Hi everyone, I am Zishuo Zhao, a first-year PhD student in UIUC, and it’s my honor to present my first work in INFORMS. Now I’m presenting my recent research: Dynamic Car Dispatching and Pricing: Revenue and Fairness for Ridesharing Platforms.

Ridesharing is a modern research topic with wide concerns, especially in the era of Covid pandemic when we don’t like taking public transport. Many researchers have designed algorithms to optimize scheduling efficiency to match drivers and riders, but there are some shortcomings that limit their application. Many traditional methods only consider current orders so they can be myopic. Some only dispatch drivers in a centralized way but don’t consider the incentive of individual drivers. Some models only work for simple cases, and some use end-to-end machine learning methods without theoretical guarantees.

In our work, we develop a dynamic method that has theoretical guarantees in both revenue and fairness with a three-fold contribution. We first propose a generalized network flow model to optimize the total revenue, and use a two-phased pricing mechanism that decouples prices on drivers’ and riders’ sides and guarantee fairness on both sides. Then we use online learning tools for exploration and exploitation in real world.

In our non-linear network model, each arc corresponds to a spatiotemporal path with a non-linear reward function. If an edge has a cost of 2 and three orders with valuations 10, 9, 8, then if we dispatch 1 driver, the price should be set to 10 and the revenue is 8. If we dispatch 2 or 3 drivers, the price should be set to 9 and 8, the revenue should be 14 and 18. If we dispatch more than 3, additional drivers would still incur the cost.

Then, to reduce the non-linear problem to a canonical max weighted flow, we can decompose a non-linear edge into several linear ones, each with capacity one and weight as the marginal reward, beside a cruise edge with infinite capacity and its weight as the negative cost.

To make the decomposition accurate to resemble the actual reward function, the reward function should be regular, which means the marginal reward is non-increasing. With the regularity condition, the optimal plan can be solved with the max weighted flow algorithm, because the network flow problem is totally unimodular and we can get an integer optimal solution. Without the regularity condition, however we have proven that the problem is NP-hard as it can solve Set Cover. Fortunately, in our experiments on real-world data, the regularity holds.

We can notice that the maximum-revenue car dispatching problem without generality condition is a subset of non-convex network flow problems. The non-convex-network flow problem is NP-hard, and we have proven that even for this special subset, the problem is still NP-hard as we constructed a reduction of it from Set Cover.

To derive the reduction, we should firstly construct an arc with riders of certain valuations, and make the edge reward function an indicator function of whether x is at least k plus a constant. As the reward for zero driver is zero but for one driver is definitely not zero, for each arc we add a driver initially at its starting point, and set the domain from one to infinity. Then, we can construct several such edges in parallel and form a virtual arc, with reward function as the minimum of x-1 and 0, plus a constant. In this way we can reduce Set Cover to this problem. When a set covers x items, it gets a reward of x-1 if x is at least 1. So, if m sets cover N items, the optimal total reward is N-m, then if we can compute the max-revenue car dispatching, we will be able to solve Set Cover. Therefore, such polynomial algorithm doesn’t exist unless P=NP.

Now we have collected the maximum revenue, and consider how to allocate it to drivers wisely so that they won’t complain.

Let’s look at an example, ~~~

To ensure this, we should satisfy four constraints: budget balance, incentive compatibility, subgame-perfectness and envy-freeness, which are common in the scope of fair division.

Here we are inspired by physics, in which when we move an object in a field, the change of energy doesn’t depend on the trajectory. In this way, we can set potentials to each node as maximum net income to get from this node. In fact, we prove an important lemma that a reward allocation is fair if and only if there exists a corresponding potential function, in which the potential of terminal states should be zero, the net income of all arcs should be at most the difference of potentials, and along all assigned paths the equality should hold with non-negative net income.

We also prove a theorem that such fair allocation plan always exists as long as the case is not degenerate, in which at least one driver starts in a non-terminal state and the total revenue is non-negative. That means our algorithm always works in practical situations. If there exist multiple possible fair allocation, as all constraints are linear, we can compute a minimum squared distortion solution via quadratic programming.

Now we consider learning in practical scenarios. In practice, we don’t actually know the number and valuations of riders, so we model them on each arc as a distribution. Given the distribution, we can still compute the edge reward function, and thus compute the optimal dispatching and pricing. While the number of riders can be observed in the backend of the APP, riders will not disclose their valuation, so we need to perform Thompson sampling to discover the distribution of valuations.

In the setting of Thompson sampling, at the beginning of each day we draw these parameters from a prior distribution, and compute the optimal plan based on the parameters, including dispatching and pricing. During the day we observe the riders’ responses to the offered prices, and compute the posterior distribution of the parameters based on their decisions. Then, we set the updated distribution of parameters as the prior distributions of the next day, and update every day in this way.

We perform simulated experiments on DiDi datasets to evaluate the performance of our methods. In the offline setting, our algorithm gets 23% more revenue than the fixed price method, and in the online setting, the regret is significantly lower than the explore then exploit scheme. For the reward re-allocation, we demonstrate on 10 drivers starting in a certain position C. Without the re-allocation the driver assigned the path C-A-D gets far more net income than the driver assigned with C-D-D, but after that they get the same, so they will feel fair.

In conclusion, we developed a theoretically guaranteed mechanism for ridesharing with both optimized revenue and ensured fairness, and we also proposed a way to learn about the parameters in real world with incomplete information.

In the early stage of our research, I’m grateful to Shiyuan Wang for discussion on several topics on mechanism design, and wish you success in your research too.

Thank you everyone for listening. Are there any questions?