Dynamic Car Dispatching and Pricing: Revenue and Fairness for Ridesharing Platforms

Zishuo Zhao[1], **Xi Chen**[2], **Xuefeng Zhang**[1] and **Yuan Zhou**[3*]
[1]University of Illinois Urbana-Champaign,
[2]New York University,
[3]Yau Mathematical Sciences Center, Tsinghua University
zishuoz2@illinois.edu, xc13@stern.nyu.edu, xuefeng8@illinois.edu, yuan-zhou@tsinghua.edu.cn

Abstract

A major challenge for ridesharing platforms is to guarantee profit and fairness simultaneously, especially in the presence of misaligned incentives of drivers and riders. We focus on the dispatching-pricing problem to maximize the total revenue while keeping both drivers and riders satisfied. We study the computational complexity of the problem, provide a novel two-phased pricing solution with revenue and fairness guarantees, extend it to stochastic settings and develop a dynamic (a.k.a., learning-while-doing) algorithm that actively collects data to learn the demand distribution during the scheduling process. We also conduct extensive experiments to demonstrate the effectiveness of our algorithms.

1 Introduction

Ridesharing is a novel form of sharing economy that utilizes mobile apps to match drivers and riders to allow riders to take trips conveniently and make profits for drivers. Compared to traditional taxi platforms, ridesharing platforms enable riders to put orders on the system in advance of the trip for drivers to take, so that the system can optimally plan the rides to make it more efficient. Previous studies on planning algorithms for ridesharing platforms adopt a variety of methodologies including combinatorial optimization [Bei and Zhang, 2018], reinforcement learning [Li *et al.*, 2019], or both [Qin *et al.*, 2020]. However, planning trips only in the centralized way does not guarantee that each individual driver and rider has the incentive to obey the plan, which calls for efficient and fair pricing mechanisms so that following the plan will be "happy" for each party and maximize their utilities.

The pricing mechanism for taxi platforms depends on distance and waiting time, but it is too simple to either well represent the cost of drivers or match the supply and demand, which may result in dissatisfaction on both sides and lead to refusal of trips. For example, a rider wants to take an important trip with a short distance and a low price. However, there is a traffic jam and it may take a long time for the driver to cover the trip. This situation will create an opportunity cost that discourages the driver to accept the order. Were the charged price higher, the rider would probably not mind the slight increase of cost but the driver will be satisfied to accept the trip, which benefits both parties. However, we should be careful about the price adjustment: if two friends take the same trip, but at different prices, the one who takes the trip with a higher price may "envy" the other and will be dissatisfied with the platform. This issue may also apply to the drivers: if two drivers initially at the same time and location are assigned different trips that earn different profits, the driver with lower profit would also be dissatisfied with the platform. Therefore, to make the platform satisfied by each agent, the algorithm should be "envy-free" (as in Definition 5). Another important property is "subgame-perfect Nash equilibrium", which means that each driver is assigned with a plan, following which he/she can get the best utility among all alternative actions given others' actions are fixed, so that no driver has the incentive to deviate from the plan (as in Definition 4).

In this paper, we propose a fairness-aware algorithmic framework for dynamic car dispatching and pricing, which consists of the following three-fold contributions:

1. We study the computational complexity of the task of dispatching and pricing for total revenue maximization, propose a versatile generalized network flow model for the task, and provide theoretical guarantees (Section 3).
2. We propose a novel two-phase pricing mechanism that decouples and sets different prices on drivers' and riders' sides, which can adapt to situations where the drivers' and riders' interests misalign[1] and guarantee fairness for both parties (Section 4).
3. We consider the stochastic nature of ridesharing orders and study the online learning setting. We natural extend the model to the stochastic setting (Section 5), enabling the use of Thompson sampling-based algorithm to learn the valuation distributions from the partial information given by the riders' responses, and balance the exploration-exploitation trade-off (Appendix E).

Finally, in Section 6, we perform extensive experimental evaluations of our assumptions and algorithms in the real-world

*Corresponding Author

[1]Please see the illustrative example in Appendix C.

datasets and demonstrate the effectiveness of our methods.

We have also shown that our algorithm runs in polynomial time. Please refer to Section 7 for detailed complexity analysis.

Related Works. There are several related works in the existing literature. Bei and Zhang; Qin *et al.*; Wang *et al.* [2018; 2020; 2018] study how to dispatch the drivers efficiently in a centralized way, and Hrnčíř *et al.*; Li *et al.* [2015; 2019] study the dispatching problem via multiagent systems, but they do not consider pricing which is essential for the application in platforms. Riquelme *et al.* [2015] study optimal pricing via queue theory, but they assume a single location, which is too simple for application. Castillo *et al.* [2017] discuss the phenomenon of "wild goose chase" in which drivers spent most time driving to take a distant order in unbalanced supply and demand, and propose the method of adjusting price to avoid its detriment to efficiency, but they do not consider fairness. Bimpikis *et al.* [2019] look into the effects of pricing to supply-demand balance, revenue and consumers' surplus, but adopt an over-simplified model of n pairwise equidistant locations, which is not even geometrically possible for large n. Yan *et al.* [2018] also provide an algorithm for dynamic matching and pricing, but the matching and pricing algorithms are decoupled, making the performance suboptimal. In particular, the very recent work [Ma *et al.*, 2020], which shares a similar motivation as our work, studies how to maximize social welfare, i.e. the summation of riders' valuations minus drivers' costs among all trips, via an bidding-based dispatching and pricing algorithm. In that paper, each rider should bid a maximally acceptable price for them, and the truthful mechanism guarantees that it is in each rider's interest to report their true valuation. However, there are some gaps from their mechanism to the reality. First of all, it is not practical for riders to bid their valuation like an auction. Second, the mechanism maximizes total social welfare, not drivers' revenue, but ride-sharing platforms are indeed interested in their profits. Also, it assumes that all future orders is known at the beginning, which is not realistic. In contrast, our algorithm optimizes the total revenue via dynamically learning the order distribution from the riders' responses on our carefully designed prices.

2 Preliminaries

We assume the service zone is divided into a family L of discrete locations, and the planning horizon is a family T of discrete time slots. Therefore, there are $|L| \cdot |T|$ spatiotemporal *states*, denoted by $S = L \times T$.

We also assume that the travelling time from one state $s = (l, t) \in S$ to another location s' is deterministically defined by the known function $\delta(l, l', t) \in \mathbb{Z}^+$. We call each pair of the spatiotemporal states (s, s') a spatiotemporal *arc*. For each $s = (l, t)$ and $s' = (l', t')$, we say the arc (s, s') is *admissible* if $t' \geq t + \delta(l, l', t)$. We denote by Q the set of all admissible arcs.

Each admissible arc $(s, s') \in Q$ is associated with a known deterministic cost $c(s, s')$ which is incurred to any driver that drives along this arc. The order of the i-th rider is described by an admissible arc $(s_i, s'_i) \in Q$ and a valuation v_i which is the maximum amount the rider would like to pay for the ride. Since v_i is not revealed to the ridesharing platform, we call $o_i = (s_i, s'_i, v_i)$ the i-th *latent order*, and denote $R = \{o_i, \forall i\}$ the set of latent orders.

The task of the scheduling algorithm for the ride-sharing platform involves the decision of a rider-side pricing function $p : S \times S \to [0, +\infty)$ (which has to be independent of the rider to ensure envy-freeness). For each rider i with latent order $o_i = (s_i, s'_i, v_i)$, the scheduling algorithm offers the price $p(s_i, s'_i)$. The rider only accepts the offer if $v_i \geq p(s_i, s'_i)$ in which case the platform receives $p(s_i, s'_i)$ as income. Serving the order also incurs the driver cost according to $c(\cdot, \cdot)$ along the arcs. After all trips, the drivers will leave the platform.

The first goal of the scheduling algorithm is to maximize the total revenue which is defined to be the total income (collected from the riders) minus the total cost (incurred by the drivers). Then, the second goal of the scheduling algorithm is to compute the driver-side payment function $y : S \times S \to [0, +\infty)$ to distribute the income to the drivers in a subgame-perfect and envy-free manner. (Note that the payment function also has to be independent of the drivers to ensure envy-freeness.)

The above-described scheduling problem involves the complex optimization of multiple sets of decision variables. The unknown latent order set introduces further challenges to the task. To approach this complex problem, we will first consider the deterministic setting where the latent order set R is fully revealed to the scheduling algorithm, and the scheduling problem becomes a pure static optimization task. Then, we assume that R is drawn from a latent distribution, and design an online learning algorithm that simultaneously learns the latent distribution and optimizes the total revenue.

In the following two sections, we describe each phase of the problem with more details and mathematical rigor, and propose our algorithms to achieve the optimal policy.

3 Phase 1 of the Deterministic Setting: Maximum Revenue Car Dispatching

In this section, we introduce our algorithm to the maximum revenue car dispatching problem in the deterministic setting (i.e., when the set of latent orders R is known to the platform). For convenience, we first introduce the following non-linearly weighted circulation (NLWC) problem, and the maximum revenue car dispatching problem can be formulated based on NLWC definition.

Definition 1. *In the* non-linearly weighted circulation (NLWC) *problem, there is a directed graph $G = (V, E)$. For each directed edge $e \in E$, we associate it with the* flow lower bound $\ell(e)$, *the* flow upper bound $u(e)$ *and the* reward function $r(\cdot; e) : \mathbb{N} \to \mathbb{R}$. *The goal is to find a flow $f : E \to \mathbb{N}$ so that f satisfies the flow upper and lower bounds (i.e., $\ell(e) \leq f(e) \leq u(e), \forall e \in E$) and flow conservation (i.e., $\sum_{e \text{ going out of } s} f(e) = \sum_{e \text{ going into } s} f(e), \forall s \in V$), and the total reward $\sum_{e \in E} r(f(e); e)$ is maximized.*

Observe that when the reward functions are linear (i.e., $r(x; e) = w(e) \cdot x$), the NLWC problem becomes the canonical minimum cost circulation problem, which admits a polynomial time algorithm [Tardos, 1985] (with the signs of the linear coefficients flipped).

With the formulation of the NLWC problem in place, we are ready to describe our maximum revenue car dispatching problem in the deterministic setting. Here, we assume that the platform knows all the riders' information; i.e., for each rider i, we know that his/her latent order $o_i = (s_i, s'_i, v_i)$. Based on this information, for each arc $(s, s') \in Q$, we calculate the number of latent orders following the arc and denote it by $o(s, s')$; then, for each $1 \leq i \leq o(s, s')$, we define $v_i(s, s')$ to be the i-th largest valuation among all latent orders following (s, s'). Note that if the platform plans to accept k orders on the arc (s, s'), to maximize the income, the price should be set as $p(s, s') = v_k(s, s')$, and the total income generated from this arc becomes $k \cdot v_k(s, s')$.

In light of the discussion above, we will construct a directed graph (V_0, E_0) so that the maximum revenue car dispatching problem becomes calculating NLWC on the graph, where the flow along each arc indicates the number of drivers the platform plans to dispatch.

We first let the vertex set $V_0 = S \cup \{I, O\}$ where I is the artificial source and O is the artificial sink; together, a directed edge $e_{O,I}$ that goes from O to I is set up with $\ell(e_{O,I}) = 0$ flow lower bound and $u(e_{O,I}) = +\infty$ flow upper bound and the constant-zero reward function: $r(\cdot; e_{O,I}) \equiv 0$. We then set up the following sets of edges.

- (**Initialize drivers.**) For each spatiotemporal state s with n_s initial drivers, we set up a directed edge $e_{I,s}$ going from I to s with both flow upper and lower bounds equal to $\ell(e_{I,s}) = u(e_{I,s}) = n_s$, and the constant-zero reward function $r(\cdot; e_{I,s}) \equiv 0$. The flow $f(e_{I,s})$ represents the number of drivers to start working from the state s.
- (**Leaving drivers.**) For any spatiotemporal state s, we set up a directed edge $e_{s,O}$ going from s to O with $\ell(e_{s,O}) = 0$ lower bound, $u(e_{s,O}) = +\infty$ upper bound, and the constant-zero reward function $r(\cdot; e_{s,O}) \equiv 0$. The flow $f(e_{s,O})$ represents the number of drivers to leave the system at the state s.
- (**Driving without a rider.**) For any admissible arc $(s, s') \in Q$, we set up a directed edge $e^{(\mathrm{o})}_{s,s'}$ going from s to s' with $\ell(e^{(\mathrm{o})}_{s,s'}) = 0$ lower bound, $u(e^{(\mathrm{o})}_{s,s'}) = +\infty$ upper bound. The flow $f = f(e^{(\mathrm{o})}_{s,s'})$ represents the number of drivers to drive through the arc (s, s') without carrying a rider. Therefore, we set up the linear reward function $r(f; e^{(\mathrm{o})}_{s,s'}) = -c(s, s') \cdot f$.
- (**Driving with a rider.**) For any admissible arc $(s, s') \in Q$, we set up a directed edge $e^{(\mathrm{w})}_{s,s'}$ going from s to s' with $\ell(e^{(\mathrm{w})}_{s,s'}) = 0$ lower bound, $u(e^{(\mathrm{w})}_{s,s'}) = o(s, s')$ upper bound. The flow $f = f(e^{(\mathrm{w})}_{s,s'})$ represents the number of drivers to drive through the arc (s, s') with a rider. Therefore, we define the non-linear reward function $r(f; e^{(\mathrm{w})}_{s,s'}) = [v_f(s, s') - c(s, s')] \cdot f$.

Given (V_0, E_0), the maximum revenue car dispatching problem in the deterministic setting is equivalent to finding the optimal solution to NLWC on the directed graph (V_0, E_0). Formally, we directly have the proposition below.

Proposition 1. *Let f^* be the optimal solution to NLWC on the directed graph (V_0, E_0). To achieve the maximum revenue in the car dispatching task, the platform may direct the drivers to drive with/without carrying a rider or leave the platform based on the flow value on the corresponding sets of edges. The total weight of f^* is the maximum revenue the platform may collect.*

Proposition 1 also enables us to design the *routing plan* for each individual driver based on the NLWC solution. Formally, a *route* $A = (a_1, a_2, \ldots, a_z)$ is a sequence of spatiotemporal arcs such that the ending state of each arc a_i is the same as the beginning state of the next arc a_{i+1} (for all $i \in \{1, 2, \ldots, z-1\}$). At each time step and for each driver q, the *routing plan* A_q is just a route which starts at the driver's current state.

While the general NLWC problem is computationally intractable, the maximum revenue car dispatching problem, as a special case of NLWC, is unfortunately not easier. Formally, we present the following negative result for the maximum revenue car dispatching problem. The proof of Theorem 1 is deferred to Appendix A.1. Note that since maximum revenue car dispatching is a special case, we may not directly use the NP-Hardness proof of NLWC, and have to design a new hardness instance instead.

Theorem 1. *The maximum revenue car dispatching problem, even in the deterministic setting, is NP-hard.*

On the positive side, we propose a natural *regularity condition* in Definition 2. We will show that when the condition is satisfied, the maximum revenue car dispatching problem can be solved in polynomial time.

Definition 2 (Regularity)**.** *We say that a maximum revenue car dispatching problem instance satisfies the* regularity condition *if for each admissible spatiotemporal arc (s, s'), and each $k \in \{1, 2, \ldots, o(s, s')\}$, the sequence $v'_k(s, s')$ is monotonically non-increasing with k, where we define*

$$v'_k(s, s') := \begin{cases} v_1(s, s') & (k = 1) \\ k \cdot v_k(s, s') - (k-1) v_{k-1}(s, s') & (k \geq 2) \end{cases} .$$

In the definition, $v'_k(s, s')$ can be interpreted as the *marginal reward* of accepting the k-th highest price order on arc (s, s'). The regularity condition then requires that the marginal reward sequence is not increasing with the increasing number of accepted orders on any arc, which is a standard assumption in economics literature (see, e.g., [Al *et al.*, 2005; Zhao *et al.*, 2015; Wang and Zhang, 2011]). Indeed, in our empirical evaluation, we verify that the regularity condition holds in the real-world data.

We now present our edge decomposition algorithm (details in Algorithm 1) for the maximum revenue car dispatching problem. At a higher level, Algorithm 1 first manages to decompose each non-linear directed edge in (V_0, E_0) to a family of edges with linear costs and creates a minimum linear-cost circulation problem instance $(V_0, \tilde{E}, \tilde{\ell}, \tilde{u}, -\tilde{w})$, then invokes the existing polynomial-time time algorithm for the minimum linear-cost circulation problem, and finally aggregates

Algorithm 1: The Edge Decomposition Algorithm

1: Construct the NLWC instance (V_0, E_0, ℓ, u, r);
2: $E_1 \leftarrow \{e^{(\mathrm{w})}_{s,s'} \in E_0\}$, $E_2 \leftarrow E_0 - E_1$; $\tilde{E} \leftarrow \emptyset$;
3: **for** $e^{(\mathrm{w})}_{s,s'} \in E_1$ **do**
4: **for** $i \in \{1, 2, \ldots, o(s, s')\}$ **do**
5: $\tilde{E} \leftarrow \tilde{E} \cup e^{(\mathrm{w},i)}_{s,s'}$; $(\tilde{\ell}(e^{(\mathrm{w},i)}_{s,s'}), \tilde{u}(e^{(\mathrm{w},i)}_{s,s'})) \leftarrow (0, 1)$;
6: $w(e^{(\mathrm{w},i)}_{s,s'}) \leftarrow r(i; e^{(\mathrm{w})}_{s,s'}) - r(i-1; e^{(\mathrm{w})}_{s,s'})$
7: **end for**
8: **end for**
9: **for** $e \in E_2$ **do**
10: $\tilde{E} \leftarrow \tilde{E} \cup e$; $\left(\tilde{\ell}(e), \tilde{u}(e)\right) \leftarrow (\ell(e), u(e))$;
11: $\tilde{w}(e) \leftarrow \begin{cases} -c(s, s') & \text{(if both } s, s' \in S) \\ 0 & \text{(otherwise)} \end{cases}$;
12: **end for**
13: Invoke the polynomial-time algorithm [Tardos, 1985] to compute the minimum cost circulation of $(V_0, \tilde{E}, \tilde{\ell}, \tilde{u}, -\tilde{w})$ where $-\tilde{w}$ is the coefficient function of the linear costs, denote the optimal flow by $\tilde{f}$;
14: **for** $e \in E_0$ **do**
15: **if** $e = e^{(\mathrm{w})}_{s,s'} \in E_1$ **then** $f(e) \leftarrow \sum_i \tilde{f}(e^{(\mathrm{w},i)}_{s,s'})$;
16: **else** $f(e) \leftarrow \tilde{f}(e)$;
17: **end for**
18: **return** f;

the flows in each family to construct the optimal solution to the original problem.

In Algorithm 1, the edge set E_1 denotes the edges corresponding to "driving with a rider" and E_2 the rest of the edges. We also observe that the only non-linear edges are the ones to drive with a rider (in E_1), while the rest edges (in E_2) already have linear costs. For the edges in E_1, the algorithm decomposes them from Line 3 to Line 8: since the flow on each edge in E_1 represents the amount of the rider orders accepted along the corresponding spatiotemporal arc, the algorithm assigns each decomposed edge with unitary capacity, and the corresponding flow represents an additional order to be accepted along the arc, and naturally the weight function is defined based on the marginal reward function $v'_k(\cdot, \cdot)$. Also note that the algorithm always returns an integral flow because of the integrality property of the minimum linear-cost circulation problem. Regarding the theoretical guarantee of Algorithm 1, we prove the following theorem:

Theorem 2. *Algorithm 1 runs in polynomial time, and when the regularity condition is met, the returned flow f achieves the maximum revenue of the car dispatching problem on the directed graph (V_0, E_0).*

Proof. We only need to prove that in the NLWC problem with regularity, each non-linear edge in E_1 with finite capacity can be substituted by a finite number of linear edges.

Consider an edge $e = e^{(\mathrm{w})}_{s,s'} \in E_1$, then $\ell(e) = 0$. Then, for each $i \in N$ s.t. $1 \leq i \leq u(e)$, we add to $\tilde{E}$ an linear edge $e_i(s, s', 0, 1, w(i) - w(i-1))$. Since $r(i; e^{(\mathrm{w})}_{s,s'}) - r(i - 1; e^{(\mathrm{w})}_{s,s'})$ decreases with i, when we should put t amount of flow from s, s' in G_1, the optimal plan is to saturate edges $e^{(\mathrm{w},1)}_{s,s'}, e^{(\mathrm{w},2)}_{s,s'}, \cdots, e^{(\mathrm{w},t)}_{s,s'}$, with total reward $r(t; e^{(\mathrm{w})}_{s,s'})$, identical to the NLWC model.

Therefore, we realize the same edge-reward function as the NLWC model with a minimum cost circulation model. While the Maximum Revenue Car Dispatching problem needs integer solutions, from the total unimodularity property of the minimum cost circulation problem, it is guaranteed that our algorithm outputs an integer basic solution. Therefore, we can indeed solve regular Maximum Revenue Car Dispatching via the minimum cost circulation problem. □

We also remark that even when in the general scenario (without the regularity condition), a simple variation of Algorithm 1 also serves as a good approximation to the optimal solution. It virtually approximates the edge reward function by its concave envelope to "iron" it to a concave function [Chawla *et al.*, 2007]. Please refer to Appendix B for details.

4 Phase 2 of the Deterministic Setting: Fair Reward Re-allocation to Drivers

Recall that in Phase 1 we have found the maximum revenue that can be achieved by any dispatching plan in the deterministic setting. Along the way, we have also figured out how many drivers are needed for a spatiotemporal arc $(s, s') \in Q$ with a rider (namely $f(e^{(\mathrm{w})}_{s,s'})$) and without carrying a rider (namely $f(e^{(\mathrm{o})}_{s,s'})$). For convenience, we define $F(s, s') := f(e^{(\mathrm{w})}_{s,s'}) + f(e^{(\mathrm{o})}_{s,s'})$ to be the total number of drivers we plan to dispatch along the arc (s, s'). In this section, we develop methods to figure out the fair payment scheme $y : S \times S \to [0, +\infty)$ for driving along each spatiotemporal arc to ensure that the drivers are well incentivized to cooperate with the platform and execute the optimal-revenue dispatching plan. Formally, we define the fairness condition as follows.

Definition 3 (Fair re-allocation)**.** *A re-allocation scheme is* fair *if and only if following conditions are satisfied:*

- *Budget-balance. Let $\mathcal{I}$ be the total income collected from the riders. This should also be the exact amount to be distributed to the drivers.*[2] *Formally, it is required that $\sum_{(s,s') \in Q} y(s, s') \cdot F(s, s') = \mathcal{I}$.*
- *Non-negative producer surplus. For each arc driven, the payment should be at least the cost; i.e., for each $(s, s') \in Q$ so that $F(s, s') > 0$, it is required that $y(s, s') \geq c(s, s')$.*
- *Subgame-perfectness. This is formally defined soon in Definition 4 which, together with the non-negative producer surplus condition, makes sure that the drivers do not have the incentive to refuse and deviate from the dispatching plan.*
- *Envy-freeness. This is formally defined in Definition 5 which makes sure that the drivers do not complain that the dispatching plan is more favorable to others than themselves.*

[2] We omit the amount that the platform would like to keep for profit, which can be easily added to the constraint w.l.o.g.

Note that we need to define subgame-perfectness and envy-freeness in details. Before doing this, we need to introduce a few new notations and definitions.

We will model the drivers' behavior as an *extensive game* [Glazer and Rubinstein, 1996], where, at each state, each driver has the freedom to choose any route starting from the current state. At any time step, let A_q denote the routing plan given by the platform for the driver q, let $\mathcal{A} := \{A_1, A_2, \dots\}$ denote the set of routing plans for all drivers, and let $A_{-q} := \mathcal{A} \setminus \{A_q\}$. For each driver q, let $u_q(\mathcal{A})$ denote the utility (i.e., net profit) of driver q if all drivers follow the routing plan $\mathcal{A}$. In particular, we have that $u_q(\mathcal{A}) = \sum_{(s,s') \in A_q}(y(s,s') - c(s,s'))$.

The subgame-perfectness condition requires that given reward re-allocation scheme and the set of routing plans for all drivers by the platform, any driver q does not have the incentive to deviate from the routing plan given to him/her. Formally, we make the following definition.

Definition 4 (Subgame-perfectness)**.** *A reward re-allocation scheme is* subgame-perfect *if at any time step, let* $\mathcal{A} := \{A_1, A_2, \dots\}$ *be the routing plans decided by the platform, and for any driver q, and for each route* A'_q *sharing the same starting state as* A_q*, it holds that* $u_q(A_q, A_{-q}) \geq u_q(A'_q, A_{-q})$.

Note that in game theory, a subgame-perfect Nash equilibrium in a extensive game is a strategy profile for the agents such that at any point of the game, the agents' strategies form a Nash equilibrium for the continuation of the game. Definition 4 requires that reward re-allocation scheme makes sure that the routing plan given by Proposition 1 is a subgame-perfect Nash equilibrium.

We would also like to make sure that each driver does not feel comparably inferior than others at the same state. Formally, we define the envy-freeness condition as follows.

Definition 5 (Envy-freeness)**.** *A reward re-allocation scheme is* envy-free *if at any time step, let* $\mathcal{A} := \{A_1, A_2, \dots\}$ *be the routing plans decided by the platform, and for any two drivers q and* q' *staying at the same state, it holds that* $u_q(\mathcal{A}) = u_{q'}(\mathcal{A})$.

Now we have completed the formal definition of a fair re-allocation scheme. The following lemma provides an elegant characterization of all fair re-allocation schemes and enables us to find such schemes only among the potential-based re-allocation algorithms. The proof of Lemma 1 can be found in Appendix A.2.

Lemma 1. *Given a routing plan* $\mathcal{A}$*, a reward re-allocation is fair if and only if there exists a corresponding* potential function $P : S \to \mathbb{R}^{\geq 0}$ *such that*

1. *For any* $s \in S$ *where* $\mathcal{A}$ *directs at least one driver to leave at state s (we call such states the* terminal states*), it holds that* $P(s) = 0$.
2. $\forall (s,s') \in Q$, $y(s,s') - c(s,s') \leq P(s) - P(s')$.
3. $\forall (s,s') \in Q : F(s,s') > 0$, $y(s,s') - c(s,s') = P(s) - P(s') \geq 0$.
4. $\sum_{s \in S} P(s)(\deg_i(s) - \deg_o(s)) = \sum_{(s,s') \in Q} F(s,s')(p(s,s') - c(s,s'))$*, where* $\deg_i(s)$ *and* $\deg_o(s)$ *are the number of drivers to enter and leave the platform at the state s respectively.*

Leveraging the power of Lemma 1, we are able to prove the following theorem stating that a fair reward re-allocation scheme always exists in all non-degenerating scenarios (i.e., the total revenue is non-negative and at least one driver starts from a non-terminal state).

Theorem 3. *Let* $S_\# \subseteq S$ *denote the set of terminal states. If there exist* $s_1 \in S \setminus S_\#$ *and* $s_2 \in S$ *such that* $F(s_1, s_2) > 0$ *and* $\mathcal{I} \geq \sum_{(s,s') \in Q} F(s,s') \cdot c(s,s')$ *(recall* $\mathcal{I}$ *is the total income collected from the riders), then there exists a fair reward allocation plan.*

Proof. We define a directed graph G' on vertex set $V(G') = (S - S_\#) \cup \{t\}$, in which all states in $S_\#$ are contracted in a single vertex t. For each order from $s \notin S_\#$ to s' we add a directed edge (s,s') with length 1 if $s' \notin S_\#$, or (s,t) with length 1 if $s' \in S_\#$, and for each possible cruise arc from s to s' we add an edge with length 0.

As all arcs advance in time, the graph is a directed acyclic graph (DAG). Therefore, we can define $\tilde{P}(s)$ as the maximum distance of all paths from s to t, or 0 if $s \in S_\#$. Then we let $R_* = \{(s,s') \in Q : f(s,s') > 0\}$, define $\mu(s,s') = \tilde{P}(s) - \tilde{P}(s')$, and then we allocate the revenue proportional to μ, i.e. let

$$P(s) = \tilde{P}(s) \cdot \frac{\mathcal{I} - \sum_{(s,s') \in Q} F(s,s')c(s,s')}{\sum_{(s,s') \in Q} F(s,s')\mu(s,s')}. \qquad (1)$$

Because of the assumption that $\mathcal{I} \geq \sum_{(s,s') \in Q} F(s,s')c(s,s')$, we are ensured that $r(s,s') - c(s,s')$ is proportional to $\mu(s,s')$ with a non-negative ratio. We can see all constraints are satisfied. ☐

When the fair re-allocation scheme is not unique, we solve the quadratic program in Figure 1 to find the scheme to minimize the total squared distortion between the price paid by the rider and the reward allocated to the driver among all trips. In this way, we try the best to let the reward re-allocation reasonably reflects the real income generated by driving through each arc. It is straightforward to see that the constraints (3,4,5,6,7) in the quadratic program implement the conditions stated in Lemma 1.

5 The Stochastic-Demand Setting

In the previous sections, we studied the optimal car dispatching and reward allocation task assuming the access to the full list R of latent orders, which is not realistic in practice. In this section, we assume that R is drawn from an *unknown* distribution $\{\mathcal{D}(s,s')\}$ and address the problem with techniques combining both learning and optimization. To achieve this goal, we study the optimal car dispatching and reward allocation task with the distribution $\{\mathcal{D}(s,s')\}$ *known*. We will refer to this task as the *stochastic-demand setting*.

Our algorithm for the stochastic-demand setting is a natural extension of that for the deterministic setting presented in the previous sections. Below we describe the adaptation

$$
\begin{aligned}
&\text{Minimize} \sum_{F(s,s')>0} F(s,s')(p(s,s')-y(s,s'))^2 \\
&\text{Subject to } P(s) \geq 0,\ \forall s \in S && (2)\\
&\quad y(s,s') = P(s) - P(s') + c(s,s'),\ \forall F(s,s')>0 && (3)\\
&\quad y(s,s') \leq P(s) - P(s') + c(s,s'),\ \forall (s,s') \in Q && (4)\\
&\quad P(s) = 0,\ \forall s \in S_{\#} && (5)\\
&\quad y(s,s') \geq c(s,s'),\ \forall F(s,s')>0 && (6)\\
&\quad \sum_{s\in S} P(s)(\deg_i(s) - \deg_o(s)) \\
&\qquad\qquad = \sum_{(s,s')\in Q} F(s,s')(p(s,s')-c(s,s')) && (7)
\end{aligned}
$$

Figure 1: Quad. Prog. with decision variables $\{P(s)\}_{s\in S}$

we make for each phase in the deterministic setting. We will also introduce a special parametric demand distribution (Gaussian-Poisson distribution) for the learning algorithm in Appendix D.1.

Phase 1: Revenue Optimization. For each arc (s, s'), we denote $x_{s,s'}$ as the random variable for the number of latent orders, $\{v_t\}_{t\in[x_{s,s'}]}$ as the random variables for the valuations, and denote $\mathcal{D}(s, s')$ as the distribution of $(x_{s,s'}, \{v_t\}_{t\in[x_{s,s'}]})$, with the assumption that each v_t are *i.i.d.* variables.

For each arc (s, s'), if we fix the price to be p and plan to dispatch n drivers to the arc, the number of the fulfilled latent orders will be the smaller value of n and the number of orders of valuations at least p. Therefore, given $\mathcal{D}(s, s')$, we may compute the following quantities:

- The probability mass function $\mathcal{P}(i; s, s', p) : \mathbb{N} \to \mathbb{R}$ for the number of qualified orders (orders with valuation at least p): $\mathcal{P}(i; s, s', p) = \sum_{j=0}^{\infty} b(i, j; \Pr[v_t \geq p]) \Pr[x_{s,s'} = j]$, where $b(k, n; P) = \binom{n}{k} P^k (1-P)^{n-k}$ computes the binomial distribution.
- Let $\tilde{u}(n; s, s', p)$ be the number of the fulfilled latent orders; its expectation: $\mathbb{E}[\tilde{u}(n; s, s', p)] = \sum_{i=0}^{\infty} \mathcal{P}(i; s, s', p) \min\{i, n\}$.
- The expected revenue on (s, s') at price p and n drivers: $\mathcal{R}(n, p; s, s') = p \cdot \mathbb{E}[\tilde{u}(n; s, s', p)] - c(s, s') \cdot n$.

The following definition states the optimization problem we have to solve in order to maximize the revenue in car dispatching in the stochastic-demand setting.

Definition 6. *Given $\mathcal{D}(s, s')$ for all arcs (s, s'), the* Stochastic Maximum Revenue Car Dispatching *problem is to find the optimal solution to the NLWC problem on the directed graph (V_0, E_0), where (V_0, E_0) is constructed in a similar way as described above Proposition 1, and the only difference is that for the edges corresponding to driving with a rider, we set the corresponding reward function $r(n; e^{(\mathrm{w})}_{s,s'}) = \max_{p\in\mathbb{R}^{\geq 0}}\{\mathcal{R}(n, p; s, s')\}$.*

In Definition 6, $r(n, e^{(w)}_{s,s'})$ is re-defined so as to equal the maximum possible (over all candidate prices) expected revenue generated by dispatching n drivers to the arc (s, s'). Therefore, the optimal solution to the stochastic maximum revenue car dispatching problem is the maximum possible expected revenue achieved by any dispatching plan.

Note that in Definition 6, the only quantity that specifically depends on the form of the demand distribution is the nonlinear reward function on the edges $e^{(\mathrm{w})}_{s,s'}$.

Phase 2: Fair Re-allocation. After solving the NLWC problem on (V_0, E_0), we obtain the number of drivers to dispatch and the price for each arc (s, s'). With this information, we may invoke the quadratic program in Figure 1 to find out the potential-based reward re-allocation scheme for the drivers. We are able to show the following the fairness guarantees in the stochastic-demand setting, while the detailed proof is omitted since it is almost the same as the proof in Phase 2 of the deterministic setting.

Theorem 4. *In the stochastic-demand setting, the potential-based reward re-allocation scheme obtained by the QP in Figure 1 satisfies the fairness conditions in Definition 3, except for that the budget-balance condition is changed to the following expectation version.*

- Expected-*budget-balance.* *The expected income collected from the riders should equal to the amount to be distributed to the drivers.*[3] *Formally, it is required that $\sum_{(s,s')\in Q} y(s, s') \cdot F(s, s') = \mathbb{E}[\mathcal{I}]$, where $\mathcal{I}$ is the collected income.*

Online Learning. We use a Thompson sampling-based algorithm to learn the demand distributions from riders' responses to given prices. The details are deferred to Appendix E.

6 Experimental Evaluation

Due to space constraints, we defer many of the experiments to Appendix G. For example, we empirically verify the regularity of Gaussian-Poisson distributions in Appendix G.1, and evaluate the online learning algorithm in Appendix G.2; we also show an illustrative example of our fair re-allocation algorithm on the real-world dataset in Appendix G.3.

Experiments are run on an Intel i7-8750H, 24GB RAM computer with MATLAB 2021b.

6.1 Model Setting

We now evaluate our algorithm by simulated experiments on the DiDi Chuxing public dataset [Didi Chuxing, 2021] collected from the real-world ridesharing in Chengdu, China. For one day, we extract all orders and driver initial positions and discretize the locations into $10 \times 10 = 100$ squares with dimension $2\text{km} \times 2\text{km}$, and divide the time interval between 8am and 1pm in a day into 20 slots, each of which spans 15 minutes. Therefore, there are 2000 spatio-temporal states in a day, and we use the reward column in the dataset as the rider's

[3] Similarly, here we also omit the amount that the platform would like to keep for profit.

valuation for the trip. Finally, we assume the latent orders follow the Gaussian-Poisson distribution (Appendix D.1), and collect the data for 30 days and fit the numbers and valuations of orders in any arc into the Gaussian-Poisson distribution, as the true model parameters.

Robustness. To evaluate the generalization ability of our algorithm, we modify the following two key parameters in experiments: the number of drivers and the standard deviations of the riders' valuations. Here we report the experimental results showing that our algorithms still perform well under these different experimental environments.

In Table 1, we modify the number of drivers. In the 50% drivers setting we remove each driver with 50% independent probability and in the 200% drivers setting we duplicate every driver. In Table 2, we modify the standard deviations of the riders' valuations by 0.5 and 1.5 times respectively.

6.2 Revenue Evaluation

Given the true model parameters, we invoke the algorithms described in Section 5 to find out the offline (model parameters known) optimal revenue of the Stochastic Maximum Revenue Car Dispatching problem. We refer to this value as the *two-phase value* (2P). For comparison, we introduce the baseline *distance-based fix-price algorithm* (FP) where the price for each arc is proportional to the distance of the trip with a globally fixed (but tuned) ratio, and the dispatching is done via the same network-flow-based planning algorithm.

6.3 Fairness Evaluation

To evaluate the fairness, we define the $A(s)$ as the *average net income* of all drivers initially at state s. For a driver $q \in Q$, we denote s_q as the initial state of q and u_q as the total net income of q. Then, we define the absolute unfairness $\Xi = \sqrt{\frac{\sum_{q\in Q}(u_q - A(s_q))^2}{|Q|}}$ and relative unfairness $\xi = \Xi / \frac{\sum_{q\in Q} u_q}{|Q|}$, which can be interpreted as the absolute and relative fluctuation of drivers' net incomes from given initial states. We have proven that our two-phased algorithm guarantees *zero* unfairness, and evaluate the unfairness of baseline pricing algorithms. To show the contribution of re-allocation, we refer to the result of only Phase 1 as P1.

6.4 Results

We report the revenue (Rev) and relative unfairness (Unf) of different settings in following tables.

We see that our algorithm achieves higher revenue than the fixed-price algorithm, and our re-allocation phase eliminates the unfairness that would typically range from 10% to 25% of drivers' incomes, which increases with numbers of drivers.

Intuitively, a large number of drivers would tend to result in increased unfairness as they fulfill a large portion of latent orders with a wider spread of profits (analysis in Appendix G.4). Therefore, the re-allocation phase becomes essential for satisfaction of drivers especially in this scenario.

7 Computational Complexity Analysis

Let n, m, a be the number of states, latent orders and admissible arcs, respectively. Our Phase 1 essentially solves a linear program of size $O(m + a)$, which runs in $\tilde{O}((m + a)^{2.373})$ time [Cohen *et al.*, 2019]. Our Phase 2 solves a quadratic problem of $O(n)$ variables and $O(a)$ input size, which can be transformed into a semidefinite program that runs in $\tilde{O}(\sqrt{n}(an^2 + a^{2.373} + n^{2.373}))$ time [Jiang *et al.*, 2020].

#drivers	6655		13411		26822	
	Rev	Unf	Rev	Unf	Rev	Unf
2P	**6.82**	**0.000**	**9.32**	**0.000**	**11.17**	**0.000**
P1	**6.82**	0.114	**9.32**	0.172	**11.17**	0.243
FP	5.54	0.108	7.56	0.168	9.02	0.244

Table 1: Rev($\times 10^4$)/Unf with different numbers of drivers .

stddev	0.5σ		1.0σ		1.5σ	
	Rev	Unf	Rev	Unf	Rev	Unf
2P	**10.36**	**0.000**	**9.32**	**0.000**	**8.61**	**0.000**
P1	**10.36**	0.167	**9.32**	0.172	**8.61**	0.178
FP	7.90	0.162	7.56	0.168	7.25	0.172

Table 2: Rev($\times 10^4$)/Unf with modified standard deviations .

8 Conclusion

In this paper, we present an algorithmic framework for car dispatching and pricing with both revenue and fairness guarantees. Empirical evaluation shows that our method performs better than the baseline alternatives in the real-world dataset. For future directions, it is interesting to prove the regularity of edge demand functions in Gaussian-Poisson distribution and explore the regularity property of other distributions, and mathematically prove the guarantees of our Thompson Sampling algorithm (e.g., its convergence property and finite-sample regret bound).

Acknowledgments

Zishuo Zhao would appreciate Shiyuan Wang for discussions in relevant topics and Shuran Zheng for valuable comments in improvement of a preliminary version. Zishuo Zhao is supported in part by NSF CCF-2006526.

References

[Al *et al.*, 2005] Maiwenn J. Al, Talitha L. Feenstra, and Ben A. van Hout. Optimal allocation of resources over health care programmes: dealing with decreasing marginal utility and uncertainty. *Health Economics*, 14(7):655–667, 2005.

[Bei and Zhang, 2018] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[Bimpikis *et al.*, 2019] K. Bimpikis, O. Candogan, and D. Saban. Spatial pricing in ride-sharing networks. *Operations Research*, 67, 2019.

[Castillo *et al.*, 2017] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *ACM Conference on Economics and Computation*, pages 241–242, 2017.

[Chawla *et al.*, 2007] Shuchi Chawla, Jason D Hartline, and Robert Kleinberg. Algorithmic pricing via virtual valuations. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 243–251, 2007.

[Cohen *et al.*, 2019] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, page 938–942, New York, NY, USA, 2019. Association for Computing Machinery.

[Didi Chuxing, 2021] Didi Chuxing. The GAIA Public Dataset. https://outreach.didichuxing.com/research/opendata/en/, 2021. Accessed: 2021-08-31.

[Glazer and Rubinstein, 1996] Jacob Glazer and Ariel Rubinstein. An extensive game as a guide for solving a normal game. *journal of economic theory*, 70(1):32–42, 1996.

[Hrnčíř *et al.*, 2015] Jan Hrnčíř, Michael Rovatsos, and Michal Jakob. Ridesharing on timetabled transport services: A multiagent planning approach. *Journal of Intelligent Transportation Systems*, 19(1):89–105, 2015.

[Jiang *et al.*, 2020] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–918. IEEE, 2020.

[Li *et al.*, 2019] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. WWW '19, 2019.

[Ma *et al.*, 2020] Hongyao Ma, Fei Fang, and David C. Parkes. Spatio-temporal pricing for ridesharing platforms. *SIGecom Exch.*, 18(2):53–57, December 2020.

[Qin *et al.*, 2020] T. Qin, X. Tang, Y. Jiao, F. Zhang, and J. Ye. Ride-hailing order dispatching at didi via reinforcement learning. *Interface*, 50(5):272–286, 2020.

[Riquelme *et al.*, 2015] Carlos Riquelme, Siddhartha Banerjee, and Ramesh Johari. Pricing in ride-sharing platforms: A queueing-theoretic approach. In *ACM Conference on Economics and Computation*, page 639, 2015.

[Tardos, 1985] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[Wang and Zhang, 2011] Jian Wang and Yi Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1003–1012, 2011.

[Wang *et al.*, 2018] Zhaodong Wang, Zhiwei Qin, Xiaocheng Tang, Jieping Ye, and Hongtu Zhu. Deep reinforcement learning with knowledge transfer for online rides order dispatching. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 617–626. IEEE, 2018.

[Yan *et al.*, 2018] Chiwei Yan, Helin Zhu, Nikita Korolko, and Dawn Woodard. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics, Forthcoming*, 2018.

[Zhao *et al.*, 2015] Tongtiegang Zhao, Jianshi Zhao, Pan Liu, and Xiaohui Lei. Evaluating the marginal utility principle for long-term hydropower scheduling. *Energy Conversion and Management*, 106:213–223, 2015.