Journal of Artificial Intelligence Research 75 (2022) 985-1002 Submitted 03/2022; published 11/2022

Reinforcement Learning from Optimization Proxy for Ride-Hailing Vehicle Relocation

Enpeng Yuan EYUAN8@GATECH.EDU
Wenbo Chen WCHEN616@GATECH.EDU
Pascal Van Hentenryck PVH@GATECH.EDU
School of Industrial and Systems Engineering,
Georgia Institute of Technology,
Atlanta, GA 30332 USA

Abstract

Idle vehicle relocation is crucial for addressing demand-supply imbalance that frequently arises in the ride-hailing system. Current mainstream methodologies - optimization and reinforcement learning - suffer from obvious computational drawbacks. Optimization models need to be solved in real-time and often trade off model fidelity (hence quality of solutions) for computational efficiency. Reinforcement learning is expensive to train and often struggles to achieve coordination among a large fleet. This paper designs a hybrid approach that leverages the strengths of the two while overcoming their drawbacks. Specifically, it trains an optimization proxy, i.e., a machine-learning model that approximates an optimization model, and then refines the proxy with reinforcement learning. This Reinforcement Learning from Optimization Proxy (RLOP) approach is computationally efficient to train and deploy, and achieves better results than RL or optimization alone. Numerical experiments on the New York City dataset show that the RLOP approach reduces both the relocation costs and computation time significantly compared to the optimization model, while pure reinforcement learning fails to converge due to computational complexity.

1. Introduction

The rapid growth of ride-hailing markets has transformed urban mobility, offering on-demand mobility services via mobile applications. While major ride-hailing platforms such as Uber and Didi leverage centralized dispatching algorithms to find good matching between drivers and riders, operational challenges persist due to frequent imbalances between demand and supply. Consider morning rush hours as an example: most trips originate from residential areas to business districts where a large number of vehicles accumulate and remain idle. Relocating these vehicles back to the demand areas is crucial to maintaining quality of service and income for the drivers.

Extensive studies have focused on vehicle relocation problems in real time. Existing methodologies fit broadly into two categories: model-based and model-free approaches. Model-based approaches, e.g., Model Predictive Control (MPC), involve the solving of an optimization program using expected demand and supply information over a future horizon. Model-free approaches (predominantly Reinforcement Learning (RL)) train a state-based decision policy by interacting with the environment and observing the rewards. While both approaches have demonstrated promising performance in simulation and (in some cases) real-world deployment (Jiao et al., 2021), they have obvious drawbacks: the optimization

needs to be solved in real time and often trades off model fidelity (and hence solution quality) for computational efficiency. Reinforcement learning does not require a model but needs a large number of samples to train. Consequently, most works simplify the problem (e.g., by restricting relocations to nearby regions and/or limiting coordination among the fleet) to reduce computational complexity.

This paper addresses these challenges by proposing a Reinforcement Learning from Optimization Proxy (RLOP) approach that combines optimization, supervised learning, and reinforcement learning. The RLOP framework is a special case of Reinforcement Learning from Expert Demonstration (RLED) where the expert is an optimization algorithm (Ramírez, Yu, & Perrusquía, 2021). The RLOP approach consists of two main steps:

1. It first applies supervised learning to obtain an *optimization proxy* for an MPC optimization, i.e., it trains a machine learning model that approximates the mapping between the inputs of the MPC optimization and its actionable decisions (i.e., the outputs of the MPC for the first epoch).

2. It then seeds an RL component with the optimization proxy as the initial policy. The RL component further improves this policy by interacting with the environment, which captures the real system dynamics and long-term effects that are beyond the capabilities of the model-based optimization.

To the best of the authors' knowledge, this paper is the first application of an RLED framework to tackle vehicle relocation problems, and one of the few RL models with a fully centralized policy. The RLOP framework has three important benefits:

1. The optimization proxy approximates the model-based optimization with high fidelity and is order of magnitude faster.

2. The RL component is significantly easier to train since it starts with a high-quality policy.

3. The RL component improves the optimization proxy by capturing long-term effects and real system dynamics present in sample trajectories.

The application of RLOP framework to relocation problems comes with several challenges, which require additional methodological contributions.

1. The relocation decisions are typically high-dimensional (e.g., number of vehicles to relocate between each zone) and sparse (most vehicles relocate to only a few popular zones). This creates great difficulty for supervised and reinforcement learning.

2. The predicted relocation decisions may be infeasible since most learning algorithms cannot enforce integrality or physical constraints that the decisions need to satisfy.

To tackle these challenges, this paper proposes an *aggregation-restoration-disaggregation procedure* which predicts the relocation decisions at an aggregated level, restores them back to feasible solutions, and then disaggregates them to the original granularity by applying a *polynomial-time* transportation optimization. As a result, the dimensionality and sparsity of the decisions are reduced considerably, and the approach remains computationally efficient.

The proposed RLOP framework is evaluated on the New York Taxi data set, using the optimization and simulation architecture presented by Riley, Van Hentenryck, and Yuan (2020). The experimental results reveal two interesting findings:

1. The optimization proxy learns the relocation optimization with high fidelity, producing similar objective values at a fraction of the optimization's computing time.

2. the RL component further reduces the relocation costs by 10% compared to the optimization proxy, whereas pure centralized reinforcement learning is too expensive computationally to be applied.

These results suggest that the RLOP framework provides a promising approach for the real-time operations of ride-hailing systems. It is also important to stress that the RLOP framework is general and can be applied with any relocation optimization and RL techniques.

The rest of the paper is organized as follows. Section 2 summarizes the existing literature. Section 3 defines the relocation problem. Section 4 reviews the existing relocation model used in the simulation experiments. Section 5 presents the learning framework. Section 6 reports the experimental results on a large-scale case study in New York City.

2. Related Work

Prior works on real-time idle vehicle relocation problem fit broadly into two frameworks: *model predictive control (MPC)* and *reinforcement learning (RL)*. MPC is an online control procedure that repeatedly solves an optimization problem over a moving time window to find the best control actions. System dynamics, i.e., the interplay between demand and supply, are explicitly modeled as mathematical constraints. Due to computational complexity, almost all the MPC models in the literature work at a discrete spatial-temporal scale (i.e., the dispatch area is partitioned into zones and time is divided into epochs) and use a relatively coarse granularity with a small number of zones or epochs (Miao et al., 2015, 2017; Iglesias et al., 2017; Tsao et al., 2018; Riley et al., 2020).

Reinforcement learning, on the contrary, does not explicitly model the system dynamics. It trains a decision policy by interacting with the environment and observing the rewards. It can be divided into three streams: single-agent RL (Wen, Zhao, & Jaillet, 2017), decentralized multi-agent RL (Oda & Joe-Wong, 2018; Guériau & Dusparic, 2018; Holler et al., 2019; Lin et al., 2018; Jiao et al., 2021; Liang et al., 2021), and centralized multi-agent RL (Mao, Liu, & Shen, 2020). The single-agent framework maximizes reward of an individual agent, while the multi-agent framework maximizes system-level benefits. A main challenge of RL is training complexity since the state and action spaces are typically high-dimensional (often infinite-dimensional) due to the complex demand-supply dynamics. Sampling in high-dimensional spaces makes the training computationally expensive and unstable. Consequently, many works simplify the problem by enforcing agents within the same region to follow the same policy (Verma et al., 2017; Lin et al., 2018), or restricting relocations to only neighboring regions (Wen et al., 2017; Holler et al., 2019; Oda & Joe-Wong, 2018; Guériau & Dusparic, 2018; Lin et al., 2018; Jiao et al., 2021). Another challenge is promoting coordination among a large number of agents (vehicles). Single-agent framework

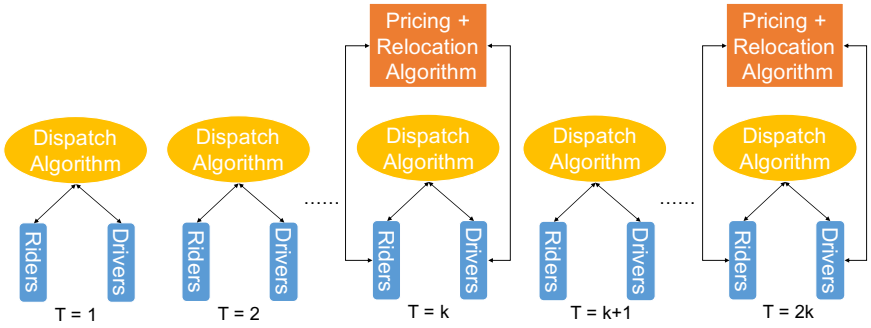

Figure 1: The Real-Time Ride-Hailing Operations.

focuses on a single vehicle and ignores group-level reward. Decentralized multi-agent framework considers group-level benefits only in a limited fashion since the state/action/reward of the individual agents are modeled separately. Centralized multi-agent framework considers the state and action of the agents jointly and has the potential to achieve maximal cooperation. However, the joint state-action spaces are extremely high-dimensional and make the problem computationally prohibitive. Mao et al. (2020) propose the only paper using a fully-centralized formulation. Similar to this paper, it models each dispatch zone instead of vehicle as an agent to simplify the state-action space. Nevertheless, the approach is demonstrated in a simple setting with a small number of zones due to computational complexity. The RLOP approach in this paper is able to demonstrate the fully-centralized approach on a much larger scale because of its computational efficiency.

The RLED approach has been applied extensively in robotics and games and achieved promising performance, a famous example being AlphaGo (Silver et al., 2016). However, it has not been employed to solve planning problems in ride-hailing systems. This paper explores this possibility through the RLOP framework.

3. Problem Definition

The real-time ride-hailing system, depicted in Figure 1, has three key components: vehicle routing, idle vehicle relocation, and dynamic pricing. The vehicle routing algorithm matches requests to vehicles and chooses the vehicle routes. It operates at the individual request level with high frequency (e.g., every $15-60$ seconds). Because of the tight time constraints and large number of requests, the routing algorithm is usually myopic, taking only the current demand and supply into account. Idle vehicle relocation and dynamic pricing, on the other hand, are forward-looking in nature. Idle vehicle relocation repositions the vehicles preemptively to anticipate demand, and dynamic pricing balances expected demand and supply in a future horizon. The two decisions also take place at a lower frequency (e.g., every $5-20$ minutes). In addition, the three components are interconnected. Take vehicle relocation as an example: the vehicle relocations depend on future demand as well as how the requests are scheduled, which are determined by the vehicle routing and pricing decisions. This paper focuses on idle vehicle relocation and abstracts away the other two components. The goal is to reduce rider waiting time by relocating idle vehicles while

accounting for the relocation costs. This paper assumes that the ride-hailing platform uses a fleet of autonomous vehicles or their own pool of drivers who follow instructions exactly - the platform can thus relocate the vehicles at will.

The Vehicle Routing Component To illustrate the relocation problem, it is helpful to review the vehicle routing component briefly. The simulation experiments in this paper use the routing algorithm in Riley, Legrain, and Van Hentenryck (2019) which is reviewed here as an example. The algorithm batches requests into a time window and optimizes every 30 seconds. Its objective is to minimize a weighted sum of passenger waiting times and penalties for unserved requests. Each time a request is not scheduled by the routing optimization, its penalty is increased in the next time window giving the request a higher priority. The routing algorithm is solved by column generation: it iterates between solving a restricted master problem (RMP), which assigns a route (sequence of pickups and dropoffs) to each vehicle, and a pricing subproblem, which generates feasible routes for the vehicles. The RMP is depicted below.

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

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

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

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

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

R denotes the set of routes. V the set of vehicles, and P the set of passengers. R_v denotes the subset of feasible routes for vehicle v. A route is feasible for a vehicle if it does not exceed the vehicle capacity and does not incur too much of a detour for its passengers due to ride-sharing. c_r represents the wait times incurred by all customers served by route r. p_i is the penalty of not scheduling request i, and $a_i^r = 1$ iff request i is served by route r. Decision variable $y_r \in [0, 1]$ 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. The column generation process terminates when the pricing subproblem cannot generate new routes to improve the solution of the RMP or the solution time limit is met.

4. The Relocation MPC Model

The first step of the RLOP framework uses supervised learning to approximate the first-stage decisions of an relocation optimization. The optimization model makes decisions at the zone-to-zone level, i.e., the number of vehicles to relocate from one zone to another. The discussion of the learning methodology and the simulation experiments are based on the model proposed by Yuan and Van Hentenryck (2021) reviewed in this section. Note

Model Input	
V_{it}	Number of vehicles that will become idle in i during t
D_{ijt}	Number of vehicles needed to serve riders from i to j who places their requests during t
λ_{ij}	Number of epochs to travel from i to j
Model Parameters	
s	Number of epochs that a rider remains in the system
W_{ij}	Average number of riders from i to j that a vehicle carries
$q^p(t,\rho)$	Weight of a rider served at ρ whose request was placed at t
$q^r_{ij}(t)$	Relocation cost between i and j in t
Decision Variables	
$x^r_{ijt} \in \mathbb{Z}_+$	Number of vehicles starting to relocate from i to j during t
Auxiliary Variables	
$x^p_{ijt\rho} \in \mathbb{Z}_+$	Number of vehicles that start to serve at time ρ riders going from i to j whose requests were placed at t
$l_{it} \in \{0,1\}$	Whether there is unserved demand in i at the end of epoch t

Table 1: The Nomenclature for the MPC Optimization.

however that the RLOP framework is general and not confined to any particular relocation optimization.

Yuan and Van Hentenryck (2021) follows the Model Predictive Control (MPC) framework. The MPC is a rolling time horizon approach that discretizes time into epochs of equal length and performs three tasks at each decision epoch: (1) it predicts the demand and supply for the next T epochs; (2) it optimizes relocation decisions over these epochs; and (3) it implements the decisions of the *first* epoch only. The dispatch area is partitioned into zones (not necessarily of equal size or shape) and relocation decisions are made at the zone-to-zone level. The model assumes that vehicles only pick up demand in the same zone and that vehicles, once they start delivering passengers or relocating, must finish their current trip before taking another assignment. These assumptions help the MPC model *approximate* the behavior of the underlying vehicle-routing algorithm, but the routing algorithm does not have to obey these constraints. The only interactions between the routing optimization and the MPC are the relocation decisions. To model reasonable waiting times, riders can only be picked up within s epochs of their requests: they would drop out if waiting more than s epochs.

The nomenclature of the model is summarized in Table 1. In the formulation, i and j denote zones, and t_0, t, and ρ are epochs. $\mathcal{Z}$ denotes the set of zones in the dispatch area and $\mathcal{T} = \{1, 2, ..., T\}$ the set of time epochs in the planning horizon. The ride-sharing coefficient W_{ij} represents the average number of riders traveling from i to j that a vehicle carries accounting for ride-sharing. Expected supply V_{it} can be estimated based on the

current route of the vehicles and travel time. Expected demand D_{ijt} can be forecasted based on historical demand. The time-dependent weights $q^p(t,\rho)$ and $q^r_{ij}(t)$ are designed to favor serving requests and performing relocations early: they are decreasing in t and ρ.

The decision variables x^r_{ijt} capture the relocation decisions. Although decisions are made for each epoch in the time horizon, only the first epoch's decisions are actionable and implemented: the next MPC execution will reconsider the decisions for subsequent epochs. Note that the auxiliary variables $x^p_{ijt\rho}$ are only defined for a subset of the subscripts, since riders drop out if they are not served in reasonable time. The valid subscripts for variables $x^p_{ijt\rho}$ must satisfy the constraint $1 \le t \le \rho \le \min(T, t+s-1)$. These conditions are implicit in the model for simplicity. Furthermore, $\phi(t) = \{\rho \in \mathcal{T} : t \le \rho \le t+s-1\}$ denotes the set of valid pick-up epochs for riders placing their requests in epoch t.

$$\max \quad \sum_{i,j}\sum_{t,\rho} q^p(t,\rho) W_{ij} x^p_{ijt\rho} - \sum_{i,j}\sum_{t} q^r_{ij}(t) x^r_{ijt}$$

$$\text{s.t.} \quad \sum_{\rho\in\phi(t)} x^p_{ijt\rho} \le D_{ijt}, \qquad \forall i,j,t \qquad (2a)$$

$$\sum_{j,t_0} x^p_{ijt_0t} + \sum_{j} x^r_{ijt} = V_{it} + \sum_{j,t_0} x^p_{jit_0(t-\lambda_{ji})} + \sum_{j} x^r_{ji(t-\lambda_{ji})}, \quad \forall i,t \qquad (2b)$$

$$\sum_{j} x^r_{ijt} \le M l_{it}, \qquad \forall i,t \qquad (2c)$$

$$\sum_{j} \sum_{t_0:\, t\in\phi(t_0)} \left(D_{ijt} - \sum_{\rho=t_0}^{t} x^p_{ijt_0\rho} \right) \le M(1-l_{it}), \qquad \forall i,t \qquad (2d)$$

$$x^p_{ijt\rho}, x^r_{ijt} \in \mathbb{Z}_+, \qquad \forall i,j,t,\rho \qquad (2e)$$

$$l_{it} \in \{0,1\}, \qquad \forall i,t \qquad (2f)$$

The model formulation is given above. The objective maximizes the weighted sum of riders served and minimizes the relocation costs. Constraint (2a) makes sure that the served demand does not exceed the true demand. Constraint (2b) is the flow balance constraint for each zone and epoch. Big-M constraints (2c) and (2d) prevent vehicles from relocating unless all demand in the zone is served, approximating the behavior of the routing algorithm which favors scheduling vehicles to nearby requests. Constraints (2e) and (2f) specify the ranges of the variables. The model is always feasible since all vehicles can remain idle and not serve any requests.

The model is a mixed-integer program (MIP), which is challenging to solve at high fidelity when the number of zones $|\mathcal{Z}|$ or the length of time horizon $|\mathcal{T}|$ is large. In addition, it is difficult to gauge how accurately the model (or any optimization model) approximates the true dynamics of the system. *This is the key motivation for approximating the MPC optimization by a computationally efficient machine-learning policy and refining it by reinforcement learning which interacts with the real system dynamics.*

5. The RLOP Framework

The RLOP framework has two stages: supervised learning and reinforcement learning. The supervised-learning stage trains an optimization proxy, i.e., a machine-learning model that approximates the actionable decisions of an optimization model. The reinforcement-learning stage takes the optimization proxy as the initial policy and refines it by a policy gradient method.

5.1 The Optimization Proxy

The supervised-learning stage trains a machine-learning model to predict the actionable decisions of an relocation model $\mathcal{M} : \mathcal{S} \to \mathcal{W}$, where $\mathcal{S}$ is the model input and $\mathcal{W}$ is the model decision, i.e., the number of vehicles to relocate between each zone in the dispatch area. Hence $|\mathcal{W}| = |\mathcal{Z}|^2$, where $|\mathcal{Z}|$ is the number of zones in the dispatch area. The training data can be generated by running $\mathcal{M}$ on a set of problem instances and extracting its results. *Without loss of generality, the presentation illustrates the supervised-learning methodology based on the MPC model from Section 4, but the framework applies to any relocation model as long as the decisions are at the zone-to-zone level.*

The machine-learning model takes the MPC model's input $\mathbf{s} = [D_{ijt}, V_{it}]_{i,j\in\mathcal{Z},t\in\mathcal{T}}$ - expected demand and supply in each zone and epoch - and predicts its first epoch decisions $\mathbf{w} = [x^r_{ij1}]_{i,j\in\mathcal{Z}}$ (only these decisions are actionable after each MPC execution). In reality, $\mathbf{w}$ is high-dimensional ($|\mathcal{W}| = |\mathcal{Z}|^2$) and sparse, since most vehicles relocate to a few high-demand zones. The high-dimensionality and sparsity makes supervised learning difficult. It also imposes significant challenges for RL in the second stage since sampling in high-dimensional action space $\mathcal{W}$ is expensive and makes the training unstable (see Section 5.2 for details). Therefore, this paper designs an aggregation-disaggregation procedure which predicts $\mathbf{w}$ at the aggregated (zone) level and then disaggregates the predictions via an efficient optimization procedure.

More precisely, the zone-level relocation decision $\mathbf{a} \in \mathcal{A}$ is predicted by the machine-learning model $\hat{\mathcal{O}}_\theta : \mathcal{S} \to \mathcal{A}$ and disaggregated to zone-to-zone level by an efficient optimization problem $\mathcal{TO} : \mathcal{A} \to \mathcal{W}$. To ensure that the machine-learning model can be refined by the policy gradient method in the RL stage, $\hat{\mathcal{O}}_\theta$ needs to be **differentiable** with respect to its parameters θ. For example, $\hat{\mathcal{O}}_\theta$ can be an artificial neural network or a linear regression parametrized by θ. The RLOP framework however is general and can accommodate any other machine-learning model.

5.1.1 Aggregation and Prediction

The zone-to-zone level decision $\mathbf{w} = [x^r_{ij1}]_{i,j\in\mathcal{Z}}$ is first aggregated to, and predicted at the zone level. More specifically, two metrics are predicted for each zone i:

1. the number of vehicles relocating from i to other zones, i.e., $x^o_i := \sum_{j\in\mathcal{Z},j\neq i} x^r_{ij1}$;
2. the number of vehicles relocating to i from other zones, i.e., $x^d_i := \sum_{j\in\mathcal{Z},j\neq i} x^r_{ji1}$.

These two metrics can be both non-zero at the same time: an idle vehicle might be relocated from i to another zone for serving a request in the near future, and another vehicle could come to i to serve a later request. The aggregated decisions $\mathbf{a} = [x^d_i, x^o_i]_{i\in\mathcal{Z}}$ are then

predicted by the chosen machine-learning model. This aggregation step reduces the label dimension from $|\mathcal{W}| = |\mathcal{Z}|^2$ to $|\mathcal{A}| = 2|\mathcal{Z}|$.

5.1.2 Disaggregation and Feasibility Restoration

The predicted relocation decisions $\hat{\mathbf{a}} = [\hat{x}_i^o, \hat{x}_i^d]_{i\in\mathcal{Z}}$ must be transformed to feasible solutions that are integer and obey flow balance constraints. This is performed in three steps:

1. $\hat{x}_i^o$ and $\hat{x}_i^d$ are rounded to their nearest non-negative integers;
2. to make sure that there are not more relocations than idle vehicles, take $\hat{x}_i^o = \min\{\hat{x}_i^o, V_{i1}\}$ where V_{i1} is expected number of idle vehicles in i in the first epoch;
3. $\hat{x}_i^o$ and $\hat{x}_i^d$ must satisfy the flow balance constraint, e.g., $\sum_{i\in\mathcal{Z}} \hat{x}_i^o = \sum_{i\in\mathcal{Z}} \hat{x}_i^d$: this is achieved by setting the two terms to be the minimum of the two, by randomly decreasing some non-zero elements of the larger term.

After a feasible relocation plan is constructed at the zone level, the disaggregation step reconstructs the zone-to-zone relocation via a transportation optimization $\mathcal{TO} : \mathcal{A} \to \mathcal{W}$. The model formulation is given below. Variable w_{ij} denotes the number of vehicles to relocate from zone i to zone j, and constant c_{ij} represents the corresponding relocation cost. The model minimizes the total relocation costs to consolidate the relocation plan. The solution w_{ij} will be implemented by the ride-hailing platform in the same way as x_{ij1}^r from the MPC. Note that w_{ii} should be 0 since $\hat{\mathbf{w}}$ denotes relocations into and out of each zone. However, the problem in that form may be infeasible. By allowing the w_{ii}'s to be positive and assigning a large value to the relocation costs c_{ii}, *the problem is always feasible, totally unimodular, and polynomial-time solvable* (Rebman, 1974).

$$\mathcal{TO}(\hat{\mathbf{a}}) = \arg\min_{\mathbf{w}} \quad \sum_{i,j\in\mathcal{Z}} c_{ij} w_{ij} \tag{3a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{Z}} w_{ij} = \hat{x}_i^o, \qquad \forall i \in \mathcal{Z} \tag{3b}$$

$$\sum_{j\in\mathcal{Z}} w_{ji} = \hat{x}_i^d \qquad \forall i \in \mathcal{Z} \tag{3c}$$

$$w_{ij} \in \mathbb{Z}^+ \qquad \forall i, j \in \mathcal{Z} \tag{3d}$$

5.2 Reinforcement Learning

The supervised-learning stage trains an optimization proxy $\hat{\mathcal{O}}_\theta : \mathcal{S} \to \mathcal{A}$ from an relocation model. The RL process starts from $\hat{\mathcal{O}}_\theta$ and improves it by a policy gradient method. Specifically, the RL step models the relocation problem as a Markov Decision Process (MDP). MDP is characterized by a tuple $< \mathcal{S}, \mathcal{A}, R, P, \gamma >$, which consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function $R(\mathbf{s}, \mathbf{a})$, a transition function $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, and a discount factor $\gamma \in [0, 1]$. At each decision epoch t in the planning horizon $\{0, 1, ..., T_e\}$, the agent observes the state of the system $\mathbf{s}_t \in \mathcal{S}$, takes an action $\mathbf{a}_t \in \mathcal{A}$, receives an immediate reward $R(\mathbf{s}_t, \mathbf{a}_t)$, and transitions to the next state according to the transition probability

$P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The goal is to find a stochastic decision policy $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ parametrized by θ, i.e., a mapping from the state space to a probability distribution over the action space that maximizes the total expected discounted reward

$$J(\theta) = \mathbb{E}_{P,\pi_\theta}\left[\sum_{t=0}^{T_e} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{4}$$

For the present application, the state and action space are the same as the input and output space of the optimization proxy $\hat{\mathcal{O}}_\theta : \mathcal{S} \to \mathcal{A}$ so that $\hat{\mathcal{O}}_\theta$ can be transformed into an initial policy for RL. The details of this transformation will be presented shortly. The reward function $R(\mathbf{s}_t, \mathbf{a}_t) = -u_t - \beta v_t$ is a weighted average of customer satisfaction and system cost, where u_t is the total waiting time of riders who emerges in epoch t and v_t is the expected time that vehicles will relocate due to action $\mathbf{a}_t$. Both u_t and v_t are in minutes. Parameter β denotes the relative importance of system cost compared to customer satisfaction, e.g., $\beta = 0.5$ implies that the platform is willing to relocate up to 2 minutes for a 1 minute reduction in waiting time. β depends on the platform's underlying objective and is taken as an input. The transition function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ depends on the underlying vehicle-routing algorithm, travel times, and demand arrival, and does not have a closed-form expression.

The policy is trained iteratively based on the policy gradient theorem (Sutton & Barto, 2018)

$$\nabla_\theta J(\theta) = \mathbb{E}_{P,\pi_\theta}\left[\sum_{t=0}^{T_e} G_t \nabla_\theta \log P_{\pi_\theta}(\mathbf{a}_t|\mathbf{s}_t)\right] \tag{5}$$

where $G_t = \sum_{\tau=t}^{T_e} \gamma^{\tau-t} R_\tau$ is the total (discounted) reward since epoch t in the trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, R_0, ..., \mathbf{s}_{T_e}, \mathbf{a}_{T_e}, R_{T_e})$ and $P_{\pi_\theta}(\mathbf{a}_t|\mathbf{s}_t)$ is the probability of taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$ under the decision policy π_θ. In reality, computing the expectation in (5) is intractable since P does not have a closed-form expression. The gradient is approximated by Monte-Carlo sampling, i.e.,

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_e} G_t^i \nabla_\theta \log P_{\pi_\theta}(\mathbf{a}_t^i|\mathbf{s}_t^i) \tag{6}$$

where $\{\tau_i\}_{i=1}^N = \{(\mathbf{s}_0^i, \mathbf{a}_0^i, R_0^i, ..., \mathbf{s}_{T_e}^i, \mathbf{a}_{T_e}^i, R_{T_e}^i)\}_{i=1}^N$ are trajectories generated by applying π_θ in the simulation environment.

It remains to specify how the optimization proxy $\hat{\mathcal{O}}_\theta$ can be turned into an initial policy for RL. Recall that $\hat{\mathcal{O}}_\theta : \mathcal{S} \to \mathcal{A}$ is a deterministic mapping from the state space to the action space. In the RLOP framework, the policy gradient optimization starts from a Gaussian policy $\pi_\theta^0(\cdot) = \mathcal{N}(\hat{\mathcal{O}}_\theta(\cdot), \Sigma)$ centered around $\hat{\mathcal{O}}_\theta$ with covariance Σ. The covariance matrix Σ is a diagonal matrix whose diagonal entry Σ_{ii} is the (sampling) variance of an relocation action a_i (a_i is an entry of $\mathbf{a} \in \mathcal{A}$). Note that a_i is one of the prediction labels of $\hat{\mathcal{O}}_\theta$, so its empirical distribution can be estimated in the supervised-learning stage. Therefore, Σ_{ii} can be taken as a certain percentage of a_i's characteristic statistics such as its empirical mean or median in the supervised-learning dataset. Prior knowledge on Σ is extremely valuable since a well-chosen Σ can lead to a more efficient exploration during training.

The policy gradient algorithm is summarized in Algorithm 1. Note that, after sampling action $\mathbf{a}$ from π_θ, $\mathbf{a}$ should be rounded and restored to zone-to-zone level by the transportation optimization $\mathcal{TO} : \mathcal{A} \to \mathcal{W}$ in Section 5.1.2. Again, note that the RLOP framework is

Algorithm 1: RLOP

Train a differential optimization proxy $\hat{\mathcal{O}}_\theta$ to approximate a given relocation model;
Choose learning rate α, discount factor γ, trade-off parameter β and covariance Σ ;
for *Episode* $= 1, 2, ...$ **do**
  for $i = 1, ..., N$ **do**
    for $t = 0, 1, ...T_e$ **do**
      Observe current state $\mathbf{s}_t^i$ and sample an action $\mathbf{a}_t^i$ from current policy $\pi_\theta(\mathbf{s}_t^i) = \mathcal{N}(\hat{\mathcal{O}}_\theta(\mathbf{s}_t^i), \Sigma)$;
      Round and disaggregate $\mathbf{a}_t^i$ to feasible zone-to-zone level action $\mathbf{w}_t^i$ by transportation optimization $\mathcal{TO}$;
      Implement $\mathbf{w}_t^i$ in the simulator and observe reward R_t^i;
    end
  end
  Compute total discounted reward $G_t^i = \sum_{\tau=t}^{T_e} \gamma^{\tau-t} R_\tau^i$ for all i, t;
  Compute the policy gradient $\nabla_\theta J(\theta)$ by Eq. (6);
  $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
end

general and can incorporate any specific reinforcement-learning techniques (e.g., actor-critic, PPO, off-policy sampling, etc.) appropriate for the problem at hand.

6. Simulation Study

The RLOP framework is evaluated on Yellow Taxi Data in Manhattan, New York City (NYC, 2019). It is trained from 2017/01 to 2017/05 and evaluated in 2017/06 during morning rush hours on weekdays. Section 6.1 reviews the simulation environment, Section 6.2 presents the supervised-learning results, Section 6.3 presents the reinforcement-learning results, and Section 6.4 evaluates the performance of the policy.

6.1 Simulation Environment

The experiments use the end-to-end simulation framework in Riley et al. (2020). The Manhattan area is partitioned into a grid of cells of 200 squared meters and each cell represents a pickup/dropoff location. Travel times between the cells are queried from OpenStreetMap (2017). The fleet is fixed to be 1800 vehicles with capacity 4, distributed randomly among the cells at the beginning of the simulation.

The simulator has two main components: the ride-sharing routing algorithm reviewed in Section 3 and the relocation MPC model reviewed in Section 4. The routing algorithm batches riders into a time window and optimizes every 30 seconds. The relocation MPC model is executed every 5 minutes. It partitions the Manhattan area into 60 zones (Figure 2) and time into 5-minute epochs. Its planning horizon contains 4 epochs. The number of idle vehicles in each epoch is estimated by the simulator based on the current route of each vehicle and the travel times. The ride-share ratio is $W_{ij} = 1.5$ for all $i, j \in \mathcal{Z}$. Service weight and relocation penalty are $q^p(t, \rho) = 0.5^t 0.75^{\rho-t}$ and $q_{ij}^r(t) = 0.001 * 0.5^t \eta_{ij}$ where

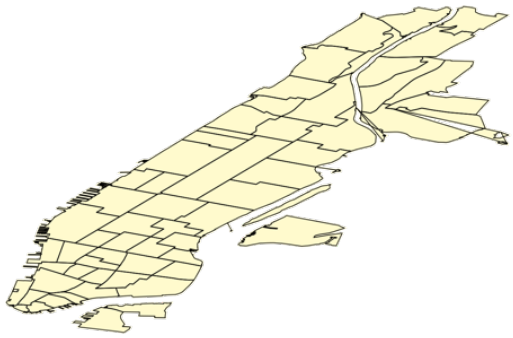

Figure 2: The Manhattan Area.

η_{ij} is travel time between zone i and zone j in seconds. The zone-to-zone demand D_{ijt} is forecasted based on historical data. The design of demand forecasting techniques is beyond the scope of this work. This paper first forecasts zone-level demand $D_{it} = \sum_{j \in \mathcal{Z}} D_{ijt}$ and then assigns the destinations based on historical distribution. The reason for doing zone-level prediction is to reduce sparsity in D_{ijt}, since most trips travel between a few popular regions. The forecasting model is a 2-layer fully-connected neural network with $(256, 256)$ hidden units and RELU activation functions. The loss function is MSE with l_1-regularization. It is trained from 2017/01 to 2017/05, 8am–9am, and tested on 2017/06, 8am–9am. The original time series data is augmented by injecting white noise sampled from a uniform distribution $U(-5, 5)$ to create more training data. To predict zone-level demand in the MPC horizon $\{D_{it}\}_{i \in \mathcal{Z}, t \in \mathcal{T}}$, the model uses the demand observed in the previous 3 epochs, as well as data observed a week ago during the same period to account for seasonality. For example, when forecasting demand from 8am to 8:20am (4 epochs) on 2017/06/08, the model uses demand from 7:45am to 8:00am on 2017/06/08 and demand from 7:45am to 8:20am on 2017/06/01. After zone-level demand is predicted, it is assigned to zone-to-zone level based on the historical distribution of the trip's destination. For example, if μ_{ij} proportion of trips from zone i goes to zone j during the hour of the prediction and $\hat{D}_{it}$ is the demand prediction for zone i, the final zone-to-zone prediction is $\hat{D}_{ijt} = \hat{D}_{it} \times \mu_{ij}$ rounded to the nearest integer. Overall, the mean squared error of the zone-to-zone level forecast in 2017/06 is 0.86.

After the MPC decides zone-to-zone level relocations, a vehicle assignment optimization determines which individual vehicles to relocate by minimizing total traveling distances (Riley et al., 2020). Of the routing, relocation, and vehicle assignment models, the routing model is the most computationally intensive since it operates on the individual (driver and rider) level as opposed to the zone level. Since all three models must be executed in the 30 seconds batch window, the experiments allocate 15 seconds to the routing optimization, 10 seconds to the MPC, and 5 seconds to the vehicle assignment. All the models are solved using Gurobi 9.1 with 24 cores of 2.1 GHz Intel Skylake Xeon CPU (Gurobi Optimization, 2021).

	Lasso	MLP	LSTM	Transformer
MSE	15.90	6.68	6.64	6.45

Table 2: Testing Loss of Machine Learning Models.

6.2 The Optimization Proxy

The optimization proxy is trained from 2017/01 to 2017/05, 8am - 9am, Monday to Friday, when the demand is at its peak and the need for relocation the greatest. The number of riders in these instances range from 22,000 to 29,000, providing a wide variety of demand distribution. The weekends and non-busy hours see much less demand and should be considered separately. The experimental study focuses on the busy hours because they need relocation the most. Each 1-hour taxi instance is perturbed by randomly adding/deleting a certain percentage of requests to generate more instances, where the percentages are sampled from a uniform distribution $U(-5, 5)$. These instances are run by the simulator and the MPC model's inputs and outputs are extracted as training data. In total, $15,000$ data points are used in training and 2500 data points are held out for testing.

Several machine learning models are trained to learn the relocation decisions. The model inputs are expected demand $D_{it} = \sum_{j \in \mathcal{Z}} D_{ijt}$, expected idle vehicles V_{it}, and expected vehicle shortage $(D_{it} - V_{it})$ in each zone i and epoch t. The target is zone-level relocation decisions $\mathbf{a} = [x_i^d, x_i^o]_{i \in \mathcal{Z}}$. All the models use MSE loss with l_1-regularization. Their loss on the testing set are given in Table 2. Of the tested models, multilayer perceptron (MLP), long short-term memory (LSTM), and Transformer achieve similar level of accuracy and outperform Lasso regression. The MLP is selected as the final optimization-proxy because it has fewer parameters. Specifically, the MLP has two hidden layers of (128, 128) units with hyperbolic tangent (tanh) activation functions. It is trained in Pytorch by Adam optimizer with batch size 32 and learning rate 10^{-3} (Kingma & Ba, 2015; Paszke et al., 2019). The loss of each zone is reported in Figure 3. The errors for all zones are reasonable, although a few zones exhibit higher losses than others. In addition, the optimization proxy achieves similar performance as the MPC in simulation: the detailed results are presented in Section 6.4. Overall, these results indicate that the optimization proxy successfully learned the MPC decisions.

6.3 Reinforcement Learning

The optimization proxy is refined by reinforcement learning in 2017/05. Since the number of riders in most daily instances ranges from 22,000 to 29,000, four instances with [23960, 25768, 27117, 28312] riders are selected and the policy is trained on these representative instances. To stabilize training, it is common practice to subtract a baseline from the reward to distinguish good and bad actions when computing the policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_e} (G_t^i - b_t^i) \nabla_\theta \log P_{\pi_\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) \tag{7}$$

where b_t^i is the baseline representing the expected reward since t following the current policy. $(G_t^i - b_t^i)$ therefore measures the "advantage" of this trajectory's decisions over the current

policy. The baseline b_t^i can be estimated in many different ways (Weng, 2018). This paper employs the sample average method: it samples $K = 10$ trajectories for each training instance and takes the sample average as baseline, i.e., $b_t^i = \frac{1}{K}\sum_{k=1}^{K} G_t^{ik}$ if trajectories for instance i are indexed by $\{i_1, ...i_K\}$. Therefore each policy gradient update is based on $4K = 40$ sample trajectories.

Algorithm 1 with the baseline is run with $\alpha = 0.005$, $\beta = 0.75$ and $\gamma = 0.75$. The sampling variance Σ_{ii} is taken as $0.05a_i^{0.75}$ where $a_i^{0.75}$ is the 75th percentile of action a_i in the supervised-learning data set (recall that a_i is a prediction label for the optimization proxy). To make sure that RL does not overfit on the selected representative instances, the policy is validated on other instances in 2017/05 after each training episode and the algorithm stops when the average reward on the validation set fails to improve for 5 consecutive episodes. The training and validation curves (broken down into waiting and relocation time) in Figure 4 show that the relocation costs drop dramatically, while the waiting times stay about the same. The algorithm converges in 55 episodes: the training is significantly more efficient computationally than pure reinforcement learning algorithms, which typically converge in tens of thousands of episodes.

6.4 Evaluation Results

The trained policy is evaluated on weekdays in 2017/06. The proposed RLOP approach is compared with the optimization proxy, as well as the MPC optimization. Pure reinforcement learning without an initial policy seeded with the optimization proxy (Algorithm 1 without step 1) fails to converge due to the high-dimensional state and action spaces: it is too expensive computationally to be applied in this setting. Figure 5 reports the average rider waiting time and the average vehicle relocation time of each weekday in 2017/06, and Table 3 reports their monthly averages as well as model run times. The optimization proxy and the MPC optimization achieve similar performance on all daily instances. The

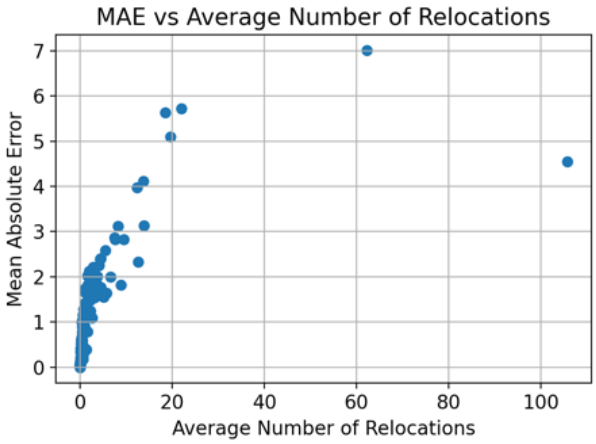

Figure 3: MPC Decision Predictions: Each Point in the Plot Denotes the Average Number of Relocations and the Mean Absolute Error of Relocation Predictions for a Zone.

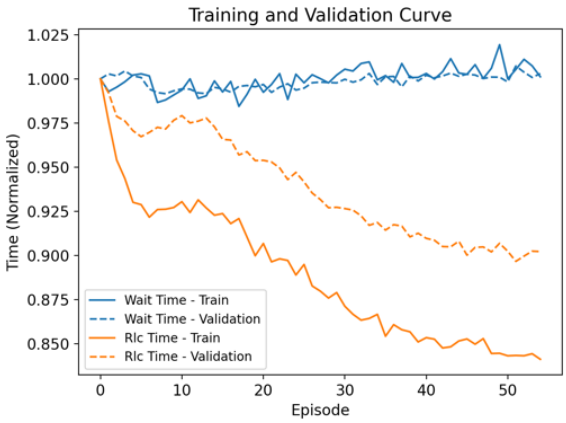

Figure 4: Training and Validation Curve of Reinforcement Learning (Normalized).

	Avg Wait Time (mins)	Avg Reloc. Time (mins)	Avg Run Time (s)	Max Run Time (s)
MPC	2.21	4.03	1.603	9.729
Opt. Proxy	2.18	4.06	0.016	0.048
RLOP	2.21	3.62	0.019	0.338

Table 3: Summary Statistics of Tested Models.

optimization proxy did slightly better than the MPC on certain metrics since the MPC optimization is based on an approximation of the ride-sharing system - its decisions are optimal for the approximation but not necessarily for the real system. The optimization proxy's performance probably benefits from the small deviation caused by the prediction. RLOP achieves similar rider waiting time as the other two models but with less relocation cost. In particular, its relocation time is 10.1% lower than the MPC and 10.8% lower than the optimization proxy. The waiting time performance of the MPC and optimization proxy are probably already near-optimal, leaving little room for improvements. Moreover, the optimization proxy and the RLOP are much faster than the MPC and are guaranteed to run in polynomial time. The longest MPC instance takes 9.73s, almost exceeding the 10s solver time limit. The optimization proxy and the RLOP framework take fractions of a second on all instances. The main computational cost of the RLOP framework lies in the offline stage where data for supervised learning and RL are generated through simulation. Nevertheless, RLOP is still more efficient than RL which requires a prohibitively large number of samples to train starting from a random policy. *Overall, these promising results show that the RLOP is an efficient and effective approach for idle vehicle relocation in real-time settings.*

7. Conclusion

Preemptively relocating idle vehicles is crucial for addressing demand-supply imbalance that frequently arises in the ride-hailing system. Current mainstream methodologies - optimization and reinforcement learning - suffer from computational complexity in either offline

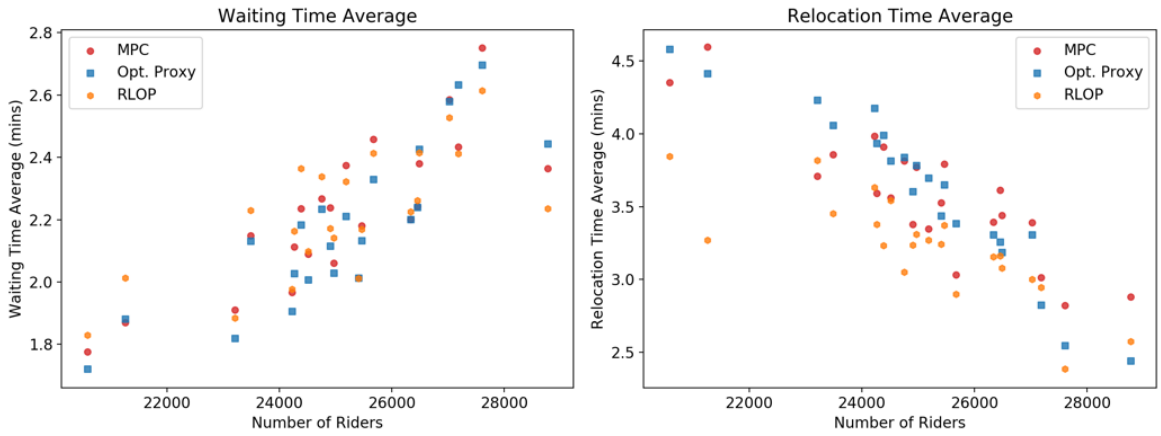

Figure 5: Evaluation Results of the RLOP, the Optimization Proxy, and the MPC Optimization in 2017/06.

training or online deployment. This paper proposes a reinforcement learning from Optimization Proxy (RLOP) approach to alleviate their computational burden and search for better policies. It trains a machine-learning policy to approximate an optimization model and then refines the policy by reinforcement learning. To reduce dimensionality and sparsity of the prediction and action space, this paper presents an aggregation-disaggregation procedure which predicts relocation actions at the aggregated level and disaggregates the predictions via a polynomial-time optimization. On the New York City dataset, the RLOP approach achieves significantly lower relocation costs and computation time compared to the optimization model, while pure reinforcement learning is too expensive computationally for practical purposes.

A key assumption behind the RLOP framework is the ability to forecast future demand and supply accurately, which may be challenging due to the volatile nature of real-time ride-hailing dynamics. Therefore, future work should focus on designing solutions that are robust to input uncertainty, possibly exploring stochastic optimization and robust training techniques.

Acknowledgments

This research is partly supported by NSF Awards 2112533 and 1854684. Many thanks to Professor Yao Xie for insightful comments on the paper.

References

Guériau, M., & Dusparic, I. (2018). Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1558–1563.

Gurobi Optimization (2021). Gurobi optimizer reference manual..

Holler, J., Vuorio, R., Qin, Z. T., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C., & Ye, J. (2019). Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In Wang, J., Shim, K., & Wu, X. (Eds.), *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pp. 1090–1095. IEEE.

Iglesias, R., Rossi, F., Wang, K., Hallac, D., Leskovec, J., & Pavone, M. (2017). Data-driven model predictive control of autonomous mobility-on-demand systems. *CoRR*, *abs/1709.07032*.

Jiao, Y., Tang, X., Qin, Z., Li, S., Zhang, F., Zhu, H., & ping Ye, J. (2021). Real-world ride-hailing vehicle repositioning using deep reinforcement learning. *ArXiv*, *abs/2103.04555*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y., & LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Liang, E., Wen, K., Lam, W. H. K., Sumalee, A., & Zhong, R. (2021). An integrated reinforcement learning and centralized programming approach for online taxi dispatching..

Lin, K., Zhao, R., Xu, Z., & Zhou, J. (2018). Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, p. 1774–1783, New York, NY, USA. Association for Computing Machinery.

Mao, C., Liu, Y., & Shen, Z.-J. M. (2020). Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *115*, 102626.

Miao, F., Han, S., Hendawi, A. M., Khalefa, M. E., Stankovic, J. A., & Pappas, G. J. (2017). Data-driven distributionally robust vehicle balancing using dynamic region partitions. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*.

Miao, F., Lin, S., Munir, S., Stankovic, J., Huang, H., Zhang, D., He, T., & Pappas, G. (2015). Taxi dispatch with real-time sensing data in metropolitan areas — a receding horizon control approach. *IEEE Transactions on Automation Science and Engineering*, *13*.

NYC (2019). Nyc taxi & limousine commission - trip record data.. Accessed: 2020-10-01.

Oda, T., & Joe-Wong, C. (2018). Movi: A model-free approach to dynamic fleet management. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 2708–2716.

OpenStreetMap (2017). Planet dump retrieved from https://planet.osm.org.. Accessed: 2020-10-01.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach,

H., Larochelle, H., Beygelzimer, A., d'Alche Buc, F., Fox, E., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc.

Ramírez, J., Yu, W., & Perrusquía, A. (2021). Model-free reinforcement learning from expert demonstrations: a survey. *Artificial Intelligence Review*, *1*(1).

Rebman, K. R. (1974). Total unimodularity and the transportation problem: a generalization. *Linear Algebra and its Applications*, *8*(1), 11–24.

Riley, C., Legrain, A., & Van Hentenryck, P. (2019). Column generation for real-time ride-sharing operations. In Rousseau, L.-M., & Stergiou, K. (Eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 472–487. Springer International Publishing.

Riley, C., Van Hentenryck, P., & Yuan, E. (2020). Real-time dispatching of large-scale ride-sharing systems: Integrating optimization, machine learning, and model predictive control. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4417–4423. International Joint Conferences on Artificial Intelligence Organization.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*, 484–489.

Sutton, R., & Barto, A. (2018). *Reinforcement Learning: An Introduction*. The MIT Press., Cambridge, Massachusetts.

Tsao, M., Iglesias, R., & Pavone, M. (2018). Stochastic model predictive control for autonomous mobility on demand. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3941–3948.

Verma, T., Varakantham, P., Kraus, S., & Lau, H. C. (2017). Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In *ICAPS*.

Wen, J., Zhao, J., & Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 220–225.

Weng, L. (2018). Policy gradient algorithms. In *lilianweng.github.io/lil-log*.

Yuan, E., & Van Hentenryck, P. (2021). Real-time pricing optimization for ride-hailing quality of service. In Zhou, Z. (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 3742–3748. ijcai.org.