Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Special Track on AI for Computational Sustainability and Human Well-being

Real-Time Dispatching of Large-Scale Ride-Sharing Systems: Integrating Optimization, Machine Learning, and Model Predictive Control

Connor Riley , **Pascal Van Hentenryck** and **Enpeng Yuan**
Georgia Institute of Technology, Atlanta, USA
{ctriley,pvh,eyuan8}@isye.gatech.edu

Abstract

This paper considers the dispatching of large-scale real-time ride-sharing systems to address congestion issues faced by many cities. The goal is to serve all customers (service guarantees) with a small number of vehicles while minimizing waiting times under constraints on ride duration. This paper proposes an end-to-end approach that tightly integrates a state-of-the-art dispatching algorithm, a machine-learning model to predict zone-to-zone demand over time, and a model predictive control optimization to relocate idle vehicles. Experiments using historic taxi trips in New York City indicate that this integration decreases average waiting times by about 30% over all test cases and reaches close to 55% for high-demand zones.

1 Introduction

Transportation Network Companies (TNCs) like Uber and Lyft have fundamentally transformed mobility in many cities, providing on-demand door-to-door transportation through mobile applications. They have also increased traffic in many cities: a recent study by Erhardt *et al.* [2019] showed that, between 2010 and 2016, weekday vehicle hours of traffic delay have increased by 62% in San Francisco. In contrast, it was estimated that the delay increase would have been 22% without TNCs. To address this issue, several cities have begun limiting the number of TNC vehicles on the road. Another way to tackle the underlying congestion and pollution issues is to build mobility systems that utilize ride-sharing systematically. A study by Alonso-Mora *et al.* [2017] showed that systematic ride-sharing may significantly reduce the number of vehicles needed to serve requests. Their results indicate that 98% of the historic demand for taxi services in NYC could be served with a much smaller taxi fleet, while maintaining short wait times. This paper continues this line of research and focuses on how to build a real-time dispatching and routing architecture that serves the needs of large-scale ride-sharing systems. It is envisioned that, in the future, these ride-sharing systems will be deployed using autonomous vehicles, guarantee service to all customers, and leverage advanced AI systems. In the transition period, they can be supported by human drivers provided that these drivers follow the instructions of the ride-sharing system.

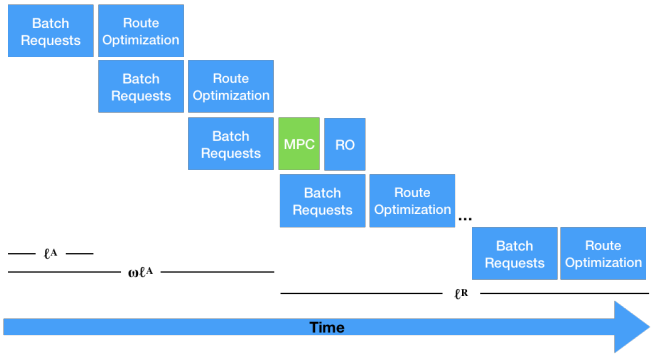

Figure 1: The A-RTRS Architecture for Real-Time Dispatching.

The value of stochastic information in real-time vehicle routing has been demonstrated previously by Bent and Van Hentenryck [2004]. However, incorporating stochastic information in large-scale ride-sharing systems, where requests may arrive every tenth of a second during peak times, is a challenge. It is thus not surprising that state-of-the-art approaches are purely myopic [Alonso-Mora *et al.*, 2017; Riley *et al.*, 2019]. These systems batch requests and optimize frequently to account for real-time demand. Other approaches to real-time dispatching (e.g., [Holler *et al.*, 2019]) use deep reinforcement learning, but they ignore ride-sharing and do not leverage advanced routing algorithms, focusing only on customer assignment. To incorporate stochastic information, this paper proposes a novel end-to-end framework (A-RTRS) for real-time ride-sharing systems that tightly integrates state-of-the-art optimization techniques, machine learning, and model predictive control.

The A-RTRS architecture is illustrated in Figure 1. Time is divided into epochs and, during epoch t, A-RTRS optimizes the routing of the requests that were batched in epoch $t-1$ as well as unserved requests from earlier epochs. Moreover, at a lower frequency and prior to the routing optimization, A-RTRS relocates idle vehicles using a Model Predictive Control (MPC) step. The MPC step does not operate on individual requests for scalability reasons. Rather it works with longer time periods and at a coarser zone level (e.g., taxi zones in New York City), and relies on a machine-learning

model to predict the number of requests between each pair of zones over time. *The main contribution of* A-RTRS *is to demonstrate that, in large-scale real-time ride-sharing systems, hybridizing state-of-the-art optimization algorithms for fine-grained routing decisions with model predictive control for idle vehicle relocation at a coarser space and time granularity provides significant operational benefits.* Indeed, results on historic taxi trips from the New York City Taxi and Limousine Commission [NYC, 2019], indicate that this tight integration decreases average waiting times by about 30% over all test cases and reaches close to a 55% reduction in average waiting times for high-demand zones.

This paper is organized as follows. Section 2 presents the problem. Section 3 presents related work. Section 4 gives an overview of A-RTRS. Section 5 describes the dial-a-ride optimization. Sections 6 and 7 present the core conceptual contributions of the paper: the demand forecasting model and the vehicle relocation scheme. Section 8 reports the experimental results and Section 9 concludes the paper.

2 Problem Statement

Operating a real-time ride-sharing system requires solving large-scale dial-a-ride problems, where each request corresponds to a trip for a number of riders from an origin to a destination that must take place after a specified pickup time. Constraints limit the time a rider can spend inside a vehicle (ride duration constraints) and the number of riders in a vehicle at any one time (vehicle capacity constraints). The goal is to serve all requests and minimize the average waiting time, while satisfying the ride duration and capacity constraints. Special attention is also devoted to ensure that no request is left unserved indefinitely. The systems studied in this paper either use a fleet of autonomous vehicles or their own pool of drivers who follow routing instructions exactly. The system can thus relocate the vehicles at will in order to anticipate demand. It is assumed that significant historical data is available and can be used to forecast demand at the zone level.

3 Related Work

A comprehensive review of popular dial-a-ride formulations which serve as the foundation of A-RTRS can be found in [Cordeau and Laporte, 2007]. A-RTRS uses a rolling horizon, alternating request batching and optimization, as traditionally used in taxi pooling [Ota *et al.*, 2017; Ota *et al.*, 2015]. Alonso-Mora *et al.* [2017] were first to demonstrate the value of ride-sharing in NYC: they showed that 98% of the historic demand could be served with a smaller taxi fleet and short wait times. Their anytime algorithm uses cliques to generate vehicle routes and hard-time windows to discard requests which cannot be served efficiently. A linear program is employed to move idle vehicles towards discarded requests in order to better serve those areas in the future. Lowalekar *et al.* [2018] improve over [Alonso-Mora *et al.*, 2017] by partitioning the region into zones and assigning vehicles to zone paths. Riley *et al.* [2019] is the first algorithm designed to serve *all requests* with small waiting times: they use column generation to serve all requests with smaller number of vehicles and shorter waiting times. Their dial-a-ride algorithm is used as the dispatching engine of A-RTRS. *To the author's knowledge, no algorithm other than the one of Riley* et al. *[2019] provides guarantees to serve all requests: They can decide to ignore arbitrary requests.* Note that these three algorithms are myopic: they do not exploit information about future requests. Iglesias *et al.* [2017] proposed a model predictive control approach for serving individual requests at the zone level, combining a machine-learning model (based on deep learning) and a mixed-integer program for request assignments and vehicle relocation. They did not consider ride-sharing and their dispatching algorithm is performed at a coarse granularity. This paper leverages and generalizes their model predictive control approach. Ma *et al.* [2019] integrated dispatching optimization and model predictive control for scheduling requests in a multimodal transit systems: they do not batch requests, use a single period for vehicle relocation, and assume Poisson arrivals for each zone. The dispatching of each request uses local search and insertion heuristics. The benefits of demand prediction and vehicle relocation has been demonstrated by Bent and Van Hentenryck [2007; 2004] for various types of vehicle routing problems (using online stochastic optimization) and by Yu and Shen [2019] for on-demand ride-pooling, using approximate dynamic programming. Holler *et al.* [2019] used deep learning and bipartite matching for dispatching and vehicle relocation: their approach does not support ride sharing. Shah *et al.* [2020] enhance [Alonso-Mora *et al.*, 2017] with an approximation of the future reward learned using a deep neural network. They provide improvements over [Alonso-Mora *et al.*, 2017] when the ride duration is twice as long as the shortest path. However, the approach does not provide service guarantees and does not minimize waiting times. It also rejects requests even when vehicles are available, which can be problematic to justify in practice. In contrast, this paper serves all requests with an average waiting time of 2.5 minutes with 2,000 vehicles during peak times and a more realistic ride-duration constraint (50% increase). The socio-economic benefits of ride-sharing systems is explored by Bistaffa *et al.* [2019]. To the authors' knowledge, this paper is the first integration of advanced optimization techniques, machine learning, and model predictive control for the real-time vehicle dispatching and relocation of large-scale ride-sharing systems.

4 Overall Organization

This section gives an overview of the A-RTRS architecture. As depicted in Figure 1, A-RTRS divides time into epochs of length ℓ^A, i.e., $[0, \ell^A), [\ell^A, 2\ell^A), [2\ell^A, 3\ell^A), \ldots$. During epoch τ, A-RTRS batches incoming requests and performs an optimization that assigns prior requests to vehicles and routes them. The requests considered in this optimization are those batched in epoch $\tau - 1$, as well as unserved requests from earlier epochs. Periodically, A-RTRS performs a relocation optimization, which exploits a forecasting model to direct idle vehicles towards expected demand.

The Optimization Problem

The optimization problem receives as inputs a set of requests, each of which is characterized by its origin and destination, its

earliest pickup time, and its number of riders. The optimization has at its disposal a number of vehicles. Each vehicle is characterized by its departure location, its earliest departure time, its capacity, and its set of riders. Each rider is characterized by her dropoff location and the time she has already spent in the vehicle.

The starting location and departure time of a vehicle are given by the current state of the mobility system. If a vehicle is idle in the existing schedule, its starting location is its current position and its departure time is the beginning of the epoch (i.e., $\tau\ell^A$). If a vehicle is serving customers, its starting location is the first location it visits after the start of the epoch and the departure time is specified accordingly. The riders associated with a vehicle are those who have already been picked up and need to be dropped off. Hence, for every epoch, the optimization problem considers all the requests whose riders have not been picked up yet, while also scheduling the dropoffs of existing riders. Note that the optimization problems associated with two successive epochs may schedule a request differently as long as the request's riders have not been picked up. This gives a lot of flexibility to the real-time system at the cost of more complex optimization problems.

Given the computational complexity of the dial-a-ride problem that must be solved in real time, the optimization may not be able to serve all requests for some epochs. Hence, following Riley *et al.* [2019], A-RTRS associates a penalty with each request to ensure that the request is served in reasonable time. The penalty is increased after each epoch in which the request is not served. The optimization model minimizes a weighted sum of the average waiting time and the penalties associated with unserved requests.

Vehicle Relocation

Every ω epochs, A-RTRS performs a relocation of vehicles at the zonal level (e.g., taxi zones, census tracks, or traffic analysis zones). The goal is to determine the desired number of idle vehicles to move from zone i to zone j over the next period $(\tau\ell^A, \tau\ell^A + \ell^R)$, where ℓ^R is the length of the relocation period and is significantly larger than the epoch length ℓ^A. As a result, the relocation optimization operates at a much coarser granularity both in space and time.

This combination of micro- and macro-decisions for routing and relocation is one of the salient features of A-RTRS and is driven by the reality of the large-scale real-time ride-sharing systems, where the number of requests in each epoch makes it difficult computationally to exploit forecasting information during the routing decisions.

Forecasting the Demand

To inform the vehicle relocation, A-RTRS is assumed to have at its disposal historical data for the number of requests from zone i to zone j for every time period of length ℓ^R. This historical data is used to train a forecasting model of the demand.

5 The Dial-A-Ride Optimization

During each epoch, A-RTRS solves a generalized dial-a-ride optimization specified in Section 4. To perform this task, it borrows the algorithm from Riley *et al.* [2019], which is briefly reviewed in this section. The dial-a-ride optimization is based on a column generation that operates at the route level. A vehicle route specifies a sequence of pickups and dropoffs which satisfies the ride duration constraints and the vehicle capacity. The column generation interleaves the solving of (the linear relaxation of) a Restricted Master Problem (RMP), which selects routes, and pricing subproblems which generate new routes for each vehicle. The process terminates when no new routes can improve the solution of the RMP or the time limit for the column generation is met. The last stage of the dial-a-ride optimization is a mixed-integer program that solves the RMP exactly for the generated routes. The pricing subproblems are complex due to their objective of minimizing waiting times. As a result, traditional dynamic programming formulations are not effective and Riley *et al.* [2019] use an anytime exact algorithm that generates routes of increasing lengths.

$$\min \quad \sum_{r\in R} c_r y_r + \sum_{i\in P} p_i z_i \tag{1a}$$

$$\text{subject to} \left(\sum_{r\in R} y_r a_i^r\right) + z_i = 1 \qquad \forall i \in P \tag{1b}$$

$$\sum_{r\in R_v} y_r = 1 \qquad \forall v \in V \tag{1c}$$

$$z_i \in \mathbb{N} \qquad \forall i \in P \tag{1d}$$

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

Figure 2: The Resricted Master Problem Formulation.

The RMP is depicted in Figure 2. In the formulation, R denotes the set of routes, R_v denotes the subset of possible routes for vehicle v, c_r represents the wait times incurred by all customers served by route r, p_i is the penalty of not scheduling request i for this epoch, and $a_i^r = 1$ iff request i is served by route r. The RMP uses the following decision variables: $y_r \in [0,1]$ is 1 iff route r is selected and $z_i \in [0,1]$ is 1 iff request i is not served by any of the selected routes. The objective function minimizes the waiting times of the served customers and the penalties for the unserved customers. Constraints (1b) ensure that z_i is set to 1 if request i is not served by any of the selected routes and constraints (1c) ensure that only one route is selected per vehicle.

Since the dial-a-ride optimization may not schedule all the requests, it is important to update the penalty of unserved requests to ensure that they will not be delayed too long. For the penalty for an unserved request c in epoch τ, Riley *et al.* [2019] use $p_c = \rho 2^{(\tau\ell^A - e_c)/(10\ell^A)}$, where e_c is the earliest possible pickup time for request c. The ρ parameter was tuned to incentivize the algorithm to schedule each incoming request in its first available epoch.

6 Demand Forecasting

Forecasting the demand from zone i to zone j over time may be challenging in some settings, since this demand may be sparse for some zones and historical data may be limited. To

address this difficulty, A-RTRS proceeds in two steps: it first predicts the number of requests in a zone z in time period t and then approximates the zone-to-zone demand.

Preprocessing
Let d_{zt} be the demand for zone z during period t. In the case study, the time series $\{d_{zt}\}_t$ is strongly non-stationary (the mean and variance vary over time). As a result, the forecasting model first stationarizes the time series by differencing it over a week period. More precisely, the forecasting model defines $\delta_{zt} = d_{zt} - d_{z(t-n_r\times 7)}$, where n_r is the number of periods in a day, and predicts the differenced demand δ_{zt} instead of d_{zt}.

Vector Autoregression
To forecast the time series $\{\delta_{zt}\}_t$, A-RTRS uses Vector Autoregression (VAR), a multivariate generalization of Autoregression (AR). In VAR, the expected value of a multivariate time series at a particular period is assumed to be a linear function of the value of the time series at previous time steps.

The prediction for the differenced demand in zone z in period t uses not only z's demand in prior periods but also the differenced demands of z's adjacent zones. Let $N(z)$ denote the zones adjacent to z and $d = |N(z)| + 1$. Let vector $\Delta_{zt} \in \mathbb{R}^d$ denote the weekly-differenced demands of z and its adjacent zones in period t. Each element in Δ_{zt} is an element in $\{\delta_{zt}\}_{z\in N(z)\cup\{z\}}$. δ_{zt} can then be modeled as

$$\delta_{zt} = \phi_{zt-1}\Delta_{zt-1} + \cdots + \phi_{zt-k}\Delta_{zt-k} + \eta \qquad (2)$$

where ϕ_{zt} is a row vector in $\mathbb{R}^d$ and η is a white noise with zero mean. The coefficients ϕ_{zt} are estimated using least square regression and the order k is selected based on the *Akaike information criterion (AIC)*.

Once the parameters have been estimated, the prediction for the differenced demand of z in period t is given by

$$\bar{\delta}_{zt} = \bar{\phi}_{zt-1}\Delta_{zt-1} + \cdots + \bar{\phi}_{zt-k}\Delta_{zt-k}$$

The demand prediction for zone z at time t is then given by

$$\bar{\lambda}_{zt} = d_{z(t-n_r\times 7)} + \bar{\delta}_{zt}.$$

Destination Assignment
Given the number of requests for zone z in period t, the trip destinations for these requests are assigned using historical distributions. A-RTRS uses an historical distribution for each hour during the weekdays and the weekend days. For example, when predicting the demand in zone z during the 7–8am period on a Wednesday, if 70 percent of the trips originating from z during this period on weekdays have their destination in zone b in historical data, then the number of trips going from z to b will be $0.7\bar{\lambda}_{zt}$ rounded to the nearest non-negative integer. This returns the final demand prediction $\bar{\lambda}_{ijt}$ for the requests from zone i to zone j at t.

7 Idle Vehicle Relocation

The idle vehicle relocation process is run every ω epochs and considers periods of length ℓ^R, i.e., $[0, \ell^R), [\ell^R, 2\ell^R), \ldots$ It has at its disposal the zone to zone demand forecasts for each period. It proceeds in two steps: (1) It first uses a Model Predictive Control (MPC) approach to find the desired number of vehicles in each zone; (2) It then selects the vehicle relocation assignments to minimize the relocation cost.

$$\min \sum_{t=0}^{T-1}\sum_{i,j\in Z}(T-t)u_{ijt} + \sum_{t=0}^{T-1}\sum_{i,j\in Z} tt_{ij}x^r_{ijt} \qquad (3a)$$

subject to

$$x^p_{ij0} + u_{ij0} = \lceil \bar{\lambda}_{ij0}/w_{ij} \rceil \quad (\forall i,j) \qquad (3b)$$

$$x^p_{ijt} + u_{ijt} = \lceil \bar{\lambda}_{ijt}/w_{ij} \rceil + u_{ijt-1} \quad (\forall i,j,t) \qquad (3c)$$

$$\textstyle\sum_j x^p_{ijt} + x^r_{ijt} = |A_{it-1}| + \sum_j x^p_{jit-tt_{ji}} + x^r_{jit-tt_{ji}} \ (\forall i,t) \qquad (3d)$$

$$x^p_{ijt} \in \mathbb{Z}, x^r_{ijt} \in \mathbb{Z}, u_{ijt} \in \mathbb{Z} \qquad (3e)$$

Figure 3: The MPC-MIP Model for Zone Balancing.

Zone Rebalancing
The MPC approach in this section is borrowed from Iglesias *et al.* [2017] and generalized to real-time ride-sharing systems, where multiple riders can share a vehicle. Its goal is to determine the number of idle vehicles to move from zone i to zone j during the next period $(\tau\ell^A, \tau\ell^A + \ell^R)$ in order to minimize the average waiting time in the dial-a-ride optimizations. To achieve this goal, the MPC approach solves an assignment optimization at the zone level over multiple time periods. Hence, in contrast to the dial-a-ride optimization, it works at a coarser granularity and looks ahead in the future using the demand forecasting module.

The MPC approach uses a MIP model (MPC-MIP) over T periods, each of length ℓ^R. Let $\mathcal{T} = \{0, \ldots, T-1\}$. For each period $t \in \mathcal{T}$, MPC-MIP takes as input $\bar{\lambda}_{ijt}$, the forecasted demand originating in zone i with a destination in zone j at time t, as well as a variety of information about the current state of the system. In particular, w_{ij} is the current sharing ratio for requests from zone i to zone j in the system (e.g., $w_{ij} = 1$ means that riders are alone in the vehicle, while $w_{ij} = 1.5$ means an average of 1.5 passengers per vehicle); A_{it} is the set of vehicles that will become idle in zone i during period t (estimated from routes of current vehicles) and tt_{ij} is the average number of periods it takes to move from a stop in zone i to a stop in zone j.

The MPC-MIP decision variables are: the number x^r_{ijt} of empty vehicles to move from zone i to zone j starting at period t; the number x^p_{ijt} of vehicles with passengers moving from i to j starting at period t; and the number u_{ijt} of requests originating in zone i and ending in zone j which are not served at period t. The objective (3a) minimizes the number of unserved requests while also enforcing a small penalty on moving vehicles. This penalty ensures that vehicles prefer to stay in their current zone if that current zone is expected to need them in the future. Constraints (3c) and (3c) are the flow balance constraints for requests. Constraints (3d) are the flow conservation constraints for vehicles.

Vehicle Relocation
MPC-MIP returns the number $\bar{x}^r_{ijt}$ of vehicles that should move from zone i to zone j in period t. Only the relocations in period $\bar{x}^r_{ij0}$ are relevant for A-RTRS at this point in time. However, A-RTRS must now identify $\bar{x}^r_{ij0}$ specific ve-

$$\min \quad \sum_{v \in A_{i0}} \sum_{j} c_{vj} y_{vj} \tag{4a}$$

$$\text{subject to} \sum_{v \in A_{i0}} y_{vj} = \bar{x}^r_{ij0} \qquad \forall j \in Z \tag{4b}$$

$$\sum_{j} y_{vj} \leq 1 \qquad \forall v \in A_{i0} \tag{4c}$$

$$y_{vj} \in \{0,1\} \qquad \forall v \in A_{i0}, j \in Z \tag{4d}$$

Figure 4: The Vehicle Relocation Optimization (VR-MIP).

hicles to relocate. This is performed by another MIP model (VR-MIP), which receives the following inputs: $\bar{x}^r_{ij0}$, the sets A_{i0} of idle vehicles in zone i, the time c_{vj} to move vehicle v to its closest stop in zone j. VR-MIP decides whether a vehicle is relocated to a zone: Variable y_{vj} is 1 if vehicle v is chosen to relocate to zone j. The VR-MIP objective (4a) minimizes the sum of the traveling times, Constraints (4b) ensure the correct number of relocations from zone i to zone j, and Constraints (4c) ensure that a vehicle does not relocate to more than one zone.

8 Experimental Results

Case Study A-RTRS is evaluated on the yellow trip data obtained via the New York City Taxi and Limousine Commission (NYCTLC) [NYC, 2019]. The NYCTLC dataset provides the number of passengers for each trip and the start time of each trip, which is used as both the request time and the lower bound on the earliest pickup time. This section reports results on 24 instances, two hours each day for two days per month from July 2015 through June 2016. Rush hours (7–9am) were selected to obtain instances that are computationally challenging. The instances have an average of 48,100.5 customers and range from 19,276 to 59,820 customers. Individual requests with more customers than the vehicle capacity are split into several trips. Following Riley *et al.* [2019], in order to ease ride sharing, avoid curb management issues, and reduce the number of stops, Manhattan is represented by a grid with cells of 200 squared meters. Each such cell represents a pickup/dropoff location. The travel time matrix for the network of locations was then precomputed by querying OpenStreetMap [OpenStreetMap contributors, 2017]. For every trip, the locations of the origin and destination were obtained by selecting the closest locations to their pickup and dropoff points in the NYCTLC dataset.

Runtime Configurations A-RTRS is compared to its myopic version M-RTRS which has no idle vehicle relocation and is essentially the approach proposed by Riley *et al.* [2019]. Unless otherwise specified, all experiments were performed with the following default parameters for A-RTRS: 2000 vehicles of capacity 4, a maximum deviation from the shortest path determined by $\max\{\alpha t_c, \beta + t_c\}$, where t_c is the shortest possible path from customer c's origin to their destination, $w_{ij} = 1.2$, $\rho = 420$, $\alpha = 1.5$, $\beta = 240$ seconds, $\ell^A = 30$ seconds, $\ell^R = 300$ seconds, and $\omega = 10$ epochs. In other words, requests are accumulated for, and optimized over, each period of 30 seconds, simulating real-time operations. Empty vehicles are initially distributed evenly over the locations. The demand was forecasted at the hour level and scaled down uniformly due to the sparse demand in some of the zones. All MIP models are solved using Gurobi 8.1 [Gurobi Optimization Inc, 2016]. The experiments were performed on a virtual configuration composed of 24 intel skylake cores clocked at 2.1GHz and with 32GB of memory.

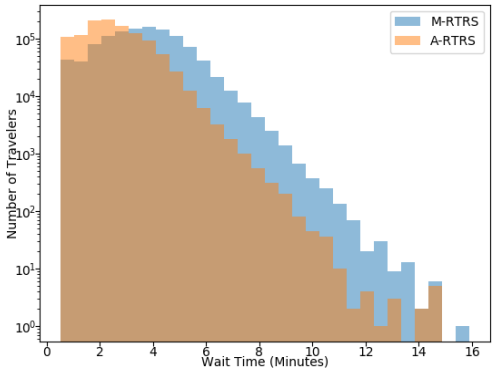

Figure 5: Histograms of Waiting Times for A-RTRS and M-RTRS (Logarithmic Y-Scale).

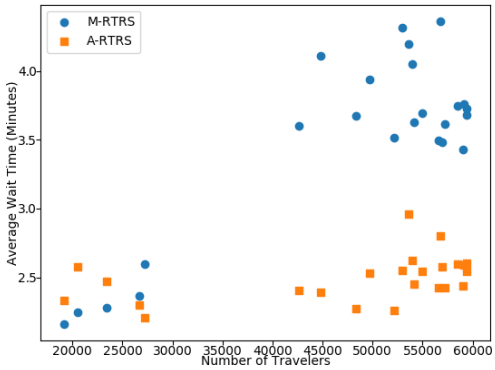

Figure 6: Waiting Times for A-RTRS and M-RTRS.

Number of Requests	M-RTRS	A-RTRS
$< 40{,}000$	2.33	2.37
$40{,}000 - 50{,}000$	3.83	2.40
$50{,}000 <$	3.78	2.56

Figure 7: Average Waiting Times by Instance Sizes.

Reduction in Waiting Times Figure 5 reports the distributions of the waiting times incurred by all customers across all instances with a logarithmic y-axis. The results demonstrate that A-RTRS reduces waiting times across the vast majority of trips. It reduces the average waiting times from 3.64 to 2.51 minutes (a 30% improvement) while also decreasing the standard deviation from 1.48 to 1.16. Figures 6 and 7 go into more details and report results on each instance and by instance sizes.

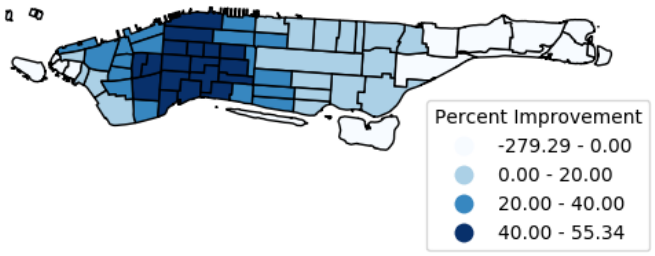

Figure 8: Percentage Improvement in Waiting Times by Origin.

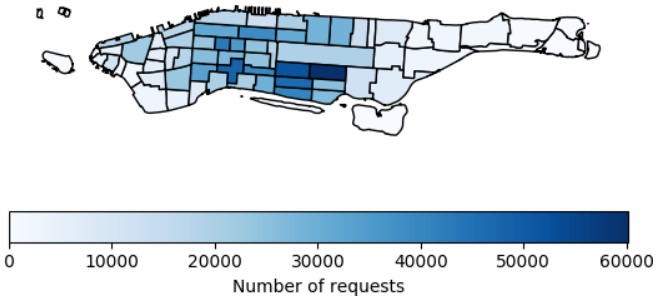

Figure 9: Total Number of Requests By Zone (over all instances).

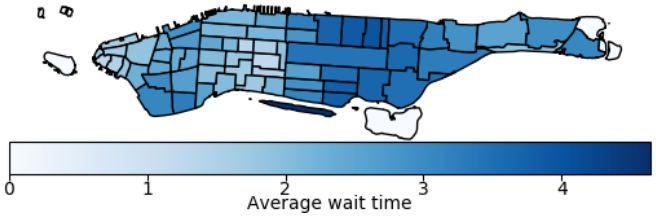

Figure 10: Average Waiting Times by Zone.

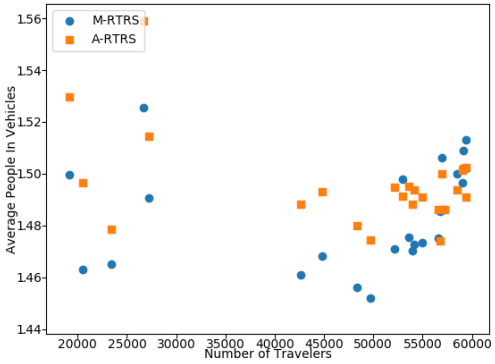

Figure 11: Average Number of Passengers per Vehicle.

Zonal Information Figures 8 and 9 describe where the reductions in waiting times occur. Together, they show that relocation benefits most the zones with significant demand, where the improvements can reach almost 55%. Figure 10 is reassuring from a fairness standpoint: Zones with low demand keep low average waiting times under relocation.

Vehicle Information Figure 12 depicts the histograms of idle times per vehicles. It shows that M-RTRS has more vehicles with no idle time and more vehicles with high idle times. A-RTRS is more balanced.

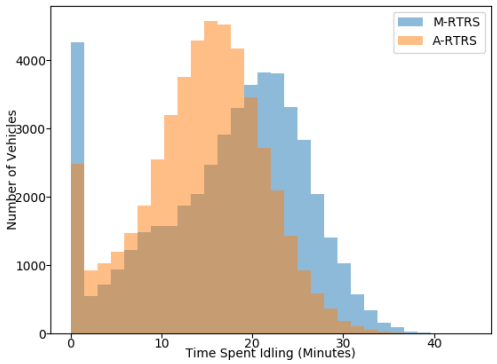

Figure 12: Idling Times per Vehicle.

9 Conclusion

This paper proposed A-RTRS, an end-to-end framework for real-time optimization of large-scale ride-sharing systems. A-RTRS combines demand forecasting, state-of-the-art optimization, and model predictive control to dispatch, route, and relocate vehicles in real-time, minimizing average waiting times. The mobility system provides service guarantees (i.e, it serves all requests), enforces a ride-duration constraint (i.e., no passenger travels more than 50% over their shortest path), minimizes waiting times, while achieving reasonable waiting times through penalties increasing over time. Experiments using historic taxi trips in New York City indicate that this integration decreases average waiting times by about 30% over all test cases and reaches close to 55% on the largest instances for high-demand zones compared to a base line without relocation. On the NYC case study, A-RTRS serves all requests in reasonable time and with an average waiting of 2.51 minutes with a standard deviation of 1.16, using a fleet of 2,000 vehicles of capacity 4. The results also demonstrate that, while zones with large demand see the most benefits, zones with low demand maintain low waiting times and that the vast majority of vehicles spend less than 17% of their operating time relocating. In summary, A-RTRS demonstrates that, in large-scale real-time ride-sharing systems, hybridizing state-of-the-art optimization algorithms for fine-grained routing decisions with model predictive control for idle vehicle relocation at a coarser space and time granularity provides significant operational benefits. Future research will be devoted in improving the machine-learning algorithm, since more accurate predictions will enable a better performance on instances with relatively fewer requests.

Acknowledgements

This research was partly supported by Department of Energy Research Award 7F-30154 and NSF Leap-HI Award NSF-1854684.

References

[Alonso-Mora *et al.*, 2017] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela

Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[Bent and Van Hentenryck, 2004] Russell W. Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

[Bent and Van Hentenryck, 2007] Russell Bent and Pascal Van Hentenryck. Waiting and relocation strategies in online stochastic vehicle routing. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1816–1821, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[Bistaffa *et al.*, 2019] Filippo Bistaffa, Chistian Blum, Jesuus Cerquides, Alessandro Farinelli, and Juan A. Rodríguez-Aguilar. A computational approach to quantify the benefits of ridesharing for policy makers and travellers. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2019.

[Cordeau and Laporte, 2007] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, Sep 2007.

[Erhardt *et al.*, 2019] Gregory D. Erhardt, Sneha Roy, Drew Cooper, Bhargava Sana, Mei Chen, and Joe Castiglione. Do transportation network companies decrease or increase congestion? *Science Advances*, 5(5), 2019.

[Gurobi Optimization Inc, 2016] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016.

[Holler *et al.*, 2019] John Holler, Risto Vuorio, Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jie ping Ye. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. *ArXiv*, abs/1911.11260, 2019.

[Iglesias *et al.*, 2017] Ramón Iglesias, Federico Rossi, Kevin Wang, David Hallac, Jure Leskovec, and Marco Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. *CoRR*, abs/1709.07032, 2017.

[Lowalekar *et al.*, 2018] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Zac: A zone path construction approach for effective real-time ridesharing. In *Proceedings of the 2018 Conference on Automated Planning and Scheduling*, 2018.

[Ma *et al.*, 2019] Tai-Yu Ma, Saeid Rasulkhani, Joseph Y.J. Chow, and Sylvain Klein. A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. *Transportation Research Part E: Logistics and Transportation Review*, 128:417 – 442, 2019.

[NYC, 2019] NYC. Nyc taxi & limousine commission - trip record data, 2019.

[OpenStreetMap contributors, 2017] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2017.

[Ota *et al.*, 2015] Masayo Ota, Huy Vo, Claudio Silva, and Juliana Freire. A scalable approach for data-driven taxi ride-sharing simulation. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 888–897, Oct 2015.

[Ota *et al.*, 2017] Masayo Ota, Huy Vo, Claudio Silva, and Juliana Freire. Stars: Simulating taxi ride sharing at scale. *IEEE Transactions on Big Data*, 3(3):349–361, Sept 2017.

[Riley *et al.*, 2019] Connor Riley, Antoine Legrain, and Pascal Van Hentenryck. Column generation for real-time ridesharing operations. In Louis-Martin Rousseau and Kostas Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 472–487. Springer International Publishing, 2019.

[Shah *et al.*, 2020] Sanket Shah, Meghna Lowalekar, and Pradeep Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *AAAI-2020*, 2020.

[Yu and Shen, 2019] Xian Yu and Siqian Shen. An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2019.