Data-Driven Methods for Balancing Fairness and Efficiency in Ride-Pooling

Naveen Raman
University of Maryland
nraman1@umd.edu

Sanket Shah
Harvard University
sanketshah@g.harvard.edu

John P. Dickerson
University of Maryland
john@cs.umd.edu

Abstract

Rideshare and ride-pooling platforms use matching algorithms to maximize profit when pairing riders and drivers. However, these platforms can induce unfairness either through an unequal income distribution or by biasing toward certain neighborhoods when determining which riders to service. We investigate two methods to reduce forms of inequality in ride-pooling platforms: by varying the matching policy and redistributing income. We construct new policies that optimize for a combination of profit and some form of inequality. We vary the number of drivers and riders to see how policies perform in different environments; we find that policies that minimize the variance of income also maximize riders served when the number of drivers is small, indicating that reducing some forms of inequality can also boost utility. We explore income redistribution as a method to combat income inequality by having drivers contribute a portion of income to a redistribution pool, where the amount contributed is dependent upon a risk factor. For certain values of the risk factor, we maintain near 0 income variance, while still incentivizing drivers to maximize income, thereby avoiding the tragedy of the commons.

1 Introduction

In recent years, ride-pooling services such as UberPool, Lyft-Line, and Didi Chuxing have exploded in popularity. Ride-pooling services have the benefit that riding costs are split amongst multiple riders, making it cheaper for riders, while increasing the profit for drivers. Coinciding with this is an increase in attention from researchers into optimizing how Rideshare companies operate.

Rideshare companies match drivers and riders by employing policies that optimize for an objective function, typically some function of profit. However, these policies have led to inequalities such as a gender pay gap, with female drivers earning 7% less than their male counterparts [9]. Many drivers have expressed a desire for a minimum wage, which is complicated by variation in driver pay even within the same time of day [23]. Currently, most rideshare matching policies offer no guarantees of fairness or minimum wage.

Prior research into rideshare and ride-pooling has been split into three primary fields. The first is to view rideshare as an online matching problem, which can build on prior work in economics and CS [19, 20]. Online matching has also been used to maximize not just immediate, but also future profits by employing dynamic programming [28, 24]. There is also past research into incorporating fairness into rideshare objectives. This includes different notions of fairness, such as driver side, rider side, and inter-group fairness [21, 25]. Another area of research is an investigation of the rideshare market, including surge pricing [3]. There has been game theory and economics based research that uses income redistribution and insurance as a method to mitigate risk and inequality [8].

We deal with the problem of fairness in ride-*pooling* services, which has not been dealt with prior and is more difficult than fairness in rideshare due to the increase in combinations of riders and drivers. We consider fairness on the rider side by considering discrimination between neighborhoods, which can be generalized to protection amongst protected groups. We pursue two methods of incorporating fairness: through new matching policies, and through income redistribution methods. Our contributions are the following

1. We investigate how inequality varies based on environment parameters such as the number of drivers and number of riders
2. We propose new policies that aim to maintain profitability while minimizing inequality on the driver and rider sides, and evaluate their performance on utility and fairness metrics. These policies optimize for certain measures of fairness, and in certain situations, manage to outperform other policies on utility metrics.
3. We create two income redistribution strategies by using two different definitions for the value of a driver. The first is the amount of profit a driver generates, and the second uses the Shapley value to determine how much profit a driver adds to each subset of drivers. We show that using income redistribution with certain levels of risk tolerance minimizes income inequality while incentivizing drivers to maximize income.

2 Problem Statement

Ridepool Matching Problem

Rideshare companies such as Uber and Lyft serve as intermediaries between riders and drivers. Each rider has a requested trip (request), which has a starting and ending point. Each driver has a current capacity (the number of passengers currently in their vehicle), a maximum capacity (the number of seats in their car), and a current location. After a certain time interval (for example, one minute), rideshare platforms centrally match all incoming ride requests to available drivers, subject to capacity and wait time constraints. Riders who cannot be matched in one batching instance can either be dropped from the system or be carried over to the next batching process, depending on the details of the system. For our formulation, riders who cannot be matched are dropped.

We can formally define the problem using the following tuple: $(G, U, R, O, D, \Delta, P)$.

1. $G(L, E)$ represents the graph upon which the drivers reside upon. This consists of a set of locations, L, and the set of travel times between locations, E. Four our problem, the travel time between two locations, i and j, is represented as $E_{i,j}$.
2. U is the set of rider requests. Each user, $u_i \in U$ is represented as a tuple $u_i = (s_i, e_i, t_i)$, which represents the starting and ending locations (both elements of L), and the time that the request originated.
3. R is the set of all drivers. Each $r_i \in R$ is defined by the tuple $r_i = (m_i, c_i, d_i, p_i, s_i)$, where m_i is the maximum capacity of driver i, and c_i is the current capacity (how many riders driver i is currently driving). d_i is the location of the driver (which is an element of L), p_i is the location of where driver i is going to, and s_i is the set of previously completed requests.
4. O is the objective function, which aims to balance utility and fairness.
5. D is the set of constraints on matching. These constraints include capacity constraints (which say that drivers with a car that seats 5 can't drive 7 people) and lateness constraints (drivers can't be more than say five minutes late to pick up a passenger)
6. $\Delta$is the epoch duration, which is how often requests are batched and matched. For all our experiments, Δ is 60 seconds.
7. P is the pricing or profit function, which maps a request to some real, positive number denoting the price charged for the request. In many scenarios, the utility in the objective function is the pricing function. Throughout this paper, we set $p(x) = x + 5$, where $p(x)$ is in dollars, and x is in minutes. We select this because it approximates both the upfront cost of rideshare companies and the cost per time aspect.

Simulating Requests

We evaluate our policies using real-world data from New York City taxi cabs using a simulator designed by Shah et al. [24]. Our value of $G(L, E)$ is based on the Yellow Cabs New York taxi data [7]. The nodes in the graph represent street intersections in Manhattan, while the edges represent travel time between intersections (in seconds). We fix $\Delta = 60$, and so simulate 1440 epochs each day.

Our dataset ranges from March 23rd to April 1st, and from April 4th to 8th. We define our training data to be data collected from March 23rd to April 1st, and our testing data to be from April 4th. Each epoch, t, has a set of user requests, U_t. Because all policies will be tested on April 4th, they all have the same sequence of user requests $U_1 \cdots U_{1440}$. Each epoch, some subset of U_t is assigned to the drivers.

We match riders and drivers based on an objective function. The details on how riders and drivers are matched, based on the objective function, can be found in Appendix A.

3 Definitions of Utility and Fairness

We define utility and fairness for both drivers and riders. We note that, as with all operationalizations of societally-relevant concepts such as fairness, the decision of *which* definition of fairness to use—if one is appropriate to use at all—is a morally-laden decision, and one that should be made with the explicit input of stakeholders. Our definitions are drawn in part from recent reports of (primarily rider-side) unfairness in fielded rideshare applications [see, eg., 11, 10, 5, 22]; still, we acknowledge that these need not be one-size-fits-all formalizations of a complicated concept, and view them rather as illustrative examples of a class of fairness definition that could be incorporated into modern automated matching algorithms used by rideshare platforms.

Drivers

Recent news has noted wage discrepancies among rideshare drivers [4]. As such, we define the utility on the driver side as being the total collection of driver profits. Formally, we define the profit for each iteration, for each driver, as $p_{i,t}$, and then denote the total profit for each driver as $p_i = \sum_{t=1}^{T} p_{i,t}$. The total profit is therefore $\sum_{i=1}^{N} p_i$.

We define the inequality of a set of driver profits, by using two different metrics: the variance of income, and the value of the 25^{th} percentile of income. Rideshare drivers have complained about inequality in pay amongst drivers, and also have complained about not earning a livable wage [13], and so develop metrics to measure these. We use standard deviation because it's a good measure for the spread of income, while the 25^{th} percentile income is a good measure of how much those on the lower end of wages earned. These metrics capture both the idea of inequality amongst drivers and how well drivers on the lower end of the spectrum are doing.

Riders

Inequality results from differential treatment amongst groups of people. For riders, their utility originates from whether or not their request gets serviced. As a result, we define rider utility as the total percent of rider requests that get serviced, $\frac{\sum_{i=1}^{1440} |l_i}{\sum_{i=1}^{1440} |b_i}$

Formally, we define this by clustering L using KMeans into 10 different regions, corresponding to 10 different neighborhoods (4). We denote the total requests for each of these regions as $b_1 \cdots b_{10}$, and the number of requests accepted as $l_1 \cdots l_{10}$.

The overall utility for all riders is formally defined as $\frac{\sum_{i=1}^{10} l_i}{\sum_{i=1}^{10} b_i}$, which is the total proportion of riders serviced. The inequality for riders is defined similarly to driver-side inequality through two metrics: the standard deviation in the distribution of rider success rates, by neighborhood, which we formally define as $\mathrm{Var}(\frac{l_1}{b_1}) \cdots \frac{l_{10}}{b_{10}}$, and the minimum success rate of all neighborhoods, or formally, $\min(\frac{l_1}{b_1})$ Lower variance means that all neighborhoods have equal success rates, so we aim to minimize variance. (We use the lowest, as opposed to the 25th percentile, neighborhood here because there are only 10 neighborhoods.)

4 Policies

We formally define a policy as a function that maps an action (which is a combination of requests) and a driver to a real value score. After rating all all feasible combinations of riders and drivers, an integer linear program (ILP) attempts to maximized the combined score, subject to constraints D. Formally this is represented as

$$\sum_{i=1}^{N} \sum_{f \in F_i} o(f, r_{i,t}) a_t^{i,f} \qquad (1)$$

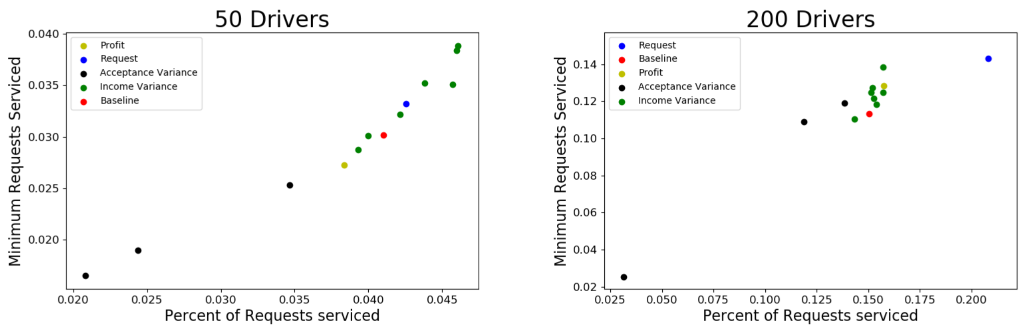

Figure 1: Comparison of Requests serviced vs. Fairness at 50 drivers and 200 drivers. We see that at 50 drivers, Income Variance performs optimally, while at 200 drivers, Request performs optimally.

where o is the policy function. We perform preliminary experiments by varying the objective function, o, and the constraints, D. A complete list of policies from the preliminary experiments are listed in Appendix B.
We compare the following policies:

1. **Baseline** - We use $o(f, r_{i,t}) = |f|$, where $|f|$ is the number of requests in action f. This policy aims to maximize the number of requests without the help of deep learning
2. **Requests** - This policy is from NeurADP [24], and has $o(f, r_{i,t}) = |f|$, but uses deep learning.
3. **Profit** - To maximize profit, let $R = \sum_{u=(s,e,t)\in f} P(E_{s,e})$ denote the profit (reward) from action f. Then $o(f, r_{i,t}) = R$.
4. **Income Variance** - This policy aims to maintain profitability while minimizing driver side inequality by including a term for the variance of income. Defining p as before, let $p_{1,t-1} \cdots p_{n,t}$ denote the income distribution prior to action f, and let $p'_{1,t} \cdots p'_{n,t}$ be the distribution after including action f. $p'_{j,t} = p_{j,t-1} + R$ if $j = i$, and $p'_{j,t} = p_{j,t-1}$ otherwise. Then $o(f, r_{i,t}) = R - \lambda(\text{Var}(p'_{1,t} \cdots p'_{n,t}) - \text{Var}(p_{1,t} \cdots p_{n,t}))$, where λ is a hyperparameter determining how much to weight minimizing inequality vs. utility. For large values of t, p does not change much between epochs, and so p' is a good approximation of p. Note that the inequality term is negative, as we aim to minimize income variance.
5. **Acceptance Variance** - This policy aims to maintain profit while minimizing rider side inequality by including a term for the variance of request acceptance percentage. Let the request acceptance percentages prior to action f be $\frac{l_{1,t}}{b_{1,t}} \cdots \frac{l_{10,t}}{b_{10,t}}$, for each of the 10 neighborhoods, and let the acceptance percentages if we accept action f be $\frac{l'_{1,t}}{b'_{1,t}} \cdots \frac{l'_{10,t}}{b'_{10,t}}$. Then $o(f, r_{i,t}) = R - \lambda(\text{Var}(\frac{l'_{1,t}}{b'_{1,t}} \cdots \frac{l'_{10,t}}{b'_{10,t}}) - \text{Var}(\frac{l_{1,t}}{b_{1,t}} \cdots \frac{l_{10,t}}{b_{10,t}}))$. In other applications, the neighborhoods could be other types of groups, such as ensuring equal treatment for riders with different genders or races.

We use deep learning to learn the value function for each of these policies outside of the baseline policy. We use information about driver location and capacity as inputs to the value function. The details of the Neural Network are in [24].

5 Policy Experiments

We test each of the policies on New York City taxicab data to compare the utility and fairness for each policy. We then explore how different environment parameters affect the performance of each policy by varying the number of riders and drivers. We find which policies perform optimally for certain numbers of riders and drivers.

Experiment Setup

The goal of the experiments is to compare how each policy performs at different numbers of riders and drivers. We compare how the following five policies perform when using New York City taxi data:

Baseline, Requests, Profit, Income Variance, Request Variance. These policies represent a variety of different objective functions, with some optimizing for utility, and others aiming to minimize inequality. We train all policies except baseline for three days and test all policies for one day. We vary the number of riders in $[10, 50, 100, 200]$ and vary the number of riders by downsampling $[\frac{1}{4}, \frac{1}{2}, 1]$ of all riders. We additionally vary the hyperparameter λ for Income Variance and Request Variance policies. We run a grid search over all combinations of environment parameters and policies (with every hyperparameter), to see how policies evolve with a changing environment. The details of the hyperparameters are in Appendix C.

Our metrics of comparison are the amount of profit, percent of rides serviced, distribution of income, and distribution of acceptance percentage.

Effect of varying parameters

We compare how changing the number of riders or drivers affects different metrics for different policies. We list how each parameter affects utility below:

1. **Effect of increasing the number of drivers**: Larger numbers of drivers increases the amount of profit, while the profit per driver fluctuates with an increasing number of drivers. With downsampling at 1, profit is maximized for 10 drivers using the Baseline policy, while for 50, 100, and 200 drivers, the profit is maximized by the Request policy.

   This fluctuation is due to two competing forces: the profit dips due to increased competition, while the profit increases due to an increased number of training samples, improving model performance. When the number of drivers increases to 50, the profit decreases because of the increased competition, but when the number of drivers jumps up to 200, the profit increases because of the increase in training samples.
2. **Effect of downsampling**: Increasing the down-sampling ratio increases the number of riders, and thereby increases the ratio of riders to drivers. When the ratio of riders to drivers is high, the Income Variance policy performs better, while the Request policy performs better when the ratio of riders to drivers is low. The Income Variance policy achieves income equality by giving drivers shorter trips with less profit and giving each driver more trips, which reduces fluctuation in income due to variance in the length of trips. It is easier to give each driver more trips when the ratio of riders to drivers is high, which is why Income Variance achieves more requests serviced than the Request policy when the ratio of riders to drivers is large.
3. **Effect of increasing lambda:** On the driver side, the profit is maximized by minimizing λ for the Income Variance policy, while on the ride side, the requests serviced is maximized by minimizing λ. On the rider side, $\lambda = 10^8$ achieves the most requests, and $\lambda = 10^{10}$ achieves the least requests in all combinations of riders and drivers.

Next, we list how each parameter affects fairness:

1. **Effect of increasing the number of drivers**: As the number of drivers increases, the spread of both the income and the request acceptance percentage increases, due to the larger number of matches between drivers and riders. Increasing the number of drivers also increases the minimum request acceptance percentage, due to more requests being serviced in all neighborhoods. Additional results are given in Appendix C.
2. **Effect of downsampling**: Increasing the down-sampling rate increases the number of riders, which increases income across the board, but leads to lower request acceptance percentages for the worst-off neighborhood.
3. **Effect of increasing lambda:** Larger λ correlates with a higher weight to minimizing inequality for both the Income Variance and Request Variance policies. Higher λ decreases the spread of outcomes but also lowers the utility, leading to worse outcomes for those worst off, in terms of 25^{th} percentile income and lowest request acceptance percentage by neighborhood. This exemplifies the trade-off between utility and fairness, as when the spread of outcomes decreases, so does the utility.

Comparison of Policies

We additionally compare the performance of all policies to see how they perform with differing numbers of riders and drivers. We find that rider side performance can be improved by minimizing driver-side inequality, while driver-side performance cannot be improved through fairness constraints.

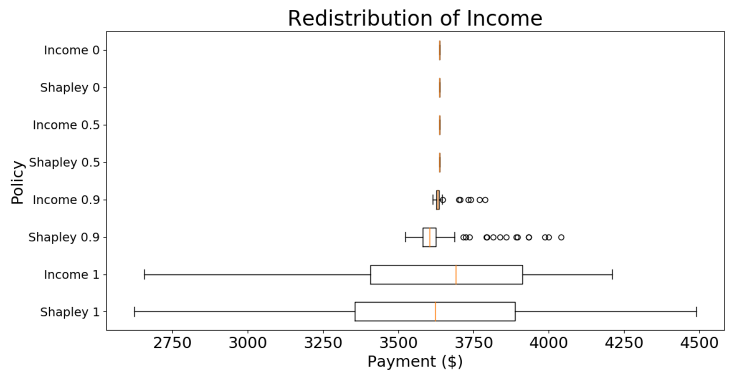

Figure 2: Comparison of various methods to redistribute income. Even at high values of r, the spread of income is small. We also see that the Shapley method tends to decrease the median income, and skews toward people earning high amounts of income.

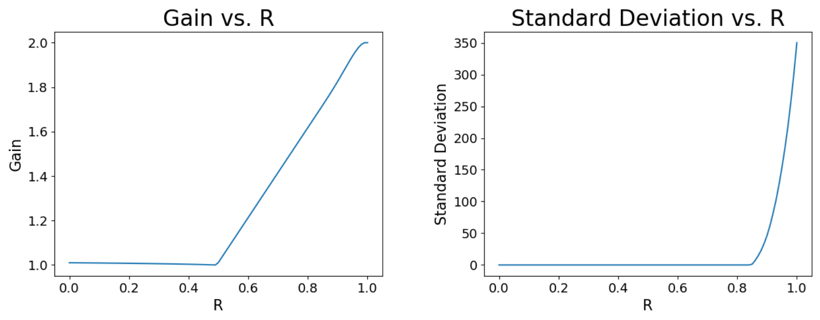

Figure 3: Comparing the gain and standard deviation for different values of r. We find that the region between $r = 0.5$ and $r = 0.85$ has the gain increasing without an increase in the spread of income, meaning that drivers are incentivized to earn more without negatively affecting the income inequality.

This is shown by Income Variance policy achieving more requests at 50 drivers than Request policy (Figure 5) We also find a tradeoff between reducing the spread of outcomes, and improving the worst outcomes, on both the rider and driver sides (see also Figure 5). Higher utility is correlated with better outcomes for those at the bottom, and so policies that achieve higher profit also help raise the income of those at the bottom. The Income Variance and Request Variance succeed in reducing the spread, but suffer consequences in utility, exemplifying the utility-fairness trade-off.

6 Income Redistribution

We propose a second method to encode a variant of driver-side fairness, though instead of encoding it at the matching level, we instead attempt to redistribute income at the end of the day. We essentially view this as a form of profit-sharing, where some of the total profits are redistributed to drivers. Each driver takes a certain commission from each rider, and the rest is redistributed at the end of each day. This also serves as a form of risk sharing, allowing for drivers to be inundated from risk on days where they do not drive as many passengers. This reduces risk for both high earning drivers and low earning drivers, and reduced income fluctuation.

Value of Driver

We introduce two methods of determining the value of a driver, which we denote by A_i, $1 \leq i \leq n$. A_i is a measure of the worth of the driver to the whole system. We consider two definitions for A_i: the first is that $A_i = p_x$, or the profit earned by driver x. The second is that A_i is the Shapley value of driver x, which is a way of measuring the marginal contribution of a driver to every subset of drivers. We will give an example, and define it rigorously below.

As an example of how the Shapley value works, suppose we had 2 drivers, each in the same location, and one rider who pays \$50. Suppose that the rider is matched with driver 1, then the profit driver 1 gets is \$50, while the profit driver 2 gets is \$0. However, the Shapley value of each driver would be \$25, as if any driver left, the other would cover the rider, so they would split the pay equally. We use the Shapley value as a method to determine effort because it smooths out profit from a rider amongst multiple nearby drivers, and rewards drivers who drive in less populated areas, rather than those who simply got lucky and got a lot of requests.

The Shapley Value, $S(x)$ is defined as the marginal contribution of the driver to each combination of drivers: that is, if we let $N = 1 \cdots n$, then

$$S(x) = \sum_{s \subset N \setminus x} \frac{|s|!(n-|s|-1)!}{n!} v(s \cup x) - v(s), \tag{2}$$

where v is the total profit generated by a set of drivers. Note that $\sum_{i=1}^{n} S(x) = \sum_{i=1}^{n} p_i$, that is, the sum of Shapley values is the total profit.

To calculate v, we would need to re-run all the epochs for each subset of drivers, for a total of 1440×2^n matching algorithm iterations, which would be computationally infeasible. Instead, we make some approximations to compute v quickly.

1. We only consider Shapley values on a per-match basis, so we would consider the value of each driver for each epoch. That is, instead of calculating v for each day for all epochs, we instead calculate v for each epoch.
2. Instead of computing the exact value for the Shapley value, we instead approximate it through a Monte Carlo simulation

We consider three different methods to approximate the Shapley value, and find that all three converge quickly to the actual Shapley value (Appendix C), so we using the truncated Shapley method [12].

Formalization

Let A_i denote the value of a driver, which is either the Shapley Value or the actual income. We aim to smooth out the income so that the spread of income is reduced, while still incentivizing drivers to maximize effort. To do this, we consider a risk parameter, r, $0 \leq r \leq 1$, which designates what fraction of their income drivers are willing to forgo to offset risk. When $r = 1.0$, drivers are risk-tolerant, while $r = 0.0$ means drivers are risk-averse.

Our aim is to strike a balance between paying each driver their value and uniformly spreading income across all drivers. We let drivers keep $r \times A_i$, and redistribute the rest of the money, with P_i denoting the amount that each driver gets post redistribution.

Letting the total profit be $T = \sum_{i=1}^{n} p_i$. We then collect all other profits into a central pool which totals to $(1-r)T$. If all income were distributed uniformly, then each driver would get $\frac{T}{n}$, and so we distribute to those who earn less than that (that is, all drivers with $r \times A_i < \frac{T}{n}$

For each driver, we distribute an amount proportional to $\min(0, r \times A_i - \frac{T}{n})$. Because we have $(1-r)T$ overall to redistribute, each driver gets

$$P_i = r \times A_i + \frac{\min(0, \frac{T}{n} - r \times A_i)}{\sum_{i=1}^{n} \min(0, \frac{T}{n} - r \times A_i)} \times (1-r) \times T \tag{3}$$

This redistribution strategy helps all who earn less than the uniform distribution, while proportionally helping those who earned less.

Experiments

We vary the risk parameter, the number of drivers, and the policy used to evaluate how income redistribution performs (Appendix C). Our aim is a fair income redistribution strategy, while still incentivizing drivers to work. We determine if an income redistribution is fair by looking at the 25^{th} percentile income, and the standard deviation of income. We additionally wish to avoid the tragedy of the commons, where agents are incentivized act against the common good, and in this case, incentivized to put in less effort. To determine this, we look at, for a particular driver, what the change in income would be, if they generated twice the value. Let x denote the median driver, then let $A'_x = 2 * A_x$ and $A'_i = A_i, i \neq x$. We define gain to be

$$\text{Gain} = \frac{P'_x}{P_x} \tag{4}$$

which represents their increase in income due to putting in twice the effort. The closer that the gain is to two, the more drivers are incentivized to drive; indeed as if they have twice the value, they get close to twice the profit in the end. We list our observations below:

1. **Effect of increasing the number of drivers** - As the number of drivers increases, the distribution of income tends to spread out more, for a constant value of r. This is because as the number of drivers increases, there is a larger variety of incomes, spreading out the distribution.
2. **Effect of changing the policy** - For all policies, the income redistribution methods have similar effects, as the spread of income shrinks. In the Income Variance policy, the effect is especially pronounced, as the spread of income was already small, to begin with, and adding in income redistribution makes it even smaller.
3. **Effect of changing risk factor** - As the risk factor increases, so does the spread of income. This makes sense, as increased risk factor means there's less in the central pot to redistribute, meaning more income inequality.

We find that even using high values of r, such as $r = 0.9$, significantly reduces the income disparity between drivers. To determine this, we evaluate the effect that changing the r value has upon the Request policy through the aforementioned metrics. We use the Request Policy at 100 drivers because it achieves the most profit, and vary r from 0 to 1 in increments of 0.01. To determine the gain, we take the median driver, and double their wage, and determine the ratio of P_i. Using other drivers, such as the lowest-earning driver, does not change the findings; see also Appendix C.

While the Gain increases linearly after $r = 0.5$, the standard deviation of income only starts to significantly increase around $r = 0.8$. What this means is that $r = 0.8$ has a gain of 1.6 , while still minimizing income inequality, and keeping the 25^{th} percentile income high. Such redistribution strategies are useful in practice to avoid income inequality while also incentivizing drivers to work.

Discussion

We find that the two different definitions of fairness, the 25^{th} percentile income and the standard deviation—in addition to the analogous metrics on the rider side—are generally opposed to each other. This is perhaps unsurprising given the classic and ubiquitous fairness-efficiency tradeoff found in many economic applications. In our setting, policies that reduce spread tend to also lower income for everyone. However, interestingly, policies that aimed to reduce driver side inequality ended up not just reducing the spread of driver side incomes, but also increasing the utility on the rider side (in terms of the number of rides serviced). This implies that introducing some types of fairness into the objective function can potentially also positively impact the utility—at least for some aggregate measures of impact on utility.

The income redistribution strategies can naturally be combined with fairness-focused policies. For example, for small numbers of drivers, the Request Variance policy achieves more profit than other policies, while also achieving high rider side fairness. To add in driver-side fairness, it could be combined with income redistribution so that there is some consideration for both rider-side and driver-side fairness encoded into the policy, while also maintaining a large amount of profit at the system level.

7 Conclusion

With the rise of ride-pooling services arises the importance of considering fairness—toward both sides of the market, riders and drivers alike. By adopting state-of-the-art deep-learning-based policies, we explored policies that increase the number of requests serviced for certain environment parameters, and also increased fairness across groups. We additionally proposed addressing income inequality by using profit redistribution to allocate income and alleviate fluctuation in income between drivers. We developed different methods to redistribute income and found that it is possible to avoid free-riding while keeping income inequality low.

Relevance to Workshop

Our work deals with two different methods of incorporating fairness to ride pooling matching policies. This is done by testing a variety of matching policies that affect which riders and drivers are matched. Our second method is to implement economic policies that redistribute income to avoid large income disparities between riders. This is done by using game theory to assess the contributions of the rider and then redistributes to account for random fluctuations. Our work deals with fair representation in economic outcomes; our goal is to equalize request acceptance percentages across neighborhoods, so that everyone has an equal chance at getting serviced in ride pooling. Additionally, our work tackles the issue of economic equality between drivers by proposing a variety of mechanisms by which inequality on the driver side is reduced. On the machine learning side, our work utilizes reinforcement learning to learn equitable policies that reduce inequality in ride pooling matching algorithms.

Our work has not been published or made available prior to January 1 2017, and is original work.

Potential Ethical Impact of our Work

We address a specific application of the classic fairness-efficiency tradeoff found in numerous economic systems. As discussed in Section 3, our method relies on a *quantitative definition* of fairness. While we did derive our proposed definitions of fairness from reports of inequality in rideshare systems [see, eg., 11, 10, 5, 22], we acknowledge that this was a prescriptive decision and, as technicians rather than domain experts in the greater rideshare space, that decision may not be the proper solution for all settings where rideshare is deployed. Indeed, measuring, defining, and incorporating definitions of fairness into automated systems is an area of extremely active research within the FATE community—and one that is, rightfully, increasingly involving cross-talk between communities outside of AI and machine learning. We view the potential ethical impact of our work as largely positive. Indeed, we confidently make the prescriptive statement that considerations of fairness at *both* the driver and rider level *should* be taken into account in some way in rideshare platforms. That said, as techniques such as those presented in our paper move toward deployment in real rideshare platforms, an open dialogue with stakeholders—e.g., local governments, riders and drivers hailing from various backgrounds—is imperative to understand the desires of those stakeholders [see, e.g., 15].

References

[1] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[2] K. Ando and K. Takase. Monte carlo algorithm for calculating the Shapley values of minimum cost spanning tree games. *Journal of the Operations Research Society of Japan*, 63(1):31–40, 2020.

[3] A. Asadpour, I. Lobel, and G. van Ryzin. Minimum earnings regulation and the stability of marketplaces. *Available at SSRN*, 2019.

[4] E. Bokányi and A. Hannák. Ride-share matching algorithms generate income inequality. *arXiv preprint arXiv:1905.12535*, 2019.

[5] A. E. Brown. *Ridehail revolution: Ridehail travel and equity in Los Angeles*. PhD thesis, UCLA, 2018.

[6] A. E. Brown. *Ridehail revolution: Ridehail travel and equity in Los Angeles*. PhD thesis, UCLA, 2018.

[7] N. Y. City. New york yellow taxzi data, 2016.

[8] S. Cole, D. Stein, and J. Tobacman. What is rainfall index insurance worth? a comparison of valuation techniques. *Mimeo*, 2011.

[9] C. Cook, R. Diamond, J. Hall, J. A. List, and P. Oyer. The gender earnings gap in the gig economy: Evidence from over a million rideshare drivers. Technical report, National Bureau of Economic Research, 2018.

[10] T. R. Dillahunt, V. Kameswaran, L. Li, and T. Rosenblat. Uncovering the values and constraints of real-time ridesharing for low-resource populations. In *Conference on Human Factors in Computing Systems (CHI)*, pages 2757–2769, 2017.

[11] Y. Ge, C. R. Knittel, D. MacKenzie, and S. Zoepf. Racial and gender discrimination in transportation network companies. Technical report, National Bureau of Economic Research, 2016.

[12] A. Ghorbani and J. Zou. Data Shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*, 2019. Full version: arXiv:1904.02868.

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

[14] F. H. Gutierrez. *Labor contracts and risk sharing*. World Bank, Washington, DC, 2013.

[15] K. Holstein, J. Wortman Vaughan, H. Daumé III, M. Dudik, and H. Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *Conference on Human Factors in Computing Systems (CHI)*, pages 1–16, 2019.

[16] D. Krueger and H. Uhlig. Competitive risk sharing contracts with one-sided commitment. *Journal of Monetary Economics*, 53(7):1661–1691, 2006.

[17] N. S. Lesmana, X. Zhang, and X. Bei. Balancing efficiency and fairness in on-demand ridesourcing. In *Advances in Neural Information Processing Systems*, pages 5309–5319, 2019.

[18] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pages 983–994, 2019.

[19] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783. ACM, 2018.

[20] A. Mehta et al. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

[21] P. Nanda, Anant Xu, K. A. Sankararaman, J. P. Dickerson, and A. Srinivasan. Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours. In *Conference on Artificial Intelligence*, 2020.

[22] A. Pandey and A. Caliskan. Iterative effect-size bias in ridehailing: Measuring social bias in dynamic pricing of 100 million rides. *arXiv preprint arXiv:2006.04599*, 2020.

[23] J. Prassl and M. Risak. Uber, taskrabbit, and co.: Platforms as employers-rethinking the legal analysis of crowdwork. *Comp. Lab. L. & Pol'y J.*, 37:619, 2015.

[24] S. Shah, M. Lowalekar, and P. Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *Conference on Artificial Intelligence (AAAI)*, 2020. Full version: arXiv:1911.08842.

[25] T. Sühr, A. J. Biega, M. Zehlike, K. P. Gummadi, and A. Chakraborty. Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3082–3092. ACM, 2019.

[26] Uber. Chicago: An uber case study. 2015.

[27] O. Wolfson and J. Lin. Fairness versus optimality in ridesharing. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pages 118–123. IEEE, 2017.

[28] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913. ACM, 2018.

A Simulating Requests

To simulate and match requests between riders and drivers, we use a technique [1] to generate feasible trips between riders and drivers. Each combination is scored according to the policy used, which is denoted by the objective function $o(s_t, a_t)$, which scores some action (which is a matching of a set of requests and a driver) for a state at time t. We then use an Integer Linear Program to match riders to drivers.

If we let F_t^i represent all feasible matchings for driver i at time t, where each matching, $f \in F_t^i$ is a set of requests matched to driver i (including the null matching). We let $a_t^{i,f}$ denote an indicator value, determining whether the set of requests f is matched to driver i at time t. We aim to maximize:

$$\max \sum_{i=1}^{N} \sum_{f \in F^i} a_t^{i,f} o(s_t, f) \tag{5}$$

$$Subjectto \forall i, \sum_{f \in F^i} a_t^{i,f} = 1 \tag{6}$$

$$\forall j \in U_t, \sum_{i=1}^{N} \sum_{f \in F^i} a_t^{i,f} \leq 1, \tag{7}$$

in addition to the additional constraints in D. Those constraints include the following:

1. One action per vehicle - $\forall i, \sum_{f \in F^i} a_t^{i,f} = 1$
2. One request, one action, $\forall j \in U_t, \sum_{i=1}^{N} \sum_{f \in F^i} a_t^{i,f} \leq 1$
3. Delay - All requests must be serviced within time $w$nnn. In our case $w = 300$.
4. Capacity - No driver can be filled past capacity - $\forall i, c_i + \sum_{f \in F^i} a_t^{i,f} |f| \leq m_i$, where $|f|$ is the number of riders in request f. We assume $m_i = 4$ for our experiments.

We experiment with several policies in this paper that attempt to ensure fairness through additional constraints (see Appendix B). Any request not matched is dropped. After matching, we recalculate motion, and employ an algorithm [1] for empty drivers. At the end of each epoch, we calculate the profit of each driver, $p_{i,t}$, the locations accepted, L_t, and all the locations of requests, B_t. After all epochs, we aggregate the data, and calculate profit per driver as $p_i = \sum_{t=1}^{1440} p_{i,t}$, total profit as $p = \sum_{i=1}^{N} p_i$ and overall acceptance percentage as $\frac{\sum_{i=1}^{1440} |L_i|}{\sum_{i=1}^{1440} |B_i|}$

B List of policies

We perform preliminary experiments with the following policies. We split up our policies into those that include deep learning, and those that don't.

Without Deep Learning

We will enumerate all the policies without deep learning.

1. **Baseline**: Our baseline policy aims to maximize the number of requests, or in other words, $o(f, r_{i,t}) = |f|$, where $|f|$ is the number of requests in action f.
2. **Nearest Driver**: This policy aims to match drivers with the closest driver by giving higher scores to closer drivers. Formally, this is $o(f, r_{i,t}) = \frac{1}{\min_{u=(s,e,t) \in f} E_{s,P_{i,t}}}$. Recall that $P_{i,t}$ is the position of the driver, and s is the starting point for request $u \in f$, so the policy is inversely proportional to the closest request.
3. **Income Entropy**: Consider rider incomes $p_{1,t}, p_{2,t}, \ldots, p_{n,t}$. The entropy of $p_{1,t}, p_{2,t}, \ldots, p_{n,t}$, written as $S(p_{1,t}, p_{2,t}, \ldots, p_{n,t})$, is indicative of the spread of the profits; the higher the entropy, the larger the variation in income. We therefore aim to minimize entropy in order to reduce inequality, while maximizing profit. We use the generalized entropy index with $\alpha = 0$. For a particular feasible action f, the total profit (or reward) from

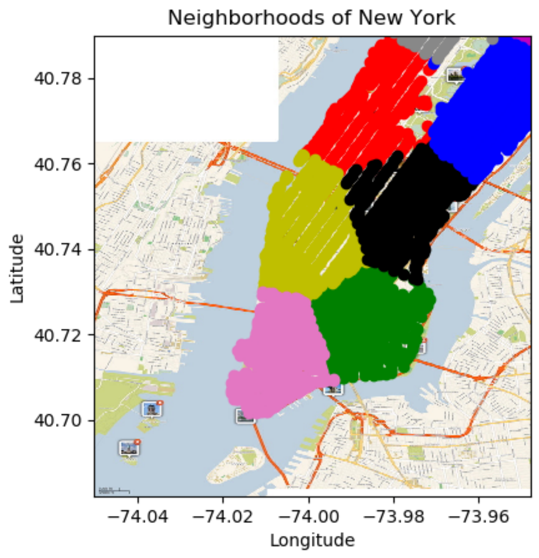

Figure 4: The neighborhoods of New York City. Note that only some of the neighborhoods are shown, and that further neighborhoods are situated North.

f is $R = \sum_{u=(s,e,t)\in f} P(s,e)$, where P is the profit function. After accepting action f, the total profit of the system goes up by R, while the total entropy of the system changes by $S(p_{1,t} \dots p_{i,t} + R \dots p_{n,t}) - S(p_{1,t} \dots p_{n,t})$. which we represent as $\Delta S(p_{1,t} \dots p_{n,t})$. Our objective function is a linear combination of the change in profit and change in entropy, regularized by the hyperparameter λ, or in other words, $o(f, r_{i,t}) = R - \lambda \Delta S(p_{1,t} \dots p_{n,t})$. Higher values of lambda skew the objective function towards lower entropy profit distributions, while lower values of lambda skew the objective function towards higher profit.

4. **Income Variance**: We aim to minimize income inequality by minimizing the variance of the income. As before, $R = \sum_{u=(s,e,t)\in f} P(s,e)$, and we let Var represent the variance function. Similar to the entropy case, our objective function is $o(f, r_{i,t}) = R - \lambda \Delta \text{Var}(p_{1,t} \dots p_{n,t})$.

5. **Request Entropy**: To allow for drivers to service all neighborhoods equally, we aim to minimize the spread of request acceptance percentage. The request acceptance percentages are written as $\frac{l_{i,t}}{b_{i,t}} 1 \leq i \leq 10$, for each of the 10 neighborhoods. One method of minimizing the spread is to minimize the entropy of the request acceptance percentages. We let $\frac{l_{i,t}}{b_{i,t}}$ be the acceptance percentages prior to accepting action f, and $\frac{l'_{i,t}}{b'_{i,t}}$ be the acceptance percentages after accepting action f. Therefore, our objective function is $o(f, r_{i,t}) = R - \lambda(S(\frac{l'_{i,t}}{b'_{i,t}}) - S(\frac{l_{i,t}}{b_{i,t}})$.

6. **Request Variance**: Similarly, to minimize the spread of request percentages, we aim to minimize the variance. This makes the objective function $o(f, r_{i,t}) = R - \lambda(\text{Var}(\frac{l'_{i,t}}{b'_{i,t}}) - \text{Var}(\frac{l_{i,t}}{b_{i,t}})$

7. **Income Hard Constraint**: When matching, we add an additional constraint to enforce income inequality. We sort the profits $p'_{1,t} \dots p'_{n,t}, p'_{1,t} \leq p'_{2,t} \leq \dots p'_{n,t}$, and add a constraint that $p_{n,t} - p_{n/10,t} \leq \lambda * p_{n/10} + 500$, where λ is a hyperparameter. This requires that the largest income and the 10th percentile income have to stay close, with larger values of lambda allowing for more inequality, while smaller values of lambda allow for less inequality.

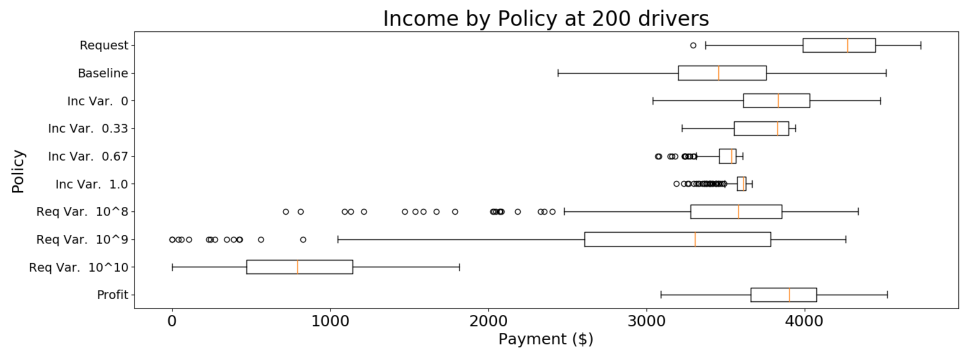

Figure 5: Distribution of income for a variety of policies. We see that Request policy achieves the most profit, while Income Variance policies tend to shrink the spread of income

Neural Policies

We use Neural Networks to develop policies that estimate the value function for a state using the data from matches. We use information about driver location and capacity as inputs to the value function. Our policies using neural networks vary in their objective function, as well as using additional inputs for the neural network. We list policies that use deep learning

1. **Request**: We use the Request value function as one policy [24]. This policy uses an objective function of $o(f, r_{i,t}) = |f|$, prioritizing the number of requests serviced, and is essentially the baseline policy with a neural component.
2. **Income Entropy**: This is the neural version of the equivalent non-neural policy.
3. **Income Variance**: This is the neural version of the equivalent non-neural policy.
4. **Request Entropy**: This is the neural version of the equivalent non-neural policy.
5. **Request Variance**: This is the neural version of the equivalent non-neural policy.
6. **Profit**: This policy optimizes only for profit, and is similar to Request policy, except for profit

Additionally, for the income entropy and variance policies we add an extra input to the neural network, called "driverz", which calculates the z-score for the income of a driver.
After performing preliminary experiments on these policies, we find that Variance based policies achieve higher fairness than those involving entropy.

C Experimental Results

Hyperparameters

We run experiments to investigate the performance of each policy with a variety of λ. We list hyperparameters for each policy

1. **Baseline**: We run the baseline policy without training (as it's a non-neural policy), and testing for 1 day.
2. **Requests**: We run the requests policy by training for 3 days and testing for 1 day.
3. **Income Variance** - We run the Income Variance policy by training for 3 days and testing for 1 day. We use $\lambda = \{0, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, \frac{6}{6}\}$
4. **Request Variance**: We run the Request Variance policy by training for 2 days and testing for 1 day. We use $\lambda = \{10^8, 10^9, 10^{10}\}$
5. **Profit**: We run the Profit policy by training for 3 days and testing for 1 day.

Experiments

We find that as the number of drivers increases, the request policy achieves more profit. We plot the distribution of income for each of the 4 levels of drivers with downsampling at 1.

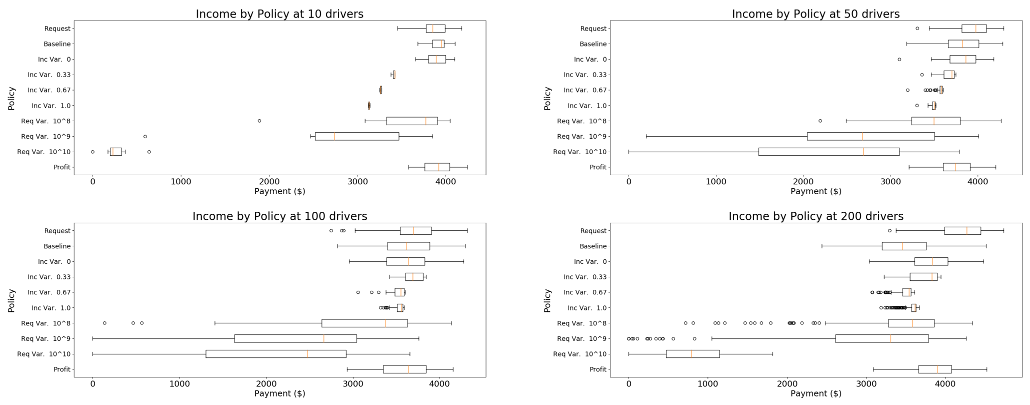

Figure 6: Plots of income distribution with varying numbers of drivers

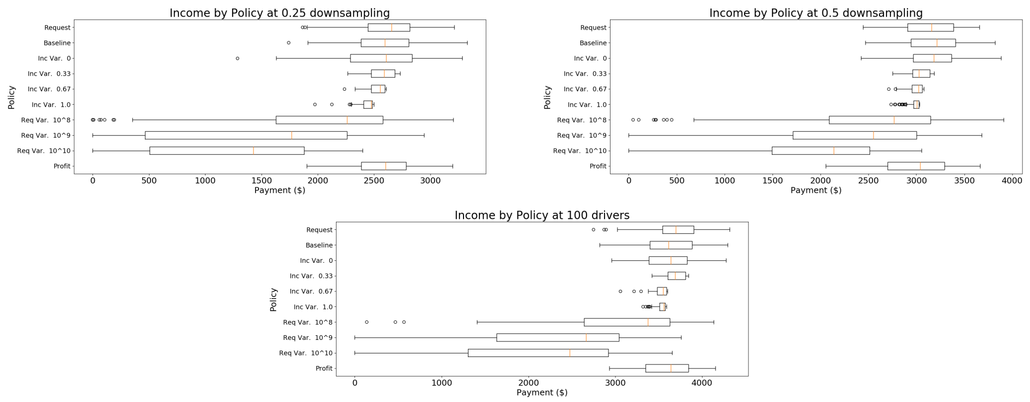

Figure 7: Plots of income distribution with varying amount of downsampling

We also note that the spread of income decreases with increasing lambda when using the Income Variance policy. Also, note that the spread of the Request policy increases as the number of drivers increases.

We additionally plot the income distribution by changing the downsampling rate. We find that as the downsampling rate increases, there is more profit due to more riders being available for drivers. The plots use 100 drivers and vary the downsampling rate.

Shapley Experiments

We try three different methods of approximating the Shapley value. The first is from the truncation method presented by [12]. The second, "random," is done by randomly simulating twice for each driver over a certain number of iterations, once with the driver, and once without. The third is due to [2]. These values are averaged to predict the Shapley value for each driver. We compare these with the true Shapley value (on a small number of drivers, where we can manually compute the Shapley value), and find that both converge relatively quickly. We then compare how quickly they converge for large values of the number of drivers, and find that error decreases rapidly. Figure 8 shows the empirical convergence of the approximation.

When approximating the Shapley distribution, we use each of the 3 algorithms, and stop after reaching $2n$ iterations.

Redistribution Experiments

For Neural Policies, we use the following hyperparameters: $\lambda = \frac{1}{3}$ for income variance, and $\lambda = 10^8$ for request variance, and train each for 3 days. We then see how varying environment parameters affect the distribution of income. We compare the distributions of policies at different number of

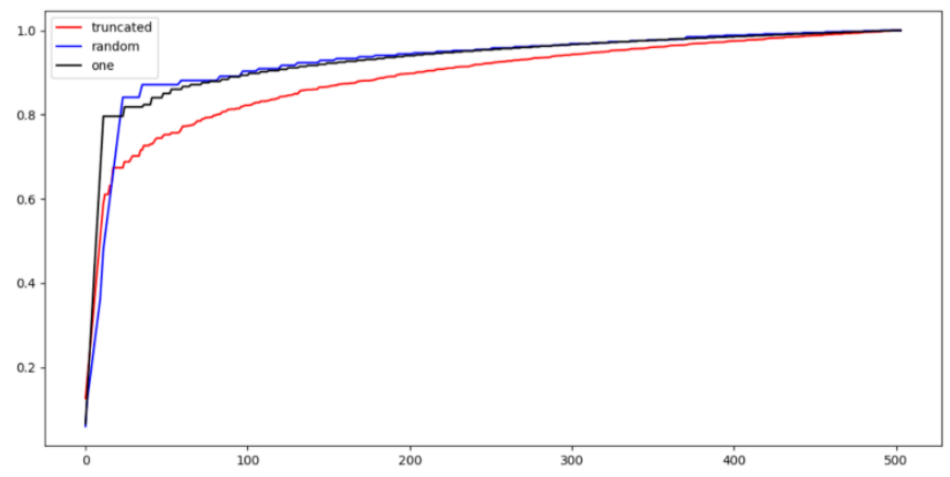

Figure 8: Convergence of computational approximation of Shapley value to the true, exhaustively-computed Shapley value.

drivers, and find that the distribution of Shapley values tends to widen with an increasing number of drivers.

Comparing Gain for Different Drivers

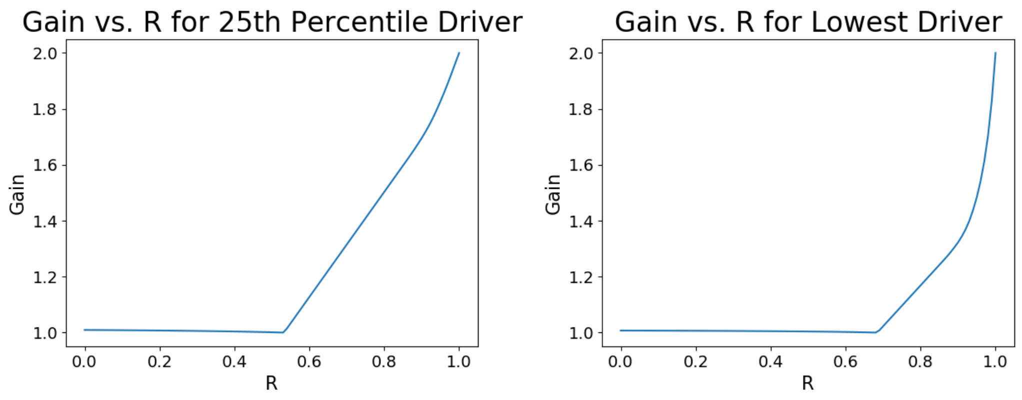

Figure 9: Gain vs. R using the 25^{th} and lowest drivers

Figure 9 plots gain versus R for the 25th percentile and lowest drivers. We exhibit that the distributions are similar to the distribution using the median driver, with the range of values for which the gain increases varying based on which driver is used.

D Related Work

We build on prior work in matching in Rideshare and ride-pooling, fairness in Rideshare, and risk-sharing. Past work on matching helps us develop a framework to evaluate policies, and we use work on fairness and risk sharing to construct new policies and methods of income redistribution.

Matching in Rideshare

Prior work has viewed the rideshare and ride-pool matching problems as an instance of the Markov Decision Process framework. Much of this work has approached the problem from an online matching perspective, where deep learning is used offline to learn the value of the state, and is used to optimize matches online [19, 18]. To deal with the complexity of the ride-pool matching problem, past work has used Approximate Dynamic Programming has been used in conjunction with deep learning to efficiently match riders and drivers [24]. Additionally, there have been algorithms that deal with what unmatched drivers should do [1].

Fairness

Fairness in rideshare and ride-pooling has been studied from both an algorithmic perspective and a social science perspective. Past research discusses a trade-off between profit and certain definitions of fairness in rideshare, which can be adjusted through a parameter [21, 17]. These works also prove bounds on how much profit can be sacrificed for a certain amount of fairness. Our work builds on these past work by considering the empirical aspect, and also considers fairness among both drivers and riders. Additionally, past work has looked at matching riders with other preferred riders, while still maintaining optimality [27].
While rideshare companies have increased the rate of service for passengers [26], fairness amongst drivers is still an issue. There have been reports of disparate treatment by rideshare companies for certain sub-populations in terms of differences in wait-time between black and non-black riders [6]. There are also cases where rideshare drivers cannot achieve a liveable wage due to income inequality in rideshare [13].

Risk Sharing

There has been a variety of scenarios where risk sharing is used as a mechanism to avoid fluctuations in risk. Rainfall insurance is used to account for income loss due to fluctuations in how much it rains [8]. This is done by equalizing the amount of utility while reducing the variation in outcomes. Risk-sharing has also been done through contracts to minimize risk in labor [14]. These contracts have been shown to help achieve equilibrium in certain marketplaces [16].