assume multi-hop traveling. Because we deal with a dynamic environment, we do not assume any prior information about the fleet size and the arrival of passengers. Moreover, we assume that when passengers travel on the same route together, they pay an equal share of the fare for that route.

The purpose of this study is to propose an improved algorithm based on optimization and to provide a valid upper bound for it. Therefore, the algorithm's performance can be evaluated. Another desired outcome of the dynamic ridesharing system is its capability to match a large number of users in a short time. In practice, there can be thousands of transportation requests in urban areas, especially with high traffic. Therefore, by differing the size of instances, we demonstrate that our algorithm is also scalable to medium or large amounts of data.

This study is composed as follows: In Section 2, we introduce literature related to dial-a-ride problems (DARPs) and ridesharing. Section 3 explains our model and provides the solution approach. In Section 4, we show the results of computational experiments and offer a sensitivity analysis. And finally, in Section 5, we present the conclusions.

Literature review

The history of ridesharing goes back to World War II, when the US government led regulatory policy to reduce fuel consumption (Fagnant and Kockelman 2014). Since then, ridesharing has been widely studied because of the ongoing necessity of finding efficient models for human transportation. Dumas, Desrosiers, and Soumis (1991) provides one of the earliest exact algorithms for multi-vehicle pickup and delivery vehicle routing problems with time windows (VRPPDTWs). Baldacci, Maniezzo, and Mingozzi 2004) suggested an exact algorithm for scheduled car pooling problems (CPPs) with bounding procedures based on Lagrangian decomposition. Cordeau 2006) suggested an exact algorithm for DARPs based on the brand-and-cut technique. However, most exact approaches for DARPs are confined only to small-scale instances that contain less than 300 customers. More recently, Yan and Chen (2011) provided an algorithm based on Lagrangian relaxation and heuristics, which can be adapted to large-scale many-to-many car-pooling problems.

For large-scale problems, the cluster-first-routing-second strategy has been successfully applied (Dumas, Desrosiers, and Soumis 1989), Ioachim et al. (1995). People who are close in terms of time and distance are clustered together using column generation, and then the path for each cluster is determined later. Another well-adopted approach is based on insertion heuristics (Jaw et al. (1986), Diana and Dessouky 2004), Xiang, Chu, and Chen (2006)). In this type of algorithm, customers are inserted into each vehicle route by a certain criterion (e.g., minimum cost, maximum profit). Meta-heuristics, including tabu search approaches, are also popular options for solving complex DARPs (Cordeau and Laporte (2003), Nanry and Barnes (2000)). For more details on the various types and solution approaches of DARPs, there exist numerous review papers (Cordeau and Laporte (2007), Toth and Vigo (2014)).

Compared to the static DARPs, the dynamic DARPs have been relatively understudied. Various types and characteristics of dynamic DARPs are summarized in Furuhata et al. (2013). Another study by Savelsbergh and Sol (1998) provides an iterative algorithm based on the set-partitioning formulation and branch-and-price technique for dynamic DARPs. Chen and Xu (2006) proposed a dynamic column generation approach to solve dynamic vehicle routing problems with time windows (DVRPTWs). This study applied the column generation procedure repeatedly, combined with the rolling horizon framework. Tabu search is another well-adopted approach (Attanasio et al. (2004), Berbeglia, Cordeau, and Laporte (2012)). Berbeglia, Cordeau, and Laporte (2010) provides an extensive review on dynamic DARPs and their solution approaches.

Recent advances in mobile technology led to the emergence of a new class of problems: dynamic ridesharing. Agatz et al. (2011) is one of the earliest works on dynamic ridesharing. In real-time ridesharing, it is necessary to match a large number of requests and find an optimal path in a very short time. An agent-based simulation is one of the most popular tools because it is useful for capturing complex interactions between agents on real systems. Several studies have shown that the simulation results fit well with real-world data (Martinez, Correia, and Viegas (2015), Nourinejad and Roorda (2016)). While most simulation studies aim to achieve maximum utility for each individual from a distributed viewpoint, there is also a centralized approach to maximize system-wide benefits, such as a reduction in total travel distance or the number of rides matched. Hosni, Naoum-Sawaya, and Artail (2014) and Masoud and Jayakrishnan (2017) suggested decomposition algorithms for large-scale problems. They formulated their problems as mixed-integer programs and divided them into smaller subproblems, which were then solved in a parallel manner. Ma, Zheng, and Wolfson (2015) and Alonso-Mora et al. (2017) designed heuristics for a real-time, large-scale taxi-sharing system, while Bertsimas, Jaillet, and Martin (2019) suggested an algorithm based on mixed-integer programming and the max-flow heuristic.

Different objectives have been proposed, depending on the priority of decision makers. Santos and Xavier (2015) suggested a ridesharing system that maximizes the number of served requests and minimizes the total cost, while Santi et al. (2014) maximized the number of shared trips. Chen, Liu, and Chen (2010) suggested a congestion-aware scheduling system to reduce fuel consumption. Wang, Dessouky, and Ordonez (2016) proposed an incentive system for high-occupancy vehicles (HOVs) by using dedicated lanes and toll discount policies to improve time savings and matching rates. Li et al. (2015) and Cheng, Xin, and Chen (2017) reflected the preference of riders to drivers to promote the satisfaction and safety of the service.

A reasonable pricing scheme is one of the main design issues of ridesharing systems. Kleiner, Nebel, and Ziparo (2011) and Asghari et al. (2016) designed pricing schemes based on parallel auctions to meet the criteria of both drivers and riders. Stiglic et al. (2016) designed an incentive system that gives more benefit to those who offer more flexibility to the matching system. Qian et al. (2017) solved a taxi group ride (TGR) problem and designed a discount-incentive mechanism to maximize the total saved mileage. Sayarshad and Chow (2015) proposed a non-myopic pricing scheme, based on queuing theory, to make use of real-time data and anticipation of future states. Wang, Agatz, and Erera (2018) and Peng et al. (2018) introduced pricing policies motivated by stable matching, ensuring some degree of satisfaction from the perspective of both drivers and riders, while only incurring a small degradation of the system-wide objective. However, there are not sufficient studies for dynamic ridesharing systems that incorporate a reasonable pricing scheme with many-to-one or one-to-many matching and routing scheme at the same time. Table 1 summarizes the differences between this paper and other literature on dynamic ridesharing. In the table, ridehailing refers to a specific system in which drivers aim to maximize their profit on their own, while ridesharing refers to a system in which participants aim to minimize their costs by sharing rides.

Table 1. Literature on dynamic ridesharing.

Reference	Problem	One-to-one matching	One-to-many matching	Routing	Pricing	Solution approach
Savelsbergh and Sol (1998)	Pickup and delivery		✓	✓		Branch and price
Chen and Xu (2006)	Pickup and delivery		✓	✓		Column generation
Hosni, Naoum-Sawaya, and Artail (2014)	Ridehailing		✓	✓		Lagrangian relaxation
Bertsimas, Jaillet, and Martin (2019)	Ridehailing	✓				MIP and heuristic
Kleiner, Nebel, and Ziparo (2011)	Ridehailing	✓			✓	Auctions
Wang, Agatz, and Erera (2018)	Ridehailing	✓			✓	Stable matching
Peng et al. 2018)	Ridehailing	✓			✓	Stable matching
Yan and Chen (2011)	Ridehailing	✓			✓	Dynamic pricing
Santos and Xavier 2015)	Ridesharing		✓	✓	✓	Heuristic
This study	Ridesharing		✓	✓	✓	Column generation

Dynamic column generation

Introduction

In reviewing previous studies, we found a lack of research on dynamic ridesharing problems. Among the prior studies, Santos and Xavier (2015) is the closest to our work. The study formulated the problem as a mixed-integer program, but did not suggest a tractable solution to it. It instead solved the problem heuristically, using the greedy randomized adaptive search procedure (GRASP). What differentiates this problem from others is the cost structure. Santos and Xavier (2015) assumes that if passengers have traveled together, they share the travel cost of the trip. The amount to be paid is determined by dividing the cost of the arc equally into the number of passengers inside the vehicle. The objective of the problem is to find a solution that maximizes the number of requests served, minus the amount of the payment made by passengers. Santos and Xavier (2015) proposes a model, namely the *dynamic dial-a-ride problem with money as an incentive* (DARP-M).

In the DARP-M, the information of the ridesharing participants is not known in advance. Instead, it is gradually updated over the planning horizon. At the beginning of the planning horizon, there is no request or server in the system, and therefore, both M and N are empty. When a request, r, enters the system, the passengers list their information $(e_r, l_r, r^+, r^-, p_r, b_r)$ on a mobile application. Without loss of generality, we assume that the time at which request r is entered is the same as that at which e_r is entered. Likewise, every server, s, enters the system at time e_s and provides its information $(e_s, l_s, s^+, s^-, q_s)$.

If a server and a request are matched, the server departs from its origin to serve the first request. If a server is not matched with any request, we assume that each waits at its origin until being matched. We call those servers and requests waiting to be matched *idle*. We also refer to a request as idle if it was matched with a server but is still waiting for a pickup. However, if requests cannot be matched within the deadline of reaching their destinations, they have become *critical*. When servers and passengers become critical, they are considered to leave the system. Finally, if a server or request is traveling on the network, we call it *en-route*. When a server or a request is either idle or en-route, it is called an *active* server or a request.

In Santos and Xavier (2015), the authors assumed that a route is not allowed to be modified unless the vehicle is empty, in order to prevent the passengers from being changed. In this study, however, we allow a path of any server to be modified at any time, even if the server is en-route. Any request can be added or removed, as long as the path remains feasible, and the system-wide objective may increase by the change. For the real-time route modification to be possible, the position of every vehicle in the system should be tracked at all times. Due to the advances in telecommunications technology, we assume that this can be done without difficulty. Also, we assume the P2P sharing environment, in which each participant tries to minimize travel expenses, not to maximize profit. In our model, therefore, when a server drops the last passenger off, the server then heads directly to its destination, leaving the system as soon as arriving at that destination. In Santos and Xavier (2015), the servers do not leave the system after their planned trips are complete because the authors regarded them as taxi drivers whose main purpose was to maximize profit.

Framework

This section discusses the dynamic column generation framework to solve the DARP-M. The rolling horizon is a common approach for dynamic vehicle routing problems. Our approach is based on the rolling horizon approach, and it was also motivated by the method suggested by Chen and Xu (2006). Prior to a detailed explanation of the approach, we first give some definitions. Let $M(t)$ and $N(t)$ be the set of active requests and servers at time t. Also, we denote the set of feasible paths of the server $s \in N(t)$ that have been found by time t as $P_s(t)$. The set of all the feasible paths found can be expressed as $P(t) = \bigcup_{s \in N(t)} P_s(t)$. The key idea of the rolling horizon approach is to divide the planning period into multiple epochs of the same length, and solve a static snapshot of the dynamic problem in each epoch as if it were static. Let t_i be the time at which epoch i starts, where $t_1 = 0$.

We denote the static problem of epoch i as $[SP_i]$. Each epoch consists of two phases: One is the *column generation phase*, which is divided into several iterations, and the other is the *optimization and implementation phase*. In the column generation phase, we have several iterations where in each iteration we solve a restricted set covering problem and with the duals of this generate new feasible paths (columns) of the active servers. This is repeated until time $t_{i+1} - \tau$, where τ is the time allowance for the optimization and implementation phase. In the optimization and implementation phase, we solve the integer program $[SP_i]$ consisting of the columns that have been found so far. Let S_i be the solution to $[SP_i]$. Then, S_i should contain exactly one feasible path for each server, $s \in N(t_{i+1})$. At the end of the epoch, we implement the solution (i.e., notice all the servers and passengers of the schedule implied by S_i).

Note that in the column generation phase, servers and requests are still entering the system while the subproblems are being solved. Let η_0 and η_1 be the starting times of the previous and current iterations of the column generation phase, where η_1 equals t_i if it is the first iteration of the epoch. Let $\Theta[\eta_0, \eta_1]$ and $\Sigma[\eta_0, \eta_1]$ be the set of requests and servers that have arrived in time interval $[\eta_0, \eta_1]$.

After the subproblems are solved, we update the sets of active severs and requests (i.e., $M(\eta_1) = M(\eta_0) \cup \Theta[\eta_0, \eta_1]$ and $N(\eta_1) = N(\eta_1) \cup \Sigma[\eta_0, \eta_1]$) and proceed to the next iteration. Whenever a server is updated, we add an empty path for the server to $P(t)$. This is because every server must have one path by formulation [F].

After the solution, S_i, is implemented, we discard all the *obsolete* entities and paths. A request or a server is said to be obsolete if it has become critical by time t_{i+1}. Also, if the itinerary of a server is complete at time t_{i+1} (i.e., the server has arrived at its destination), the requests and the server associated with the route also are considered obsolete. Let $\hat{\Theta}_{i+1}$ and $\hat{\Sigma}_{i+1}$ each be the set of requests and servers that have become obsolete by time t_{i+1}. Then, we start the next epoch with $M(t_{i+1}) = M(t_{i+1}) \backslash \hat{\Theta}_{i+1}$ and $N(t_{i+1}) = N(t_{i+1}) \backslash \hat{\Sigma}_{i+1}$. Let $\hat{S}_i$ be the set of complete routes at time t_{i+1}. We also remove those complete routes from S_i and we start the next epoch with initial paths $P(t_{i+1}) = S_i \backslash \hat{S}_i$. The sequence of events that has been described so far is illustrated in Figure 1.

Set-Partitioning reformulation

In this section, we describe the static problem $[SP_i]$ solved in epoch i. $[SP_i]$ is formulated as a set-partitioning type problem and solved with the column generation procedure. This is a widely used approach to solve various types of vehicle routing problems because the set-partitioning formulation can imply a variety of complex constraints defining a feasible route.

Given a route $l \in P_s(t)$, let $\alpha_{l,r} = 1$ if request r was served by driver s, 0 otherwise. Also, let $\rho_{l,r}$ be the sum of the payment made by the passengers of request r in the vehicle. The binary decision variable ξ_l equals 1 if route $l \in P(t)$ is selected, or 0 otherwise. Then, the set-partitioning reformulation $[SP_i(t)]$ to be solved at time t is defined as follows:

$$[SP_i(t)] \quad \max \sum_{s \in N(t)} \sum_{l \in P_s(t)} \sum_{r \in \mathcal{R}(l)} \left(p_r - \lambda \rho_{l,r}\right) \xi_l \tag{1}$$

$$\text{s.t.} \sum_{s \in N(t)} \sum_{l \in P_s(t)} \alpha_{l,r} \xi_l \leq 1 \qquad \forall r \in M(t) \tag{2}$$

$$\sum_{l \in P_s(t)} \xi_l = 1 \qquad \forall s \in N(t) \tag{3}$$

$$\xi_l \in \{0,1\} \qquad \forall l \in P(t) \tag{4}$$

The symbol $\mathcal{R}(l)$ in the objective function (1) represents the set of requests contained in path l. Constraint (2) requires that every request is contained in, at most, one feasible route, while Constraint (3) requires that every driver is assigned exactly one feasible route. If all the feasible paths for the servers are included in $P(t)$, then problem $[SP_i(t)]$ is equal to problem $[F]$, consisting of $M(t)$ and $N(t)$. However, there are a significant number of feasible routes for the servers, and it is impossible to enumerate them all. Therefore, we start initially with $P(t_i)$, which contains exactly one path for each server, $s \in N(t_i)$. If a server was not matched with any path, it has an empty path. We call such a problem with a partial subset of paths a *restricted master problem* (RMP).

$[SP_i(t)]$ is solved by alternating the optimization of the RMP and the column generation procedure. By solving the linearly relaxed RMP, we obtain the dual values required for constituting the subproblems. The missing paths (columns) with positive reduced costs are generated by solving the subproblems, and then, they are added to $P(t)$. Exactly one subproblem is defined for each server, and therefore, there are $|N(t)|$ subproblems to be solved in a parallel fashion. Let $\bar{u}_r$ and $\bar{v}_s$ be the dual variables associated with Constraints (2) and (3), respectively. Then, the subproblem $[PP_{i,s}(t)]$ of driver s at time t can be expressed as follows:

$$\left[PP_{i,s}(t)\right] \quad \max \sum_{r \in M(t)} \left(h_r - \lambda \rho_r - \bar{u}_r\right) z_r - \bar{v}_s \tag{5}$$

$$\text{s.t.} \sum_{s \in N} \sum_{v \in V} x^s_{r^+,v} \leq 1 \qquad \forall r \in M \tag{6}$$

$$\sum_{v \in U \cup \{s^-\}} x^s_{s^+,v} = 1 \qquad \forall s \in N \tag{7}$$

$$\sum_{v \in U \cup \{s^+\}} x^s_{v,s^-} = 1 \qquad \forall s \in N \tag{8}$$

$$\sum_{v \in V} x^s_{r^+,v} - \sum_{v \in V} x^s_{v,r^-} = 0 \qquad \forall r \in M, \forall s \in N \tag{9}$$

$$\sum_{v \in U \cup \{s^+\}} x^s_{v,u} - \sum_{v \in U \cup \{s^-\}} x^s_{u,v} = 0 \qquad \forall u \in V, \forall s \in N \tag{10}$$

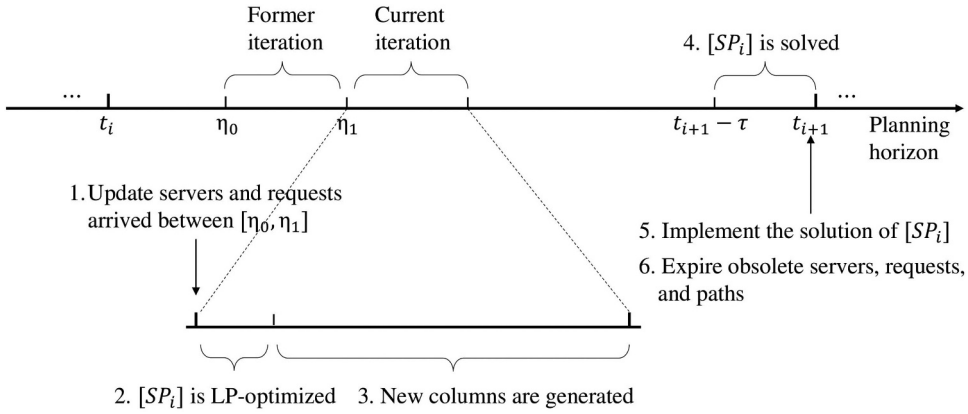

Figure 1. An illustration of the dynamic column generation framework.

$$x^s_{u,v} \in \{0,1\} \quad \forall (u,v) \in U \cup \{s^+\} \times U \cup \{s^-\}, \forall s \in N \tag{11}$$

$$\left(x^s_{u,v} = 1\right) \Rightarrow T^s_v \geq T^s_u + t_{u,v} \quad \forall u,v \in U \cup \{s^+, s^-\}, \forall s \in N \tag{12}$$

$$e_r \leq T^s_{r^+} \leq T^s_{r^-} \leq l_r \quad \forall r \in M, \forall s \in N \tag{13}$$

$$e_s \leq T^s_{s^+} \leq T^s_{s^-} \leq l_s \quad \forall s \in N \tag{14}$$

$$\left(x^s_{u,v} = 1\right) \Rightarrow L^s_v = L^s_u + p_v \quad \forall u,v \in S \cup \{s^+, s^-\}, \forall s \in N \tag{15}$$

$$0 \leq L^s_u \leq q_s \quad \forall u \in V, \forall s \in N \tag{16}$$

$$L^s_{s^+} = L^s_{s^-} = 0 \quad \forall s \in N \tag{17}$$

$$\left(x^s_{u,v} = 1\right) \wedge \left(T^s_{r^+} \leq T^s_u < T^s_{r^-}\right) \Rightarrow C^s_{r,u} = \frac{c_{u,v} p_r}{L^s_u} \quad \forall r \in M, \forall u,v \in V, \forall s \in N \tag{18}$$

$$\sum_{s \in N} \sum_{v \in V} C^s_{r,v} \leq b_r \quad \forall r \in M \tag{19}$$

$$C^s_{r,v} \geq 0 \quad \forall r \in M, \forall v \in V, \forall s \in N \tag{20}$$

$$\rho_r = \sum_{u \in V} h_r \frac{C_{r,u}}{b_r} \quad \forall r \in M(t) \tag{21}$$

$$z_r = \sum_{u \in U} x_{r^+ u} \quad \forall r \in M(t) \tag{22}$$

$$z_r \in \{0,1\} \quad \forall r \in M(t) \tag{23}$$

By Constraint (22), the binary decision variable z_r equals 1 if, and only if, request r is included in the path of server s. Constraint (21) represents the relation between ρ_r and $C_{r,u}$. If the objective is positive, we have found a feasible path with a positive reduced cost. Then, we define a new column corresponding to the path and add it to $P_s(t)$. If there is no path with a positive objective for every server, the RMP is LP-optimal. To solve $[SP_i]$ exactly, we branch the fractional variables and repeat the same procedures. Solving $[SP_i]$ exactly, however, is not considered in this study, as we are solving the dynamic problem. We terminate the procedure at time $t_i - \tau$, even if the RMP is not LP-optimal. At the optimization phase, the integer program $[SP_i]$ is solved with an IP solver. Note that the coefficients of the variables of $[SP_i]$ are either 0 or 1, and because of this special structure, the problem can be solved efficiently using commercial solvers, even if $P(t)$ contains thousands of columns.

Labeling algorithm

The subproblems in the previous section are the longest path version of the *elementary shortest path problem with resource constraints* (ESPPRC). A feasible path must be elementary (i.e., there is no cycle) because a driver cannot visit an already served request again. The ESPPRC is known to be NP-hard (Dror 1994). However, when the problem is highly constrained by resources (e.g., time, budget), it can be solved efficiently with dynamic programming. The ESPPRC is usually solved through the labeling algorithm (Irnich and Desaulniers (2005)), which is an extension of Dijkstra's algorithm to a higher dimension. There can be multiple Pareto optimal paths due to the different types of resources. The algorithm aims to find a set of Pareto optimal paths, while excluding possible non-Pareto optimal ones.

Consider the subproblem defined for server s at time t. Let $U(t) = \{r^+, r^- : r \in M(t)\}$ and $V_s(t) = U(t) \cup \{s^+, s^-\}$. In the labeling algorithm, we construct a complete directed network, $G_s(t) = (V_s(t), A_s(t))$, where $A_s(t) = V_s(t) \times V_s(t)$. The costs of arcs in $A_s(t)$ are called residual costs and depend on the dual values of Constraints (2) and (3). Moreover, they also depend on the status of the vehicle, including the number of passengers inside, and the budgets of those passengers, by Constraint (21).

In the labeling algorithm, partial paths are constructed on a network progressively. Starting from s^+, partial paths are extended along the arcs until reaching s^-. Whenever a new partial path is created to a node, we map a label that records the status of the path. Each node, $p \in V_s(t)$, has a bucket of labels, Λ_p, and each label in it corresponds to a partial path from s^+ to p. We denote l^{th} label of Λ_p as $L^l_p = (R^l_p, Z^l_p, Y^l_p)$, where R^l_p is a resource consumption vector, Z^l_p is the reduced cost of the partial path, and Y^l_p is the set of unfinished requests (i.e., those which not have arrived at their destinations).

The resource consumption vector is represented as (S^l_p, T^l_p, π^l_p), where S^l_p is the set of visited nodes, T^l_p is the elapsed time, and $\pi^l_p = \{\pi^l_{p,r} : r \in Y^l_p\}$ is the expenditure vector, where each $\pi^l_{p,r}$ corresponds to the money paid by the passengers of request r. If the server is idle, the initial label $L^1_{s^+}$ is expressed as $((\{s^+\}, T^1_p, \emptyset), 0, \{s\})$. As the solution to $[SP_i]$ will be implemented at time t_{i+1}, the initial elapsed time T^l_p equals $\max\{t_{i+1}, e_s\}$. Assume the dual variables $\{\bar{u}_r : r \in M(t)\}$ and $\{\bar{v}_s : s \in N(t)\}$ are given. For each $p \in V_s(t)$, define σ_p as follows:

$$\sigma_p = \begin{cases} -\bar{v}_s, & \text{if } p = s^+ \\ h_r - \bar{u}_r, & \text{if } p = r^+ \text{ for some } r \in M(t) \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

Suppose a partial path is located on node p. We define the number of passengers in the vehicle as $||Y^l_p||$, i.e., $||Y^l_p|| = \sum_{r \in Y^l_p} h_r$. Then, given a label, L^l_p, mapped to the path, the residual arc cost $\bar{t}_{pq}(L^l_p)$ induced by extending from p to another node $q \in V_s(t)$ is defined as follows:

$$\bar{t}_{pq}(L^l_p) = \begin{cases} \sigma_p + t_{pq} g(Y^l_p), & \text{if } Y^l_p \neq \emptyset \\ \sigma_p, & \text{if } Y^l_p = \emptyset \end{cases} \tag{25}$$

$$\text{where } g(Y^l_p) = -\frac{\lambda}{||Y^l_p||} \sum_{r \in Y^l_p} \frac{h_r}{b_r} \tag{26}$$

The function $g(\cdot)$ in Equation (26) can be translated as the cost-sharing factor associated with the status of the label. It strictly increases with respect to $|Y^l_p|$ and b_r, resulting in an increased residual cost.

Node q can be associated with either some request $r' \in M(t)$, or server s. Given a label, L_p^l, on node p, the extension to q yields a new label, $L_q^{l'}$, if the following conditions hold:

1. (Precedence) $q \notin S_p^l$ and either (a) or (b) holds:

(a) $(q = r'^+) \vee ((q = r'^-) \wedge (r' \in Y_p^l))$.

(b) $(q = s^-) \wedge (Y_p^l = \emptyset)$

2. (Time windows) either (a) or (b) holds:

(a) $(T_p^l + t_{pq} \le l_{r'}) \wedge (q = r'^-)$

(b) $(T_p^l + t_{pq} \le l_s) \wedge (q = s^-)$

3. (Capacity) either (a) or (b) holds:

(a) $(q = r'^+) \wedge (||Y_p^l|| + h_{r'} \le q_s)$

(b) $q = r'^-$

4. (Budget limit) $\pi_{p,r}^l + \delta(L_p^l, t_{pq}) \le b_r, \quad r \in Y_p^l$ where

$$\delta(L_p^l, t_{pq}) = \begin{cases} t_{pq}/||Y_p^l||, & \text{if } Y_p^l \neq \emptyset \\ t_{pq}, & \text{if } Y_p^l = \emptyset \end{cases} \tag{27}$$

Function $\delta(\cdot)$ defines the cost to be charged to each passenger inside the vehicle. When all three conditions hold, the resulting label $L_q^{l'} = (R_q^{l'}, Z_q^{l'}, Y_q^{l'})$, where $R_q^{l'} = (S_q^{l'}, T_q^{l'} \pi_q^{l'})$, can be expressed as follows:

$$S_q^{l'} = S_p^l \cup \{q\}, \tag{28}$$

$$T_q^{l'} = \begin{cases} \max\{T_p^l + t_{pq}, e_{r'}\}, & \text{if } q = r'^+ \\ T_p^l + t_{pq}, & \text{otherwise} \end{cases} \tag{29}$$

$$\pi_q^{l'} = \begin{cases} \{\pi_{p,r}^{l'} = \pi_{p,r}^l + \delta(Y_p^l, t_{pq}) : r \in Y_p^l\} \cup \{\pi_{r'}^{l'} = 0\}, & \text{if } q = r'^+ \\ \{\pi_{p,r}^{l'} = \pi_{p,r}^l + \delta(Y_p^l, t_{pq}) : r \in Y_p^l \backslash \{r'\}\}, & \text{if } q = r'^- \end{cases} \tag{30}$$

$$Z_q^{l'} = Z_p^l + \bar{t}_{pq}(Y_p^l), \text{and} \tag{31}$$

$$Y_q^{l'} = \begin{cases} Y_p^l \cup \{r'\}, & \text{if } q = r'^+ \\ Y_p^l \backslash \{r'\}, & \text{if } q = r'^- \end{cases} \tag{32}$$

Equations (28) to (30) are also referred to as resource extension functions (REFs). Whenever a label is extended from node p to node q, we add it to bucket Λ_q. We also save the pointer of the parent label so that we can restore the sequence of nodes constituting the feasible path by tracing the pointers backward. However, as the size of the problem grows, the number of labels created can become extremely large. Therefore, additional conditions must be met to extend the label, in order to prevent excessive proliferation of the labels. Propositions 1 and 2 are those conditions that are also explained in Dumas, Desrosiers, and Soumis (1991) explicitly, and therefore will not be proved here.

Proposition 1. Given a label $L_p^l = \left(R_p^l, Z_p^l, Y_p^l\right)$, suppose $r \in Y_p^l$ for some $r \in N$. L_p^l can be eliminated if the extension $p \Rightarrow r^- \Rightarrow s^-$ is not feasible.

Proposition 2. Given a label $L_p^l = \left(R_p^l, Z_p^l, Y_p^l\right)$, suppose $r_1, r_2 \in Y_p^l$ for some $r_1, r_2 \in N$. L_p^l can be eliminated if both the extensions $p \Rightarrow r_1{}^- \Rightarrow r_2{}^- \Rightarrow s^-$ and $p \Rightarrow r_2{}^- \Rightarrow r_1{}^- \Rightarrow s^-$ are not feasible.

It is also common practice to exclude unnecessary labels that cannot be considered Pareto optimal by applying dominance rules, in order to avoid enumerating all possible paths. Label L_p^l is said to dominate another label, $L_q^{l'}$, if all the feasible paths that can be generated from $L_q^{l'}$ can also be generated from L_p^l with greater or equal reduced cost. Proposition 3 states this dominance rule, and we remove a label whenever it is dominated by the other.

Proposition 3. Given two labels $L_p^l = (R_p^l, Z_p^l, Y_p^l)$ and $L_p^{l'} = (R_p^{l'}, Z_p^{l'}, Y_p^{l'})$ where $R_p^l = (S_p^l, T_p^l, \pi_p^l)$, and $R_p^{l'} = (S_p^{l'}, T_p^{l'}, \pi_p^{l'})$, suppose $T_p^l \le T_p^{l'}$, $Z_p^l \ge Z_p^{l'}$, $Y_p^l = Y_p^{l'}$, and $\pi_{p,r}^l \le \pi_{p,r}^{l'} \quad r \in Y_p^l$. Then, L_p^l dominates $L_p^{l'}$.

The labeling algorithm is terminated whenever any non-dominated label is not generated. However, when the size of the problem is big, the algorithm slows down because of the enormous amount of labels that exist in the system. Therefore, we adopt the *truncated labeling algorithm*, which sets a limit on the maximum number of labels that we treat at a time. If the total number of labels exceeds a predefined value, Ω, we sort out the labels with the smallest reduced costs and remove them from the queue until it reaches Ω. If the algorithm does not produce a solution, we increase the size of Ω and repeat the algorithm. We denote the labeling algorithm applied to subproblem $[PP_{i,s}(t)]$ as ELPPRC(i, s, t), as described in Algorithm 1. The function *Extend*(L, q) returns a new label, defined on node q, if the extension is legal. Otherwise, it is $\emptyset$.Another point to be noted is when the server is en-route at time t_i. In this case, one can expect the future position p_0 of the server at time t_{i+1} because servers are assumed to travel through the shortest paths between nodes. In Figure 2, for instance, request r_1 will have arrived at its destination at time t_{i+1}, and therefore r_1 does not have to be considered in this case. Therefore, nodes r_1^+ and r_1^- can be excluded from $V_s(t)$ for all $s \in N(t)$. Request r_2, on the other hand, was picked up, but has not reached its destination at time t_{i+1}. It follows that node r_2^+ can be ignored, but node r_2^- should be considered when solving $[PP_{i,s}(t)]$. We can also calculate $T_{p_0}^1$, $Z_{p_0}^1$, and $\pi_{p_0}^1$, based on the Equations (24) to (32).

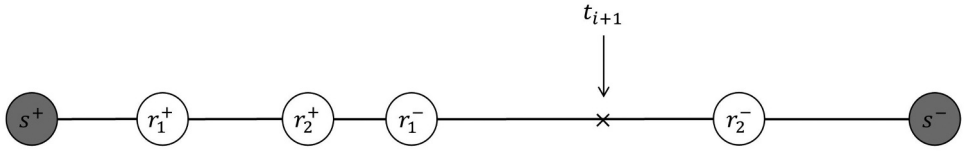

Figure 2. An example of an en-route server at time t_i.

Computational experiments

In this section, we present the results and analyses of computational experiments conducted on simulated data. The experimental setting is an Intel i5 3.20 GHz CPU with 8.0 GB of memory, and the code was written in JAVA language and Xpress-mp optimization API.[1] To simulate real-world ridesharing, instances were generated based on existing node-arc data. The data was collected from the government website[2] and consists of 7,455 nodes and 19,856 arcs. Each arc has a maximum speed limit that each vehicle can travel, and the travel time of each arc was calculated by dividing the length of the arc with the maximum travel speed. We computed the length of the shortest path between every two nodes using the Floyd-Warshall algorithm. The minimum and maximum travel time between nodes each turned out to be 0.82 and 2,217.58 seconds, and the average travel time was 840.33 seconds. The exact monetary values of arc costs are not required in our problem because the arc costs are proportionate to the distances, and we only need the percentage of money spent from the budget.

Two categories of instances were generated: one with longer time windows (class 'A'), and the other with shorter time windows (class 'B'). Let g be the shortest travel time from a server's or a request's origin to its destination. Class 'A' instances have relatively shorter time allowances between the earliest departure times and the latest arrival times of the servers ($1.5g$), and requests ($1.3g$), while class 'B' instances have longer time allowances, $2.0g$ and $1.5g$, respectively. The instances and their features are described in Table 2, where $|M|$ and $|N|$ represent the number of requests and servers, and H represents the length of the planning horizon of each instance.

To provide the relative measure on the effectiveness of our method, the upper bound of the objective value of each instance was computed. Let [SP] be the problem, which has the same structure as [SP] in Section 3.2, whose $M(t)$, $N(t)$, and $P(t)$ are substituted with M, N, and P, where P is the set of all feasible paths of all servers. Then, its objective is the same as that of the static problem, [F], and its LP-bound is also an upper bound of the DARP-M because the objective value of the static problem is always at least greater than the dynamic problem. The column generation procedure described in Section 3.3 was applied to obtain the LP-bound of each instance. It was repeated until the increase of the objective value after each iteration became less than 0.5% of the previous iteration. In the standard column generation, exactly one column with the highest reduced cost for each subproblem is added to the master problem. However, we took the strategy of adding all the columns generated in each iteration, in order to speed up the convergence of the objective value.

As a default, we used 0.5 for the value of parameter λ. For the parameter Ω, which is the maximum number of columns that can exist in the system, we took a three-stage approach. We first tried to solve each subproblem with $\Omega = 100$. If we found at least one column with a positive reduced cost, we terminated the procedure and added those columns to the RMP. If no such column was found, we increased Ω to 500 and 2,000 gradually. This approach has been shown to be effective in greatly reducing the computation time required for convergence. The upper bounds (*UB*) and number of columns generated, *nCol*, are also given in Table 2.

To compare the performance of the suggested algorithm to the existing algorithm by Santos and Xavier (2015), we conducted experiments on each instance, differing the length of an epoch, E, by 15, 30, and 60 seconds. We denote our algorithm as DYCOL (dynamic column generation) and the other as GRASP. Parameter τ, the time allowed to optimize an integer program at the end of each period, was fixed to five seconds. Tables 3 and 4 summarize the results of comparative experiments for class 'A' instances and class 'B' instances, respectively. Symbol *Obj* denotes the objective value obtained by each method, *nMat* denotes the number of rides matched, and *nSer* denotes the number of requests served. The maximum values within an instance are highlighted as bold symbols.

For class 'A' instances, the objective values turned out to be the greatest when E was 15 seconds, regardless of instance sizes. In the case of class 'B' instances, however, performance was better when E was larger as the instance size grew bigger. This shows that there is a trade-off between the responsiveness to the changes in the system's status and the computational opportunities for column generation. When E is short, it is possible to reflect the arrivals of servers and requests to the decision process more quickly. When the instance size is big, on the other hand, it is advantageous to allow for a longer computation time because of more feasible paths, due to large numbers of servers and requests in the system. For both classes, DYCOL outperformed GRASP, which is almost twice as large as the maximum objective value.

To see the impact of the balancing parameter λ, we conducted additional experiments, differing the size of λ. We tested these on three values of λ, 0.1, 0.5, and 0.99. The results are described in Tables 5 and 6, where *%Shared* represents the percentage of shared rides (i.e., rides that contain more than two requests at a time). For both problem classes, *%Shared* increased as λ increased. On the other hand, *nMat* and *nSer* tended to decrease as λ increased. This phenomenon explains the role of λ; for a small value of λ, the algorithm tries to make as many matches as possible because the penalty for budget consumption is not very important. When λ is bigger, however, it is more advantageous to pack as many requests into a vehicle as possible, in order to maximize the cost-sharing effect and thereby minimize the penalty for budget consumption, which leads to the decrease in the number of rides matched and requests served.

Theoretically, the upper bounds are expected to be poor because they give only LP-bounds. They were also computed in a static environment, which results in the introduction of an additional gap. Overall, the LP-bounds obtained from above turned out to be not very tight when the time windows were as narrow as class 'A' instances. When the time windows were as wide as class 'B' instances; however, the bounds were relatively tighter. We concluded that the bound obtained by the labeling algorithm seems to be affected by the length of time windows, and it gets tighter as time windows get longer. In addition, we introduced the limitation on the maximum number of labels that are considered while applying the Ω algorithm. We believe that if we increase the size of Ω, it could yield much larger upper bound results because there is a higher chance of discovering more feasible paths in each iteration of the labeling algorithm. However, we confined Ω to 2,000, at maximum, to terminate the algorithm within the allowed computation time, E. To improve the performance of the algorithm, we believe that acceleration techniques, such as bidirectional search (Righini and Salani (2006)) and decremental space relaxation (Boland, Dethridge, and Dumitrescu (2006)), could do this. These techniques have been proved to be effective in reducing the number of labels to be considered, and therefore, the algorithm is terminated more quickly. There are two parameters that should be considered. The first parameter, E, decides the computation time of each epoch. It was observed that the optimal length of E depends on the size of the problem and time windows. We tested only three values of E, but we

Table 2. Problem classes (*SP*).

Probl	SP_i	$\|N\|$	*H* (hr.)	*UB*	*nCol*
A1	1000	1500	1.0	501.2	3525
A2	1000	2000	1.0	591.3	3735
A3	1500	2250	1.0	861.8	6437
A4	1500	3000	1.0	1040.7	7006
A5	2000	3000	1.0	1354.8	11,102
A6	2000	4000	1.0	1601.5	12,277
A7	4000	6000	1.5	2862.4	24,658
A8	4000	8000	1.5	3422.5	26,712
A9	4500	6750	1.5	3429.8	28,642
A10	4500	9000	1.5	4119.1	31,993
A11	5000	7500	1.5	3876.2	32,973
A12	5000	10,000	1.5	4643.9	35,203
B1	1000	1500	1.0	981.0	9737
B2	1000	2000	1.0	1229.0	10,575
B3	1500	2250	1.0	1581.6	15,603
B4	1500	3000	1.0	1969.7	18,163
B5	2000	3000	1.0	2145.3	22,010
B6	2000	4000	1.0	2789.1	25,292
B7	4000	6000	1.5	4462.3	44,029
B8	4000	8000	1.5	5859.0	54,383
B9	4500	6750	1.5	5011.3	49,560
B10	4500	9000	1.5	6613.7	59,837
B11	5000	7500	1.5	5632.1	56,728
B12	5000	10,000	1.5	7470.6	67,910

Table 3. Comparison with the GRASP heuristic, problem class 'A'.

		Obj		*nMat*		*nSer*	
probl	*E* (sec.)	DYCOL	GRASP	DYCOL	GRASP	DYCOL	GRASP
A1	15	**319.5** (0.62)	**131.7** (0.26)	**403**	**218**	**539**	**245**
	30	289.6	111.4	386	191	497	210
	60	274.5	75.4	375	132	476	143
A2	15	**353.8** (0.60)	**149.2** (0.25)	**448**	**240**	**596**	**275**
	30	327.7	130.4	429	211	557	240
	60	298.7	89.6	411	161	518	172
A3	15	**547.6** (0.63)	**258.4** (0.30)	**669**	**417**	**913**	**476**
	30	525.4	228.8	658	366	883	419
	60	487.0	159.8	636	281	831	303
A4	15	**631.2** (0.61)	**291.9** (0.28)	**712**	**456**	**1022**	**531**
	30	603.9	253.9	696	399	984	462
	60	570.3	174.7	681	295	942	326
A5	15	**843.5** (0.62)	**431.5** (0.32)	**994**	**673**	**1386**	**787**
	30	808.4	372.3	969	604	1336	687
	60	762.8	279.9	953	482	1279	527
A6	15	**958.6** (0.60)	**492.4** (0.31)	**1076**	**740**	**1551**	**883**
	30	916.9	427.2	1064	682	1498	781
	60	870.2	302.8	1045	523	1436	571
A7	15	**1771.8** (0.62)	**928.4** (0.32)	**1977**	**1355**	**2860**	**1650**
	30	1725.0	806.8	1945	1253	2794	1463
	60	1634.4	607.0	1876	1003	2660	1123
A8	15	**2085.0** (0.61)	**1002.9** (0.29)	**2134**	**1471**	**3270**	**1785**
	30	2020.2	923.9	2102	1399	3191	1658
	60	1860.7	697.0	2032	1115	2978	1274
A9	15	**2198.1** (0.64)	**1103.8** (0.32)	**2311**	**1612**	**3485**	**1959**
	30	2090.1	1004.4	2283	1524	3348	1806
	60	1971.3	762.4	2214	1214	3193	1394
A10	15	**2649.4** (0.64)	**1242.0** (0.30)	**2504**	**1729**	**4064**	**2171**
	30	2515.2	1160.6	2464	1655	3897	2038
	60	2361.6	860.3	2394	1346	3694	1560
A11	15	**2457.2** (0.63)	**1243.3** (0.32)	**2566**	**1833**	**3888**	**2218**
	30	2363.3	1167.6	2545	1749	3771	2092
	60	2202.0	854.0	2458	1367	3552	1564
A12	15	**2909.8** (0.63)	**1401.1** (0.30)	**2808**	**1953**	**4486**	**2455**
	30	2801.0	1288.9	2761	1861	4348	2275
	60	2633.5	977.3	2689	1518	4131	1769

Table 4. Comparison with the GRASP heuristic, problem class 'B'.

		Obj		*nMat*		*nSer*	
probl	*E* (sec.)	DYCOL	GRASP	DYCOL	GRASP	DYCOL	GRASP
B1	15	738.0	**481.0** (0.49)	**646**	**583**	1128	**813**
	30	**749.2** (0.76)	446.4	**646**	566	**1140**	768
	60	725.0	407.8	640	531	1110	702
B2	15	**893.0** (0.72)	**532.3** (0.43)	694	**624**	**1322**	**891**
	30	886.6	505.0	**702**	609	1317	853
	60	870.3	459.1	692	579	1295	784
B3	15	1235.4	**785.8** (0.50)	994	**917**	1843	**1324**
	30	**1236.7** (0.78)	770.4	**995**	903	**1845**	1290
	60	1204.3	665.2	989	849	1807	1141
B4	15	1458.2	**885.4** (0.45)	**1062**	**959**	2126	**1446**
	30	**1467.0** (0.74)	839.8	1061	957	**2135**	1396
	60	1431.1	758.0	1051	887	2087	1274
B5	15	1703.8	**1143.6** (0.53)	1362	**1299**	2537	**1905**
	30	**1759.8** (0.82)	1134.6	1343	1283	**2594**	1879
	60	1726.8	1018.9	**1348**	1224	2559	1712
B6	15	1916.1	987.5	**1465**	1134	2816	1638
	30	**2057.5** (0.74)	**1252.8** (0.49)	1455	**1334**	**2971**	**2033**
	60	2037.9	1130.3	1456	1279	2944	1864
B7	15	3165.6	1881.5	2714	2099	4800	3099
	30	3605.8	**2397.3**	2700	**2591**	5286	**3923**
	60	**3620.5** (0.81)	2167.9	**2717**	2512	**5317**	3611
B8	15	2875.1	1527.0	**2889**	1773	4501	2522
	30	4027.4	**2569.6** (0.44)	2870	**2674**	5813	**4147**
	60	**4165.2** (0.71)	2350.8	2860	2604	**5956**	3850
B9	15	3176.2	2177.8	**3112**	2425	4970	3579
	30	3996.9	**2780.7** (0.55)	3037	**2981**	5884	**4541**
	60	**4123.6** (0.82)	2589.9	3064	2852	**6033**	4247
B10	15	3449.6	1673.0	3170	1923	5279	2759
	30	4164.1	**3066.3** (0.46)	3214	**3066**	6126	**4898**
	60	**4709.3** (0.71)	2859.8	**3221**	2970	**6734**	4601
B11	15	4229.1	2168.5	3409	2463	6389	3577
	30	4454.5	**3229.5** (0.57)	3428	**3392**	6570	**5255**
	60	**4651.6** (0.83)	3033.6	**3449**	3283	**6816**	4945
B12	15	3348.5	1853.2	3598	2158	5325	3055
	30	4445.9	**3530.0** (0.47)	**3675**	**3493**	6646	**5630**
	60	**5296.1** (0.71)	3287.4	3653	3392	**7596**	5282

believe that it can be coordinated dynamically, according to the traffic volume. The other parameter, λ, which balances the number of rides matched and the penalty for budget consumption, should be adjusted as well. It is desirable that as many drivers and riders as possible are matched for the efficiency of the system. On the other hand, each individual might wish to save travel costs by sharing a ride with others. It is the manager's decision which aspect of ridesharing should be emphasized. By adjusting the value of λ properly, it seems likely that balancing user satisfaction with system efficiency would be possible. As the size of λ grew, the penalty for travel costs was highlighted, which led to the rise of the proportion of shared rides and the decrease in the number of matches made. Although it was not reported in this study, the size of λ also affected the computation time for solving subproblems. When the traffic load is high, it is not appropriate to set the value of λ low, both for technical and practical reasons because it will reduce the number of rides and raise the computation burden at the same time.

Conclusions

In this study, we designed an alternative solution approach for the dynamic ridesharing problem to the solution suggested by Santos and Xavier (2015), the DARP-M. We formulated it as a set-partitioning problem and applied the dynamic column generation framework of Chen and Xu (2006), in which the planning horizon was divided into multiple periods and static problems were solved repeatedly. The subproblem related to the static problems was the elementarily longest path problem with resource constraints, which is known to be NP-hard, and we designed the labeling algorithm to solve the subproblem. Computational experiments have proved that our algorithm outperforms the existing algorithm by Santos and Xavier (2015). We also conducted a sensitivity analysis to capture the impact of the balancing parameter between the system efficiency and the penalty for budget consumption. We concluded that it requires managerial foresight to decide an appropriate size of the balancing parameter.

This study offers policy implications for ridesharing operators and governments. As the sharing economy develops, the number of ridesharing platform operators will increase, and there are many concerns about billing and passenger allocation. The allocation method considering money incentives enables effective and efficient cost calculation and settlement. In addition, as the market is growing, passengers can receive transportation services at a lower cost because the number of drivers increases. Our model allows users to use mobility at a fair and affordable price. On the government side, ridesharing can flexibly change the supply of transportation options available according to demand, unlike existing transportation options such as buses and taxis can. Therefore, ridesharing can

Table 5. Sensitivity analysis on Ω, problem class 'A'.

probl	*Ω*	*nMat*	*nSer*	*%Shared*
A1	0.1	395	516	0.51
	0.5	**403**	**539**	0.58
	0.99	390	520	**0.61**
A2	0.1	447	**605**	0.53
	0.5	**448**	596	0.60
	0.99	441	598	**0.63**
A3	0.1	664	893	0.55
	0.5	**669**	**913**	0.58
	0.99	649	869	**0.67**
A4	0.1	707	1017	0.54
	0.5	**712**	**1022**	0.60
	0.99	700	999	**0.64**
A5	0.1	980	1360	0.55
	0.5	**994**	**1386**	0.57
	0.99	974	1365	**0.69**
A6	0.1	**1104**	**1613**	0.59
	0.5	1076	1551	0.60
	0.99	1063	1527	**0.67**
A7	0.1	**1981**	**2871**	0.57
	0.5	1977	2860	0.61
	0.99	1936	2815	**0.69**
A8	0.1	**2137**	**3317**	0.62
	0.5	2134	3270	0.63
	0.99	2105	3244	**0.70**
A9	0.1	2137	3317	0.60
	0.5	**2311**	**3485**	0.64
	0.99	2270	3398	**0.67**
A10	0.1	2494	**4085**	0.65
	0.5	**2504**	4064	0.64
	0.99	2457	3987	**0.73**
A11	0.1	**2589**	**3888**	0.58
	0.5	2566	**3888**	0.63
	0.99	2424	3524	**0.69**
A12	0.1	2807	4483	0.62
	0.5	**2808**	**4486**	0.65
	0.99	2772	4454	**0.72**

Table 6. Sensitivity analysis on λ, problem class 'B'.

probl	*λ*	*nMat*	*nSer*	*%Shared*
B1	0.1	**652**	**1142**	0.65
	0.5	646	1140	0.67
	0.99	621	1120	**0.77**
B2	0.1	**700**	**1340**	0.70
	0.5	694	1322	0.76
	0.99	681	1320	**0.77**
B3	0.1	**1008**	**1849**	0.68
	0.5	995	1845	0.73
	0.99	943	1808	**0.78**
B4	0.1	**1062**	2134	0.74
	0.5	1061	**2135**	0.74
	0.99	1011	2104	**0.81**
B5	0.1	1391	**2601**	0.70
	0.5	**1343**	2594	0.75
	0.99	1285	2541	**0.79**
B6	0.1	**1466**	**3016**	0.72
	0.5	1455	2971	0.75
	0.99	1412	2924	**0.82**
B7	0.1	**2795**	**5352**	0.67
	0.5	2717	5317	0.70
	0.99	2522	5163	**0.79**
B8	0.1	**2874**	**5993**	0.72
	0.5	2860	5956	0.73
	0.99	2751	5830	**0.82**
B9	0.1	**3131**	**6181**	0.71
	0.5	3064	6033	0.72
	0.99	2855	5816	**0.80**
B10	0.1	**3285**	**7162**	0.73
	0.5	3221	6734	0.76
	0.99	3102	6398	**0.80**
B11	0.1	**3586**	**7039**	0.67
	0.5	3449	6816	0.73
	0.99	3191	6629	**0.81**
B12	0.1	**3669**	7598	0.74
	0.5	3653	7596	0.74
	0.99	3538	**7860**	**0.83**

increase citizens' satisfaction in a situation where the number of public transportation options and taxis in each country has decreased since COVID-19. Moreover, ridesharing is more effective in reducing greenhouse gases and in boosting road efficiency than is the transportation mode of people using their own cars. In keeping with these social benefits, the government can also develop policies to introduce incentives for ridesharing companies. In this way, whether from a government's or a platform operator's point of view, ridesharing is an attractive market and will expand in the future. We expect our research to be helpful in various cost problems that may arise at that time. We believe that related research should be conducted, and in particular, social values and incentives are expected to be an attractive field when explored in future research.

The limitations of this study are as follows: (1) we considered only two main criteria to evaluate the performance-the number of requests served and the money paid by the riders. In reality, however, there are numerous other criteria to be considered, such as security, user satisfaction, and maximum travel time. These constraints can be taken care of by introducing additional constraints, but it will affect the computation time of the column generation procedures; (2) in recent studies on ridesharing, there have been many concerns about the design aspect, such as ridesharing with designated meeting points, multi-hop (one-to-many or many-to-many) ridesharing, and multimodal ridesharing. We only considered the many-to-one ridesharing problem. Designing effective ridesharing policies is an attractive research topic because it can vary from city to city around the world. (3) finally, as Wang, Agatz, and Erera (2018) pointed out, the system-wide optimum might not coincide with each individual's optimum because each individual might have found better options in terms of travel cost and travel time if they had scheduled trips on their own that were not dictated by the centralized system. Such problems, whose objective is to ensure the best option for every individual, are known as *stable matching problems*. In our study, we did not consider stable matching, and therefore, the fairness of the proposed system can be taken into account in future studies.

Notes

1. https://www.fico.com/en/products/fico-xpress-optimization
2. https://its.go.kr/itsinfo/snl.do

Acknowledgement

The authors are grateful for the valuable comments from the editor and anonymous reviewers. This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (grant number NRF-2019R1A2C2084616).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning [grant number NRF-2019R1A2C2084616].

ORCID

Ilkyeong Moon http://orcid.org/0000-0002-7072-1351

References

Agatz, N., A. L. Erera, M. Savelsbergh, and X. Wang. 2011 . "Dynamic ride-sharing: A Simulation Study in Metro Atlanta ." *Procedia-Social and Behavioral Sciences* 17: 532–550. doi:10.1016/j.sbspro.2011.04.530.

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang. 2012. ". Optimization for Dynamic ride-sharing: A Review ." *European Journal of Operational Research* 223 (2): 259–303. doi:10.1016/j.ejor.2012.05.028.

Alonso-Mora, J., S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, 2017 . On-demand high-capacity ride-sharing via Dynamic trip-vehicle Assignment . Proceedings of the National Academy of Sciences 114, 462–467. doi:10.1073/pnas.1611675114.

Asghari, M., D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, 2016 . "Price-aware real-time ride-sharing at Scale: An auction-based Approach." in: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Francisco Bay Area, California, USA, pp. 3.

Attanasio, A., J. F. Cordeau, G. Ghiani, and G. Laporte. 2004 . "Parallel Tabu Search Heuristics for the Dynamic multi-vehicle dial-a-ride Problem." *Parallel Computing* 30 (3): 377–387. doi:10.1016/j.parco.2003.12.001.

Baldacci, R., V. Maniezzo, and A. Mingozzi. 2004 . "An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation ." *Operations Research* 52 (3): 422–439. doi:10.1287/opre.1030.0106.

Berbeglia, G., J. F. Cordeau, and G. Laporte. 2010 . "Dynamic Pickup and Delivery Problems ." *European Journal of Operational Research* 202 (1): 8–15. doi:10.1016/j.ejor.2009.04.024.

Berbeglia, G., J. F. Cordeau, and G. Laporte. 2012 . "A Hybrid Tabu Search and Constraint Programming Algorithm for the Dynamic dial-A-ride Problem." *INFORMS Journal on Computing* 24 (3): 343–355. doi:10.1287/ijoc.1110.0454.

Bertsimas, D., P. Jaillet, and S. Martin. 2019 . "Online Vehicle Routing: The Edge of Optimization in large-scale Applications." *Operations Research* 67 (1): 143–162. doi:10.1287/opre.2018.1763.

Boland, N., J. Dethridge, and I. Dumitrescu. 2006 . "Accelerated Label Setting Algorithms for the Elementary Resource Constrained Shortest Path Problem." *Operations Research Letters* 34 (1): 58–68. doi:10.1016/j.orl.2004.11.011.

Bruck, B. P., V. Incerti, M. Iori, and M. Vignoli. 2017. ". Minimizing co2 Emissions in a Practical Daily Carpooling Problem." *Computers & Operations Research* 81: 40–50. doi:10.1016/j.cor.2016.12.003.

Chan, N. D., and S. A. Shaheen. 2012 . "Ridesharing in North America: Past, Present, and Future." *Transport Reviews* 32 (1): 93–112. doi:10.1080/01441647.2011.621557.

Cheng, P., H. Xin, and L. Chen, 2017. "Utility-aware Ridesharing on Road Networks." in: Proceedings of the 2017 ACM International Conference on Management of Data, Chicago, Illinois, USA, pp. 1197–1210.

Chen, P. Y., J. W. Liu, and W. T. Chen, 2010 . "A fuel-saving and pollution-reducing Dynamic taxi-sharing Protocol in Vanets." in: 2010 IEEE 72nd Vehicular Technology Conference-Fall, Ottawa, ON, Canada, pp. 1–5.

Chen, Z. L., and H. Xu. 2006. "Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows." *Transportation Science* 40 (1): 74–88. doi:10.1287/trsc.1050.0133.

Cordeau, J. F. 2006. "A branch-and-cut Algorithm for the dial-A-ride Problem." *Operations Research* 54 (3): 573–586. doi:10.1287/opre.1060.0283.

Cordeau, J. F., and G. Laporte. 2003. "A Tabu Search Heuristic for the Static multi-vehicle dial-A-ride Problem." *Transportation Research Part B: Methodological* 37 (6): 579–594. doi:10.1016/S0191-2615(02)00045-0.

Cordeau, J. F., and G. Laporte. 2007. "The dial-a-ride Problem: Models and Algorithms." *Annals of Operations Research* 153 (1): 29–46. doi:10.1007/s10479-007-0170-8.

Delhomme, P., and A. Gheorghiu. 2016. "Comparing French Carpoolers and non-carpoolers: Which Factors Contribute the Most to Carpooling?" *Transportation Research Part D: Transport and Environment* 42: 1–15. doi:10.1016/j.trd.2015.10.014.

Diana, M., and M. M. Dessouky. 2004. "A New Regret Insertion Heuristic for Solving large-scale dial-A-ride Problems with Time Windows." *Transportation Research Part B: Methodological* 38 (6): 539–557. doi:10.1016/j.trb.2003.07.001.

Dror, M. 1994. "Note on the Complexity of the Shortest Path Models for Column Generation in Vrptw." *Operations Research* 42 (5): 977–978. doi:10.1287/opre.42.5.977.

Dumas, Y., J. Desrosiers, and F. Soumis. 1989. "Large scale multi-vehicle dial-a-ride problems . École des hautes études commerciales, Groupe d'études et de recherche en analyse des décisions."

Dumas, Y., J. Desrosiers, and F. Soumis. 1991. "The Pickup and Delivery Problem with Time Windows." *European Journal of Operational Research* 54 (1): 7–22. doi:10.1016/0377-2217(91)90319-Q.

Fagnant, D. J., and K. M. Kockelman. 2014. "The Travel and Environmental Implications of Shared Autonomous Vehicles, Using agent-based Model Scenarios." *Transportation Research Part C: Emerging Technologies* 40: 1–13. doi:10.1016/j.trc.2013.12.001.

Furuhata, M., M. Dessouky, F. Ordóñez, M. E. Brunet, X. Wang, and S. Koenig. 2013. "Ridesharing: The state-of-the-art and Future Directions." *Transportation Research Part B: Methodological* 57: 28–46. doi:10.1016/j.trb.2013.08.012.

Hosni, H., J. Naoum-Sawaya, and H. Artail. 2014. "The shared-taxi Problem: Formulation and Solution Methods." *Transportation Research Part B: Methodological* 70: 303–318. doi:10.1016/j.trb.2014.09.011.

Ioachim, I., J. Desrosiers, Y. Dumas, M. M. Solomon, and D. Villeneuve. 1995. "A Request Clustering Algorithm for door-to-door Handicapped Transportation." *Transportation Science* 29 (1): 63–78. doi:10.1287/trsc.29.1.63.

Irnich, S., and G. Desaulniers. 2005. "Shortest Path Problems with Resource Constraints." In *Column Generation*, edited by Desaulniers G, Desrosiers J, and Solomon M, 33–65. Boston, MA, USA: Springer.

Jaw, J. J., A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. 1986. "A Heuristic Algorithm for the multi-vehicle Advance Request dial-A-ride Problem with Time Windows." *Transportation Research Part B: Methodological* 20 (3): 243–257. doi:10.1016/0191-2615(86)90020-2.

JOINS, 2019. "Tada Achieves 600,000 Members in Just Seven Months of Operation. How Is It Different from Regular Taxis?" https://news.joins.com/article/23480123 . Accessed: 2019-08-27 .

Kleiner, A., B. Nebel, and V. A. Ziparo, 2011. "A Mechanism for Dynamic Ride Sharing Based on Parallel Auctions." in: Twenty-Second International Joint Conference on Artificial Intelligence, Barcelona, Spain.

Korean Transport Database, 2016. Passenger Traffic Status Survey. https://www.ktdb.go.kr/www/contents.do?key=16 . Accessed: 2019-08-27 .

Li, Y., R. Chen, L. Chen, and J. Xu. 2015. "Towards social-aware Ridesharing Group Query Services." *IEEE Transactions on Services Computing* 10 (4): 646–659. doi:10.1109/TSC.2015.2508440.

Martinez, L. M., G. H. Correia, and J. M. Viegas. 2015. "An agent-based Simulation Model to Assess the Impacts of Introducing a shared-taxi System: An Application to Lisbon (Portugal)." *Journal of Advanced Transportation* 49 (3): 475–495. doi:10.1002/atr.1283.

Masoud, N., and R. Jayakrishnan. 2017. ". A Decomposition Algorithm to Solve the multi-hop peer-to-peer ride-matching Problem." *Transportation Research Part B: Methodological* 99: 1–29. doi:10.1016/j.trb.2017.01.004.

Ma, S., Y. Zheng, and O. Wolfson. 2015. "Real-time city-scale Taxi Ridesharing." *IEEE Transactions on Knowledge and Data Engineering* 27 (7): 1782–1795. doi:10.1109/TKDE.2014.2334313.

Ministry of Land, Infrastructure and Transport, 2017. "A Study on Complementing and Improving Taxi Total Amount System." 3rd edition. https://www.prism.go.kr/homepage/origin/retrieveOriginDetail.do?cond_organ_id=1613000research_id=1613000-201600139pageIndex=1.leftMenuLevel=120. Accessed 27 august 2019.

Mobility, K., 2019. "Kakao mobility report 2018." https://brunch.co.kr/@kakaomobility/19. Accessed 27 august 2019.

Nanry, W. P., and J. W. Barnes. 2000. "Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search." *Transportation Research Part B: Methodological* 34 (2): 107–121. doi:10.1016/S0191-2615(99)00016-8.

Nourinejad, M., and M. J. Roorda. 2016. "Agent Based Model for Dynamic Ridesharing." *Transportation Research Part C: Emerging Technologies* 64: 117–132. doi:10.1016/j.trc.2015.07.016.

Peng, Z., W. Shan, P. Jia, B. Yu, Y. Jiang, and B. Yao. 2018. "Stable ride-sharing Matching for the Commuters with Payment Design." *Transportation* 45 (1): 1–21. doi:10.1007/s11116-016-9716-4.

Qian, X., W. Zhang, S. V. Ukkusuri, and C. Yang. 2017. ". Optimal Assignment and Incentive Design in the Taxi Group Ride Problem." *Transportation Research Part B: Methodological* 103: 208–226. doi:10.1016/j.trb.2017.03.001.

Righini, G., and M. Salani. 2006. "Symmetry Helps: Bounded bi-directional Dynamic Programming for the Elementary Shortest Path Problem with Resource Constraints." *Discrete Optimization* 3 (3): 255–273. doi:10.1016/j.disopt.2006.05.007.

Santi, P., G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, 2014. Quantifying the Benefits of Vehicle Pooling with Shareability Networks. Proceedings of the National Academy of Sciences 111, 13290–13294. doi:10.1073/pnas.1403657111.

Santos, D. O., and E. C. Xavier. 2015. "Taxi and Ride Sharing: A Dynamic dial-A-ride Problem with Money as an Incentive." *Expert Systems with Applications* 42 (19): 6728–6737. doi:10.1016/j.eswa.2015.04.060.

Savelsbergh, M., and M. Sol. 1998. "Drive: Dynamic Routing of Independent Vehicles." *Operations Research* 46 (4): 474–490. doi:10.1287/opre.46.4.474.

Sayarshad, H. R., and J. Y. Chow. 2015. "A Scalable non-myopic Dynamic dial-A-ride and Pricing Problem." *Transportation Research Part B: Methodological* 81: 539–554. doi:10.1016/j.trb.2015.06.008.

Sivak, M., and B. Schoettle. 2012. "Eco-driving: Strategic, Tactical, and Operational Decisions of the Driver that Influence Vehicle Fuel Economy." *Transport Policy* 22: 96–99. doi:10.1016/j.tranpol.2012.05.010.

Stiglic, M., N. Agatz, M. Savelsbergh, and M. Gradisar. 2016. "Making Dynamic ride-sharing Work: The Impact of Driver and Rider Flexibility." *Transportation Research Part E: Logistics and Transportation Review* 91: 190–207. doi:10.1016/j.tre.2016.04.010.

Techcrunch, 2016. Uber Says that 20% of Its Rides Globally are Now on Uberpool. https://techcrunch.com/2016/05/10/uber-says-that-20-of-its-rides-globally-are-now-on-uber-pool . Accessed: 2019-08-27 .

Toth, P., and D. Vigo. 2014. *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM.

Wang, X., N. Agatz, and A. Erera. 2018. ". Stable Matching for Dynamic ride-sharing Systems." *Transportation Science* 52 (4): 850–867. doi:10.1287/trsc.2017.0768.

Wang, X., M. Dessouky, and F. Ordonez. 2016. "A Pickup and Delivery Problem for Ridesharing considering Congestion." *Transportation Letters* 8: 259–269.

Xiang, Z., C. Chu, and H. Chen. 2006. "A Fast Heuristic for Solving A large-scale Static dial-A-ride Problem under Complex Constraints." *European Journal of Operational Research* 174 (2): 1117–1139. doi:10.1016/j.ejor.2004.09.060.

Yan, S., and C.-Y. Chen. 2011. "An Optimization Model and a Solution Algorithm for the many-to-many Car Pooling Problem." *Annals of Operations Research* 191 (1): 37–71. doi:10.1007/s10479-011-0948-6.