LEAD: Towards Learning-Based Equity-Aware Decarbonization in Ridesharing Platforms

Mahsa Sahebdel[1], Ali Zeynali[1], Noman Bashir[2], Prashant Shenoy[1], and Mohammad Hajiesmaili[1]

[1]University of Massachusetts Amherst
[2]Massachusetts Institute of Technology

Abstract

Ridesharing platforms such as Uber, Lyft, and DiDi have grown in popularity due to their on-demand availability, ease of use, and commute cost reductions, among other benefits. However, not all ridesharing promises have panned out. Recent studies demonstrate that the expected drop in traffic congestion and reduction in greenhouse gas (GHG) emissions have not materialized. This is primarily due to the substantial distances traveled by the ridesharing vehicles without passengers between rides, known as deadhead miles. Recent work has focused on reducing the impact of deadhead miles while considering additional metrics such as rider waiting time, GHG emissions from deadhead miles, or driver earnings. Unfortunately, prior studies consider these environmental and equity-based metrics individually despite them being interrelated.

In this paper, we propose a Learning-based Equity-Aware Decarabonization approach, `LEAD`, for ridesharing platforms. `LEAD` targets minimizing emissions while ensuring that the driver's utility, defined as the difference between the trip distance and the deadhead miles, is fairly distributed. `LEAD` uses reinforcement learning to match riders to drivers based on the expected future utility of drivers and the expected carbon emissions of the platform without increasing the rider waiting times. Extensive experiments based on a real-world ride-sharing dataset show that `LEAD` improves fairness by 2× when compared to emission-aware ride-assignment and reduces emissions by 70% while ensuring fairness within 66% of the fair baseline. It also reduces the rider wait time, by at least 40%, compared to various baselines. Additionally, `LEAD` corrects the imbalance in previous emission-aware ride assignment algorithms that overassigned rides to low-emission vehicles.

1 Introduction

Road transportation significantly contributes to global energy consumption and greenhouse gas (GHG) emissions [8]. Worldwide, the transportation sector is the fourth-largest source of GHG emissions and is the largest in the United States [29, 42]. In 2021, transportation accounted for 37% of global CO2 emissions [17]. That same year, the United States emitted approximately 5 billion metric tons of carbon dioxide, accounting for about 13.49% of global emissions, surpassing the combined emissions of all 28 European Union countries [7]. Due to these environmental impacts, there is an effort towards exploring transportation systems that transport people in an eco-friendly manner [33].

In recent years, shared mobility options have emerged as an attractive transport solution, thanks to the widespread integration of smartphones into daily life. These services enable users to access transportation on demand without owning a vehicle [4, 26, 27, 34]. As a result, ridesharing is seen as a sustainable transportation option that can help reduce emissions and traffic congestion. Unfortunately, these expectations have not materialized and ridesharing services have been linked to increased traffic congestion, higher emissions, distracted driving, and negative social equity implications [3, 14, 30]. The increase in traffic congestion and emissions results from two key by-products of ridesharing. First, ridesharing availability can reduce the use of public transportation. Second, ridesharing vehicles travel a significant fraction of their total miles without a passenger on their way to pick up a passenger or come back after dropping one [23, 31, 36]. These non-passenger miles are referred to as *deadhead* miles. Previous studies show that deadhead miles can account for 19% to 41% of total miles while increasing emissions by up to 90% as compared to the personal vehicle use [15, 46].

There has been significant recent work on optimizing the deadhead miles in ridesharing ecosystems [18, 32] while considering additional rider-specific metrics such as the wait time [19, 47] or platform-wide objectives such as GHG emissions [32, 49]. Other studies aim to enhance overall rider satisfaction by minimizing travel costs and waiting times [6, 20, 37]. However, these studies often ignore fairness from the drivers' perspective, which can exacerbate social inequities. The studies from a diver's perspective typically focus on enhancing driver satisfaction and operational efficiency by optimizing routes and maximizing a given definition of utility [21, 40]. While some of those studies also target fairness in earnings across drivers [35], they ignore the rider's wait time and system-wide emissions. As a result, the ride-assignment algorithms proposed by prior cannot be used in practice to promote sustainable and equitable growth in ridesharing platforms.

In this work, we consider the problem of holistic ride assignment that balances the objectives of the system (low emissions), the driver (fairness in earnings), and the rider (low waiting time). The goal is to minimize system-wide emissions, including emissions from ride and deadhead miles, while maximizing fairness in total utility across all trips between drivers without impacting the rider's waiting time. The utility of a trip for a driver is defined as the trip distance minus the deadhead distance. Given this setup, [32] and [35] represent our most relevant related works. In [32], the authors propose an online threshold-based ride assignment algorithm (`TORA`) that targets minimizing the system-wide emissions while optimizing the waiting time of the riders. `TORA` achieves a highly favorable tradeoff between emissions (up to 60% reduction) and rider's wait time (4% increase). However, to reduce emissions, `TORA` assigns a disproportionate fraction of rides to low-emission vehicles, resulting in high unfairness among drivers. For example, it may assign up to 65% of the rides to EVs despite them constituting only 25% of the fleet. In [35], the authors propose `LAF` that targets maximizing utility, defined as the total revenue generated by the system while minimizing unfairness in the driver's equitable earnings. While taking a driver-centric approach, `LAF` completely ignores system-wide emissions and the rider's wait time.

In this paper, we propose the Learning-Based Equity-Aware Decarbonization (`LEAD`) algorithm, which aims to minimize total system-wide emissions and maximize fairness in drivers' accumulated utilities across trips. To reduce system-wide emissions, `LEAD` must consider not only the drivers that are available at present but also the ones that will become available in the near future. To do so, `LEAD` explicitly accounts for the dependencies among ride assignments by using reinforcement learning to develop future-aware ride assignment strategies. Using a real-world ridesharing dataset, we analyze how `LEAD` and other representative baselines [32, 35] perform using metrics, such as the reduction in emissions, the quality of service for riders measured as wait time, and fairness for drivers, and how gracefully they navigate the trade-offs between these metrics. Our contributions can be summarized as follows:

- We frame the problem of ride assignment in ridesharing services (termed `RARS`) as a bipartite matching problem and propose a new objective function that aims to minimize total carbon emissions across all trips while reducing the gap between utilities of different drivers.
- We present `LEAD`, an online learning-based algorithm that reduces emissions and ensures fairness for drivers, as measured by utility. To the best of our knowledge, this is the first study to leverage the dependencies between past and future ride assignments to minimize carbon emissions while ensuring a fair distribution of utility among drivers.
- We implemented `LEAD` on a simulation testbed and evaluated its performance using real data from RideAustin [28], a nonprofit ridesharing service. In addition, we compare the performance of `LEAD` with heuristic and state-of-the-art algorithms used in [32, 35]. Our experimental results show that, in practical scenarios, our algorithm not only significantly improves the fairness of utility distribution among drivers but also substantially reduces emissions, compared to the current state-of-the-art methods.

2 Related Work

Ridesharing Optimization Approaches. The development of ridesharing systems has introduced unique challenges, with the primary focus being the assignment of ride requests to drivers. Some challenges in this area include improving vehicle utilization across multiple ride requests and demands [22, 25, 48], developing algorithms for route planning [10, 43], and analyzing and integrating temporal and spatial patterns to predict the arrival time of riders' requests [11, 16]. Some other works in this domain also consider different optimization objectives such as maximizing drivers' profit [2, 45, 51], while others aim to minimize travel costs [41, 44] or reduce riders' waiting times [1, 47, 50].

In [18] , authors attempt to reduce trip-level deadhead miles by leveraging hour-ahead trip demand predictions and using a heuristic approach to driver assignment. However, this focus on reducing trip-level deadhead miles does not always result in a reduction in system-wide deadhead miles and emissions, as these outcomes also depend on factors such as the fuel efficiency of vehicles and traffic conditions. Another work [32] focuses on minimizing emissions from deadhead miles while also reducing rider waiting times, ensuring no degradation in user satisfaction. However, a ride assignment system focused only on reducing emissions can negatively impact the quality of service (QoS) for riders and drivers. For instance, prioritizing emission reduction as the primary objective can lead to a preference for electric or low-emission vehicles over high-emission ones. This bias is unfair to drivers with high-emission vehicles, who often belong to lower-income communities and may face reduced earning opportunities. Additionally, an emissions-aware ride assignment may allocate trips with inherently longer deadhead miles to low-emission vehicles. This increases the deadhead-to-trip ratio, decreasing these vehicles' overall efficiency and service quality.

Another key challenge is the lack of consideration for the temporal dependencies between current and future assignments. This oversight can impact the optimization of utility and fairness. Many algorithms in this area attempt to provide theoretical performance guarantees, but they typically rely on myopic assumptions [47]. For instance, they frequently neglect the dependency of assignments on past and future decisions, which can lead to sub-optimal assignments. This shortsightedness fails to account for the dynamic and evolving nature of ridesharing systems, where current ride assignments can significantly impact future states and the system's overall efficiency. An effective alternative approach involves leveraging historical data to optimize ride assignments alongside integrating reasonable assumptions.

Fairness in Ridesharing. Fairness is a critical factor in matching scenarios, particularly in multi-sided markets where preferences play a significant role [9, 12, 24]. In the context of ridesharing, recent literature has raised concerns about the fairness of the assignments. Brown et al. [5] shows the unfair treatment of riders by ridesharing companies, which leads to higher trip cancellation rates for a few groups of riders. Another study indicates that income inequality among ridesharing drivers may prevent them from earning a living wage [13]. Another work aims to optimize long-term efficiency and fairness in ridesharing platforms through joint order dispatching and driver repositioning [38]. Another recent work [32] showed a trade-off between reducing carbon emissions and sacrificing the fairness of utilities among drivers. Similarly, another study [35] demonstrated optimization in driver utilities and fairness without considering emissions. This study used reinforcement learning techniques to evaluate the expected achievable utilities in different city areas. Despite the success of this approach in ensuring equity among drivers' utilities, it did not address eco-friendly ride assignments with low carbon emissions.

Ensuring fair assignments in ridesharing systems is challenging due to drivers' and riders' inherent spatiotemporal dynamics. Drivers can earn varying amounts of utility based on their assignments, while riders may experience differing waiting times. These disparities can lead to perceptions of unfairness, complicating the assignment process. To tackle these challenges, our work introduces a new fairness metric for driver earnings that considers the spatiotemporal dynamics inherent in ridesharing systems. This metric is designed to ensure a more equitable distribution of earnings among drivers, considering the dynamic nature of the system. Building on this metric, we have developed a reinforcement learning-based approach assignment algorithm that reduces carbon emissions while providing a fair distribution of driver utilities compared to the state-of-the-art assignment algorithm [32]. The algorithm leverages historical data and dynamic learning to create effective and equitable assignment strategies, addressing immediate and long-term objectives in the ridesharing ecosystem.

3 Problem Formulation

We consider a ridesharing system consisting of a set of drivers, denoted as $\mathcal{V}$, and a set of ride requests, denoted as $\mathcal{R}$. In the ridesharing context, a ride $r \in \mathcal{R}$ is defined as a request that includes a rider's pickup location, denoted as p_r, a rider's dropoff location, denoted as q_r, and time of the request, denotes as c_r. A driver $v \in \mathcal{V}$ in the ridesharing system is defined by their current location, utility, and the emissions per mile of the driver's vehicle e_v.

In ridesharing platforms, riders set the pickup and dropoff locations for a ride request; then, the platform matches the request to one of the available drivers and estimates the trip duration and waiting time based on factors such as trip distance, traffic congestion, and the distance between the assigned driver and the rider's pickup location. An available driver in the system refers to a driver who is either idle or soon to be available within the specified time to serve a request. The emission associated with a ride request r is influenced by the unit emissions of the driver's vehicle, the deadhead distance, the trip distance, and additional factors such as traffic congestion, road conditions, etc. For simplicity, we estimate the carbon emissions produced during the servicing of the ride request r by driver v using the equation below.

$$E_{v,r} = (d_{v,r}^{(D)} + d_r^{(T)}) \cdot e_v, \tag{1}$$

where e_v is the carbon emission per mile of the vehicle of driver v, $d_{v,r}^{(D)}$ is the deadhead distance of the trip, and $d_r^{(T)}$ is the trip distance which is the distance between rider's pickup and dropoff location. The utility of a driver v is defined as the difference between the trip distance and deadhead distance accumulated across all served ride requests.

Table 1: Summary of important notations.

Notation	Description
$\mathcal{V}$	Set of all drivers
$\mathcal{R}$	Set of all ride requests
$\mathcal{V}_b$	Set of available drivers during batch b
$\mathcal{R}_b$	Set of ride requests in batch b
p_r	Pickup location of ride request r
q_r	Dropoff location of ride request r
c_r	Created time of ride request r
$a_{r,v}$	Dropoff time of ride request r when served by driver v
$E_{v,r}$	Emission of serving request r by driver v
$U_{v,r}$	Utility history of driver v before serving ride request r
$d_r^{(T)}$	Trip distance of ride request r
$d_{v,r}^{(D)}$	Deadhead distance of serving ride request r by driver v
$d_{t,D}$	Deadhead distance of t^{th} trip served by the agent
$d_{t,T}$	Trip distance of t^{th} trip served by the agent
$V_D(s)$	Value function at state s associated to deadhead distances
$V_T(s)$	Value function at state s associated to trip distances
e_v	Produced carbon emission per mile of the vehicle of driver v

The trip distance represents the actual distance driven to fulfill the ride request, which contributes to the driver's earnings. In contrast, the deadhead distance, the distance traveled without carrying a passenger (e.g., traveling from the driver's location to the rider's pickup location), incurs an overhead cost for the driver beyond the costs incurred during the trip, such as fuel expenses. By subtracting the deadhead distance from the trip distance, we effectively account for the revenue generated and the overhead cost incurred by the driver. This approach provides a measure of the driver's net revenue, reflecting the true utility of a ride to the driver:

$$U_{v,r} = \sum_{r' \in \mathcal{V}_r} (d_{r'}^{(T)} - d_{v,r'}^{(D)}) \cdot x_{v,r'}, \tag{2}$$

where $U_{v,r}$ is the cumulative utility of the driver v before serving ride request r, $\mathcal{V}_r$ is the set of ride requests posted before r, and $x_{v,r} \in \{0,1\}$ is a decision variable: $x_{v,r} = 1$ if ride request r is assigned to driver v; 0 otherwise. For simplicity, we denote the utility of driver v after serving all requests as U_v.

The ride-assignment problem builds on the classic bipartite matching problem, involving riders and drivers as distinct sets. The main challenge is modeling the constraints of available drivers and their costs for new requests, especially when some drivers are already occupied but may soon become available to take a ride. Our goal is to design an algorithm that assigns available vehicles to ride requests in a manner that minimizes emissions while ensuring fairness among drivers and considering the long-term effects and interdependencies of current assignments on future ones. We formulate the RARS problem as follows.

$$
\begin{aligned}
[\texttt{RARS}] \quad \min \quad & \text{Emission} - \eta \cdot \text{Fairness}, && \text{(3a)}\\
& \text{Emission} := \sum_v \sum_r E_{v,r} \cdot x_{v,r}, && \text{(3b)}\\
& \text{Fairness} := -\max_v(U_v) && \text{(3c)}\\
& \quad + \min_v(U_v), &&\\
\text{s.t.,} \quad & \sum_v x_{v,r} \in \{0,1\} \qquad \forall r, && \text{(3d)}\\
& \sum_{\{r' \neq r | c_r \leq c_{r'} \leq a_{r,v}\}} x_{v,r'} \cdot x_{v,r} = 0 \quad \forall v, r, && \text{(3e)}\\
\text{vars.,} \quad & x_{v,r} \in \{0,1\} \qquad \forall v, r. && \text{(3f)}
\end{aligned}
$$

where $a_{r,v}$ shows the dropoff time of ride request r when served by driver v. The term *Emission* includes trip emissions and deadhead miles emissions. The *Fairness* metric is defined as the difference between the minimum and maximum utilities of the drivers and is a non-positive metric. Constraint (3d) ensures that each ride request is assigned to at most one driver, and constraint (3e) ensures that each driver is assigned to at most one ride request at each time. The parameter $\eta \geq 0$ controls the balance between emissions and fairness. Specifically, it quantifies the increase in emissions in grams of carbon dioxide (g.CO2) that we will accept for a one-kilometer decrease in unfairness across drivers. A higher value of η indicates a greater emphasis on fairness. A lower value of η prioritizes reducing emissions, potentially at the cost of sacrificing fairness.

4 The LEAD Algorithm

This section introduces the LEAD algorithm that leverages reinforcement learning and batching to accumulate multiple ride requests that are jointly assigned to ensure that a long-term system-wide beneficial assignment is obtained. We present the formulation of the single batch matching problem as an integer linear programming problem whose objective includes two terms: one for the expected emission of ridesharing service and the second for the expected fairness in utility across drivers. To evaluate these expected values, LEAD employs reinforcement learning to account for temporal dependencies between current and future ride-matching. This is essential since the system operates online, with rider requests and driver availability dynamically changing in real-time.

In the following, we present the details of LEAD, which consists of two main modules: the *Learning Based* module and the *Batched Emission Aware Fair Assignment* module. These modules are discussed in detail in Section 4.1 and Section 4.2, respectively.

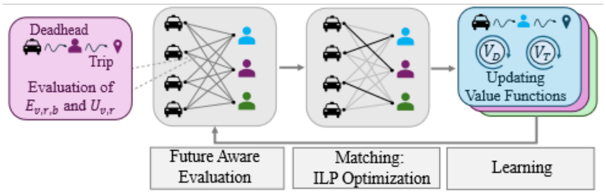


Figure 1: Overview of `LEAD`. During each batch, `LEAD` evaluates the long-term emissions and utilities and uses them to construct the weighting of the ILP problem of (10). After solving the optimization problem and matching ride requests with drivers, `LEAD` updates the value functions based on the deadhead and trip distances of the served requests in the batch.

4.1 Learning Expected Emissions and Utility

In the following, we discuss the use of online reinforcement learning to model the impact of current assignments on future assignments and evaluate the expected deadhead and trip distances of trips in different regions of the covered area. Reinforcement learning is a method where agents learn by interacting with the environment over time. At each step, the agent takes an action and receives feedback in the form of utility from the environment. This continuous interaction allows the agent to optimize its strategy based on the utility received, making it well-suited for dynamic and complex systems such as ridesharing, where decisions made at one point can influence outcomes in the future. In our work, the reinforcement learning module consists of the following elements:

- **Agent:** Each active driver v in the ridesharing system is considered as an agent.
- **State:** The state of driver v during batch b is represented by the driver's location $l_{v,b}$. To facilitate data analysis and management, city space is typically discretized into a square grid system, which results in a finite number of unique states.
- **Action:** The action of the available driver is defined as the request they choose to accept.

After taking action, each agent receives utility, which is the difference between the trip distance and the deadhead distance. To predict the expected emissions produced and the utility achieved by a driver, we introduce two value functions for the agent: one for the deadhead distance and another for the trip distance.

Due to the complexity of considering multiple dependent agents within the system, we simplify the learning module by treating different agents as a single agent. In the emission-aware fair assignment module, we leverage the evaluated value functions to take appropriate actions for the different action-correlated agents. We adopt a framework that involves sequences comprising the current state, current action, reward, and next state to estimate the value functions. One effective method for learning in this scenario is Temporal Difference (TD) learning [39], which iteratively updates value functions based on the deadhead and trip distances of each completed ride request.

Below, $V_D(s)$ shows the expected deadhead distance a driver could travel when starting from state s.

$$V_D(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot d_{t,D} | s_0 = s\right], \quad (4)$$

where $d_{t,D}$ is the deadhead distance of t^{th} trip served by the agent, s_0 denotes the starting state of the agenet, and $\gamma \leq 1$ is the discount factor. Similarly, we define $V_T(s)$ as the expected trip distance a driver will travel when starting from state s.

$$V_T(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot d_{t,T} | s_0 = s\right], \quad (5)$$

where $d_{t,T}$ is the trip distance of t^{th} trip served by the agent.

Assume the agent is located in state s_t and after serving the request, travels to the next state s_{t+1}. The TD learning update rule for the value functions based on the trip and deadhead distance of the trip are as follows:

$$V_D(s) \leftarrow V_D(s) + \alpha\left[d_{t,D} + \gamma V_D(s_{t+1}) - V_D(s_t)\right], \quad (6)$$
$$V_T(s) \leftarrow V_T(s) + \alpha\left[d_{t,T} + \gamma V_T(s_{t+1}) - V_T(s_t)\right], \quad (7)$$

where α is the learning rate. The values of $V_D(s)$ and $V_T(s)$ are initialized through a specific initialization process (such as zero initialization or random initialization). After a driver serves a ride request, the values of these functions are updated accordingly. In the next section, we demonstrate how these value functions can be used to evaluate the expected deadhead and trip distances for trips starting in different city locations, ultimately estimating the future emissions and utility of drivers.

4.2 Batched Emission-Aware Fair Assignment

In this section, we propose the process of matching ride requests to available drivers within `LEAD`. `LEAD` performs the ride matching process in a batch setting, where each batch comprises ride requests posted after the assignments from the previous batch. Let $E_{v,r,b}$ denote the long-term expected emission produced by driver v after being assigned to request r within batch b. The value of $E_{v,r,b}$ depends on the trip and deadhead distances for serving request r, as well as future deadhead and trip distances traveled by driver v. To estimate the latter, we leverage the value function

Algorithm 1: LEAD($\mathcal{R}_b$, $\mathcal{V}_b$)

```
1  forall v ∈ V_b, r ∈ R_b do
2      Calculate E_{v,r,b} using Equation (8);
3      Calculate U_{v,r,b} using Equation (9);
4  Matched-Pairs ← Solution of optimization
     problem (10);
5  foreach r ∈ R_b do
6      v ← Matched-Pairs(r);
7      s_t ← current location of v;
8      s_{t+1} ← dropoff location of the request r;
9      Update V_D(s_t) using Equation (6);
10     Update V_T(s_t) using Equation (7);
11 return Matched-Pairs;
```

described in the previous section. Specifically, the expected emission produced by driver v during and after batch b can be calculated as:

$$
\begin{aligned}
E_{v,r,b} &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t(d_{t,T}+d_{t,D})\cdot e_v \big| s_0 = l_{v,b}, s_1 = q_r\right] \\
&= \Bigg[(d_r^{(T)} + d_{v,r}^{(D)}) + \gamma(V_T(q_r)+V_D(q_r)) \\
&\quad - (V_T(l_{v,b}) + V_D(l_{v,b}))\Bigg]\cdot e_v, \qquad (8)
\end{aligned}
$$

which is directly derived from Equations (4) and (5), considering that the driver is located at state $l_{v,b}$ and will travel to the dropoff location of the request after serving of that. Similarly, let $U_{v,r,b}$ denote the expected final utility of driver v given that they were assigned to request r during batch b and had a utility history of $U_{v,r}$ by that time. We can evaluate the value of $U_{v,r,b}$ as follows:

$$
\begin{aligned}
U_{v,r,b} &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t(d_{t,T}-d_{t,D}) \big| s_0 = l_{v,b}, s_1 = q_r\right] \\
&= U_{v,r} + (d_r^{(T)} - d_{v,r}^{(D)}) + \gamma(V_T(q_r)-V_D(q_r)) \\
&\quad - (V_T(l_{v,b}) + V_D(l_{v,b})). \qquad (9)
\end{aligned}
$$

During the ride-matching of batch b, LEAD finds the set of requests $\mathcal{R}_b$, and available drivers in that batch, $\mathcal{V}_b$, and evaluates $E_{v,r,b}$, and $U_{v,r,b}$ for every pair of v and r in the batch. Then, LEAD finds the solution of the integer linear programming problem below to match each ride request to available drivers.

$$
\begin{aligned}
\min \quad & \text{Emission}_{(b)} - \eta\cdot\text{Fairness}_{(b)}, && (10a) \\
& \text{Emission}_{(b)} := \sum_{v\in\mathcal{V}_b}\sum_{r\in\mathcal{R}_b} E_{v,r,b}\cdot x_{v,r}, && (10b) \\
& \text{Fairness}_{(b)} := -\max_v \sum_{r\in\mathcal{R}_b}(U_{v,r,b}\cdot x_{v,r}) \\
& \quad + \min_v \sum_{r\in\mathcal{R}_b}(U_{v,r,b}\cdot x_{v,r}), && (10c) \\
\text{s.t.,} \quad & \sum_{v\in\mathcal{V}_b} x_{v,r} \in \{0,1\} \quad \forall r, && (10d) \\
& \sum_{r\in\mathcal{R}_b} x_{v,r} \in \{0,1\} \quad \forall v, && (10e) \\
\text{vars.,} \quad & x_{v,r} \in \{0,1\} \quad \forall v, r, && (10f)
\end{aligned}
$$

where $\text{Emission}_{(b)}$ represents the expected total emissions produced by the ridesharing service during and after batch b. $\text{Fairness}_{(b)}$ estimates the expected fairness of the system given the assignments in batch b. It is worth mentioning that if the number of ride requests in batch b exceeds the number of available drivers, only the first $|\mathcal{V}_b|$ ride requests will be assigned, with the remaining requests being moved to the next batch.

The pseudocode for LEAD is outlined in Algorithm 1. Initially, LEAD evaluates the values of $E_{v,r,b}$ and $U_{v,r,b}$ for every pair of v and r in batch b (Lines 1-3). Next, it finds the solution for the optimization problem in (10), determining which driver should serve each ride request. The final step is to update the value functions based on the actual trip and deadhead distances of the served ride requests within this batch. Specifically, for each request r in the batch, LEAD updates the value functions associated with trip and deadhead distances using the update rules presented in Equations (6) and (7) (Lines 5-10). By integrating emission and fairness into a single objective, LEAD balances reducing carbon emissions with the fairness of utilities among drivers. Additionally, by considering the long-term expected emissions and utilities instead of just the current values, LEAD optimizes these metrics with a forward-looking perspective.

5 Experimental Evaluation

In this section, we conduct comprehensive experiments to evaluate the performance of LEAD using several metrics, including emissions per trip, fairness in driver's utility, and average wait times for riders. We compare LEAD against state-of-the-art algorithms across various values for different batch durations, the fairness parameter η, and the percentages of low-emission vehicles. Below, we outline the key questions the evaluation addresses and summarize our findings.

Q1 How does LEAD reduce emissions, especially compared to other baselines targeting emissions reduction?
Outcome: *LEAD reduces emissions by at least 73.6%, 70.5%, and 58.3% compared to CD, TORA, and LAF algorithms, respectively.*

Q2 How much unfairness does LEAD incur to reduce emissions, compared to LAF that only targets fairness?
Outcome: *LEAD offers a favorable, and more importantly configurable, tradeoff between fairness and emissions reduction; it reduces emissions by 65% for a 9% drop in fairness. It also exposes the η knob to enable navigating the tradeoff gracefully.*

Q3 How does LEAD's rider wait time performance compare to the baselines that explicitly consider reducing wait time?
Outcome: LEAD achieves at least a 40% improvement over baseline algorithms while also ensuring that improving fairness for drivers does not impact the rider's wait time by more than 3%.

5.1 Experimental Setup

First, we provide details on the ridesharing dataset, baseline algorithms, metrics, and the range of evaluation parameters.

Ridesharing dataset. Our experiments use a publicly available dataset from RideAustin [28], a non-profit ridesharing service based in Austin, Texas. This dataset includes approximately 1.5 million trips over ten months in 2016 and 2017, collected in Austin. It contains detailed trip information such as pick-up and drop-off coordinates of each trip, vehicle make and model, distances traveled before, during, and after each trip.

In our experiments, we use a subset of the RideAustin dataset, focusing on trips from December 1, 2016, to December 15, 2016. This subset includes 58,866 ridesharing trips served by 150 drivers. We augment this dataset by including carbon emission data for both deadhead miles and individual trips, as well as equity information for drivers and riders, using E^2-RideKit, a publicly available toolkit [32]. We assume a rider cancels their request if it remains unassigned for two consecutive batches, and considered drivers available within 15 minutes in the set of available drivers of the batch. We also classified vehicles emitting under 135 g.CO2/km as low-emission (LEVs) and over 270 g.CO2/km as high-emission (HEVs). To assess the impact of replacing HEVs with LEVs, we created three dataset variants by randomly converting a certain percentage of the non-LEVs into electric vehicles (EVs), resulting in datasets with 10%, 15%, and 25% LEVs. We set the EV's emission intensity to 63.35 g.CO2 per kilometer.[1]

Comparison algorithms. We evaluate the performance of LEAD against a heuristic baseline algorithm and two state-of-the-art algorithms from prior works detailed below:

- **Closest Driver (CD)**: For a given batch of requests, this algorithm sequentially matches requests in the batch and greedily assigns a request to the nearest available driver to minimize the deadhead miles and the rider's wait time.

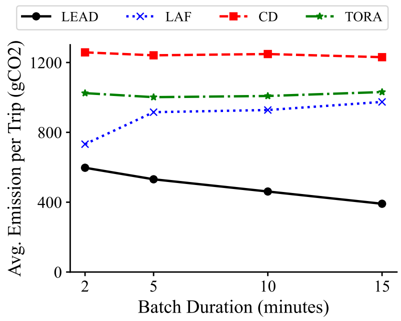
- **LAF** [35]: This algorithm has four stages: evaluating, assigning, guiding, and learning. Initially, it evaluates a bipartite graph where edge weights represent trip prices, and these weights are updated to reflect both current and future earnings. During the assigning stage, LAF utilizes a bi-objective assignment algorithm to balance the aggregate utilities of drivers and the equity in their utilities. The learning stage involves refining weight updates through reinforcement learning to guide future assignments.

  Since LAF aims to maximize total utilities and ensure equity in utilities, it uses a single value function to track the expected achievable utilities in different city regions. For adaptation to the current work, we set the utility (or price) of a ride request r served by driver v as the difference between the trip and deadhead distances and evaluated weights in Algorithm 1 of [35] accordingly. Additionally, during our experiments, the weight of trip utilities over different times of the day is the same.

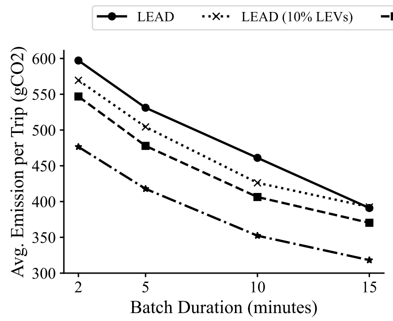
- **TORA** [32]: TORA is an online ride-assignment algorithm that aims to balance passenger waiting times with system-wide emissions. To the best of our knowledge, it is the only existing online ride assignment algorithm that explicitly considers emissions. TORA aims to reduce passenger waiting times by initially selecting the closest available driver. It then evaluates alternative available drivers by comparing their distances and the additional deadhead emissions they would generate relative to the closest driver. For each driver, TORA calculates an "Emission-to-Distance" (E2D) ratio, which is the ratio of the difference in deadhead emissions and the deadhead miles from the two drivers to the passenger. Finally, the algorithm selects the driver with the most favorable E2D ratio. Although TORA accounts for carbon emissions during ride matching, it does not address fairness in driver utilities.

Parameter ranges. We evaluate the performance of LEAD and baseline algorithms in various experimental settings. We evaluate the impact of parameter η and batch duration by varying η between 0.1 g.CO2/km and 10 g.CO2/km and varying batch duration values between 2 and 15 minutes.

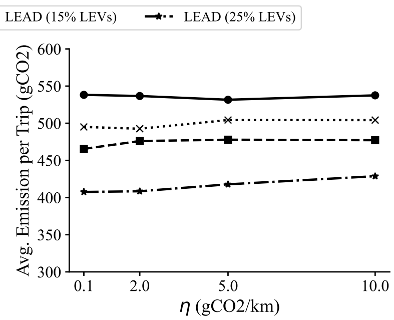
[1] We used the Tesla Model Y as a representative EV, which has an energy efficiency rating of 26 kWh/100mi. The unit distance emissions were calculated using the average carbon intensity of 408 g.CO2eq/kWh for Austin, Texas.



(a) LEAD vs. baselines



(b) Impact of LEVs



(c) Impact of Eta (η).

Figure 2: Emissions reduction performance: (a) emission per trip for different algorithms as a function of batch duration, (b) impact of the percentage of low emission vehicles (LEVs) in the fleet on LEAD, and (c) the impact of increasing emissions for an increase in fairness, captured using η. Here, η specifies extra emissions that the algorithm incurs to reduce unfairness by 1km.

To define unique states, we divide the city into square tiles with a width of 1 km, initialize value functions to 0, set the discount factor $\gamma = 0.9$, and the learning rate $\alpha = 0.025$ to align with the evaluation settings used in [35].

Performance metrics. We evaluate the performance of LEAD and comparison algorithms using three performance metrics: *per trip emissions*, *fairness in driver's utility*, and *rider waiting time*. For fairness, we normalize our results to LAF outcomes when comparing different algorithms. However, in analyzing the effect of LEV penetration, we normalize the values to the LEAD without the addition of LEVs. We also report the proportion of trips assigned to low-emission vehicles (LEVs) and high-emission vehicles (HEVs), as well as the proportion of successfully matched requests.

5.2 Impact on Per-Trip Emissions

Figure 2 shows the performance of LEAD in reducing the average emissions per trip against the baselines across various parameters.

Key results. As shown in Figure 2(a), LEAD outperforms all the baseline algorithms for all batch durations. LEAD reduces emissions per trip by at least 73.6%, 70.5%, and 58.3% when compared to CD, TORA, and LAF algorithms, respectively. LEAD does even better when the batch duration increases. While the emissions per trip for CD and TORA slightly decrease with an increase in batch duration, LEAD achieves significantly higher reductions at longer batch duration, thereby increasing the margin of improvement. Finally, LAF's performance worsens with an increase in batch duration; emissions are 9% and 3% higher at 15 minutes than at 2 and 5 minutes, respectively.

As shown in Figure 2(b), adding low-emission vehicles (LEVs) to the ridesharing fleet reduces the average emissions per trip. Adding 10% LEVs provides up to 8.2% decrease in emissions; adding 15% LEVs reduces emissions by up to 13.5%; adding 25% LEVs achieves up to 30.9% reductions in emissions. Importantly, the lowest reductions for all LEV values occur at the longest batch duration, such as adding 10% LEVs at 15 minutes yields no added benefit.

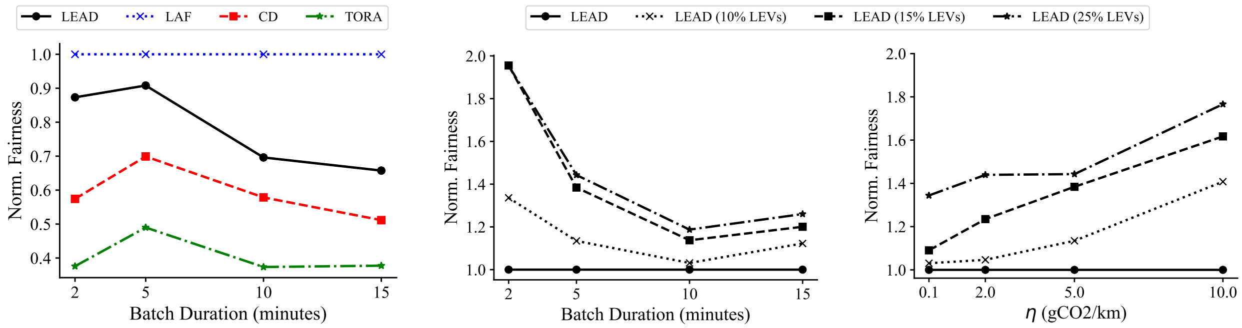
As shown in Figure 2(c), η does not have a significant impact on the emissions per trip. The highest increase in the emissions per trip is 5.15%, which is observed for an LEV percentage of 25%. As we increase η to 10gCO2/km, the emissions only increase by 1.9% and 2.5% at 10% and 15% LEV percentages, respectively.

Analysis of findings. We next take a deeper look at some of the results we have observed about LEAD's impact on emissions.

- **Why does LEAD outperforms deadhead miles- and emissions-aware algorithms CD and TORA?** The primary reason that LEAD outperforms CD is that CD does not take into account the unit emissions of the vehicles when assigning requests to drivers. Another reason that LEAD outperforms TORA and CD is that it assigns all the requests in each batch simultaneously, whereas TORA and CD assign the requests sequentially within each batch.

- **Why does adding LEVs yield fewer emissions reductions at longer batches?** As the batch duration increases, LEAD can access a bigger set of drivers and riders, allowing it to make low-emission ride assignments, even without a high LEV percentage. The effect is more acute at smaller percentages of LEVs.

- **Why do the emissions per trip for LEAD not increase with η?** The value of η shows the grams of CO2 emitted to increase fairness by 1km. LEAD makes this tradeoff highly judiciously and only chooses the trips when improvement in fairness is possible.



(a) `LEAD` vs. baselines (b) Impact of LEVs (c) Impact of Eta (η).

Figure 3: Fairness performance: (a) normalized fairness for different algorithms as a function of batch duration, (b) impact of the percentage of low emission vehicles (LEVs) in the fleet on `LEAD`, and (c) the impact of increasing emissions for an increase in fairness, captured using η. Here, η specifies extra emissions that the algorithm incurs to reduce unfairness by 1km.

Key takeaways. *`LEAD` outperforms the state-of-the-art emission-aware ride assignment algorithm `TORA` by up to 80.7%. While additional reductions may be possible by optimizing batch duration and η, it will introduce tradeoffs with our other objectives of increasing driver's fairness and reducing rider's wait time.*

5.3 Impact on Driver's Fairness

Figure 3 shows the performance of `LEAD` in improving fairness among drivers against the baselines across various parameters.

Key results. As shown in Figure 3(a), no other algorithm, including `LEAD`, beats the fairness performance of `LAF`. The next best fairness performance is shown by `LEAD`, as it reaches 90.8% of the fairness achieved by `LAF` at the batch duration of 5 minutes. Importantly, `LEAD`'s worst performance of achieving 65% of `LAF`'s fairness is only marginally surpassed by `CD` algorithm which achieves 69% of `LAF`'s fairness performance. `TORA` is the worst performing algorithm as it achieves between 37% to 49% of `LAF`'s fairness performance.

As shown in Figure 3(b), adding LEVs gives the greatest benefits at low batch duration, where having 10% and 15% LEVs can improve fairness by 36% and to almost 2$\times$, respectively, compared to baseline ridesharing fleet with `LEAD`. Increasing LEVs beyond 15% provides marginal gains in fairness across all batch durations. Finally, the gains in fairness have the lowest point at a medium batch duration of 10 minutes and increase on either side of the batch duration.

Figure 3(c) shows the effect of η on fairness, providing us with two key observations. First, increasing η increases fairness, with a 37% increase at 0.1g.CO2/km and 80% increase at 10g.CO2/km. Second, even a small percentage of LEVs provide a very big improvement in fairness at the higher values of η; 15% LEVs can improve fairness by 40% at η value of 10g.CO2/km.

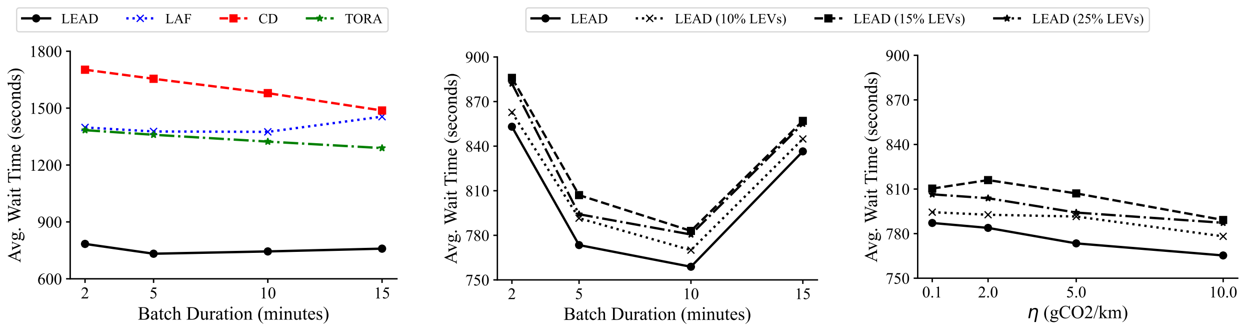
Analysis of findings. We next look deeper at the results we presented to quantify the fairness properties of `LEAD`.

- **Why does fairness first increase and then decrease with increasing batch duration?** As the batch duration increases, more opportunities exist to increase fairness and reduce emissions. The absolute fairness (not shown in the graph) for both `LEAD` and `LAF` increases. However, since `LAF` uses the benefits from batch duration only to improve fairness, while `LEAD` uses it for both increasing fairness and decreasing emissions, the normalized performance of `LEAD` with respect to `LAF` decreases.

- **Why does longer batch duration suppress the effect of LEVs on improving fairness?** When the batch duration is short, there are fewer available drivers, limiting the algorithm to a smaller pool and reducing its ability to prioritize low-emission vehicles. However, with longer batch durations, the algorithm has a wider selection, allowing it to choose low-emission vehicles more effectively, which can lead to reduced fairness.

- **How increasing η to improve fairness impacts emissions?** As outlined in the previous section, η has a marginal impact on emissions, with a maximum increase of 5.15%. Since increasing η can improve fairness by almost 80%, the tradeoff seems desirable.

Key takeaways. *`LEAD` offers a highly favorable tradeoff between improving fairness and reducing emissions. For example, at batch duration of 5 minutes, it reduces emissions by 65% while degrading fairness by less than 9%. However, using η, fairness can be made comparable with `LAF` for a less than 2% increase in emissions.*



(a) LEAD vs. baselines (b) Impact of LEVs (c) Impact of Eta (η).

Figure 4: Wait time performance: (a) wait time for different algorithms as a function of batch duration, (b) impact of the percentage of low emission vehicles (LEVs) in the fleet on LEAD performance, and (c) the impact of increasing emissions for an increase in fairness, captured using η. Here, η specifies extra emissions that the algorithm incurs to reduce unfairness by 1km.

5.4 Impact on Rider's Wait Time

Figure 4 shows the impact of LEAD on the wait time for riders against the baselines across various parameters. Note that CD targets reduce wait time and deadhead miles (or emission) by assigning the closest driver. TORA is explicitly wait time aware as it bounds the wait time experienced by the rider.

Key results. As shown in Figure 4(a), LEAD outperforms the baseline algorithms for all batch duration. The reduction in wait time over CD is the most significant, with 49% to 54% improvement. The performance improvement over LAF and TORA is always more than 40%. Finally, the increase in batch duration has a higher impact on our baseline algorithms, while LEAD achieves its best wait time performance at a batch duration of 5 minutes.

Figure 4(b) shows that the wait time for the riders gets worse at a higher penetration of LEVs. However, the increase in wait time, even at the worst point, is 4.16% (for 15% LEVs at 5-minute batch duration). Interestingly, increasing LEVs from 15% to 25% reduces the wait time, but by at most 1.6% and at least 0.2%. Figure 4(c) shows that increasing η has a small effect as well, between 2.1% to 2.8% for different penetration levels of LEVs.

Analysis of findings. We now do a deep dive into our results.

- **Why does wait time for LEAD increase after batch duration of 5 minutes?** An increase in the batch duration puts a lower bound on the wait time for all the rides in the batch. However, an initial increase in batch size duration increases the possibility of a driver coming closer to the rider and becoming available. However, beyond a certain batch duration, the effect of increasing the lower-bound dominates, and wait time goes up.

- **Why having more LEVs is bad for wait time? How does it impact the usefulness of LEAD?** As LEAD targets reducing emissions, it will prioritize EVs even if they are farther away or will only be available after finishing a ride, thus increasing the wait time. While the increase in wait time is not desirable, the increase is not significant, and the addition of LEVs provides significant emission reduction and fairness benefits.

Key takeaways. *LEAD outperforms CD by considering the future availability of the drivers, same as TORA. However, its explicit focus on reducing deadhead miles reduces wait time beyond TORA. Importantly, LEAD' wait time is less sensitive to batch duration and η, allowing for reducing emissions and increasing fairness.*

5.5 Additional Performance Benchmarking

In this section, we benchmark LEAD and other baselines using the additional metrics that evaluate system-level performance. The results are shown in Table 2 and Table 3.

LEAD outperforms baseline algorithms in improving utility by at least 19% and reducing deadhead-to-trip ratio by 15.2% over the second-best algorithm LAF. However, LAF rejects many ride requests, as outlined in the original paper, and accepts only 93.3% requests compared to 98.3% ride acceptance rate for LEAD.

Importantly, LEAD also solves TORA's problem of assigning a disproportionate number of rides to LEVs by allocating rides to LEVs in proportion to their penetration in the fleet.

6 Discussion and Future Works

In this paper, we propose LEAD, a Learning-based Equity-Aware Decarbonization approach for ridesharing platforms. LEAD minimizes emissions while ensuring fair

Table 2: Performance comparison of `LEAD` against baselines for $\eta = 5$ g.CO2/km and batch duration of 5 minutes.

Algorithm	Avg. utilities (km)	Avg. deadhead to trip ratio	Percentage of matched requests
CD	763.14	0.66	97.4%
TORA [32]	820.4	0.62	97.3%
LAF [35]	890.56	0.59	93.3%
LEAD (ours)	1065.09	0.50	98.3%

Table 3: Fraction of rides assigned to LEVs at various LEV %ages for η=5 g.CO2/km and batch duration of 5 minutes.

Algorithm	Original	10% LEV	15% LEV	25% LEV
TORA [32]	13.58	20.41	26.07	35.05
LEAD (ours)	6.9	11.60	16.61	26.16

distribution of drivers utility. `LEAD` leverages reinforcement learning to optimize rider-driver matches based on the expected future utility for drivers and projected carbon emissions for the platform without increasing rider waiting times. Extensive experiments using a public real-world ridesharing dataset demonstrate that `LEAD` outperforms state-of-the-art baselines in terms of reduction in emissions and wait time while providing a desirable tradeoff with fairness. In the future, we will explore fairness from a rider's perspective and short-term fluctuations in driver utility, in addition to the long-term effect this paper considers.

7 Acknowledgements

This research was supported in part by NSF grants CAREER 2045641, CNS-2102963, CNS-2106299, CPS-2136199, NRT-2021693, and NGSDI-2105494.

References

[1] B. M. Anthony and C. Chung. Online bottleneck matching. *Journal of Combinatorial Optimization*, 27(1):100–114, 2014.

[2] M. Asghari and C. Shahabi. Adapt-pricing: a dynamic and predictive technique for pricing to maximize revenue in ridesharing platforms. In *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 189–198, 2018.

[3] J. M. Barrios, Y. V. Hochberg, and H. Yi. The cost of convenience: Ridehailing and traffic fatalities. *Journal of Operations Management*, 69(5), 2023.

[4] C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis. A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. *Transportation Research Part E: Logistics and Transportation Review*, 165:102835, 2022.

[5] A. E. Brown. *Ridehail revolution: Ridehail travel and equity in Los Angeles*. University of California, Los Angeles, 2018.

[6] P. Cheng, H. Xin, and L. Chen. Utility-aware ridesharing on road networks. In *ACM International Conference on Management of Data*, 2017.

[7] Climate.gov. Does it matter how much the united states reduces its carbon dioxide emissions?, 2024.

[8] S. C. Davis and R. G. Boundy. Transportation energy data book: Edition 39. Technical report, Oak Ridge National Lab. (ORNL), 2021.

[9] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Price of fairness in kidney exchange. In *AAMAS*, pages 1013–1020, 2014.

[10] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3054–3063, 2020.

[11] X. Fang, J. Huang, F. Wang, L. Liu, Y. Sun, and H. Wang. Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2840–2848, 2021.

[12] R. S. Garfinkel. An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19(7):1747–1751, 1971.

[13] M. Graham and S. Joe. *Towards a fairer gig economy*. Meatspace Press, 2017.

[14] A. Henao. Impacts of ridesourcing - lyft and uber - on transportation including vmt, mode replacement, parking, and travel behavior. 2017.

[15] A. Henao and W. E. Marshall. The impact of ride-hailing on vehicle miles traveled. *Transportation*, 46(6):2173–2194, 2019.

[16] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye. Heteta: Heterogeneous information network embedding for estimating time of arrival. In

Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pages 2444–2454, 2020.

[17] International Energy Agency. World energy outlook 2022, 2022.

[18] E. Kontou, V. Garikapati, and Y. Hou. Reducing ridesourcing empty vehicle travel with future travel demand prediction. *Transportation Research Part C: Emerging Technologies*, 121:102826, 2020.

[19] D.-H. Lee, H. Wang, R. L. Cheu, and S. H. Teo. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record*, 1882(1):193–200, 2004.

[20] C. Levinger, N. Hazon, and A. Azaria. Human satisfaction as the ultimate goal in ridesharing. *Future Generation Computer Systems*, 112, 2020.

[21] X. Li, J. Gao, C. Wang, X. Huang, and Y. Nie. Ride-sharing matching under travel time uncertainty through data-driven robust optimization. *IEEE Access*, 10:116931–116941, 2022.

[22] Y. Li, J. Wan, R. Chen, J. Xu, X. Fu, H. Gu, P. Lv, and M. Xu. Top-k k vehicle matching in social ridesharing: A price-aware approach. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[23] P. Loa, S. M. Mashrur, and K. Nurul Habib. What influences the substitution of ride-sourcing for public transit and taxi services in toronto? an exploratory structural equation model-based study. *International Journal of Sustainable Transportation*, 17(1):15–28, 2023.

[24] M. O. Lorenz. Methods of measuring the concentration of wealth. *Publications of the American statistical association*, 9(70):209–219, 1905.

[25] H. Luo, Z. Bao, F. M. Choudhury, and J. S. Culpepper. Dynamic ridesharing in peak travel periods. *IEEE Transactions on Knowledge and Data Engineering*, 33(7):2888–2902, 2019.

[26] H. Ma, F. Fang, and D. C. Parkes. Spatio-temporal pricing for ridesharing platforms. *ACM SIGecom Exchanges*, 18(2):53–57, 2020.

[27] J. Ma, Q. Meng, L. Cheng, and Z. Liu. General stochastic ridesharing user equilibrium problem with elastic demand. *Transportation Research Part B: Methodological*, 162:162–194, 2022.

[28] RideAustin. Ride-austin-june6-april13. `https://data.world/ride-austin/ride-austin-june-6-april-13`, 2017.

[29] M. E. H. Sadati and B. Çatay. A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 149, 2021.

[30] M. Safaeian, R. Khayamim, E. E. Ozguven, et al. Sustainable decisions in a ridesharing system with a tri-objective optimization approach. *Transportation Research Part D: Transport and Environment*, 2023.

[31] M. Sahebdel, A. Zeynali, N. Bashir, M. H. Hajiesmaili, and J. Oke. Data-driven algorithms for reducing the carbon footprint of ride-sharing ecosystems. In *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023.

[32] M. Sahebdel, A. Zeynali, N. Bashir, P. Shenoy, and M. Hajiesmaili. A holistic approach for equity-aware carbon reduction of the ridesharing platforms. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, pages 361–372, 2024.

[33] M. Schnieder. Effective speed: Factors that influence the attractiveness of cost effective and sustainable modes of transport in cities. *Sustainability*, 15(10):8338, 2023.

[34] S. Shaheen, A. Cohen, I. Zohdy, et al. Shared mobility: current practices and guiding principles. Technical report, United States. Federal Highway Administration, 2016.

[35] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, and Q. Yang. Learning to assign: Towards fair task assignment in large-scale ride hailing. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3549–3557, 2021.

[36] K. Shi, R. Shao, J. De Vos, L. Cheng, and F. Witlox. The influence of ride-hailing on travel frequency and mode choice. *Transportation Research Part D: Transport and Environment*, 101:103125, 2021.

[37] A. Singh, A. O. Al-Abbasi, and V. Aggarwal. A distributed model-free algorithm for multi-hop ridesharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(7), 2021.

[38] J. Sun, H. Jin, Z. Yang, and L. Su. Optimizing long-term efficiency and fairness in ride-hailing under budget constraint via joint order dispatching and driver

repositioning. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[39] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[40] X. Tang, Z. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1780–1790, 2019.

[41] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu. Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the VLDB Endowment*, 9(12):1053–1064, 2016.

[42] U.S. Environmental Protection Agency. Fast facts on transportation greenhouse gas emissions, 2024.

[43] J. Wang, P. Cheng, L. Zheng, C. Feng, L. Chen, X. Lin, and Z. Wang. Demand-aware route planning for shared mobility services. *Proceedings of the VLDB Endowment*, 13(7):979–991, 2020.

[44] X. Wang, N. Agatz, and A. Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 52(4):850–867, 2018.

[45] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2019.

[46] T. Wenzel, C. Rames, E. Kontou, and A. Henao. Travel and energy implications of ridesourcing service in austin, texas. *Transportation Research Part D: Transport and Environment*, 70:18–34, 2019.

[47] P. Xu, Y. Shi, H. Cheng, J. Dickerson, K. A. Sankararaman, A. Srinivasan, Y. Tong, and L. Tsepenekas. A unified approach to online matching with conflict-aware constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 2221–2228, 2019.

[48] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li. An efficient insertion operator in dynamic ridesharing services. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3583–3596, 2020.

[49] L. Yan, X. Luo, R. Zhu, P. Santi, H. Wang, D. Wang, S. Zhang, and C. Ratti. Quantifying and analyzing traffic emission reductions from ridesharing: A case study of shanghai. *Transportation Research Part D: Transport and Environment*, 89:102629, 2020.

[50] A. S. Yengejeh and S. L. Smith. Rebalancing self-interested drivers in ride-sharing networks to improve customer wait-time. *IEEE Transactions on Control of Network Systems*, 8(4):1637–1648, 2021.

[51] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2151–2159, 2017.