Rideshare Auctioning

Masaoud Haidar, Janet Liu

December 6, 2021

1 Introduction

For our project, we took a closer look at a very common problem: ride sharing. We first analyze the status quo of ride sharing with apps like Uber or Lyft, taking a look at their design and the resultant issues with the existing model. We then discuss existing research that proposes solutions to these issues, and also hypothesize our own solution. Finally, we created a simulation that implements our solution. In this simulation, we also model different kinds of driver agents with different ride valuation calculations. This culminates in an analysis of our simulation, the different agents and valuation calculations, and how this effects the customers and ride sharing app.

2 Status Quo

2.1 Current Model

Two of the most popular ride-sharing apps, Uber and Lyft, advertise two main bonuses to drivers. They advertise letting you "be your own boss" [Lyft, 2021], earning money reliably, and having "flexible driving opportunities" [Uber Drive..., 2021]. To its consumers, these apps tout the simplicity of ride-sharing: you just need to "request a ride, hop in, and go" [Uber uber..., 2021].

To achieve driver flexibility, the interface between ride sharing apps and drivers is like a repeated bargaining game. Ride sharing apps determine the price of the ride based on distance and based on supply and demand and it asks the customer that amount. Then, it determines a price to pay drivers, which is usually set as a constant fraction of the customer's payment, and drivers can decide whether or not they want to accept the ride or decline and wait for a potential better ride offer. Drivers get to choose what kind of ride they want to do and to where, and are able to work any hours they want. Rejecting a ride hasn't always been as easy as that, but ride sharing apps have been moving towards more flexible ride offers.

A main way that apps are able to meet customer demand is through surge pricing. This dynamic pricing method increases driver and customer payments when it is particularly busy, theoretically increasing efficiency and reliability [Rheingans-Yoo et al].

2.2 Issues with the Current Model

Drivers have learned to game the current surge model pricing. This often leads to chasing the surge. Drivers can spatially chase the surge, choosing to drive to a location they anticipate will have more trips or higher paying trips. They can also temporally chase the surge, denying rides when at the temporal boundary of anticipated increased requests. Drivers also learn to strategically deny trips that take them away from profitable areas [Rheingans-Yoo et al]. The latter can lead to customers having extended wait times, especially if they come from or are headed to remote locations.

The different strategies that drivers can learn also affect their hourly wages. On average, drivers with more driving experience (2,500 lifetime trips) earn 14% more than those who have driven 100 or fewer trips. This helps contribute to a gender-wage gap that is commonly found in the gig industry [Cook et al].

Theoretically, when thinking about the current model as a repeated bargaining game, we can immediately see some problems come up. The bargaining game has some bad Nash Equilibria, such as always offering the drivers very low wages and the drivers being forced to accept them because of not having other choices.

In addition, there are some missed opportunities. Say a costumer will be charged $20 for a ride, and the ride-sharing app decides to offer the driver $10, but the driver wouldn't accept less than $12. In the current model, the driver will reject and costumer will be left unsatisfied, where if the app had a way to know that it should offer exactly 12 to the driver, it would have been able to still make profit and leave the driver and costumer better off.

2.3 Related Research: Pricing Mechanisms

Most existing research on this problem aimed to solve it by designing better algorithms for deciding payments to drivers. The idea is that if the algorithm can figure what the customer's private value is for a ride, then we can offer them that value and guarantee that they will accept it.

One suggested solution by Ma et al. is to create prices that are smooth over time and space. When we have a sudden surge price at a specific time and a specific area, then drivers right outside the boundaries of this area and time will reject their rides and chase the surge. However, with smooth pricing, there isn't a clear boundary of surge, and so, riders aren't able to game the model as much.

Another suggested solution by Rheingans-Yoo et al. is to use information elicitation from drivers and then use that to better calculate the payments.

While these solutions improve on the current model, they both keep the structure of a repeated bargaining game which limits the ability to determine payments based on the market. Our project aims to directly solve the problem of payment determination.

3 Our Proposed Solution

3.1 Overview

Our solution to this ride sharing problem is an auction. An auction was discussed in class as a simple way to garner information about everyone's individual unique values for items and decide who gets which items. Auctions can be designed to incentivize truthful bidding, increase revenue, have strategy-proofness, etc. Thus, in our problem, where every driver may have a different individual value for their own time or different values for different locations, a combinatorial auction can be a good way to figure out the appropriate pricing.

In every round of our auction, all rides requested from the same area are considered as items in an auction, and all drivers from that area can bid on them. In our simulation, a "round" of an auction was performed every 10 minutes. A shorter round time allows for a shorter wait time for customers but a longer time increases customers and drivers involved.

The ride-sharing app sets a maximum price at an amount a little below what the customer is paying for the ride, to ensure that the app still makes a profit. The drivers can bid on different rides, but each driver is only allowed to win one ride, because they can't take more than one ride at the same time. Once drivers bid for rides, the app assigns rides to drivers and decides what to pay the drivers.

One difference between our proposed ride auction solution and regular combinatorial auctions is that a driver's expected utility if they don't bid or don't win anything is not 0. This is because simply waiting a round in a popular area for better rides would have some non-zero expected utility. So, drivers should factor that into their bids.

3.2 Auction Type

An auction we discussed a lot in class was VCG. However, we looked in class at several issues with the VCG mechanism specifically in combinatorial auctions. Examples in the problem sets and chapter 11 demonstrated a few common pitfalls: low revenue, collusion by losers, and false-name manipulation.

In the real world, it's probably less likely that collusion by losers is a big problem, since that would mean that multiple disjointed drivers are interacting with each other, or that drivers are able to have the ability to communicate bid manipulations on a the short bidding time crunch.

Given the number of customers, drivers would probably also be given a randomized, limited size batch of customers, so different accounts (even if managed by the same person) would not always be able to bid on the same rides.

However, the biggest issue is low revenue. Apps like Uber and Lyft exist to turn a profit, not just to improve transportation. Without a high revenue incentive, there is no reason that these apps would exist or be sustainable.

Thus, we decided to use the first price auction. Although the first price auction does not incentive truthful bidding, it is theoretically efficient (when the Winner Determination Problem is efficiently

solved). It is also a simple auction to create with very simple payments. Rather than a second price or VCG auction where your payments are dependent on other people's bids and there is less transparency with a middle man app, the first price auction has you simply paying what you bid.

3.3 Case Study: Uber's Set-Your-Fares

In July 2020, Uber rolled out a new set-your-fares feature. This feature allowed drivers to set their own fares for a trip, with fare multipliers from .5x to 5x [Uber California Drivers, 2020]. Through increasing driver flexibility and autonomy, Uber hoped to rally driver support for Prop 22 [Hiltzik]. Prop 22, largely driven by ride sharing monoliths like Uber and Lyft, aimed to define ride-sharing/app delivery workers as independent contracters. This allowed companies to escape labor and wage policies for employees that would increase pay or force the companies to provide employee benefits [Ballotpedia].

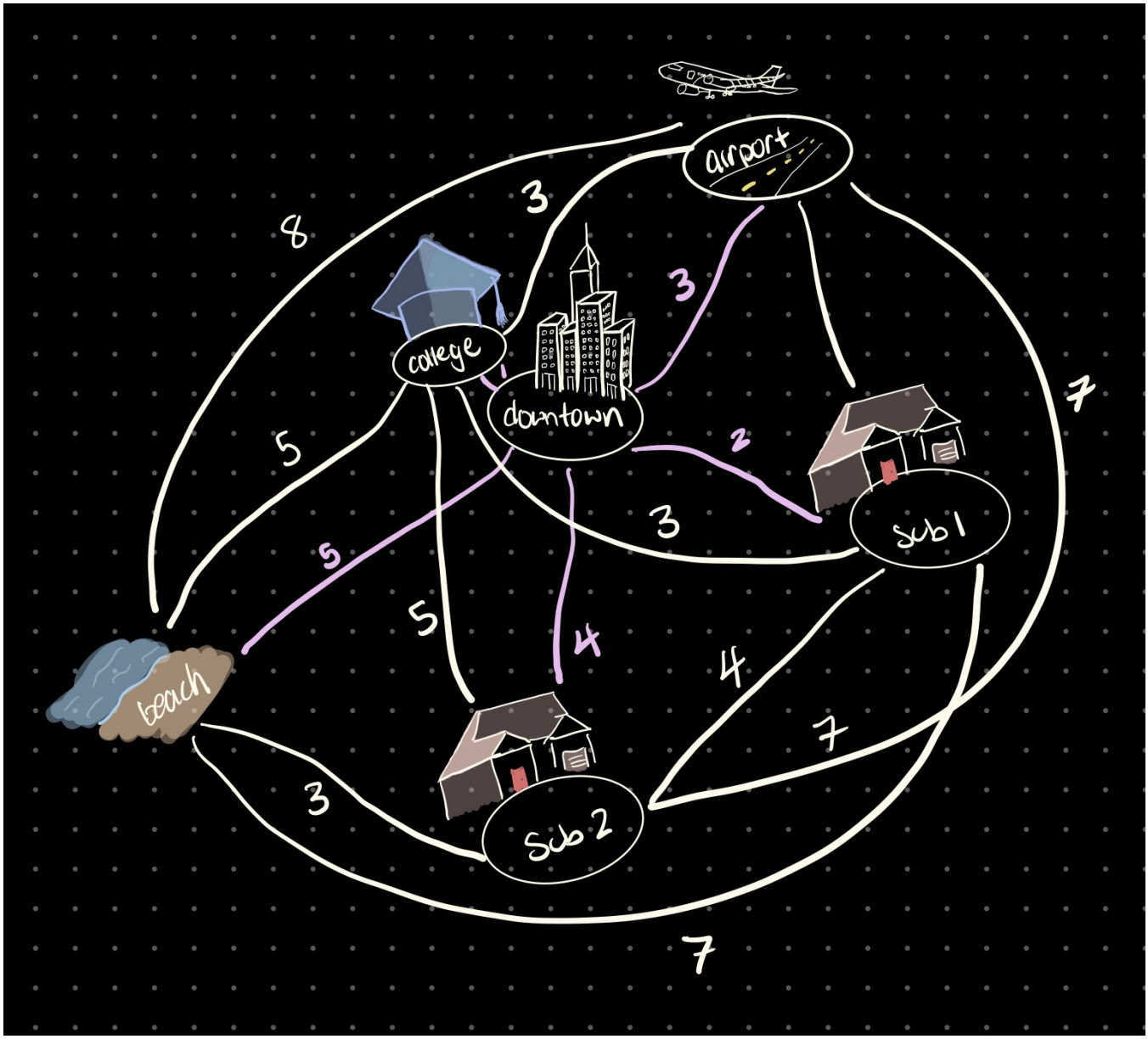
Though set-your-fares is not our proposed solution, it is a real world look into a more driver-driven price system. Less than a year after rolling out this feature, Uber took it back. They attributed this change to 117% increased rider cancellations, 56% increased customer wait times, and declining business [Uber Upcoming, 2021, Hiltzik].

Uber's previous example of set-your-fares was a failure for the company and its customer and driver experience. However, we do not anticipate our solution would have those same issues. With every driver participating in a location-based auction for the available rides, any eventual matched ride for a driver will be at least their self-reported value for it. Logical drivers would then only report a value they are willing to be paid. Thus, drivers will not be declining rides and wait times will not be affected. There will be wait time for customers as drivers participate in this auction, but this is a short, fixed time of only a few minutes.

In addition, the set-your-fares was displayed publicly to the customers, and drivers chose different set-your-fare multipliers. These multipliers made it clear to customers which drivers were asking for more (relative to everyone else), probably leading to more disgruntled customers leaving the app. With our model, the payment of the riders is constant and based only on distance and supply and demand. What changes is how much of that payment goes to the app and how much goes to the driver, which the app can practically negotiate with the driver by running an auction.

4 Simulation Implementation

4.1 Map



An image of our map is included above. Between all of the locations are edges with values. The values correspond to how many time steps it takes to travel between locations. We chose to make distance and time steps interchangeable, which is nto super realistic: often times in the real world, the same distance will take different travel times depending on the location. Traveling .5 miles in downtown is probably a lot longer than traveling .5 miles in a suburb. However, to simplify our problem, we decided to overlook that aspect for now, and treat the distances linearly to time steps.

Here are some location considerations made with the map:

1. Suburb 1 is closer than suburb 2 to downtown/college. We modelled suburb 1 with the idea that some college students live there.
2. The beach and the airport are relatively far from everything else.
3. The college and downtown are very close to each other, and we anticipated students, families, or other people coming to downtown for things like restaurants, grocery stores, clubs, etc.

In addition to the distances, we created spawn matrices that contained the mean of the normal distribution that modelled customer spawn numbers. This matrix was dependent on time and day of the week (weekend vs. weekday).

Here are some considerations made with the spawn matrix:

1. Beach spawn was higher on the weekends and during afternoon/evening, but generally still quite low.
2. Suburb spawn was quite high during weekday rush hour (8-9a), and a generally consistent trickle during other normal waking hours. There was very minimal, if any, spawn during late hours.
3. Downtown spawn was pretty consistent. There were spikes during weekday rush hours, weekend lunchtime/dinnertime, and weekend nighttime.
4. Airport spawn was consistently low, and we had spawn during all hours of the days to account for red eye flights/late arrivals.
5. College spawn was also pretty consistent, with spikes during school hours, dinner times, and weekend night hours.

Lastly, we created a movement matrix. This matrix contained all the probabilities for destinations for each source. For simplicity, we made it time independent. Realistically, there should be probably be differences in movement probabilities depending on the day/time.

Here are some considerations made with the movement matrix:

1. Beach goers were split between heading to college and the suburbs, with a higher probability for the suburb closer to the beach.
2. Suburbs had a high probability of going to downtown, either for work or running errands/seeing friends. Suburb 1 also had a high probability of heading to college, for those college students who are living off campus.
3. Downtown destinations were split between the residential areas, and a little bit of the beach.
4. Airport destinations were residential areas.
5. College destinations were split across all areas - going home to the suburbs or through an airport, or to the town/beach for a hangout.

4.2 Customer

Customers were generated at each time step. The number of customers per location were randomly sampled from a normal distribution with the mean described in the spawn matrix. Then, for each customer, we generated the trip destination by sampling from a multinomial distribution of the various destinations/probabilities, as told by the movement matrix. Every customer had the same payment calculation (see Section 4.4).

4.3 Driver: Bidder

At every stage, the driver receives info about all rides that start in their area, and can bid on some or all of these rides. The info about the ride includes the destination and the max payment possible, which equals what the customer is paying for the ride and serves as a reserve price.

In this section, we discuss the different strategies for the bidder. Because this is a first price auction (you pay what you bid), part of the strategy can be to bid a bit lower (ask for a bit higher payment) than the person's true value. However, for simplicity, we focus here on how bidders determine their own values, and we make them all bid their true values. We think our results here wouldn't change a lot if everyone follows a strategic bidding because it will preserve the efficiency of the allocation and just increase everyone's payments by a small fraction.

When modeling the drivers, every driver will have a private personal rate that they want to get paid per round of time worked. Part of this rate represents the cost of the ride, such as the gas and average car rental and maintenance. Another part of this rate will represent the minimum wage that this driver is willing to accept per round of time worked. We set this rate for every driver uniformly at random in the range $\$3 - \$5$ per round of time, or equivalently $\$30 - \$50$ per hour.

If a driver doesn't receive any rides to bid on during a round, most likely because of being stuck on a distant area with no riders, then they drive without a rider to the middle of the city. We did this to avoid all drivers being stuck for a lot of time on unpopular areas. We count the time it takes to move back into the city as part of the time worked, so, this lowers the drivers' rate if they keep getting stuck.

Now when determining the ride's value for the driver, we designed three main driver profiles:

- **Charge-dependent Driver:** This driver ignores their own rate and they bid half of the payment that the customer will be charges. This constant fraction somewhat represents the current state of price determination in apps like Uber of Lyft.
- **Cost-dependent Driver:** This driver calculates their value of the ride as being the duration of the ride multiplied by their personal rate:

$$Cost = rate \times t$$

  This means that this driver bids just enough to make their desired hourly rate.

- **Future-dependent Driver:** This driver is our most strategic driver and we experiment a lot with it. First, the driver calculates their *cost* as with the previous driver and takes their base bid as the mean between the cost and the maximum payment (the price being charged from the customer):

$$base = \frac{cost + charge}{2}$$

Then, the driver calculates the expected extra utility gained at the destination. For that, they calculate the expected distance leaving from that destination, which depends on the map and equals:

$$E(Distance_{dest}) = \sum_{e=(dest,v)} Distance_e \times Pr_e$$

Which means averaging the distances of edges leaving that node, weighted by the probability of choosing each edge. Then, using that distance and the expected population at the node they can calculate the expected cost and charge of a ride leaving that node, and then, the expected extra utility will be:

$$E(U_{dest}) = \frac{E(Cost_{dest} + Charge_{dest})}{2} - E(Cost_{dest})$$

Which is the difference between the expected base bid they will have at that destination and their cost at that destination. Now how is this expected utility related to the current bid? Well, if the current base bid is, say, \$40, but it takes you to a popular area where you expect to make an extra utility of \$10. Then, you might choose to bid anywhere between \$40 to $40 - 10 = \$30$ instead of the base bid of \$40. And on the other hand, if this ride will take you to a bad area where you won't make a lot of money, then you might ask for more money than the base bid. So, the driver calculates their bid for every ride as:

$$bid = base - w \times E(U_{dest})$$

For some $w \in [0, 1]$. w represent how much more the driver cares about the future gains compared to current gains. We gave it a default value of 0.3 but we experiment with that later to compare how different planners achieve.

4.4 App: Auction Mechanism

The ride sharing app's main function is to represent the auction mechanism, but it also offers a middle man interface to connect customers and drivers.

At every step, the app simulation does the following steps:

- For every area, collect how many customers are in the area and how much to charge them. The charge for every customer is:

$$Charge = 10 \times Distance + \frac{Population}{3}$$

  The left term means that, for example, if a ride takes 2 time rounds (20 minutes) then the customer will be charges \$20. The left term represents the surge prices. If at the beginning of the round there are 15 people in an area, then this term adds \$5 to the ride's price. This price is only calculated when the customer requests the ride, and doesn't increase if later more people request rides from this area.

- After that, the app collects the status of every driver, where they are now and whether they are free to take a ride. Then it sends the info about the customers to the drivers and collects their bids.

- Finally, the mechanism determines the winner of every ride, assigns them to the ride and pays them their bid, and the difference between the charge and payment is added to the app's revenue.

4.4.1 Winner Determination Problem

One of the main problems we faced when designing the simulating is the optimal winner determination problem. We know from class that this problem is NP-hard for combinatorial auctions. We note though that for our problem, all bids are singleton XOR bids, which we know how to turn into an OR bid with two items in every clause, which can be solved for polynomial time.

For the purpose of the simulation, we didn't implement the optimal polynomial solution as our main focus wasn't on the winner determination problem. Instead, we implemented a form of randomized serial matching. When the app has all the rides and bids from a location, it shuffles the rides, and then for every ride in order, it picks the cheapest active bid on it (the free driver that agrees to get paid the least) and assigns them the ride. We expect that the random serial matching will force drivers to still make all their bids, as the randomization makes it much harder to strategies.

This winner determination solution was another reason why the first price auction was chosen for this simulation (paying the driver exactly what they bid). Here, it isn't clear what the second price is. The next driver on the list might already be assigned a ride in this turn, and picking the next active driver poses a problem with transparency, as this driver might later be assigned a ride as the algorithm is being run. So, at the end of the turn, there is no simple way for the app to show how the payment was found.

5 Results

We did two main experiments to compare drivers with different value determination methods. In the first experiment, we compare the charge-dependent, the cost-dependent, and the future-dependent

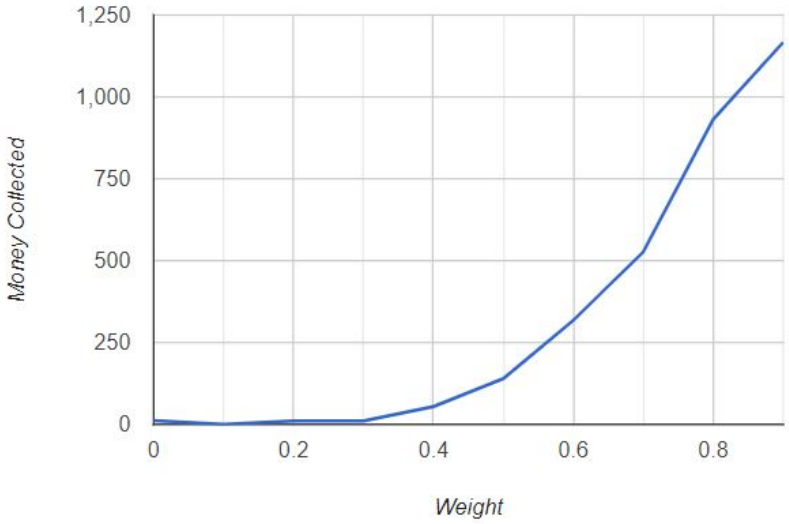
drivers (with the default weight value $w = 0.3$). In every simulation, we generate 500 total drivers (around 166 driver of each type) and find the average money, work, and rate achieved by the drivers of each type. Then, we repeat this full simulation 50 times are record the results from each one of these metrics. The final summary results are shown in the table below:

Driver List			
Driver Type	Total Money Mean (STDDEV)	Total Time Worked Mean (STDDEV)	Rate Mean (STDDEV)
Charge-Dependent	10.95 (6.28)	5.07 (1.66)	1.99 (0.57)
Cost-Dependent	752.22 (29.48)	269.87 (9.89)	2.79 (0.04)
Future-Dependent	180.77 (14.95)	48.89 (3.43)	3.7 (0.07)

We can see that the charge dependent driver makes very little money. This is mainly because they are overcharging compared to the other drivers, and so, they rarely win any rides. As for the cost and future dependent drivers, we see that the cost-dependent driver collects more total money, but the future-dependent driver has a higher rate of money collected over hours worked.

In the second experiment, we ran 500 future-dependent drivers with different values of w ranging from 0 to 1, 50 drivers at each weight $0, 0.1, ..., 0.9$. We repeated the simulation 50 times and recorded again the mean money, time worked, and rate, for each group of drivers with different weight. A graph of the money collected is shown below.

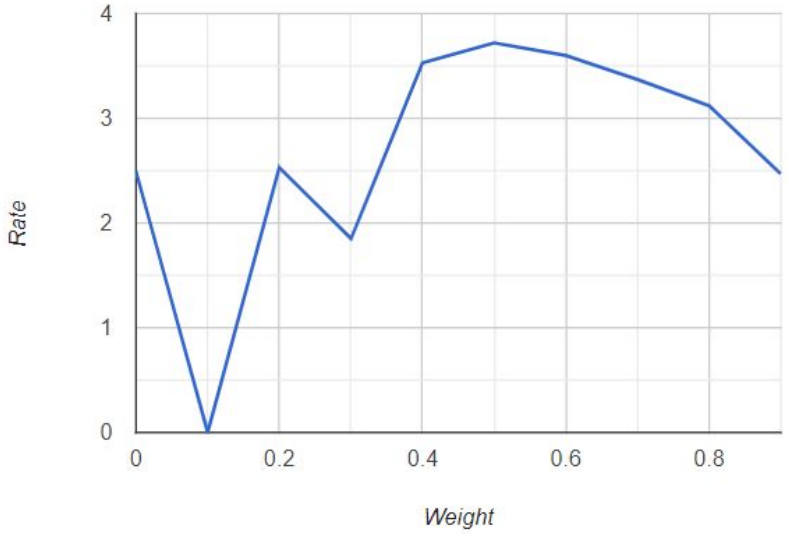
Figure 1: Graph of money collected by different future-dependent drivers



We can see that with increased weight, the drivers collect more money. We note that most expected utility in this map is positive, so, higher weight here simply means bidding less and winning more rides, and so, that explains this result. The graph below shows the mean pay rate for these different drivers.

For low weights, the bids are too high, and they rarely win rides, so, their rates aren't very interest-

Figure 2: Graph of pay rate (wage) by different future-dependent drivers



ing. However, for higher weights, we see that the optimal weight is somewhere around 0.5. This is where the driver is willing to give up half of their future expected utility. Weights higher than that give up too much utility, where they always focus on future rides and never make enough money in the present. Lower weights never win rides because they are asking to make all the money right away.

We think if this is run in real life, charge and cost-dependent drivers would make low rates and would rather turn into future-dependent drivers. In addition, within future-dependence, drivers with low weight will not win enough rides and eventually would rather switch to different weights, while higher weight drivers won't make enough wages and would rather switch to lower weight (higher prices) to make their desired hourly wage.

More generally, the auction will offer an environment where values of the rides are easier to be set by the market and where drivers will adjust their bidding strategies until the market reaches an equilibrium. This allows for a flexible price determination, which makes it less likely that drivers will completely reject some rides, which hopefully can lead to more satisfied customers.

6 Who did What

Janet: utility/map code, Sections 1, 2.1, 2.2, 3, 4.1, 4.2. Masaoud: simulation/experiment/bidding code, Sections 2.3, 4.3, 4.4, and 5.

References

1 Ballotpedia. California Proposition 22, App-Based Drivers as Contractors and Labor Policies Initiative (2020). URL https://ballotpedia.org/California_Proposition_22,_App-Based_Drivers_as_Contractors_and_Labor_Policies_Initiative_(2020). [Online; accessed 5-December-2021].

2 Cody Cook, Rebecca Diamond, Jonathan Hall, John List, Paul Oyer, et al. The gender earnings gap in the gig economy: Evidence from over a million rideshare drivers. Technical report, Stanford University, 2018.

3 Duncan Rheingans-Yoo, Scott Duke Kominers, Hongyao Ma, and David C. Parkes. Ridesharing with Driver Location Preferences. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI, 2019.

4 Hongyao Ma, Fei Fang, and David C. Parkes. Spatio-Temporal Pricing for Ridesharing Platforms. September 20, 2021.

5 Michael Hiltzik. Column: Uber reneges on the 'flexibility' it gave drivers to win their support for Proposition 22, 2021. URL https://www.latimes.com/business/story/2021-05-28/uber-flexibility-prop-22. [Online; accessed 5-December-2021].

6 Lyft. Become a Driver - Drive with Lyft. URL https://www.lyft.com/driver. [Online; accessed 5-December-2021].

7 Uber. California Drivers: Set your own fares when you drive with Uber. URL https://www.uber.com/blog/california/set-your-fares/. [Online; accessed 5-December-2021].

8 Uber. Drive with Uber: An Alternative to Traditional Driving Jobs. URL https://www.uber.com/us/en/drive/?city=boston. [Online; accessed 5-December-2021].

9 Uber. uber-sites. URL https://www.uber.com/us/en/ride/. [Online; accessed 5-December-2021].

10 Uber. Upcoming Changes to the Driver App. URL https://www.uber.com/blog/california/upcoming-changes-to-the-driver-app/. [Online; accessed 5-December-2021].