Column Generation for Real-Time Ride-Sharing Operations

Connor Riley[1], Antoine Legrain[2], and Pascal Van Hentenryck[1]

[1] Georgia Institute of Technology, Atlanta GA 30332, USA
ctriley@gatech.edu,pvh@isye.gatech.edu
[2] Polytechnique Montral, Montreal QC H3T 1J4, Canada
antoine.legrain@polymtl.ca

Abstract. This paper considers real-time dispatching for large-scale ride-sharing services over a rolling horizon. It presents RTDARS which relies on a column-generation algorithm to minimize wait times while guaranteeing short travel times and service for each customer. Experiments using historic taxi trips in New York City for instances with up to 30,000 requests per hour indicate that the algorithm scales well and provides a principled and effective way to support large-scale ride-sharing services in dense cities.

Keywords: real-time dial-a-ride · large-scale optimization.

1 Introduction

In the past decade, commercial ride-hailing services such as Didi, Uber, and Lyft have decreased reliance on personal vehicles and provided new mobility options for various population segments. More recently, ride-sharing has been introduced as an option for customers using these services. Ride-sharing has the potential for significant positive impact since it can reduce the number of cars on the roads and thus congestion, decrease greenhouse emissions, and make mobility accessible to new population segments by decreasing trip prices. However, the algorithms used by commercial ride-sharing services rarely use state-of-the-art techniques, which reduces the potential positive impact. Recent research by Alonso-Mora et al. [1] has shown the benefits of more sophisticated algorithms. Their algorithm uses shareability graphs and cliques to generate all possible routes and a MIP model to select the routes. They impose significant constraints on waiting times (e.g., 420 seconds), which reduces the potential riders to consider for each route at the cost of rejecting customers.

This paper considers large-scale ride-sharing services where *customers are always guaranteed a ride*, in contrast to prior work. The Real-Time Dial-A-Ride System (RTDARS) divides the days into short time periods called epochs, batches requests in a given epoch, and then schedules customers to minimize average waiting times. RTDARS makes a number of modeling and solving contributions. At the modeling level, RTDARS has the following innovations:

1. RTDARS follows a Lagrangian approach, relaxing the constraint that all customers must be served in the static optimization problem of each epoch. Instead, RTDARS associates a penalty with each rider, representing the cost of not serving the customer.
2. To balance the minimization of average waiting times and ensure that the waiting time of every customer is reasonable, RTDARS increases the penalty of an unserved customer in the next epoch, making it increasingly harder not to serve the waiting rider.
3. RTDARS exploits a key property of the resulting formulation to reduce the search space explored for each epoch.
4. To favor ride-sharing, RTDARS uses the concept of virtual stops used in the RITMO project [12] and being adopted by ride-sourcing services.

RTDARS solves the static optimization problem for each epoch with a column-generation algorithm based on the three-index MIP formulation [6]. The main innovation here is the pricing problem which is organized as a series of waves, first considering all the insertions of a single customer, before incrementally adding more customers.

RTDARS was evaluated on historic taxi trips from the New York City Taxi and Limousine Commission [8], which contains large-scale instances with more than 30,000 requests an hour. The results show that RTDARS can provide service guarantees while improving the state-of-the-art results. For instance, for a fleet of 2,000 vehicles of capacity 4, RTDARS obtains an average wait of 2.2 minutes and an average deviation from the shortest path of 0.62 minutes. The results also show that large-occupancy vehicles (e.g., 8-passenger vehicles) provide additional benefits in terms of waiting times with negligible increases in in-vehicle time. RTDARS is also shown to generate a small fraction of the potential columns, explaining its efficiency. The Lagrangian modeling also helps in reducing computation times significantly.

The rest of this paper is organized as follows. Section 2 presents the related work in more detail. Section 3 describes the real-time setting. Section 4 specifies the static problem and gives the MIP formulation. Section 5 describes the column generation. Section 6 specifies the real-time operations. Section 7 presents the experimental results and Section 8 concludes the paper.

2 Related Work

Dial-a-ride problems have been a popular topic in operations research for a long time. Cordeau and Laporte [6] provided a comprehensive review of many of the popular formulations and the starting point of RTDARS's column generation is their three-index formulation. Constraint programming and large neighborhood search were also proposed for dial-a-ride problems (e.g., [7] [4]). Progress in communication technologies and the emergence of ride-sourcing and ride-sharing services have stimulated further research in this area. Rolling horizons are often used to batch requests and were used in taxi pooling previously [10, 11]. In addition, stochastic scenarios along with waiting and reallocation strategies have been previously explored in [2, 3]. Bertsimas, Jaillet, and Martin [5] explored the taxi routing problem (without ride-sharing) and introduced a "backbone" algorithm which increases the sparsity of the problem by computing a set of candidate paths that are likely to be optimal. Alonso-Mora et al. proposed an anytime algorithm which uses cliques to generate vehicle paths combined with a vehicle rebalancing step to move vehicles towards demand [1]. Their "results show that 2,000 vehicles (15% of the taxi fleet) of capacity 10 or 3,000 of capacity 4 can serve 98% of the demand within a mean waiting time of 2.8 min and mean trip delay of 3.5 min." [1]. Both [1] and [5] use hard time windows to reject riders when they cannot serve them quickly enough (e.g., 420 seconds in the aforementioned results). This decision significantly reduces the search space as only close riders can be served by a vehicle. In contrast, RTDARS provides service guarantees for all riders, while still reducing the search space through a Lagrangian reformulation. The results show that RTDARS is capable of providing these guarantees while improving prior results in terms of average waiting times. Indeed, for 2,000 vehicles of capacity 4, RTDARS provides an average waiting time of 2.2 minutes with a standard deviation of 1.24 and a mean trip deviation of 0.62 minutes (standard deviation 1.13). For 3,000 vehicles of capacity 4, the average waiting time is further reduced to 1.81 minutes with a standard deviation of 1.03 and an average trip deviation of 0.23 minutes.

3 Overview of the Approach

RTDARS divides time into epochs, e.g., time periods of 30 seconds. During an epoch, RTDARS performs two tasks: It batches incoming requests and it solves the epoch optimization problem for all unserved customers from prior epochs. The epoch optimization takes, as inputs, these unserved customers and their penalties, as well as the *first* stop of each vehicle after the start of the epoch: Vehicle schedules prior to this stop are committed since, for safety reasons, RTDARS does not allow a vehicle to be re-routed once it has departed for its next customer. These first stops are called *departing stops* in this paper. All customers served before and up to the departing stops of the vehicles are considered served. All others, even if they were assigned a vehicle in the prior epoch optimization, are considered unserved.

Once the epoch is completed, a new schedule and a new set of requests are available. The schedule commits the vehicle routes for the entire next epoch and determines their next departing stops. The customer penalties are also updated to make it increasingly harder not to serve them. RTDARS then moves to the next epoch.

4 The Static Problem

This section defines and presents the static (generalized) dial-a-ride problem solved for each epoch. its objective is to schedule a set of requests on a given set of vehicles while ensuring that no customer deviates too much from their shortest trip time.

The inputs consist primarily of the vehicle and request data. The set of vehicles is denoted by V and each vehicle $v \in V$ is associated with a tuple $(u_0^v, w_0^v, I_v, T_v^B, T_v^E, Q_v)$, where u_0^v is the time the vehicle arrives at its *departing stop* for the epoch, w_0^v is the number of passengers currently in the vehicle, I_v is the set of dropoff requests for on-board passengers, T_v^B is the vehicle start time, T_v^E is the vehicle end time, and Q_v is the capacity of the vehicle. In other words, a vehicle v can only insert new requests after time u_0^v and it must serve the dropoffs in I_v. The request data is given in terms of a complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, which contains the nodes for each possible pickup and delivery. There are five types of nodes: the pickup nodes $P = \{1, \dots n\}$, their associated dropoff nodes $D = \{n+1, \dots 2n\}$, the dropoff nodes $I = \cup_{v \in V} I_v$ of the passengers inside the vehicles, the source 0, and the sink s (the last node in terms of indices). Each node i is associated with a number of people q_i to pick up ($q_i > 0$) or drop off ($q_i < 0$) and the time $\Delta_i \geq 0$ it takes to perform them. If $i \in P$, then the corresponding delivery node is $n+i$ and $q_i = -q_{n+i}$. Also, q_i and Δ_i are zero for the source and the sink. Each node $i \in P$ is associated with a request, which is a tuple of the form (e_i, o_i, d_i, q_i) where e_i is the earliest possible pickup time, o_i is the pickup location, d_i is the dropoff location, and q_i is the number of passengers. Every request i in I is associated with the time u_i^P on which the request was picked up. Every request $i \in P \cup I$ is associated with the shortest time t_i from the request origin to its destination. Finally, the input contains a matrix $(t_{i,j})_{(i,j) \in \mathcal{A}}$ of travel times from any node i to any node j satisfying the triangle inequality, the constants α and β which constrain the deviation from the shortest path, and the penalty p_i of not serving the request $i \in P$.

A MIP model for the static problem is presented in Figure 1. The MIP variables are as follows: u_i^v represents the time at which vehicle v arrives at node i, w_i^v the number of people in vehicle v when v leaves node i, x_{ij}^v denotes whether edge (i, j) is used by vehicle v, and z_i captures whether request $i \in P$ is served. Objective (1a) balances the minimization of wait times for every pickups with the penalties incurred by unserved riders. Note that the wait times for riders in I are not included in the objective because these riders are already in vehicles: only the constraints on their deviations must be satisfied. Constraints (1b) ensure that only one vehicle serves each request and that, if the request is not served, z_i is set to 1 to activate the penalty in the objective. Constraints (1c) are flow balance constraints. Constraints (1d) and (1e) are flow constraints for the source and the sink. Constraints (1f) ensure that every request is dropped off by the same vehicle that picks it up. Constraints (1g) ensure that every passenger currently in a vehicle is dropped off. Constraints (1h) define the arrival times at the nodes. Constraints (1i) and (1j) ensure that the vehicle is operational during its working hours. Constraints (1k) ensure that each rider is picked up no earlier than its lower bound. Constraints (1l) ensure that the travel time of each served passenger does not deviate too much from the shortest path between its origin and destination. Passengers are allowed to spend either $\alpha * t_i$ (a percent-

$$\min \quad \sum_{i\in P}\sum_{v\in V}(u_i^v - e_i) + \sum_{i\in P} p_i z_i \tag{1a}$$

subject to

$$\left(\sum_{v\in V}\sum_{j\in\mathcal{N}} x_{ij}^v\right) + z_i = 1 \qquad \forall i\in P \tag{1b}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^v = \sum_{j\in\mathcal{N}} x_{ji}^v \qquad \forall i\in\mathcal{N}\setminus\{0,s\}, \forall v\in V \tag{1c}$$

$$\sum_{j\in\mathcal{N}} x_{0j}^v = 1 \qquad \forall v\in V \tag{1d}$$

$$\sum_{j\in\mathcal{N}} x_{j,s}^v = 1 \qquad \forall v\in V \tag{1e}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^v - \sum_{j\in\mathcal{N}} x_{n+i,j}^v = 0 \qquad \forall i\in P, \forall v\in V \tag{1f}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^v = 1 \qquad \forall j\in I_v, \forall v\in V \tag{1g}$$

$$u_j^v \geq (u_i^v + \Delta_i + t_{ij})x_{ij}^v \qquad \forall i,j\in\mathcal{N}, \forall v\in V \tag{1h}$$

$$u_0^v \geq T_v^B \qquad \forall v\in V \tag{1i}$$

$$u_s^v \leq T_v^E \qquad \forall v\in V \tag{1j}$$

$$u_i^v \geq e_i \qquad \forall i\in P, v\in V \tag{1k}$$

$$t_i \leq u_{n+i}^v - (u_i^v + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i\in P, \forall v\in V \tag{1l}$$

$$t_i \leq u_i^v - (u_i^P + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i\in I_v, \forall v\in V \tag{1m}$$

$$w_j^v \geq (w_i^v + q_j)x_{ij}^v \qquad \forall i,j\in\mathcal{N}, \forall v\in V \tag{1n}$$

$$0 \leq w_i^v \leq Q_v \qquad \forall i\in\mathcal{N}, \forall v\in V \tag{1o}$$

$$x_{ij}^v \in \{0,1\} \qquad \forall i,j\in\mathcal{N}, \forall v\in V \tag{1p}$$

Fig. 1: The Static Formulation of the Dial-A-Ride Problem.

age of the shortest path), or $\beta + t_i$ (a constant deviation time from the shortest path) traveling in the vehicle, whichever is larger. Constraints (1m) do the same for passengers already in a vehicle. Constraints (1n) define the vehicle capacities. Lastly, constraints (1o) ensure that the vehicle capacities are not exceeded. Constraints (1h) and (1n) can be linearized using a Big M formulation.

The following theorem provides a way to prune the search space significantly. It shows that, in an optimal solution, a rider cannot be picked up by a vehicle v if the smallest possible wait time incurred using v is greater than her penalty.

Theorem 1. *A feasible solution where rider l is assigned to vehicle v such that $u_0^v + t_{0,l} - e_l > p_l$ is suboptimal.*

Proof. Suppose that there exists a feasible solution *(I)* that serves a passenger l such that $u_0^v + t_{0,l} - e_l > p_l$. Let r be the route of vehicle v (i.e., a sequence of edges in $\mathcal{A}$). Removing the pickup and dropoff of rider l from route r produces a new feasible route $\hat{r}$ since the deviation time cannot increase by the triangular inequality and the number of riders in v decreases. Solution *(II)* is derived from solution *(I)* by replacing the route r by route $\hat{r}$ and fixing z_l to 1. Using $\hat{u}$ and $\hat{z}$ to denote the variables of solution *(II)*, the cost $C_{(II)}$ of solution *(II)* is:

$$
\begin{aligned}
C_{(II)} &= \sum_{i \in P \setminus \{l\}} \sum_{v \in V} (\hat{u}_i^v - e_i) + \sum_{i \in P \setminus \{l\}} p_i \hat{z}_i + p_l && \text{(2a)}\\
&< \sum_{i \in P \setminus \{l\}} \sum_{v \in V} (\hat{u}_i^v - e_i) + \sum_{i \in P \setminus \{l\}} p_i \hat{z}_i + u_0^v + t_{0,l} - e_l && \text{(2b)}\\
&\leq \sum_{i \in P \setminus \{l\}} \sum_{v \in V} (u_i^v - e_i) + \sum_{i \in P \setminus \{l\}} p_i \hat{z}_i + u_l^v - e_l && \text{(2c)}\\
&= \sum_{i \in P} \sum_{v \in V} (u_i^v - e_i) + \sum_{i \in P} p_i z_i = C_{(I)} && \text{(2d)}
\end{aligned}
$$

Equality (2a) is just the definition of the objective of solution *(II)*. Inequality (2b) is induced by the hypothesis. Inequality (2c) is induced by the triangular inequality on the travel times. Inequality (2d) just factors the equation to get the objective of solution *(I)*. Solution *(I)* is thus suboptimal. □

5 The Column-Generation Algorithm

This section presents the column-generation algorithm, starting with the master problem before presenting the pricing subproblem, and the specifics of the column-generation process. Upon completion of the column generation, RT-DARS solves a final MIP that imposes integrality constraints on the master problem variables.

The Master Problem The restricted master problem, RMP, (presented in Figure 2) selects a route for each vehicle. In order for a route to be assigned to a vehicle, the route must contain dropoffs for every current passenger of that vehicle. The set of routes is denoted by R and its subset of routes that can be assigned to vehicle v is denoted R_v. The variables in the master problem are the following: $y_r \in [0, 1]$ is set to 1 if potential route r is selected for use and variable $z_i \in [0, 1]$ is set to 1 if request i is not served by any of the selected routes. The constants are as follows: c_r is the sum of the wait time incurred by customers served by route r, p_i is the cost of not scheduling request i for this period, and $a_i^r = 1$ if request i is served by route r. The objective minimizes the waiting times incurred by all customers on each route and the penalties for the customers not scheduled during the current period. Constraints (3c) ensure that z_i is set to 1 if request i is not served by any of the selected routes and constraints (3d) ensure that only one route is selected per vehicle. The dual variables associated with

$$\min \quad \sum_{r\in R} c_r y_r + \sum_{i\in P} p_i z_i \tag{3a}$$

subject to (3b)

$$\left(\sum_{r\in R} y_r a_i^r\right) + z_i = 1 \qquad \forall i \in P \qquad (\pi_i) \tag{3c}$$

$$\sum_{r\in R_v} y_r = 1 \qquad \forall v \in V \qquad (\sigma_v) \tag{3d}$$

$$z_i \in \mathbb{N} \qquad \forall i \in P \tag{3e}$$

$$y_r \in \{0,1\} \qquad \forall r \in R \tag{3f}$$

Fig. 2: The Master Problem Formulation.

each constraint are specified in between parentheses next to the constraint in the model.

The Pricing Problem The routes for each vehicle v are generated via a pricing problem depicted in Figure 3. The pricing problem (4) is defined for a given vehicle v. Theorem 1 makes it possible to remove some passengers from the set P to obtain the subset P_v and thus a new graph $\mathcal{G}_v = (\mathcal{N}_v, \mathcal{A}_v)$. The pricing problem minimizes the reduced cost of the route being generated. Constraints (4b) – (4o) correspond to constraints (1c) – (1p) in the static problem.

The Column Generation In traditional column generation for dial-a-ride problems, the pricing problem is formulated as a resource-constrained shortest-path problem and solved using dynamic programming. However, the minimization of waiting times, i.e., $\sum_{i\in P}(u_i - e_i)$, is particularly challenging, as it cannot be formulated as a classical resource-constrained shortest-path problem. One option is to discretize time and use time-expanded graphs. However, this raises significant computational challenges for large instances. As a result, this paper solves the pricing problem through an anytime algorithm that takes into account the real-time constraints RTDARS operates under.

The column-generation algorithm is specified in Algorithm 1: *It generates multiple columns with disjoint sets of customers*. In the algorithm, function Pricing(v, R) solves the pricing problem for a vehicle v and a set R of requests, while Route(v, R) returns the optimal route for a vehicle v and a set of request R. Lines 1–5 is the high-level column-generation procedure: It alternates the generation of columns and the solving of the master problem with the generated columns until no more columns can be generated. It proceeds in waves, first generating columns with one customer before progressively increasing the number of considered requests. Procedure GenerateColumn (lines 6–12) generates columns by increasing number of requests. Procedure GenerateSizedColumn (lines 13–18) generates columns of size k, where k is the number of requests in

$$\min \quad \sum_{i \in P_v} (u_i - e_i) - \sum_{i \in P_v} \sum_{j \in \mathcal{N}_v} x_{ij} \pi_i - \sigma_v \tag{4a}$$

subject to

$$\sum_{j \in \mathcal{N}_v} x_{ij} = \sum_{j \in \mathcal{N}_v} x_{ji} \qquad \forall i \in \mathcal{N}_v \setminus \{0, s\} \tag{4b}$$

$$\sum_{j \in \mathcal{N}_v} x_{0j} = 1 \tag{4c}$$

$$\sum_{j \in \mathcal{N}_v} x_{js} = 1 \tag{4d}$$

$$\sum_{j \in \mathcal{N}_v} x_{ij} - \sum_{j \in \mathcal{N}_v} x_{n+i,j} = 0 \qquad \forall i \in P_v \tag{4e}$$

$$\sum_{i \in \mathcal{N}_v} x_{ij} = 1 \qquad \forall j \in I_v \tag{4f}$$

$$u_j \geq (u_i + \mathit{\Delta}_i + t_{ij}) x_{ij} \qquad \forall i, j \in \mathcal{N}_v \tag{4g}$$

$$u_0 \geq T_v^B \tag{4h}$$

$$u_s \leq T_v^E \tag{4i}$$

$$u_i \geq e_i \qquad \forall i \in P_v \tag{4j}$$

$$t_i \leq u_{n+i} - (u_i + \mathit{\Delta}_i) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i \in P_v \tag{4k}$$

$$t_i \leq u_i - (u_i^P + \mathit{\Delta}_i) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i \in I_v \tag{4l}$$

$$w_j \geq (w_i + q_j) x_{ij} \qquad \forall i, j \in \mathcal{N}_v \tag{4m}$$

$$0 \leq w_i \leq Q_v \qquad \forall i \in \mathcal{N}_v \tag{4n}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{N}_v \tag{4o}$$

Fig. 3: The Pricing Problem Formulation for Vehicle v.

the column. It first computes Q, a set in which each element is a k-sized set of possible requests. It then considers the various vehicles ranked in decreasing order of their dual values σ_v. Line 15 computes the sets of requests with the smallest pricing objective value. If the pricing objective is negative (line 16), all set of requests which contains a request covered by R_v are removed from Q to ensure that RTDARS generates a set of non-overlapping columns at each iteration (line 17). Finally, line 18 returns the routes for each vehicle with negative reduced costs.

6 The Real-Time Problem

RTDARS divides the time horizon into epochs of length ℓ, i.e., $[0, \ell), [\ell, 2\ell), [2\ell, 3\ell), \ldots$ and epoct τ corresponds to the time interval $[\tau\ell, (\tau+1)\ell)$. During period τ, RTDARS batches the incoming requests into a set P_τ, which is considered in the

Algorithm 1: ColumnGeneration

while *true* **do**
  $\mathcal{C} \leftarrow$GenerateColumns()
  if $\mathcal{C} = \emptyset$ **then**
    break;
  Solve RMP after adding $\mathcal{C}$

Function GenerateColumns():
  $k \leftarrow 1$
  while $k \leq |P|$ **do**
    $\mathcal{C} \leftarrow$ GenerateSizeColumns(k)
    if $\mathcal{C} \neq \emptyset$ **then**
      return $\mathcal{C}$
    else
      k++

Function GenerateSizedColumms(k):
  $Q \leftarrow \{R \subseteq P \mid |R| = k\}$
  forall $v \in |V|$ *ordered by decreasing* σ_v
    $R_v \leftarrow \text{argmin}_{R \subset Q}$ pricing(v, R)
    if pricing$(v, R_v)\} < 0$ **then**
      $Q \leftarrow \{R \subseteq Q \mid R \cap R_v = \emptyset\}$
  return $\{$route$(v, R_v) | v \in V$ & pricing$(v, R_v) < 0\}$

next epoch. It also optimizes the static problem using the requests accummulated in $P_{\tau-1}$ and those requests not yet committed to in the epochs $\tau - 1$ and before. The optimization is performed over the interval $[(\tau + 1)\ell, \infty)$.

It remains to specify how to compute the inputs to the optimization problem, i.e., the departing stops and times for each vehicle and the various set of requests to serve. To determine the starting stop for a vehicle v, the optimization in epoch τ uses the solution $\phi_{\tau-1}$ to the static problem in epoch $\tau - 1$ and considers the first stop s_v in $\phi_{\tau-1}$ in the interval $[(\tau + 1)\ell, \infty)$ if it exists. This stop becomes the starting stop u_0^v of the vehicle and its earliest time is given by the earliest departure time of vehicle v in $\phi_{\tau-1}$. If vehicle v is idle at stop s_v in $\phi_{\tau-1}$ and not scheduled on $[(\tau + 1)\ell, \infty)$, then the departing stop is s_v and the earlierst departing time is $(\tau + 1)\ell$. Consider now the sets P, D, and I_v $(v \in V)$ for period τ. For a vehicle v, all the requests before its departing stop s_v are said to be *committed* and are not reconsidered. The set I_v are the dropoffs of the requests that have been picked up before s_v but not yet dropped off. The set P corresponds to the requests that have not been picked up by any vehicle v before s_v, as well as the requests batched in $P_{\tau-1}$. The set D simply contains the dropoffs associated with P.

Finally, since the static problem may not schedule all the requests, it is important to update the penalty of unserved requests to ensure that they will

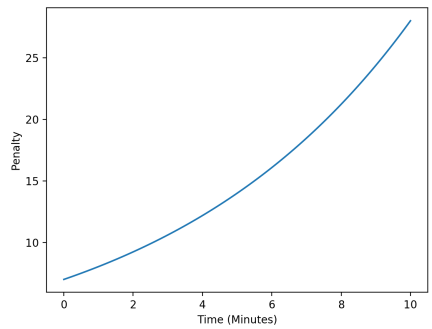


Fig. 4: The Penalty Function for Unserved Customers.

not be delayed too long. The penalty for an unserved request c in period τ is given by $p_c = \delta 2^{(\tau\ell - e_c)/(10\ell)}$ and it increases exponentially over time as shown in Figure 4. The δ parameter incentivizes the schedule of the request in its first available period. Figure 4 displays the function for $\delta = 420$ seconds and $\ell = 30$ seconds: It ensures that the penalty doubles every ten periods (in the example, every five minutes).

Observe that the static model schedules all the requests which have not been committed to any vehicle. This gives a lot of flexibility to the real-time system at the cost of more complex pricing subproblems.

7 Experimental Results

Instance Description RTDARS was evaluated on the yellow trip data provided by the New York City Taxi and Limousine Commission [8]. This data provides *pickup and dropoff locations*, which were used to match trips to the closest virtual stops, *starting times*, which were used as the request time, and the *number of passengers*. This section reports results on a representative set of 24 instances, 1 hour per day for two weekdays per month from July 2015 through June 2016. To capture the true difficulty of the problem, rush hours (7–8am) were selected. The instances have an average of 21,326 customers and range from 6,678 customers to 28,484 customers. Individual requests with more customers than the capacity of the vehicles were split into several trips. An additional test was performed on the largest instance with 32,869 customers.

Virtual Stops The evaluation assumes a dial-a-ride system using the concept of virtual stops proposed in the RITMO system [12] (Uber and Lyft are now considering similar concepts). Virtual stops are locations where vehicles can pick up and drop off customers without impeding traffic. They also ensure that customers are ready to pick up and make ride-sharing more efficient since they

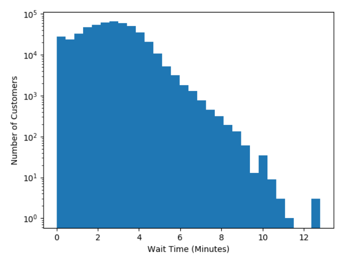


Fig. 5: The Histogram of Wait Times (Log Scale).

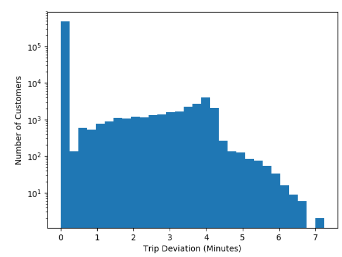


Fig. 6: The Histogram of Trip Deviations (Log Scale).

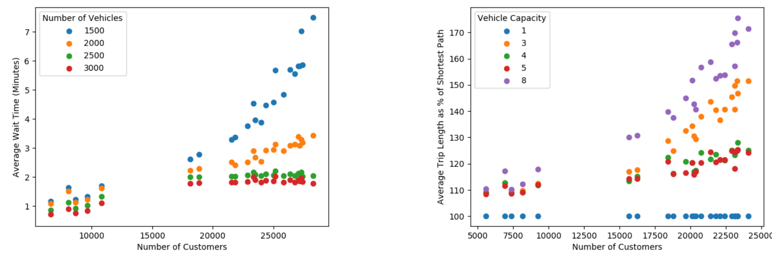
decrease the number of stops. To implement virtual stops, Manhattan was overlayed with a grid with cells of 200 squared meters and every cell had a virtual stop. The trip times were precomputed by querying OpenStreetMap for travel times between each virtual stop [9]. All customers at a virtual stop are grouped and can be picked up together.

Algorithmic Setting Both the final master problem and the restricted master problem are solved using Gurobi 8.1. Empty vehicles are initially evenly distributed over the virtual stops. The pricing problem uses parallel computing to implement line 15 of Algorithm 1, exploring potential requests simultaneously. To meet real-time constraints, the implementation greedily extends the "optimal" routes of size k to obtain routes of size $k+1$. Unless otherwise specified, all experiments are performed with the following default parameters: 2,000 vehicles of capacity 5, $\alpha = 1.5$, $\beta = 240$ seconds, and $\delta = 420$ seconds. The impact of these parameters is also studied.

Wait Times Figure 5 reports the distribution of the waiting for all customers across all instances. The results demonstrate the performance of RTDARS: The average waiting time is about 2.58 minutes with a standard deviation of 1.31. On the instance with 32,869 customers, the average waiting time is 5.42 minutes.

Trip Deviation Figure 6 depicts a histogram of trip deviations incurred because of ride-sharing. The results indicate that riders have an average trip deviation of 0.34 minutes with a standard deviation of 0.74. In percentage, this represents a deviation of about 12%. On the instance with 32,869 customers, the average trip deviation is 2.23 minutes, which shows the small overhead induced by ride-sharing.

The Impact of the Fleet Size Figure 7 studies the impact of the fleet size on the waiting times and trip deviation. The plot reports the average waiting times for various numbers of riders, where capacity is 4, $\alpha = 1$, $\beta = 840$ seconds, and $\delta = 420$ seconds to facilitate comparisons to [1]. The results show that, even with



(a) The Impact on the Average Wait Times.

(b) The Impact on the Average Trip Deviations.

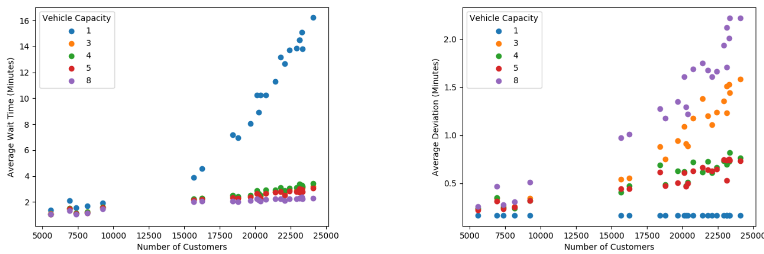
Fig. 7: The Impact of the Fleet Size on the Average Wait Times and Average Deviations on All Instances.

1,500 vehicles, the average waiting time remains below 6 minutes and the average deviation time below 40 seconds. Since RTDARS is guaranteed to serve all the requests, these results demonstrate the potential of column generation and ride-sharing for large-scale real-time dial-a-ride platforms. Adopting RTDARS has the potential to substantially reduce traffic in large cities, while still guaranteeing service within reasonable times. Recall that the approach in [1] does not serve about 2% of the requests.

The Impact of Vehicle Capacity Figure 8 studies the impact of the vehicle capacity (i.e., how many passengers a vehicle can carry) on the average waiting times and trip deviation. The parameters are set to 2,000 vehicles, $\alpha = 1$, $\beta = 840$ seconds, and $\delta = 420$ seconds to facilitate comparisons to [1]. The results on waiting times show that moving to vehicles of capacity 8 further reduces the average waiting times, especially on the large instances. On the other hand, moving from a capacity 5 to 3 does not affect the results too much. The results on deviations are more difficult to interpret. Obviously moving to a capacity 8 further increases the deviation (although it remains below one minute). However, moving to vehicles of capacity 3 also increases the deviation, which is not intuitive. This may be a consequence of myopic decisions that cannot be corrected easily given the tight capacity.

The Impact of the Penalty The penalty p_i in the model is an exponential function of the current waiting time of customer i. Constant δ controls the initial penalty: If it is too small, the penalty for not scheduling a request for the first few periods is low, which causes an increase in wait times, as can be observed in Figure 9. Once δ is large enough, the average wait times converge to the same values.

Final Vehicle Assignments As a result of re-optimization, the vehicle to which a rider is assigned can change. Figure 10 reports the amount of time until riders receive their final vehicle assignment (the vehicle which actually picks them up).



(a) The Impact on Wait Times. (b) The Impact on Trip Deviations.

Fig. 8: The Impact of the Vehicle Capacity on the Average Wait Times and the Average Trip Deviations on All Instances.

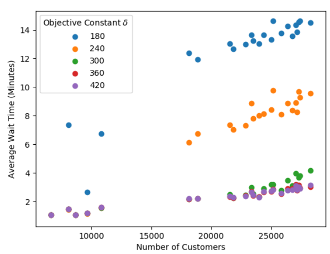


Fig. 9: The Impact of the Penalty on Average Wait Times.

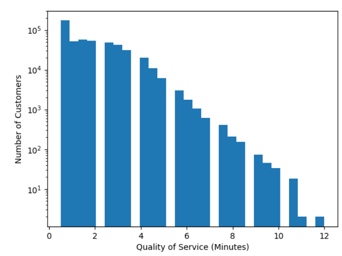


Fig. 10: Times Until Final Vehicle Assignments.

Not surprisingly, this histogram closely follows the waiting time distribution. The majority of riders receive this assignment quickly. However, it takes some riders over 10 minutes to receive their final vehicle assignment, which shows that RTDARS takes advantage of the ability to re-assign riders to vehicles which will result in better overall assignments.

The Impact of Column Generation Figure 11 depicts the impact of column generation and reports the number of columns in the final MIP as all possible columns of sizes 1 and 2 to be conservative. The results show that the algorithm only explores a small percentage of all potential columns, demonstrating the benefits of a column-generation approach.

The Impact of Pruning Figure 12 shows the impact of Theorem 1, which provides a way to prune the number of requests considered at each step of the algorithm. The figures report the total optimization time for all time periods of each instance. Each optimization must be performed in less than 30 seconds, but the graph reports the total optimization time over the entire hour. As the results indicate, the pruning benefits become substantial as the instance sizes

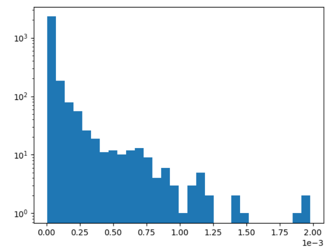


Fig. 11: The Number of Generated Columns as a Percentage of Possible Combinations of Requests/Vehicles. The x-Axis value are scaled by 10^{-3}.

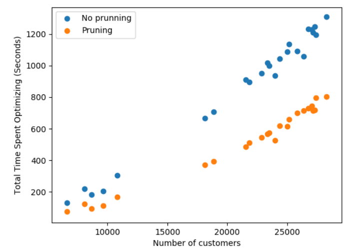
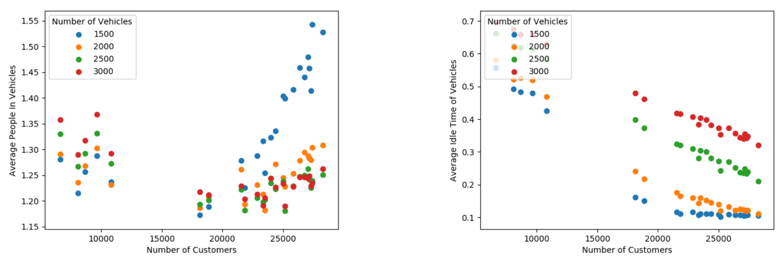


Fig. 12: Optimization Times With and Without Pruning.



(a) The Impact on Average Vehicle Utilization.

(b) The Impact on Average Vehicle Idle Time.

Fig. 13: The Impact of the Fleet Size on the Average Vehicle Utilization and Idle Time on All Instances.

grow. The results show that the pruning significantly reduces the computational time. They also show that RTDARS should be able to handle even larger instances since, after exploiting Theorem 1, RTDARS uses only about a sixth of the available time. This creates opportunities to exploit stochastic information.

The Impact of Ride Sharing Figure 13 reports the average number of people in each vehicle at all times for each instance. The results show a significant amount of ride sharing, although single trips and idle time remain a significant portion of the rides, especially when the fleet is oversized. Lastly, Figure 8 shows that wait times are reduced by a factor of 4 when moving from single-rider trips to ride-sharing for large instances while the trip deviation only increases to at most 2 minutes for vehicles of capacity 8, thus demonstrating the value of ride sharing.

Comparison with Prior Work The results of [1] “show that 2,000 vehicles (15% of the taxi fleet) of capacity 10 or 3,000 of capacity 4 can serve 98% of the demand within a mean waiting time of 2.8 min and mean trip delay of 3.5 min." RTDARS relaxes the hard time-windows present in [1] and improves on these results, yielding an average wait time of 2.2 minutes with only 2,000 vehicles, while guaranteeing service for all riders.

8 Conclusion

This paper considered the real-time dispatching of large-scale ride-sharing services over a rolling horizon. It presented RTDARS, a real-time optimization framework that divides the time horizon into epochs and uses a column-generation algorithm that minimizes wait times while guaranteeing services for every rider and a small trip deviation compared to a direct trip. This contrasts to earlier work which rejected customers when the predicted waiting time was considered too long (e.g., 7 minutes). This assumption reduced the search space at the cost of rejecting a significant number of requests.

The column-generation algorithm of RTDARS is derived from a three-index formulation [6] which is adapted for use in real-time dial-a-ride applications. In addition, to ensure that all riders are served in reasonable times, the paper proposed an optimization model that balances the minimization of waiting times with penalties for riders that are not scheduled yet. These penalties are increased after each epoch to make it increasingly harder not to serve waiting riders. The paper also presented a key property of the formulation that makes it possible to reduce the search space significantly.

RTDARS was evaluated on historic taxi trips from the New York City Taxi and Limousine Commission [8], which contains large-scale instances with more than 30,000 requests an hour. The results indicated that RTDARS enables a real-time dial-a-ride service to provide service guarantees (every rider is served in reasonable time) while improving average waiting times and average trip deviations compared to prior work. The results also showed that larger occupancy vehicles bring benefits and that the fleet size can be further reduced while preserving very reasonable waiting times.

Substantial work remains to be done to understand the strengths and limitations of the approach. The current implementation is myopic and heavily driven by the dual costs to generate the columns. Different pricing implementation, including the use of constraint programming to replace our dedicated search algorithm, and the inclusion of stochastic information are natural directions for future research.

Acknowledgments

This research was partly supported by Didi Chuxing Technology Co. and Department of Energy Research Grant 7F-30154. We would like to thank the reviewers for their detailed comments and suggestions which dramatically improved the paper, and the program chairs for a rebuttal period that was long enough to run many experiments.

References

1. Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D.: On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proceedings of the National Academy of Sciences **114**(3), 462–467 (2017)
2. Bent, R., Van Hentenryck, P.: Waiting and relocation strategies in online stochastic vehicle routing. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 1816–1821. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007), http://dl.acm.org/citation.cfm?id=1625275.1625569
3. Bent, R.W., Van Hentenryck, P.: Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research **52**(6), 977–987 (2004)
4. Berbeglia, G., Cordeau, J.F., Laporte, G.: A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. INFORMS Journal on Computing **24**(3), 343–355 (2012)
5. Bertsimas, D., Jaillet, P., Martin, S.: Online vehicle routing : The edge of optimization in large-scale applications (2018)
6. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Annals of Operations Research **153**(1), 29–46 (Sep 2007)
7. Jain, S., Van Hentenryck, P.: Large neighborhood search for dial-a-ride problems. In: International Conference on Principles and Practice of Constraint Programming. pp. 400–413. Springer (2011)
8. NYC: Nyc taxi & limousine commission - trip record data, http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
9. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org (2017)
10. Ota, M., Vo, H., Silva, C., Freire, J.: A scalable approach for data-driven taxi ride-sharing simulation. In: 2015 IEEE International Conference on Big Data (Big Data). pp. 888–897 (Oct 2015)
11. Ota, M., Vo, H., Silva, C., Freire, J.: Stars: Simulating taxi ride sharing at scale. IEEE Transactions on Big Data **3**(3), 349–361 (Sept 2017)
12. www.secondwavemedia.com/concentrate/innovationnews/ritmorollout0443.aspx: RITMO app introduces on-demand mass transit at U-M, with plans to expand. Concentrate (2018)